Digital learning platform for waste-pickers in Brazil

Building a microservices based distributed web system for creating and presenting digital learning material

Daniel Britze, Jacob Vejlin Jensen

Computer Engineering, May 26, 2021

Bachelor thesis

STUDENT REPORT

AALBORG UNIVERSITY

AALBORG UNIVERSITY
STUDENT REPORT

Computer Engineering
Aalborg University
http://www.aau.dk

Title:
Digital learning platform for waste-pickers in Brazil

Theme:
Full stack development

Project Period:
Spring Semester 2021

Project Group:
CS6-654

Participant(s):
Daniel Britze
Jacob Vejlin Jensen

Supervisor(s):
Jens Myrup Pedersen

Copies: 1

Page Numbers: 112

Date of Completion:
May 26, 2021

Abstract:

This thesis presents an engineering solution for developing digital learning material through an education platform. Initially, availability of smartphones and learning needs for individuals in low socioeconomic communities, are examined. Based on the examinations, the situation of Brazilian waste pickers struggling with financial viability after forced transition into payroll-based employment is chosen as case for the project. Subsequently, the engineering methodologies applied are discussed and a solution consisting of both frontend and backend services is conceptualised. Requirements for the solution are defined and a technical analysis is conducted to make foundation for software implementation. The system design and implementation are documented and put through acceptance testing. Conclusively, a first version of a distributed web system enabling both creation and presentation of digital learning material is developed.

AALBORG UNIVERSITET
STUDENTERRAPPORT

Internet teknologier og Computersystemer
Aalborg Universitet
http://www.aau.dk

Titel:
Digital læringsplatform for skraldere i Brasilien

Tema:
Full stack udvikling

Projektperiode:
Forårssemesteret 2021

Projektgruppe:
CS6-654

Deltager(e):
Daniel Britze
Jacob Vejlin Jensen

Vejleder(e):
Jens Myrup Pedersen

Oplagstal: 1

Sidetal: 112

Afleveringsdato:
26. maj 2021

Abstract:

Denne bacheloropgave præsenterer en løsning til udvikling og præsentation af digitalt læringsmateriale igennem en uddannelsesplatform. Initierende, undersøges adgangen til smartphones og læringsbehov for individer tilhørende lavere socioøkonomiske grupperinger. Baseret på disse undersøgelser, udvælges brasilianske skralderes økonomiske kamp efter konvertering til nye lønvilkår, som projektets case. Efterfølgende diskuteres de ingeniørfaglige metoder anvendt i projektet og en løsning bestående af både frontend og backend services konceptudvikles og fremlægges. Krav til løsningen opsættes og en teknisk analyse skaber basis for efterfølgende software implementering. Systemdesign og implementering dokumenteres og accept-testes op imod specificerede krav. Afslutningsvist, præsenteres en første version af et distribueret web system hvorigennem brugere kan oprette og præsentere digitalt læringsmateriale.

Contents

Preface

The authors, Daniel Britze and Jacob Vejlin Jensen, chose to write about the subject of improving digital learning opportunities in low socioeconomic communities, due to previous research experiences. Daniel Britze was part of the first project of the Mobile Education Projects in 2019, where he was introduced to the situation of the Brazilian waste pickers and conducted both desk and field research of the subject. Jacob Vejlin Jensen was part of a project in 2019 that worked with expanding digital learning opportunities for children living in townships in South Africa, where he conducted both desk and field research. Additionally, both authors have participated in multiple international and cross-disciplinary student collaborations working with the UN's sustainable development goals and acquired insights into management and structure of such collaborations. This experience is applied in this project by acting as managers and coordinators of the collaboration with student groups from University of Brasilia. Alongside this bachelor thesis a research article was written about the collaborative structures and sustainable vision in the project, published at the 2021 PAEE/ALE conference.

Aalborg University, May 26, 2021

Daniel Britze
<dbritz17@student.aau.dk>

Jacob Vejlin Jensen
<jvje17@student.aau.dk>

Chapter 1

Introduction

In todays digitization climate, an exceedingly substantial amount of people have access to smartphones, including those whom people traditionally would not expect to. During the Syrian refugee crisis [55] to Europe, a common way of dismissing the refugees as not being so, was due to them having smartphones [39]. Regardless on the invalidity of such claims, considering even broader terms, smartphones are becoming evermore ubiquitous around the world. For example in Brazil where 85% of the population between 18 and 34 have smartphones [46]As found by the Pew research center, and later qualitatively confirmed with boots on the ground research provided by students from University de Brasilia (UnB) [12], even those at the economic bottom, have access to smartphones.

In many ways, it can be argued, that digitization has changed the educational landscape, expanding the amount of people with access to information. However, with the rise of terms such as "fake news" this rising digitization appears as a tool for both good and bad. On the positive side, individuals are able to access much a lot of educational content as today via these digital technologies. An example of this being the Greenshoots NGO, trying to provide digital learning for grade schools around south Africa [45] But this change has not come without its fallacies.

Current social structures affect the unequal distribution of digital engagement, which influences some individuals' opportunity to acquire essential knowledge, improve skills and increase well-being. Most of the available digital educational services, focus on a user group within the middle-or upper social classes, where most individuals have proper reading and writing skills, stable internet connections as well as some monetary capital. Hereby the groups of people falling short of these requirements, are often left to either outdated learning methods, or often a disregard of learning entirely.

These are the people, that arguably are in most crucial need of education. Improvements within mathematics, personal finances, health and reading/writing skills, can help people to get jobs, take better care of themselves and their families, and bring new life opportunities, that ultimately help decrease the existing inequality. In some cases, even basic education, can be a deciding matter in lifting people from poverty.

1.1 EPIC SDG Challenge initiative & previous research

Aware of the challenging scenario regarding the waste pickers conversion to payroll employees at sorting facilities, Aalborg University and University of Brasilia committed to a cross-disciplinary collaboration, the Mobile Education Project (MEP), as part of the EPIC SDG Challenge (SDGC) initiative in 2019. The ambition of the project is to combine engineering disciplines and cross-cultural approaches in PBL based student projects, to minimize the educational gaps within waste picking communities and contribute positively to the financial security and life quality of the Brazilian waste pickers through digital learning.

Two of the projects carried out by previous student groups are: (i) Mobile Education Platform [12] and (ii) Mobile Education Platform: Finance management for waste pickers [41]. In project (i), the students identified broadband quality as a prime issue of distributing digital learning in the waste picking communities. To comply with this challenge, the student project proposed a Smart Caching system for Android smartphones, that enables offline access to digital educational content. In project (ii), the students continued upon the foundation established by project (i) and identified that especially financial education was critical to the target demographic. The project proposed and developed a mobile application dedicated to financial management learning for the waste pickers with focus on the user experience, system interaction and usability design.

This project aims to combine the outcomes of the two previous student projects by solving the issue of large-scale distribution of digital education within low socioeconomic communities. Additional details about the integration of cross-disciplinary collaboration between student groups from both universities is covered in Section 4.1.

Chapter 2

Problem Analysis

This chapter initially identifies main issues within the problem domain of distributing digital learning opportunities in lower socioeconomic communities. With basis in the concrete use case of uneducated Brazilian waste pickers transitioning into payroll based employment, Section 2.1.1 researches the adoption of smartphone technology within the demographic group. The rate of smartphone adoption is crucial for the baseline viability of digital learning within the user group, as mobile devices greatly expands the possibilities of creating learning material and increases accessibility. In Section 2.1.2, Section 2.1.3 and Section 2.1.4 additional research further explores the problem domain and target demographic which conclusively makes the basis for an initial problem statement in Section 2.1.5.

Section 2.2 examines how digital education could be specialized for the target demographic by conducting and analyzing interviews with previous waste pickers in Section 2.2.1. Additionally, currently available digital educational services are explored in Section 2.2.2 and their functionality compared with learning needs of the target demographic group. Conclusively, the analysis provides the basis for the problem statement in Chapter 3.

2.1 Problem Identification

This section explores the problem domain with aim to identify key issues related to the distribution of digital learning.

2.1.1 Smartphone availability

Access to smartphones is increasing globally in low socioeconomic communities enabling new demographic groups to benefit from opportunities such as digital learning[46]. Whilst a few years ago, many would have scuffed the idea due to obstacles such as usability issues, networking and user commitment, it has become increasing prevalent and due to the corona virus outbreak of 2020 it has become almost ubiquitous [25]. Digital classrooms and zoom meetings have become commonplace at workplaces and educational institutions and online learning sites such as Udemy, Khan Academy and Coursera continues to increase the amount high quality material available to users. However, this plethora of digital education is not necessarily a great equalizer but rather dividing based of a few unfortunate factors. In the west, a tendency to take ground schooling as a guarantee leaves a blind spot for those individuals in need of basic skills such as reading, writing and mathematical operations.

In a 2018 study[46], the pew research center found that smartphone adoption rates are on an exponential rise within emerging economies such as Brazil. Using Brazil as the example, 85 percent of adults between 18 and 34 were found to have smartphones, and this percentage is rising quickly. However this still leaves 15 percent without smartphones, and if it is those with lower socio-economic status, this would nullify the idea of a specialized learning platform utilizing the strength of mobile applications. In a collaborative effort with partner student groups from University of Brasilia (UnB), as explained in Section 1.1, interviews with 25 previous waste pickers now employed at sorting facilities is conducted. From this research it is found that 84,7% of the participants owns a smartphone. Though this small study cannot be regarded as a definite and generalized expression for smartphone adaption within low socioeconomic communities, it is sufficient for the aims of this project as a supporting argument. The result is noticeably close to the results from the Pew Research Center and therefore the hypothesis of widespread smartphone availability within the target demographic group is accepted.

2.1.2 The digital learning gap

Current selection of digital educational services reflects mostly higher level content that presumes fundamental skills such as reading, writing and basic mathematics. This is exemplified within Section 2.2.2, wherein current digital learning solutions are examined. Effectively, this creates a disparity between those able to benefit from the current selection of courses and those lacking the necessary skills to do so. Additionally, individual learning needs are highly dependent on cultural, societal and economic circumstances of individual learners. Therefore, there's a need for increasing the available digital education services, built with the learning needs of individuals living in lower socioeconomic communities, such as the Brazilian waste pickers, in mind.

A service designed for western resourceful individuals looking to advance various skills will simply not suffice for learners from contrasting demographic groups. This argument is supported by research done in previous student projects, where it was found that a digital learning solution for the waste pickers in Brazil, should have a completely different structure of content (visualization, user involvement, language etc.) to motivate and cultivate effective learning[12][41]. If learning content and platform is not sufficiently specialized to the needs and circumstances of the target demographic, there is a risk of alienating or demeaning learners, instead of supporting their learning. Therefore, it is of crucial importance to conduct solid research into the target demographic group when working with digital learning and involve first-hand users and stakeholders with local knowledge of the group, in the process.

2.1.3 Understanding learning needs of the target demographic

As previously described, this project aims to cultivate digital learning opportunities for individuals of lower socioeconomic status. Whilst this could include several user groups, due to the different needs of each group, the target demographic needs to be further specified. Learners reaction towards different teaching methods and learning content may vary greatly across even slight differences in culture, language, society etc.

Therefore, the focus of this project, is not to cultivate learning for user groups in the described demographic, but to demarcate the scope to the individuals in Brazil, transitioning from lives as waste pickers to employers at recycling facilities. This is a choice made, aware that it is just one of the user groups in need of advancements in digital learning opportunities. However, due to the possibility of continuing progress made in previous student projects and the collaborative aspects described in Section 1.1, deemed a use case with the possibility of doing real help. As a result, the target demographic is now delimited, and creating a digital service specialized to the learning needs of this specific user group is established as the focal point of this project.

2.1.4 Locals teaching locals

Developing a digital learning service requires many crucial considerations. The platform must be made to both accommodate the users potential reading / writing deficiencies and inspire them to learn without appearing demeaning or degrading. Additionally, it is important that the user interface appeals to the learners in a way that supports their preferred methods of learning. This includes finding a balance between elements such as video, image and text, but also to the specific linguistic properties of the material as well as supporting motivation through features such as gamification etc. As described in Section 2.1.3 this requires involvement of local stakeholders and first-hand users, with aims to consider cultural, societal and personal circumstances and have that reflect the development.

It is important to acknowledge that in a user centered product, a misalignment between target demographic and the digital platform can result in impactful misunderstandings. Involving users on a very local and small scale is therefore extremely crucial for the success of the project. Because learning needs for different individuals even within the specified target group, may still differ greatly, a general philosophy of "Locals Teaching Locals" is adopted. To avoid engineering a solution based on personal assumptions about the target group, this philosophy is about involving users and local stakeholders as much as possible, and make sure that learning content will be created by locals for locals.

Based on the discussions thus far an initial problem statement is formed.

2.1.5 Initial problem statement

How can digital learning be used to support baseline educational learning for the Brazilian waste pickers transitioning into employment at the local recycling centers?

2.2 Problem Examination

Having identified the problem of digital education to those with lower socioeconomic status a problem examination is done. As described in the problem identification, different use cases might bring about more specialized requirements. Bearing this in mind, two use cases are examined further: one in Brazil and one in South Africa. In both cases, the connectivity is also explored, as it showcases the viability of the education system being smartphone based.

With conclusions drawn from the use cases, a competition analysis is commenced, allowing insight into deployed solutions and further examining how well they fulfill the different use cases. As a result of this problem examination, subsidiary conclusions are drawn, leading to the problem statement.

2.2.1 Case study: Brazilian waste pickers

As described in the problem identification, the focus case within this project is waste pickers in and around Brasilia, Brazil, as representative for a demographic of individuals of lower socioeconomic status. This target group is chosen due to several factors, including previous knowledge of said use case in addition to the collaboration between Aalborg University (AAU) and the University of Brasilia (UnB)Section 1.1. As described in [12], the waste pickers have gone from literally picking waste from the largest dumpsite in Latin America, to working within waste sorting facility cooperatives. Having completed this transition from a dangerous working environment into more stable jobs, the aim is to now elevate the educational status of the waste pickers, hopefully enabling them to progress in the socioeconomic ladder.

As described earlier, due to familiarity of the case, several important factors are described in [12]. Analysis of interviews conducted with previous waste pickers now working in a sorting facility, showed that illiteracy was frequent amongst the employees but that using smartphones were an integrated part of everyday life. A key point being their use of voice via WhatsApp [12], wherein they would utilize the record and send feature to facilitate communication.

Additionally, desk research concluded that the wages at the sorting facilities are below estimates of value from waste picking, however still livable [12]. The research concludes that the economic situation of the sorting facility employees was partly due to lack of knowledge about payroll, financial management and savings - something that could be helped through proper education. Though the conclusions from this report are foundational for this project, the research suffered from issues such as limited interview persons and mainly desk research. Therefore, the aim is to conduct more in-depth research into the learning needs of the employees at the Brazilian sorting facilities in this project, with a focus on expanding results from field research (interviews).

Due to the COVID-19 pandemic during the period of this project, it has been impossible to arrange physical travels to Brazil for the field research. Luckily, since the project is part of the UnB/AAU collaboration (see Section 1.1), the responsibility of conducting additional field research is adapted by student groups from UnB. The collaboration includes multiple student groups with different perspectives and responsibilities, which is further detailed in the methodology chapter Section 4.1.

Student groups from UnB conducted interviews with a total of 25 waste pickers and 5 cooperative leaders during the project. From analyzing responses from a questionnaire it was found that 65.3% of the participating employees from the sorting facilities have not completed elementary school. Most of these 65.3% additionally expressed trouble reading / writing / interpreting, indicating that differentiating levels of illiteracy amongst learners must be considered in the project.

When asked about their preferences with regard to learning methods, 80% of the survey respondents answered that they considered a learning mobile application a viable alternative to in class learning. While this is undoubtedly an indication that digital learning on mobile devices could be part of a solution, interviews with the cooperative leaders also showed that previous experiments with digital and mobile learning has highlighted issues such as attention span, motivation and certification, has resulted in low engagement and poor performance. These issues also came up during an attempt to use in class learning, which resulted in low interest from the waste pickers.

To address this issue, one of the leaders suggested that a mobile learning experience should focus on short bursts of interactive and visual content, leaves the learner with a continuous feeling of progression. Combined with an approach to creating the educational material aligned with the "locals teaching locals" philosophy explained in Section 2.1.4, mobile learning might be able to achieve better engagement and performance for learners.

Partial conclusions

In conclusion, the qualitative field research conducted by partner student groups from UnB, has come to the following main conclusions:

- Most employees at the sorting facilities have access to smartphones and consider mobile learning a viable option for education
- A mobile learning platform for this target group must address issues of motivation, engagement and certification
- The majority of the sorting facility employees have little to no education and therefore lack knowledge within fundamental areas such as reading, writing and basic mathematics

To further qualify understanding of the problem domain and avoid creating a solution that already exist, the next section researches currently available online learning services.

2.2.2 Comparative analysis of digital learning services

To better understand the features necessary for a mobile based educational platform, this section examines functionality aspects of 7 popular digital learning frameworks. The frameworks are selected by subjective intuition about relevance based by mentions, search results and personal familiarity, and therefore cannot be regarded as a comprehensive analysis but as an indicative analysis for extraction of core and specialized features. Additionally, it gives the opportunity to obtain an overview of what exist and how features of the different platforms can be combined to create something new. Supplementary to some of the popular digital learning frameworks, the analysis also compares the functionality from the two previous student projects, working on the issue. (see [12] and [41])

The frameworks included in the comparative analysis are:

- Student project (Computer Engineering (CE)) [12]
- Student project (Product and Design Psychology (PDP)) [41])

- Khan Academy
- Udemy
- Moodle
- Rise Articulate
- Kolibri

The following sections details key functionality and purpose for each of the seven services and Section 2.2.3 summarizes conclusions derived from the analysis.

Student project (Computer Engineering (CE))

During the spring of 2019 the original mobile education project was commenced with the same goal of providing education to the Brazilian waste pickers [12]. Throughout the original project, the problem analysis showcased a need for it to be on an android based smartphone with considerations given towards connectivity. To solve this connectivity issue wherein the waste pickers do not have data on their cellphones as noted in [12], the idea of "Smart-caching" was conceived. The fundamental concept being the app recognizing whenever the user was on Wi-Fi and then calculating an amount of course material to automatically download.

The course material itself consisted of four elements: Text, Image, Audio, Video with test being declared out of scope while still valuable to the waste pickers. These types of learning materials are the core of the application with smart-caching being an extra feature. The courses themselves where to be made on the Moodle platform [12] and then automatically scraped into a database.

The final product allowed for a course to be made and put on the android smartphone via an app, however the smart-caching was never implemented. Additionally, the application was in a very early prototype stage, made even more complicated by the use of Moodle as the course creator.

Student project (Product and Design Psychology (PDP))

During the spring of 2020 a group of Product and design psychology students looked into details in creating a focused financial education app for the waste pickers [41]. Their study underlined the need to use visuals to help the waste pickers better learn. In their case they used a stoplight signal to highlight the economic choices / budgeting.

Whilst the demarcation was limited to a theoretical application, it was developed internally at AAU and can be found here: [33]

A generally more interactive approach than in the original mobile education project, the issue comes at the narrow focus. The application teaches only one thing and is not easily expandable to include other subjects or enhance the already existing materials. Again the fundamental takeaway from the application is the need to enable learning trough more visual approaches, such as images / video or interactivity.

Kahn Academy

Kahn Academy is a video and test based learning site mainly focused on mathematics, but expanding to other subject [1]. Within Kahn Academy a user can enroll into a course for free and learn via mostly videos and test allowing the user to self evaluate their understanding of the subject. The main focus of the platform is to teach a user a concept within mathematics via a short 5-10 minute video, then testing if the user has learned the subject via a short test. Once the user has completed said test they can then proceed to the next video which is usually just an expansion of the subject learned in the previous video. These series of videos / test are split into different courses covering different subjects.

For example one could be about matrix computation and another about high school integrals, each starting relatively simple before slowly opening the subject up, video by video. Text is not used as often within Kahn Academy, with the main focus being the videos and testing.

Whilst it offers great learning opportunities for mathematics there are notable issues if this was to be used with the Waste picker use case. Firstly, it is a closed system, so this project can not be made on their application unless a partnership was formed. Secondly as mentioned in the P3 demarcation, the waste pickers to not have cellphone data and the download function on Kahn Academy is rudimentary only allowing manual one by one downloads [2]. Thirdly, whilst there is a need for videos, a video / testing only approach could leave to less interest from the waste pickers due to the lack of variety.

Udemy

Udemy is course creation / taking website / app that focuses on enabling a variety of courses from external partners. [53] This means subjects from writing a novel in the style of Ernest Hemingway to Signal processing are taught on the platform. Just as in the original MEP project [12] the platform allows for text / images / audio / video. Additionally like Kahn Academy it also allows for testing and hand ins of files.

Udemy again like Kahn Academy also has a heavy focus on videos with the text and image implementation being a simple embedded pdf viewer. This again falters in the lack of variety of course materials and also has a simple download system for their smartphone app, where each video has to be downloaded manually. From a usability perspective it does provide the ability to insert subtitles into the videos, however it does not allow for text to be read aloud.

Moodle

Moodle is a learning site mainly used as an assistant to classroom learning with files being available within it for each course and course segment. [32] Moodle acts as a file storage system and course coordinator for a course but does not show then embedded in the course and is not usually used as a stand alone teaching tool. However, it still allows for many types of files, be it text / image / audio / video in addition to testing, both freeform and multiple choice testing. Whilst Moodle is not a remotely adequate solution to the waste picker use case, it still highlights the need for these five subjects as core features alongside a form of mitigating the connectivity issues from the waste pickers.

Rise Articulate

Rise Articulate is a leading provider of an extensive digital learning content creator tool. Contrary to the other frameworks included in this analysis, Rise Articulate focus on providing all necessary functionality for users to create interactive and modernly styled digital courses with built in progression tracking, video playback and much more. The framework is therefore not a library of existing content already but merely a tool for creating content as a user. The benefits of the framework are obvious in that high quality digital learning content can be created very easily and quickly, but the drawbacks are visible too. The service in itself is very expensive and aimed at companies with large amounts of available resources. Furthermore, though the framework is extensive, it is quite limited as a user since you rely solely on the platform to provide you with the components you need[5].

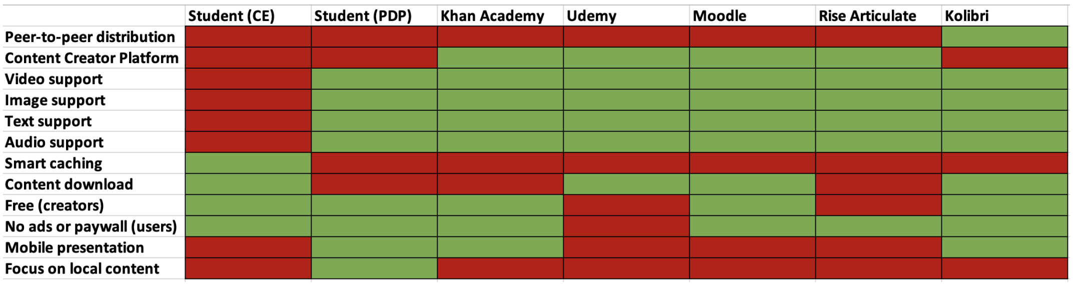
Kolibri

Kolibri is an adaptable set of open solutions specially developed to support learning for the half of the world without Internet access. [24] Essentially, Kolibri aims to solve the issues shown for both Udemy and Kahn Academy in relation to connectivity via peer to peer distribution. This allows a course from for example Kahn Academy to be cloned from one phone to another as long as the original phone already has said course downloaded. Whilst not producing the learning content itself Kolibri takes already existing platforms and adds this sharing methodology on top. Better yet, it is open source, allowing for specific features such as this offline cloning to be taken into another application.

However, it still does not solve some of the other usability focused issues established within both the Udemy and Kahn Academy examination. Additionally it also gives one more step for the user to go trough in order to clone / update a course, that smart-caching completes automatically. It seems then that Kolibri is well suited for places with little to no internet access, however for this use case as described in the Section 2.2.1, the waste pickers have access to Wi-Fi. For similar use cases wherein the internet connectivity is worse, it could be a great implementation.

2.2.3 Summary

To establish an overview of the different platforms and their feature sets, the table on Figure 2.1 summarizes which features are available for which platforms. Green cells indicate availability while red cells indicate that the feature is not implemented in the framework.



	Student (CE)	Student (PDP)	Khan Academy	Udemy	Moodle	Rise Articulate	Kolibri
Peer-to-peer distribution	🟥	🟥	🟥	🟥	🟥	🟥	🟩
Content Creator Platform	🟥	🟥	🟩	🟩	🟩	🟩	🟥
Video support	🟥	🟩	🟩	🟩	🟩	🟩	🟩
Image support	🟥	🟩	🟩	🟩	🟩	🟩	🟩
Text support	🟥	🟩	🟩	🟩	🟩	🟩	🟩
Audio support	🟥	🟩	🟩	🟩	🟩	🟩	🟩
Smart caching	🟩	🟥	🟥	🟥	🟥	🟥	🟥
Content download	🟩	🟥	🟥	🟩	🟩	🟥	🟩
Free (creators)	🟩	🟩	🟩	🟥	🟩	🟥	🟩
No ads or paywall (users)	🟩	🟩	🟩	🟥	🟩	🟩	🟩
Mobile presentation	🟥	🟩	🟩	🟥	🟥	🟥	🟩
Focus on local content	🟥	🟩	🟥	🟥	🟥	🟥	🟥

Figure 2.1: Summary table of key functionalities for digital learning services

As a result of this analysis, feature extraction of the different existing platforms is done, gaining an overview of both the core features of a digital learning application and the specialized features that are well suited to the specific use case. As described in all the course creation tools, the core features of a digital learning application are the ability to create and view: Text, Image, Audio, Video. With another commonality being the ability to take tests on the platform being a much needed feature as well.

For connectivity purposes, the smart-caching system from the original Mobile Education Project (MEP) is still seen as a viable method to get the courses onto an application. However a potential feature extracted from Kolibri, is the platforms use of caching, allowing courses to be cloned from one smartphone to another [24] through peer-to-peer distribution. The usefulness of this feature becomes apparent based on the research provided in the article by [12] wherein it is established that the courses would only be downloadable whenever the individual user is connected to Wi-Fi. By implementing the same course cloning structure, it could help alleviate this potential roadblock to entry. However, it is also clear that no existing solution perfectly fits the use case of the waste pickers, with either roadblocks such as costs of the platform, the platform being closed or to heavily focused on one thing, such as video leaving all current options seeming sub-optimal.

2.2.4 Subsidiary conclusion

From conducting and analyzing interviews with the user group in Section 2.2.1 and comparing core and specialized features from currently available digital learning services in Section 2.2.2, it is concluded that current digital learning opportunities are unsatisfactory to the learning needs of Brazilian sorting facility employees. Hereby, a problem statement is formed, based on the learnings from this chapter.

Chapter 3

Problem statement

How can a digital learning solution be built to satisfy the educational needs of individuals employed at the Brazilian sorting facilities?

Chapter 4

Methodology

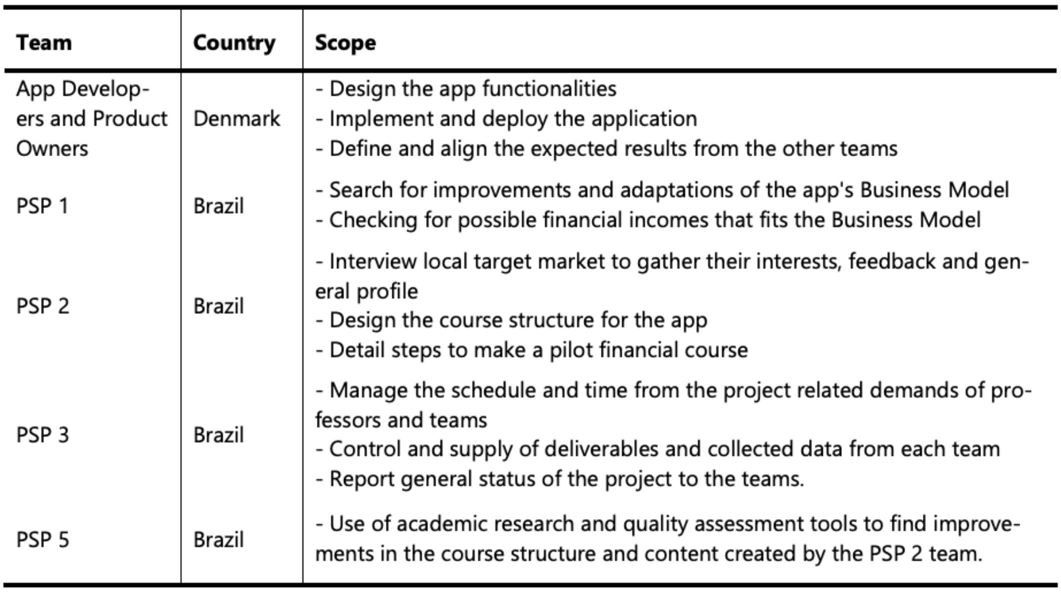
The project aim is to combine a Problem Based Learning approach [54] with aspects from Design Thinking (Section 4.1), Human-Centered Design (Section 4.3) and Structured System Development (Section 4.4), to handle the complexity of cross-disciplinary and international collaboration and ensure that the solution is developed through close involvement of user groups and stakeholders. A large part of the project is about gathering fragmented contributions from different disciplines - and ensuring that the contributions, work towards a coherent process of product development.

This chapter discuss and describes methodical approaches to above mentioned issues and presents the structures applied in the project. This involves descriptions and models of the collaborative structure between involved teams as well as technical approaches to engineering problems and testing processes.

4.1 PBL & collaboration methodology

Problem Based Learning constitutes the foundational approach to engineering in this project [54]. The socioeconomic and educational issues regarding the waste pickers transitioning from a trading based personal economy into a payroll based structure, is the problem this project aims to innovate engineering solutions to. The aim is to meet the needs of the waste pickers with a product that is both technologically feasible, includes a viable and sustainable financial strategy and acts as a connector for future progress.

From University of Brasilia, four student teams participate in the project, as part of their Bachelors Degree in Production Engineering. The four groups will be referred to as Production System Project 1, 2, 3 and 4, accordingly to related course activities. Each team has certain responsibilities in the research and development and is expected to contribute differently to the proposed solution. Figure 4.1 summarizes the division of responsibilities between the 5 teams involved in the project.



Team	Country	Scope
App Developers and Product Owners	Denmark	- Design the app functionalities - Implement and deploy the application - Define and align the expected results from the other teams
PSP 1	Brazil	- Search for improvements and adaptations of the app's Business Model - Checking for possible financial incomes that fits the Business Model
PSP 2	Brazil	- Interview local target market to gather their interests, feedback and general profile - Design the course structure for the app - Detail steps to make a pilot financial course
PSP 3	Brazil	- Manage the schedule and time from the project related demands of professors and teams - Control and supply of deliverables and collected data from each team - Report general status of the project to the teams.
PSP 5	Brazil	- Use of academic research and quality assessment tools to find improvements in the course structure and content created by the PSP 2 team.

Figure 4.1: Responsibilities of student teams

This division of responsibilities is crucial for the long term sustainable prospects of the solution. It is a core part of the methodical approach taken in this project (See Section 4.2) that a combination of focus and perspective contributes to a sustainable product design. Achieving a Minimal Viable Product solution (MVP) is the technical ambition of the project with further consideration given to long term sustainability of the solution. To achieve this the different groups are to conduct user research, research financial options with a structured approach to project management taken from this projects authors. As described in Figure 4.1, PSP1 group is responsible for researching the financial viability of the solution, PSP2 group is responsible for researching users and problem domain through qualitative research methods (interviews etc.) and creating pilot educational content, PSP3 group manages the integration between the Brazilian teams and PSP5 has the responsibility of quality assurance.

To elaborate upon the division of responsibilities and interaction between teams Figure 4.2 illustrates each of the involved teams and their interactions with each other.

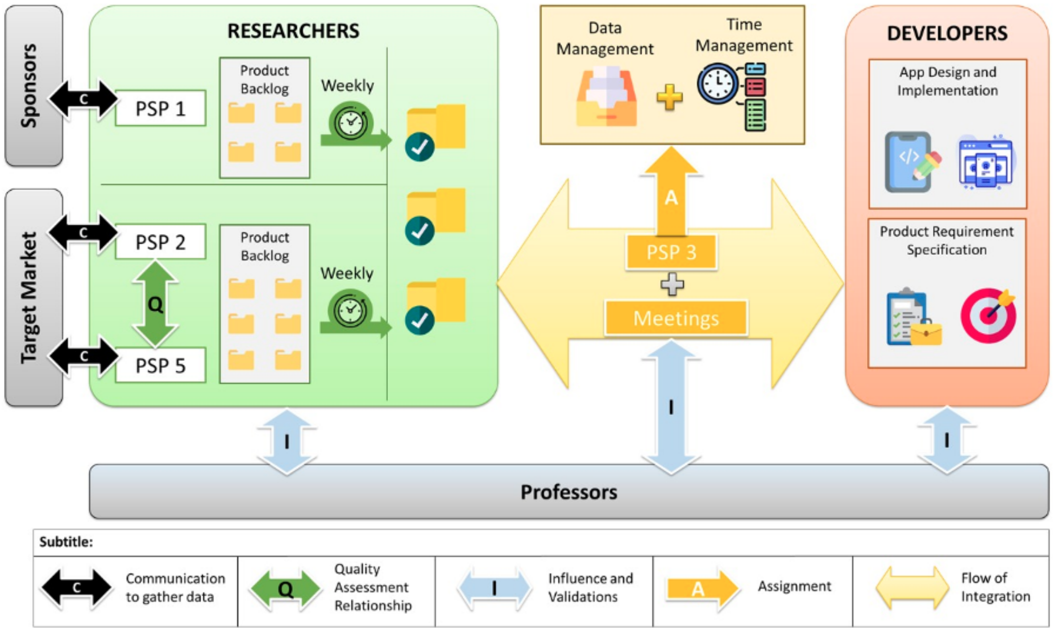


Figure 4.2: Illustration of the collaborative structures between UnB and AAU student teams

This representation outlines the collaboration between teams and division of responsibilities in the project. The focus of this thesis is not the collaboration and management perspective but additionally to this report, a research article with this focus is also written during the study term and published in the 2021 PAEE/ALE conference. See Appendix B.1 to read the full paper which also further details Figure 4.2. Another important aspect of Problem Based Learning in Engineering is a methodical approach to development. This ensures alignment of the expected flow and progression throughout the project transitioning through research, design and implementation. This project uses a framework inspired by Design Thinking.

4.2 Design Thinking in Engineering

Design Thinking was first mentioned by cognitive scientist and Nobel Prize laureate Herbert A. Simons in his book "The Sciences of the Artificial" in 1969. Since then, academics and professionals from a vast variety of fields have contributed to and shaped the original ideas, to establish a well known and widely used framework that supports development of sustainable products through human-centered design, strategic innovation and user-involvement. Engineering teams have adapted Design Thinking worldwide - including companies such as Google, Apple and Airbnb - reporting notable positive effects in innovation, usability design and capitalization of market opportunities[17].

One of the earliest acknowledged examples of Design Thinking is Thomas Edison's invention of the light bulb [51]. The bulb itself is usually thought of as the signature invention but Edison understood that isolated it was little more than a parlor trick. Contrary, if combined with a vast and complex network of electric power generation and transmission, it had the potential to revolutionize modern life and wrap an entire industry around it. But this understanding wasn't trivial for a specialized scientist as Edison. It was a result of strategically surrounding himself with a diverse group of people throughout the process of innovation and closely integrating their feedback into all stages of development. Gifted men of craftsmanship, business owners and experimenters all came together and participated in the innovation process, shattering the popular image of Edison as a 'lone genius inventor', and creating a team-based and cyclical approach to invention[13, p. 2-3].

The story of Thomas Edison reflects the strategic mindset applied in this project. By rooting the project in cross-discipline, strong involvement of stakeholders and focus on user-centered design principles, the aim is to ensure the creation of both innovative, useful and maintainable technology. Similar to Edison's notion of the relative uselessness of the light bulb if isolated, the complex socioeconomic issues that remain the focal point of the project cannot be solved without combining perspectives and thinking beyond individual functionalities or optimizations.

Adapting concepts from Design Thinking as an engineering philosophy can help mitigate the fallacy of simply validating preconceived hypotheses by strategically utilizing iteration and engagement of users, engineers and stakeholders, to create a progressive and structured flow of innovation. As described by the Interaction Design Foundation[17], Design Thinking is a "non-linear, iterative process that teams use to understand users, challenge assumptions, redefine problems and create innovative solutions". Based on research conducted by the Hasso Plattner Institute of Design at Stanford University, Design Thinking can be deconstructed into five phases: Empathize, Define, Ideate, Prototype and Test.

This project proposes a revised model of the Design Thinking process based on material from the Interaction Design Foundation[17] and graphical illustrations from Nielsen Norman Group[20]. The model consists of 3 main-phases, 6 subphases and descriptions that are fitted to the unique aspects of this project, as illustrated below on Figure 4.3:

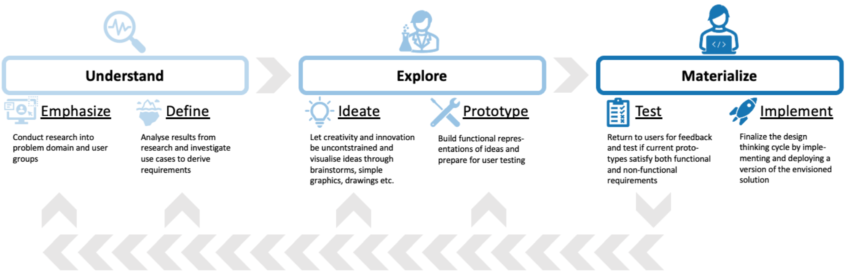

Understand
Explore
Materialize
Emphasize
Conduct research into problem domain and user groups
Define
Analyse results from research and investigate use cases to derive requirements
Ideate
Let creativity and innovation be uncontstrained and visualise ideas through brainstorms, simple graphics, drawings etc.
Prototype
Build functional repres-entations of ideas and prepare for user testing
Test
Return to users for feedback and test if current proto-types satisfy both functional and non-functional requirements
Implement
Finalize the design thinking cycle by imple-menting and deploying a version of the envisioned solution

Figure 4.3: Revised model of the Design Thinking process

The first phase of the Design Thinking process is about increasing understanding of the problem domain and user groups. Any project following Design Thinking principles should prioritize this stage greatly at the beginning of a project, as it is crucial for early decision making and will impact anything that builds on top of it. Design Thinking as a paradigm belongs to the category of Human-Centered design and this first phase focuses on engaging and empathizing with users, consulting local experts and immersing yourself into the problem domain. The result is a deeper understanding of opportunities, obstacles and circumstances involved in the project and allows engineers to set aside personal assumptions and instead gain insight into actual user needs, wants and constraints.[17][20]

The Understand main-phase should result with a solid understanding of users and their needs, a well defined problem domain and a concrete list of product requirements that determines the success of the project. With this in hand, it is time to embark upon the second main-phase: Explore. In this phase, it is all about 'thinking outside the box', exploring and debating ideas, identifying issues and design suggestions and discussing angles of innovation. A multitude of methods - such as brainstorms, wireframes and rich picture drawings - should be taken into use during this phase, and generally nothing is out of bounds. This is an important notice as it is crucial for the innovation process to stimulate free thinking and expand upon the problem space.[17]

Then comes the final main-phase Materialize. From the Explore main-phase, concrete prototypes and mapped out ideas for functionality and design should have been made. The next step is to return to the users for feedback on the creations and test if user needs and wants are met. During testing you should focus on documenting the feedback and especially notice if the ability to see, feel and/or use the prototypes, changes user perspectives or requirements. Finally, when feedback has been reviewed and prototypes and ideas may have been adjusted accordingly, it is time to implement the prototypes into a finalized product version. This does not mean that development is done - because it never will be, but it's important to release a version and complete a Design Thinking cycle, because then a new one can start, with aim to optimize and develop the functionality of the next version. And once again - start at main-phase 1, and review understanding of the problem domain and user groups.

That concludes the Design Thinking methodology of this project. The next section further details methodological aspects of Human-Centered design used in this project.

4.3 Human-Centered design

Involving users and stakeholders in the design process is crucial as it ensures that the interactive products fits with user needs, wants and requirements - instead of what engineers and designers assume. Close involvement of users and stakeholders, is especially important, when working in circumstances of a target group so drastically different from the circumstances of the engineers and designers. This is the case in this project as few things are comparable for the life of us as students in Denmark and the waste pickers in Brazil. Under normal circumstances field and user research would be a top priority to conduct physically, but due to the circumstances of the Covid-19 pandemic, traveling to Brazil has not been possible. Therefore, the responsibility of user and field research is managed by student groups from University of Brasilia as described in Section 4.1. Through qualitative investigation this research examines questions such as: (Full questionnaire in Figure A.1):

- What operating system does the users have on their smartphones?
- What are the users abilities in terms of reading, writing and listening?
- What types of learning content are most crucial for the target group?
- Do the users have access to Wi-Fi at home or work?

For all these questions, one might draw quick assumptions about the answers, but it is certain that those assumptions are correct. They might reflect part of the picture and sometimes be completely right, but there's also a prevalent risk of being completely wrong, and basing subsequent design decisions on a wrongful basis. The cost of not performing proper user involvement, is that we risk developing a product that will only frustrate users and end up not being used. This project aims to additionally build upon user research carried out in previous student projects[12][41] as well as conduct additional research in collaboration with partners from University of Brasilia.

4.3.1 Why Human-Centered design?

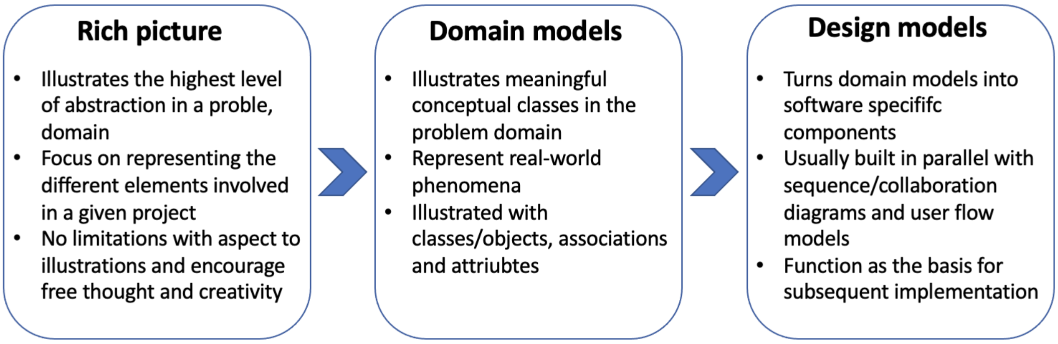
As defined by Preece, Sharp & Rogers in the widely acknowledged book "Interaction Design: beyond human-computer interaction"[40], Human-Centered design is an approach to interactive system development that focuses specifically on making systems usable through systematic work with usability and user experience (UX). This includes but is not limited to active involvement of users, use of iterative design paradigms and focus on functional and non-functional requirements. The wording Human-Centered design instead of User-Centered design, indicates that requirements originate not only from the immediate users but also from stakeholders, secondary users and others involved in the interaction with the product.

This engineering principle is adapted in this project to ensure that the developed solution fits with the learning needs of the users.

4.4 Structured System Development

Methodology of Structured System Development is used in this project as a way to manage the complexity of developing a Distributed Web System with many individual elements and interfaces. Structured System Development is about investigating the problem domain in a project and systematically establish requirements, with aim to develop a progressively more detailed conceptual solution that satisfy the requirements. It is not programming specific, but in the end does produce models that is bound up on software objects and system relations.

The general structure of the methodology can be divided into three sections, as illustrated below on Figure 4.4



Rich picture

- Illustrates the highest level of abstraction in a proble, domain
- Focus on representing the different elements involved in a given project
- No limitations with aspect to illustrations and encourage free thought and creativity

Domain models

- Illustrates meaningful conceptual classes in the problem domain
- Represent real-world phenomena
- Illustrated with classes/objects, associations and attriubtes

Design models

- Turns domain models into software specififc components
- Usually built in parallel with sequence/collaboration diagrams and user flow models
- Function as the basis for subsequent implementation

Figure 4.4: Revised model of the Design Thinking process

The methodology is generally Use Case driven, meaning that requirements are primarily recorded through analyzing use cases. Using use cases as basis for design and software implementation is a central element in the Human-Centered design process, as it encourages the engineering team to base system design on concrete needs and wants from the user groups. It is also an important structural element in the iterative planning and development cycles used (see Section 4.2) and supports a "one feature at a time" development process.

The Rich Picture phase is where the system is outlined and general components are identified. This serves as a high level overview and makes the basis for further innovation.

When transforming the rich picture basis into domain models, a vocabulary of all concepts of interest to the problem domain is developed. This gives insights into which concepts influence and/or interface to others, and provides a framework for essential abstractions required to deepen the understanding of the system and current requirements. This is a central element in a problem-solving design process.

The final step, of converting the domain models into software specific design models, is a method to gradually increase the number of software representations of classes in the problem domain. It gives developers a firm grasp of which classes are responsible for managing specific data, system operations etc. Utilizing progressive user-centered models as sequence and collaboration diagrams to map communication is also a central element for subsequent implementation. The design models can be closely connected to Wireframe prototypes and initial UI design.

By combining principles from design thinking, human-centered-design and structured system development, a methodical foundation for the project is established. In the following chapter, the product vision is explored and conceptualized, based on the methods presented in this chapter.

Chapter 5

Conceptualization and design

This chapter describes and documents the phases of conceptualizing and designing a solution to the problem statement, see Chapter 3. The methodical approach used in this section is derived mainly from principles of Structured System Development (SSD) and Object Oriented Analysis and Design (OOAD) as detailed in Chapter 4, and reflects the processes used in the product development processes in the project.

Initially, the solution is conceptualized in a Rich Picture on a high level of abstraction, mainly based on conclusions from Chapter 2 and showcased in Section 5.1. Subsequently, use cases that represent user interaction with system interfaces are defined and analyzed in Section 5.2. Based on the conclusions from use case analysis, domain models are constructed as a way to represent relations between the solution, problem domain and users. This leads into the transition from problem oriented models to design models, that reflect a representation of the user-faced interfaces of the system in Section 5.3. The models established in this chapter makes the basis for software implementation and design in Chapter 8.

5.1 Rich Picture Conceptualization

Originally developed by Peter Checkland, emeritus professor of systems as Lancaster University[57], Rich Picture modeling is a methodical approach to use simplistic free-hand drawings as a central element in the initial phase of a technical problem based project. The method is used regularly within Engineering fields as a pre-analysis stage, to illustrate ideas, obstacles and processes associated with a problem domain. As Checkland describes in his book 'Soft Systems Methodology'[14], Rich pictures describe real world issues and can be an effective tool in ensuring shared understanding of a problem domain in engineering. As explained in Section 4.1 this project unfolds in a very complex problem domain and entails both cross-disciplinary and international elements, which means that shared understanding is increasingly important for successful software development. For this reason, Rich Picture modeling is applied as the initial pre-analysis step in the solution development.

As the purpose of a Rich Picture is to be a creative, open-minded and unconstrained illustration of concepts within a problem domain, the output varies greatly. Figure 5.1 shows the Rich Picture model constructed in this project.

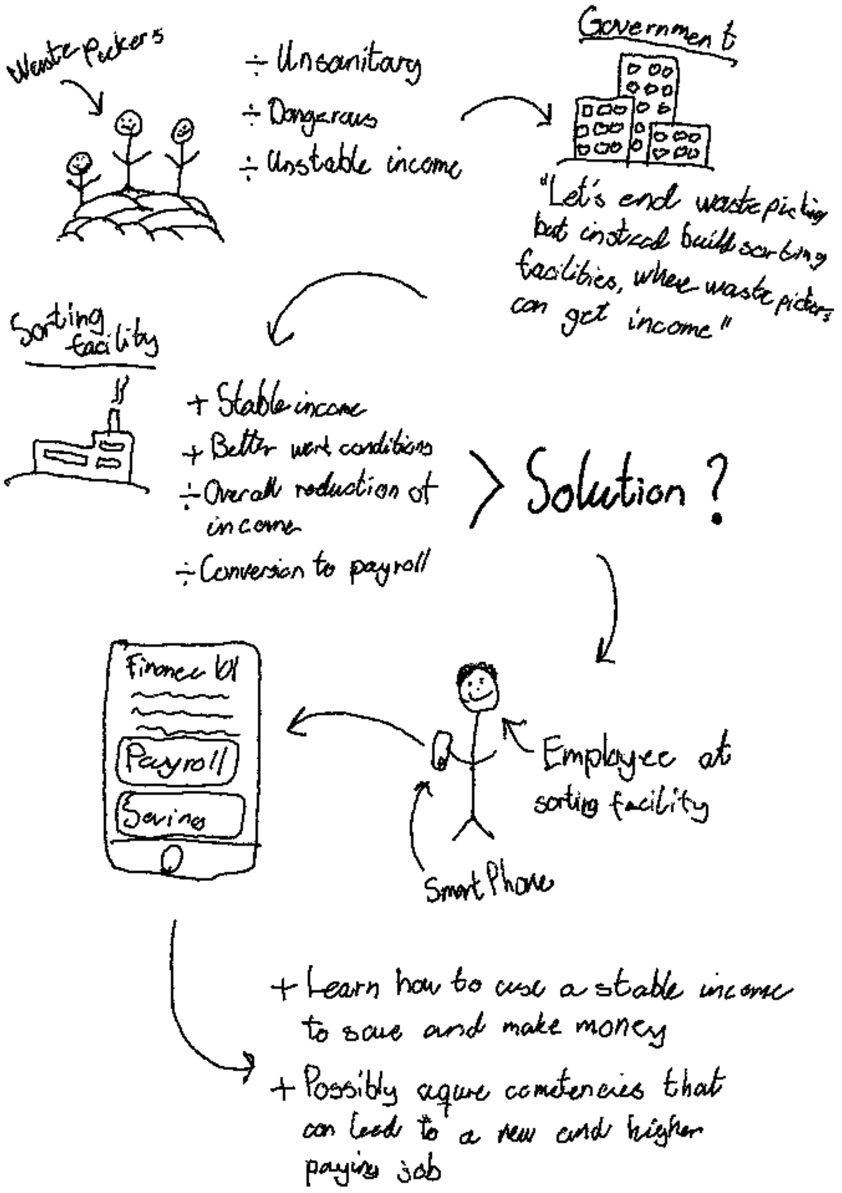

Waste Pickers
÷ Unsanitary
÷ Dangerous
÷ Unstable income
Government
"Let's end wastepicking but instead build sorting facilities, where waste pickers can get income"
Sorting facility
+ Stable income
+ Better work conditions
÷ Overall reduction of income
÷ Conversion to payroll
Solution?
Finance UI
Payroll
Saving
Employee at sorting facility
Smart Phone
+ Learn how to use a stable income to save and make money
+ Possibly aqure comtenecies that can lead to a new and higher paying job

Figure 5.1: Rich picture illustration

The model illustrates the story of Brazilian Waster Pickers transitioning into a position of payroll based employment and the problems associated with that change. Initially, as also detailed in Chapter 2, the daily life of Brazilian waste pickers was plagued by unsanitary and dangerous conditions, while creating a life of financial instability. The Brazilian government decided to solve this problem by building sorting facilities at the waste locations and hire the waste pickers as employees. Though the ambition of this initiative might have been to solve the problems associated with waste picking, this transition brought other unforeseen challenges. A more stable source of income was achieved for the individuals now working as employees and much safer and sanitary working conditions was created.

But, though it's hard to measure the exact income of waste pickers before this initiative, the projects partners in Brasilia specifically PSP2 as described in Section 4.1 researched into the new circumstances of the waste pickers. Based on the feedback received from the waste pickers the new payroll based personal economy resulted in less money pr. month than waste picking. Additionally, converting to a payroll based personal economy when used to one of trading, gathering and selling, can be extremely difficult.

Previous projects researching this transition for the waste pickers in Brazil[12][41] concluded that basic finance education focusing on managing a payroll based economy has great potential to increase the socioeconomic situations of the Brazilian Recycling Facility employees. Based on that conclusion, the rich picture model conclusively illustrates the idea of these individuals accessing personal finance educational content digitally on their smartphones.

From discussing the rich picture model a high-level system overview is constructed as shown on Figure 5.2.

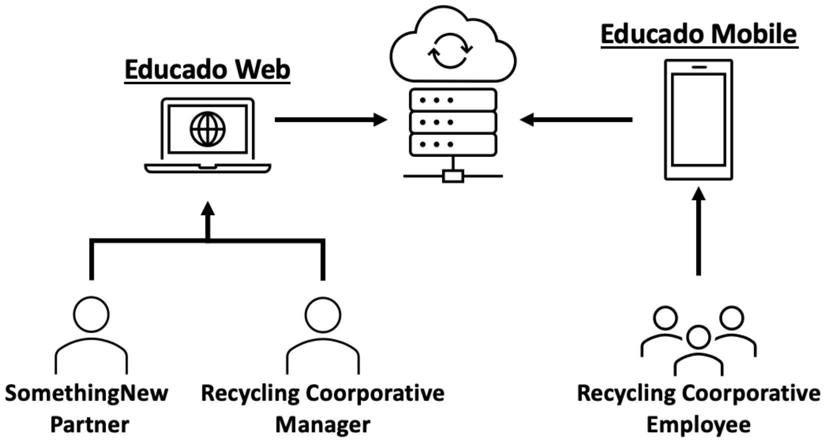

Educado Web
Educado Mobile
SomethingNew Partner
Recycling Coorporative Manager
Recycling Coorporative Employee

Figure 5.2: General overview of proposed system

The envisioned solution consists of two systems: Educado Web and Educado Mobile. Each application acts as an independent system with different functionality and user groups. Educado Web is envisioned as a web application with twofold functionality: (i) Create educational content and (ii) Track employee progression. These core features are derived from the problem analysis in Chapter 2.

Educado Mobile will be the direct user interface for the recycling facility employees and is envisioned as a smartphone application. At the center of the model, a cloud synchronization server is illustrated to represent a interconnected backend system of file storage, databases and api systems, that allows for digital content to be persisted and employee progression to be tracked.

In order to manage these applications and secure long term development, product ownership is transferred to the engineering startup SomethingNew by the end of the study term. The company is expected to maintain responsibility for the development of content through partnerships and internal development. The aim of this approach is to no longer start anew each semester as happened in previous projects [12][41], but rather have the company as an umbrella enabling continues development of the project. This includes, more field research, product development and ensuring a cross-disciplinary approach to best suit the case etc.

This concludes the rich picture conceptualization and a shared understanding of the problem domain as well as a high-level overview of the envisioned system is now achieved. The next section enters the analysis stage by researching use cases and establishing domain models.

5.2 Use case analysis

Use cases are useful to record and derive requirements for a product. As described in [23], use cases are an important part of iterative development as per Object Oriented development paradigms. By centering requirements and hereby development around concrete use cases, it is ensured that the design is based on user needs and not on assumptions made by designers or engineers.

Use cases can be analyzed in various ways but for the purpose of constructing domain models in this project, the intitial analytical step applied is highlighting of noun phrases. By highlighting the parts of a use case where a noun is connected to an action, it is possible to establish an overview of actionable content related to a certain application. Working within the problem domain perspective, the nouns and phrases won't be software specific yet, but will give indications as to which functionality requirements the system might have. Conclusively, these indications will act as the foundation for constructing domain models.

A domain model illustrates meaningful conceptual classes within a problem domain, and hereby represent real-world phenomena, and NOT software specific design objects. They consist of objects, associations and attributes, illustrated in UML standard. The purpose of domain models is to establish a vocabulary of concepts of interest to the problem domain and acquire insight into how concepts influence and relate to each other. It provides a level of abstraction and information required to understand the domain in the context of deriving requirements. Furthermore, it serves as the basis for a problem-solving product design, by building upon use case driven analysis.[23]

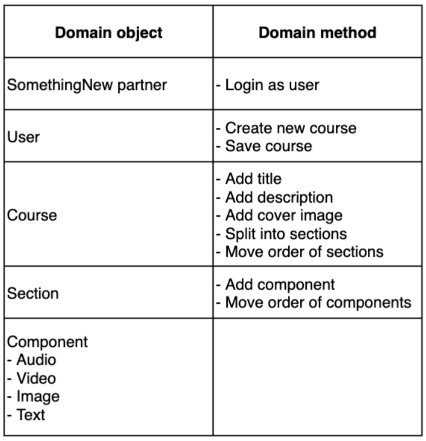
As Section 5.1 details, the proposed solution involves two different user-faced applications: (i) Educado Web and (ii) Educado Mobile. As these two applications have different purposes and user-groups a use case is defined for each of them and subsequently functionality indications are also derived individually.

5.2.1 Educado Web

For the Educado Web system, it is already known from Chapter 2 and Section 5.1, that the main functionality is to allow users to create digital content. Through discussions with the PSP teams from UnB (See Section 4.1) the following use case is defined for the system.

A **SomethingNew partner logs into the ECS system** as a **user**. The **user** then **creates a new course** and **chooses title** and **description** for the **course**. Afterwards the **user edits the course** and **add a cover image**. Subsequently the **user splits the course into different sections**, which then each breaks down into **individual components**. The **user** chooses to **add a mix of audio, video, image and text components** to each **section** and **saves the complete course**. After having completed each **section**, the user decides to **move the order of two sections** around and also **move around in the order of components** in one of the **sections**.

Based on the highlighted noun phrases in the use case, the indications are divided into two groups: (i) Domain objects and (ii) Domain methods. The domain methods are actionable functionalities that belong to certain domain objects. It is a useful analytical step to illustrate the high-level overview of relations between classes and functions in the system. The result is summarized in the table below:



Domain object	Domain method
SomethingNew partner	- Login as user
User	- Create new course - Save course
Course	- Add title - Add description - Add cover image - Split into sections - Move order of sections
Section	- Add component - Move order of components
Component - Audio - Video - Image - Text	

Figure 5.3: Summary table for use case analysis of web system

The final step is to convert the domain objects and methods into the domain model that follows UML design principles. Each of the domain objects are illustrated by class boxes in UML where associated methods are listed as list points underneath. Furthermore, the relation between classes are illustrated in traditional UML style where '*' represents 'many' and numbers represent their value. The domain model for Educado Web is illustrated below on Figure 5.4:

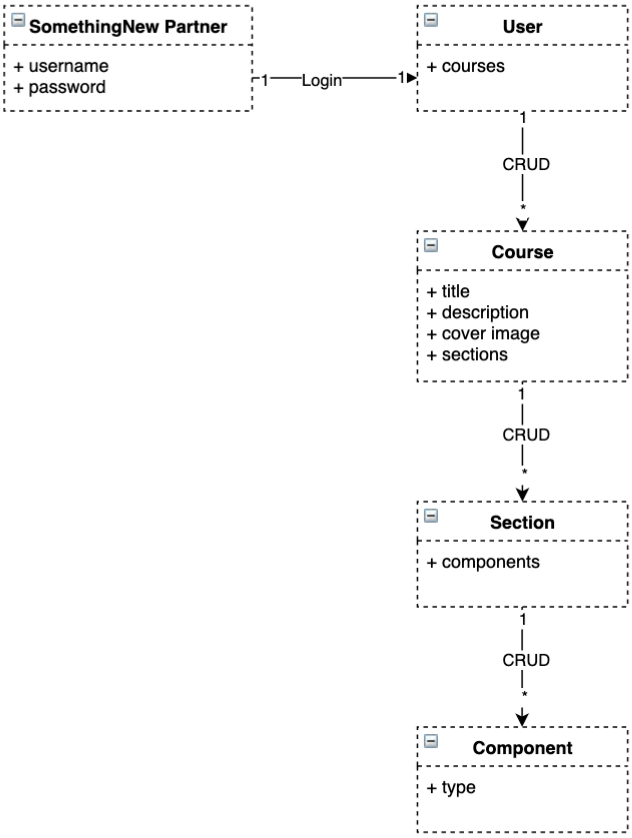


Figure 5.4: Domain model of Educado Web system

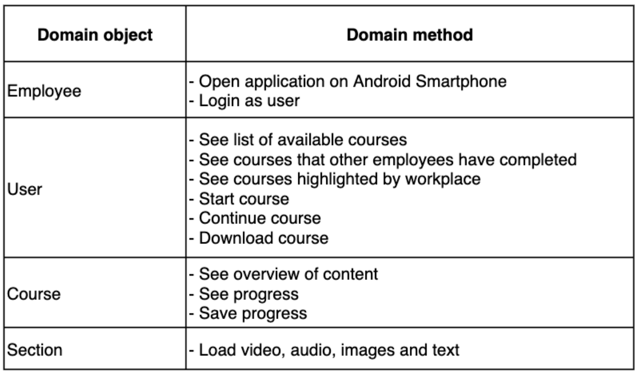
The level of detail is still limited to real-world perspective and has yet to be transformed into a software specific design model. The same analytical process is conducted for the Educado Mobile system.

5.2.2 Educado Mobile

From Chapter 2 and Section 5.1 it is known, that the main functionality of the Mobile system is to allow learners (employees as the recycling facilities), to access the digital educational material. As the material itself is created through the Web system, the two systems are dependent on each other. Through discussions with the PSP teams from UnB (See Section 4.1) the following use case is defined for the mobile system.

An **employee** at a Brazilian recycling center is taking a 30-minute break at work. Using his/her **Android Smartphone**, he/she opens the **Educado Mobile Learning application**, automatically **logs in** as a registered **user** and scrolls down the list of **available courses**. The **employee** sees that five other workers at the recycling center already has **completed a course** titled 'Personal Finance' and that it is **highlighted by the workplace administration**. The **employee** decides to **start the course** and chooses to **download the complete contents** while connected to the workplace Wi-Fi. When **download** is finished, he/she **starts the course** and **completes the first section** that consisted of a couple of **short videos, text pieces and images**. The **progress** of completing the first section is clearly visualized and **automatically saved on the phone**. As the break comes to an end, the employee closes down the application and looks forward to **continuing the course** during his/her 45-minute commute to home after work.

As with the Web system, the highlighted noun phrases in the use case leads to a division into two groups: (i) Domain objects and (ii) Domain methods. The domain methods are actionable functionalities that belong to certain domain objects. It is a useful analytical step to illustrate the high-level overview of relations between classes and functions in the system. The result is summarized in the table below:



Domain object	Domain method
Employee	- Open application on Android Smartphone - Login as user
User	- See list of available courses - See courses that other employees have completed - See courses highlighted by workplace - Start course - Continue course - Download course
Course	- See overview of content - See progress - Save progress
Section	- Load video, audio, images and text

Figure 5.5: Summary table for use case analysis of mobile system

Finally, the domain objects and methods are converted into a domain model that follows UML design principles. Each of the domain objects are illustrated by class boxes in UML where associated methods are listed as list points underneath. Furthermore, the relation between classes are illustrated in traditional UML style where '*' represents 'many' and numbers represent their value. The domain model for Educado Mobile is illustrated below on Figure 5.4:

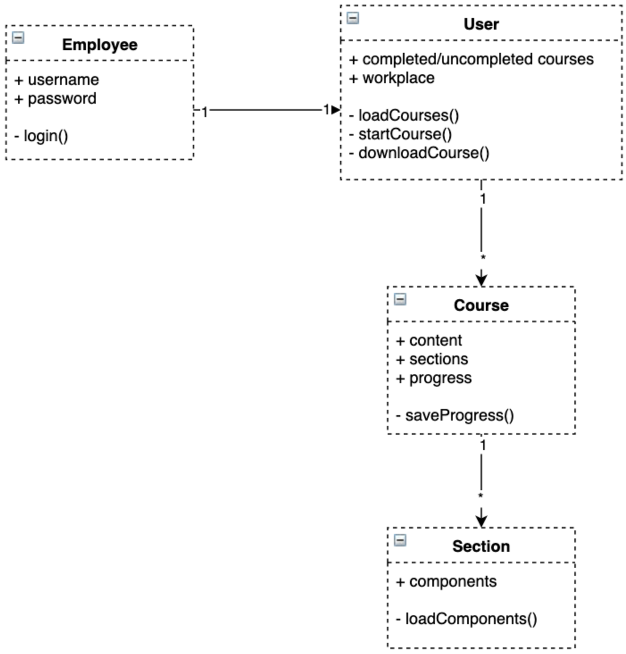


Figure 5.6: Domain model of Educado Mobile system

Domain models have now been established for both envisioned systems. As a result, a shared vocabulary of concepts of interest to both systems with roots in the problem domain, as well as insights into how these concepts influence and relate to each other is acquired. Though this analysis is not exhaustive and many other use cases could be explored, ensures an essential level of abstraction and information required to understand the initial concept of a solution within the problem domain. In the next section, the information from the use-case analysis and the domain models are converted into software specific design models reflecting applied principles from Structured System development and Object Oriented Design and Analysis.

5.3 Transitioning from Domain to Design

From establishing domain models rooted in the problem space, the acquired understanding and information can be utilized to transition into initial software specific design representations. The methods applied in this section are derived from [23] and [40] and consist of an approach to convert the UML based domain models into Wireframe prototypes representing user-interfaces that will make basis for subsequent implementation. As with the previous section, the solution in this project is still divided into two individual dependent systems: (i) Educado Web for creating digital educational material and (ii) Educado Mobile for accessing the material as a learner. This section is structured to reflect this recurring division and details the establishment of design models for each of the systems.

5.3.1 Educado Web

The use case analysis and domain model in Section 5.2.1 indicates both conceptual classes and methods for core functionality of the web system. The model represents illustrates a top-down and progressive line of communication, starting with a SomethingNew partner logging in to the application. When logged in, the user should be able to see an overview of the courses already created and have the ability to create a new one or edit existing ones. This is represented on Figure 5.4, with the 1-to-* arrow between the User and Course objects, with an attached indication of CRUD. CRUD is an abbreviation for 'Create, Read, Update and Delete', and represents that the object must have functionality to manipulate another object [16]. This flow is reproduced in the relation to both Section and Component objects, that each add an additional nested layer to the object oriented model.

Based on the above description of functionality in the web system based on the domain models and use case analysis, an initial concept of an appropriate user-faced interface is established. First the concept is visualized in free-hand drawing, and subsequently digitized in Wireframe models imitating standard design of a web application running in a browser. The illustration can be seen below on Figure 5.7:

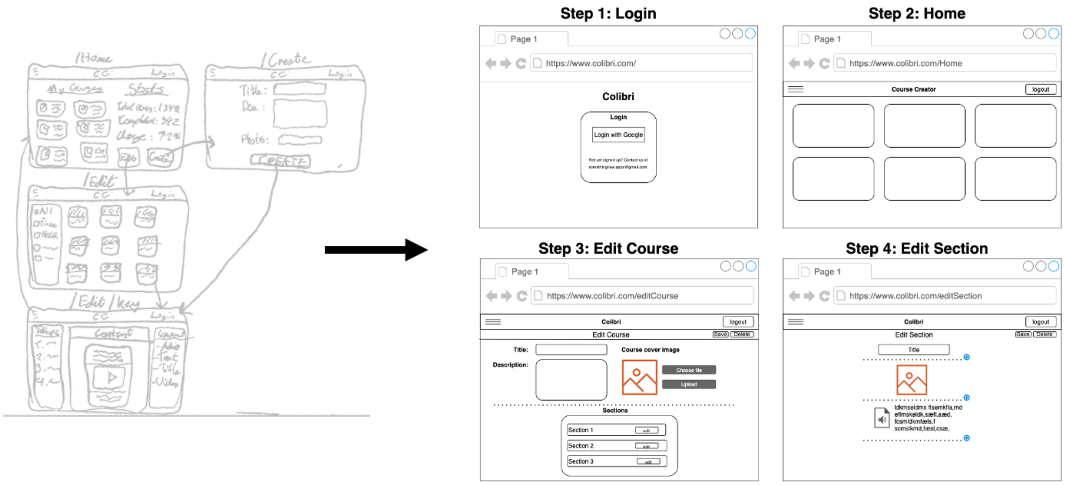


Figure 5.7: Wireframe model of Educado Web

The development of the wireframe design model is done through multiple iterations of discussion and drawing, and represents the final result of that process.

Ultimately, the result is a conceptual idea of the application's user interface, that supports the core functionality and objects derived in the use-case analysis in Section 5.2. The model consists of four different screens illustrated to the right, each representing a step in the nested relational structure of the application. The expected user journey is progressively through the steps, as logging in, seeing the home screen, editing/creating a course and editing/creating a section.

5.3.2 Educado Mobile

The use case analysis and domain model in Section 5.2.2 indicates both conceptual classes and methods for core functionality of the mobile system. The model represents illustrates a top-down and progressive line of communication, starting with an employee as the sorting facilities. From the employee there is a 1-to-1 relationship to a user for which a login method exist. The user is then able to see completed/uncompleted courses and use methods for loading available courses, starting new courses and downloading courses for offline functionality. The next relation is then between a user object and a course object. In this case, the relation is a 1-to-* relation, indicating that the user can access many courses. The course object should contain values for the contents and nested sections, but also a method for saving user progress. Finally, the course related to the section objects consisting of the lowest nested layer of audio, video, text and image components. The functionality of this object is to load the components in a visual way for the user.

Based on the above description of functionality in the mobile system, an initial concept of an appropriate user-faced interface can be established. First the concept is visualized in free-hand drawing, and subsequently digitized in Wireframe models imitating standard design of a mobile application running on a smartphone device. The illustration can be seen below on Figure 5.8:

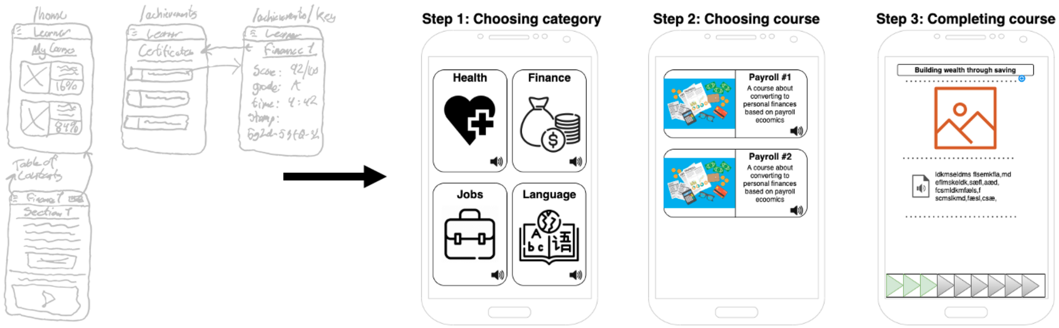

/home
Learner
My Courses
16%
84%
/achievements
Certificates
Finance 1
Score: 92/100
grade: A
time: 4:42
Table of Contents
Section 1
Step 1: Choosing category
Health
Finance
Jobs
Language
Step 2: Choosing course
Payroll #1
A course about converting to personal finances based on payroll ecoomics
Payroll #2
A course about converting to personal finances based on payroll ecoomics
Step 3: Completing course
Building wealth through saving
ldkmseldms flsemkfla,md eflmskeldk,sæfl,aæd, fcsmldkmfæls,f scmslkmd,fæsl,csæ,

Figure 5.8: Wireframe model for Educado Mobile

The development of the wireframe design model is done through multiple iterations of discussion and drawing across involved teams. Especially the PSP2 team (see Figure 4.1), focusing on the mobile learning experience, influence the initial design. Ultimately, the result is a conceptual idea of the application's user interface, that supports the core functionality and objects derived in the use-case analysis in Section 5.2. The model consists of three different screens illustrated to the right, each representing a step in the nested structure of the application. The expected user journey is loading of course category overview, seeing list of available course and completing a course by progressing through the component based contents.

5.3.3 User Journey

As described in Section 5.2 and elaborated upon in Section 5.3, the user journey is in focus for this project. This journey is summarized into the following model:

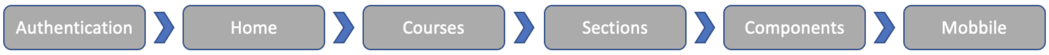


Figure 5.9: Illustration of the user journey envisioned for the system

This model is meant to give an overview of the journey and where in the system the user is. The model is used in Chapter 8 to maintain this overview and structure.

5.4 Conceptual software model

To summarize the conceptualization and design chapter, a model is created to illustrate the different software services involved and to create an overview of their interaction with each other, as seen below on Figure 5.10.

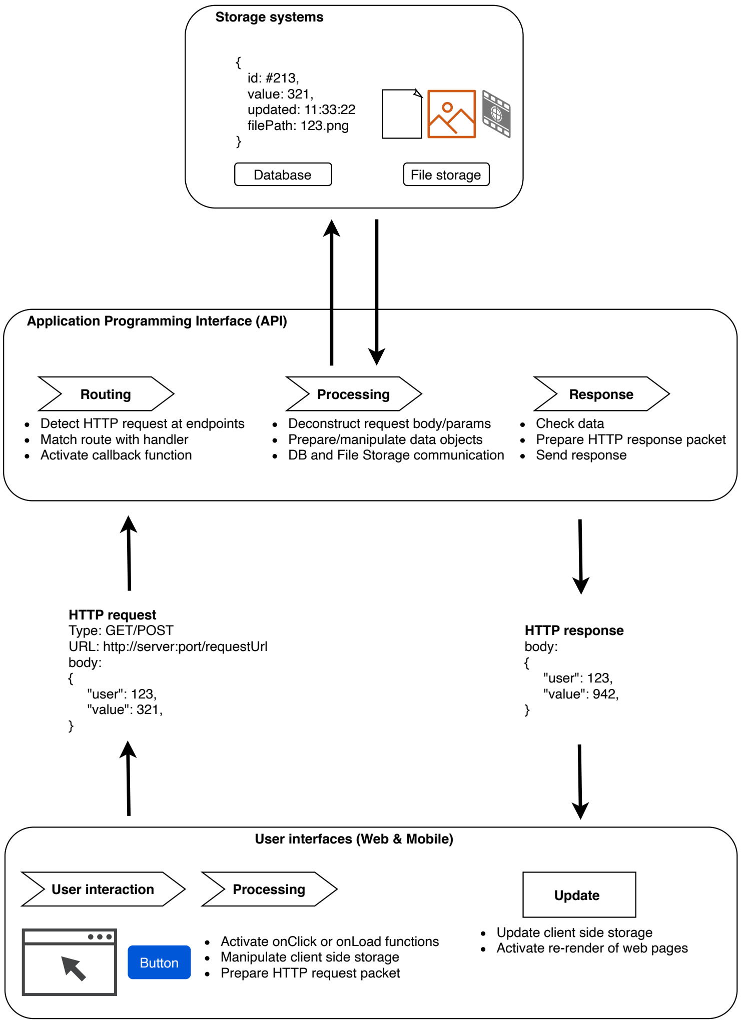

Storage systems
{
id: #213,
value: 321,
updated: 11:33:22
filePath: 123.png
}
Database
File storage
Application Programming Interface (API)
Routing
Detect HTTP request at endpoints
Match route with handler
Activate callback function
Processing
Deconstruct request body/params
Prepare/manipulate data objects
DB and File Storage communication
Response
Check data
Prepare HTTP response packet
Send response
HTTP request
Type: GET/POST
URL: http://server:port/requestUrl
body:
{
"user": 123,
"value": 321,
}
HTTP response
body:
{
"user": 123,
"value": 942,
}
User interfaces (Web & Mobile)
User interaction
Processing
Update
Button
Activate onClick or onLoad functions
Manipulate client side storage
Prepare HTTP request packet
Update client side storage
Activate re-render of web pages

Figure 5.10: Interface model

As illustrated on the model (starting in the bottom left corner), the flow of communication in the system starts with users interacting with either the web or mobile frontend. This triggers an HTTP request that is detected at the API by routing endpoints. While the API process the request it can communicate back and forth with the database and file storage systems depending on the request type and finally return an HTTP response to the client. When receiving the response packet the client will then update the local data-structures if necessary and trigger an update/re-render for the interface.

That concludes the conceptualization and design phase of the project. Conclusively, the design and domain models rooted in the problem domain through analyzing use cases are now established. A general understanding of both Educado

Web and Educado Mobile is established and expectations for the product implementation is aligned. In the next chapter, functional and non-functional requirements are established for the solution.

Chapter 6

Requirements

Having examined the problem in Chapter 2 and gotten a clearer technical view trough project conceptualization and initial design phases in Chapter 5, requirements for the system are made. Firstly, the scope and limitations of the project is considered in order to elaborate on the requirements to come and why exactly the ones chosen are in the requirement specification. Having established the scope and limitations, the methodology and testing for requirements is outlined, to adhere with the standards from structured systems development [38].

With these aspects considered, the requirement specification is drawn, with requirements being split into four categories: General, API & Storage Systems, Web frontend, and Mobile frontend as shown in Section 6.3. Lastly, the technologies chosen for the project design and implementation are considered with arguments as to these choices being presented in Section 6.4.

6.1 Scope & Limitations

Within this project, the authors are the developers of the application, effectively allowing for two people about half a year of time for design and implementation whilst managing the International teamwork as described in Section 4.1. With this timescale and amount of workforce available, the project must be scoped to be within the possible whilst getting as good a final product as possible.

With this in mind, the focus of this project is then to implement the core functionalities found within the comparative analysis in Section 2.2.2, wherein the ability to use text, image, audio and video where found to make the core of a digital education platform. Bearing this in mind, the scope of the testing for the application this project is focused on whether these functionalities are properly implemented (see Chapter 9). Properly in this context denoting both the actual functionality as well as the simplicity of future development leading to a future product, the waste pickers want to use.

In effect, the human centered design as described in Section 4.3 must be kept in mind trough the entire design and implementation, with the focus of this project stage being the core functionalities. Whilst feedback from this initial prototype from the waste pickers as a proof of concept would be a "nice to have", due to the Covid-19 situation alongside the lack of specialized features, no user testing is performed. This again leaves the requirements focused on if the functionality of the implementation allows for these core functionalities.

6.2 Requirements Methodology and Testing

"Making a robust requirement testing methodology is important to ensure they are worth pursuing. If a requirement is not put up to proper scrutiny, it can end up being impossible to judge whether or not to it can be / is accomplished. A real world example of important requirements can be found in Denmark, wherein the police and ministry of justice spent half a billion Danish crowns on a system they ended up not using, since it wasn't better than the old system [22]. As the article describes, this was in part due to poor requirements, leading to improper testing. As to avoid this issue, the requirement testing methodology as described Eric Sink is used [47]. More specifically an interpretation of Eric Sink's requirement methodology as described on the structural systems development is used [38]. A summary of the different test each requirement used in this methodology is given" [36], appendix 2:

- Correctness - "A requirement specification is correct, if and only if every requirement in the specification are not impossible"
- Unambiguousness - "A requirement specification is unambiguous if and only if every requirement has a meaning"
- Completeness - "A requirement specification us complete if and only if it includes several elements"
- Consistency - "A requirement specification is consistent if and only if no underlying requirements are in conflict with each other or overlying requirements"
- Organized after priority - "A requirement specification is prioritized if every requirement has a number allocated, which indicates the importance of the requirement"
- Verifiable - "A requirement specification us verifiable if and only if every requirement made is verifiable"

- Modifiable - "A requirement specification is Modifiable if and only if its structure is made as to enable the addition, changes and removal do not lead to inconsistencies or redundancies"
- Traceable - "A requirement is traceable if it is possible to trace from where the requirement originates from"

[36], appendix2

6.3 Requirement specification

The traceability of the requirements all connect to what is documented in the problem analysis (see Chapter 2) and the conceptualization and design (see Chapter 5).

General

1. The system must implement HTTP communication between UI applications (Web & Mobile) and an Application Programming Interface (API)
2. The system must store and synchronize data in a relational database
3. The system must store and synchronize files on a storage server

API & Storage Systems

4. The API must detect incoming HTTP requests at specified endpoints
5. The API must authenticate HTTP requests from Educado Web
6. The API must match all defined routes with an unambiguous handler function
7. All handler functions must activate a callback function that returns a HTTP response to the source
8. The API must be able to deconstruct HTTP requests and extract data from body and params objects
9. The API must be able to communicate with database server
10. The API must be able to communicate with file storage server

Web frontend

11. User must be able to access application through Google Chrome browser
12. User must be able to login through secure authentication (OAuth)
13. User must be able to see all courses related to user ID
14. User must be able to create, edit and delete the following elements:
    - Course
    - Section
    - Components (Text, Image, Audio and Video)
15. App must be able to render text, image, audio and video elements
16. User must be able to upload and download files up to 100 MB
17. User must be able to reorder position of sections in a list
18. User must be able to reorder position of components in a list
19. User must be able to persist all changes to file and database storage

Mobile frontend

20. User must be able to access application on an Android Smartphone
21. User must be able to see list of all available courses
22. User must be able to see the contents of a course
23. App must be able to render text, image, audio and video elements

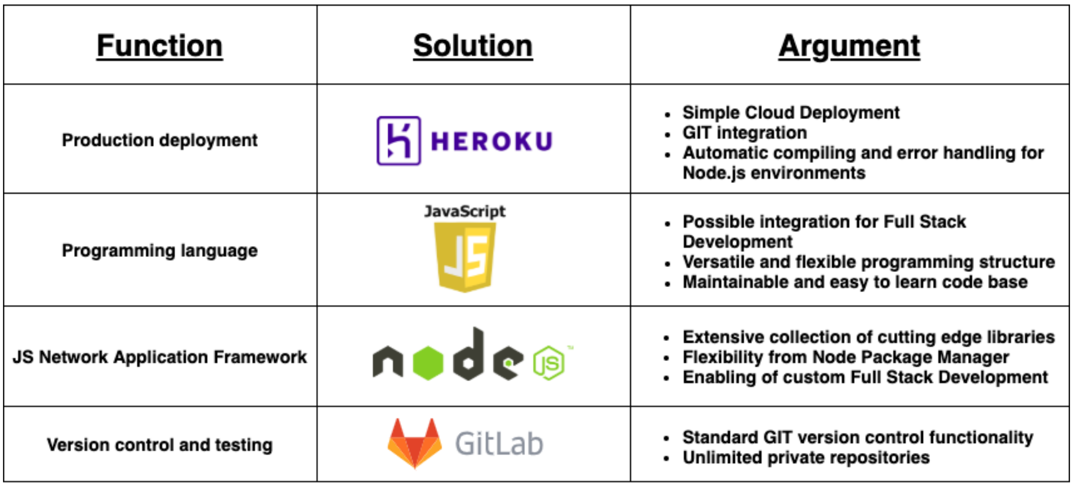
6.4 Selecting technologies

When selecting technologies to fulfill the requirements, serval key points are considered. Within this section, the aim is explain the choices of technologies made from a high level perspective, a larger deep dive into technical reasoning is provided within the technical analysis Chapter 7. Each choice made is listed within figures Figure 6.1, Figure 6.2 and Figure 6.3 alongside bullet points explaining some of the reasoning, further elaborated upon in this section. Considering, the project aim is to build a foundation on to witch future expansion is the goal, this expandability must be reflected within the technologies, not limiting the project to a prototype stage.

Furthermore, a goal of a complete ecosystem is set by the authors to keep all aspects 'in house'. Previously in the attempt to create MEP [12], the solution presented featured a modular aspect, with the 'course creation' aspect relying on Moodle staying the same so that course content could be scraped from the course website. This approach leaves dependencies for Moodle to stay the same or the web scraper would have to be re-written and does not give the course creator any idea of what the courses would look like on a smartphone.

Therefore a focus on a more streamlined and consistent experience throughout the applications with fewer dependencies on aspects such as the entire course creation part is needed.s

Due to the relative size of the application and the limited development time, ease of development to ensure deployment and completion of the prototype is also considered. This manifests itself with the choice of JavaScript as shown on figure Figure 6.1. JavaScript is chosen exactly because within it's different packages and frameworks, as seen in Figure 6.1, Figure 6.2 and Figure 6.3 and further elaborated upon in the technical analysis, enables the fulfillment of all requirements whilst staying within one programming language.



Function	Solution	Argument
Production deployment	HEROKU	• Simple Cloud Deployment • GIT integration • Automatic compiling and error handling for Node.js environments
Programming language	JavaScript JS	• Possible integration for Full Stack Development • Versatile and flexible programming structure • Maintainable and easy to learn code base
JS Network Application Framework	node JS	• Extensive collection of cutting edge libraries • Flexibility from Node Package Manager • Enabling of custom Full Stack Development
Version control and testing	GitLab	• Standard GIT version control functionality • Unlimited private repositories

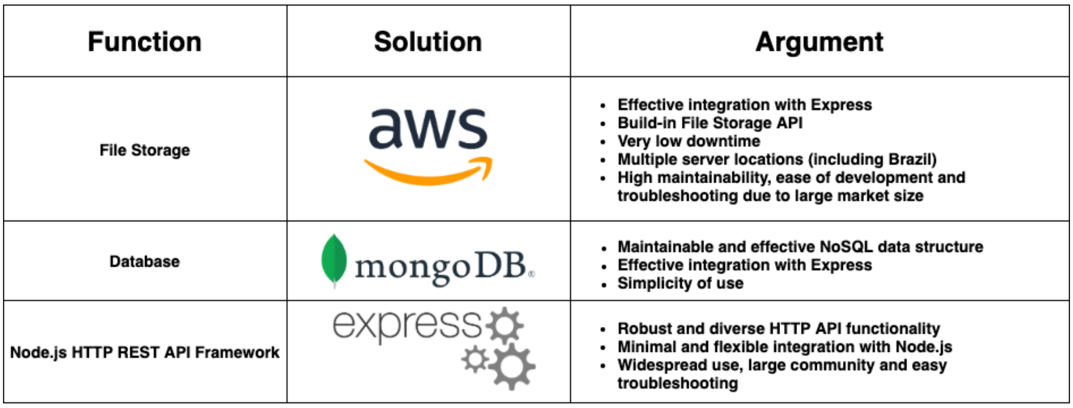
Figure 6.1: General technologies used in the project

The product deployment trough Heroku is made very simply as it allows for easy deployment trough git, other competitors are considered in section Section 7.3.1, however since Heroku fulfills the requirements for deployment well and simply, it is chosen.

Gitlab is one of many git repository hosting websites, however it was chosen due to the unlimited private repositories alongside familiarity both with git for version control and previous experience with Gitlab from the authors.

NodeJs is one of JavaScript libraries chosen due to it being within the Js development strategy and high amount of documentation, cross platform compatibility, support for the asynchronous communication needed as described within Section 6.3 and further elaborated upon in Section 7.1.

Having considered the core technologies, the backend technologies are considered. Similar to the overarching choices, the focus is the ease of development with possibility for future expansion without delegating large portions of the final implementation to things developed outside of the project.



Function	Solution	Argument
File Storage	aws	• Effective integration with Express • Build-in File Storage API • Very low downtime • Multiple server locations (including Brazil) • High maintainability, ease of development and troubleshooting due to large market size
Database	mongoDB.	• Maintainable and effective NoSQL data structure • Effective integration with Express • Simplicity of use
Node.js HTTP REST API Framework	express	• Robust and diverse HTTP API functionality • Minimal and flexible integration with Node.js • Widespread use, large community and easy troubleshooting

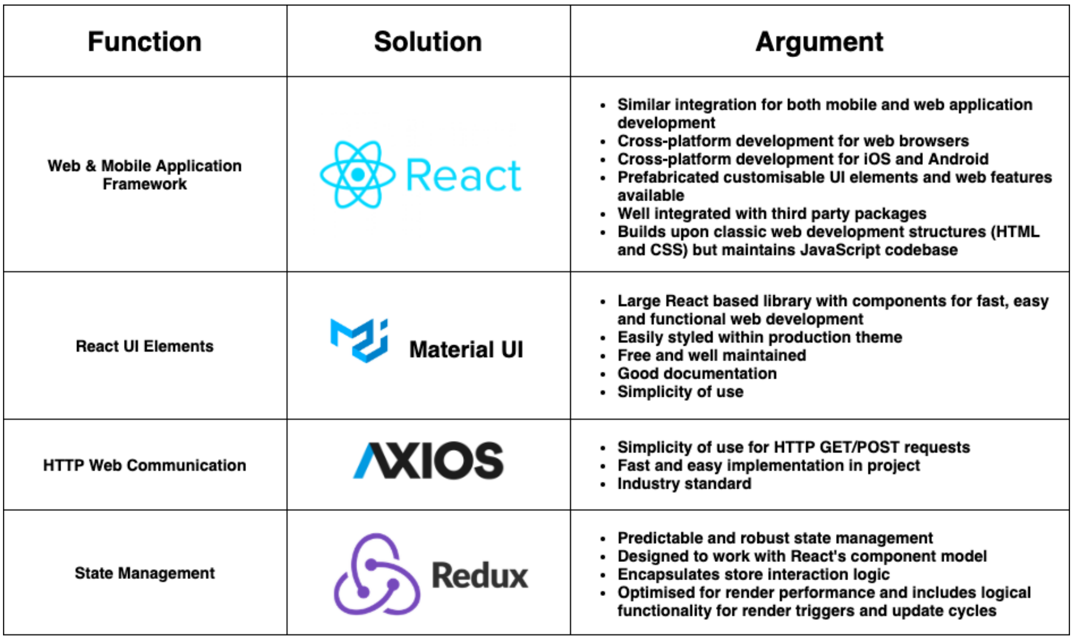
Figure 6.2: Backend technologies used in the project

AWS s3 file storage is chosen in part due to the authors previous experience with AWS in general including s3 buckets, in addition to the api offered for the s3 with features such as pre-signed urls for audio and video playback as described in detail in Section 7.3.2.

mongoDB is chosen as it allows a NoSQL database to be easily integrated with the HTTP REST API within the project and for its general simplicity of use.

The Express web framework is chosen due to how lightweight and stable it is, reducing bloat whilst enabling all the features needed whilst still staying within the JavaScript and NodeJs framework.

With the backend technologies considered, the frontend technologies are considered as they make the backbone of the user experience.



Function	Solution	Argument
Web & Mobile Application Framework	React	• Similar integration for both mobile and web application development • Cross-platform development for web browsers • Cross-platform development for iOS and Android • Prefabricated customisable UI elements and web features available • Well integrated with third party packages • Builds upon classic web development structures (HTML and CSS) but maintains JavaScript codebase
React UI Elements	Material UI	• Large React based library with components for fast, easy and functional web development • Easily styled within production theme • Free and well maintained • Good documentation • Simplicity of use
HTTP Web Communication	AXIOS	• Simplicity of use for HTTP GET/POST requests • Fast and easy implementation in project • Industry standard
State Management	Redux	• Predictable and robust state management • Designed to work with React's component model • Encapsulates store interaction logic • Optimised for render performance and includes logical functionality for render triggers and update cycles

Figure 6.3: Frontend technologies used in the project

For the overall JavaScript package, React is chosen for several key reasons. Firstly, it allows for both web and mobile application development with very similar integrations, simplifying the development of the two main frontends. Furthermore, it keeps in line with the streamlined JavaScript development and has access to packages such as Material UI.

Material UI is a library of frontend modules, simplifying development and enabling global styling alongside its excellent documentation.

For communication from the frontend to the backend, AXIOS is chosen due to the simplicity of implementing HTTP get and post requests whilst being well documented due to it being de facto industry standard.

For state management, which is described in section Section 7.2, redux is chosen, as it integrates with React frameworks in a predictable and reliable way.

Whilst this cover the high level methodology to choose technologies and the technologies chosen, a further detailed technical analysis is commenced, to enable proper design and implementation.

Chapter 7

Technical analysis

Distributed systems are computing systems consisting of many individual elements. These elements are able to behave independently of each other and communicate, but appears to the user as a single system[50]. Modern distributed systems can, and often will, be constructed of a diverse collection of sub-systems, ranging from large cloud instances to sensor networks. The sub-systems are programmed to collaborate through passing and processing messages, with aims to achieve a common goal. In this project, the computing system developed, is a distributed system consisting of nodes such as mobile and web based frontend applications, database and file storage servers and an application programming interfaces (API). As the main line of communication between nodes is web based (HTTP), the system is referred to as a Distributed Web System (DWS).

Working with systems of collaborative but independent nodes, naturally leads to issues such as synchronization, scalability, dependability and bottlenecks. Scaling the system automatically increases complexity and overhead, so maintaining a systematic and structured system architecture is crucial. In this chapter, we investigate some of these issues and presents theoretical approaches for developing a stable, efficient and timely DWS system. The theoretical material is partly based on studies from the field of Distributed Computing, but also from general web development and database theory. It is important to note that, in Section 6.4 the technologies used in the project were chosen to fit the software needs derived from the requirements described in Section 6.3. From this, it follows that this chapter to some extend reflects software specific limitations and considerations.

7.1 Distributed Web Systems

In Chapter 5 the envisioned solution is described and it is clear that the system consists of multiple independent components communicating with each other. Developing a software system in this distributed context requires an architecture that supports both flexibility, scaleability and efficiency in regards to the system requirements (see Section 6.3). Different approaches to distributed computing architecture naturally comes with different advantages and drawbacks. In Section 7.1.1, the traditional monolithic architectural structure and the increasingly popular cloud-native Microservices approach is explored and evaluated in the context of the project. Subsequently, Section 7.1.2 explores issues of concurrency, time and synchronization that arises in a distributed context and compares traditional web communication flows with Ajax based ones. Conclusively, fault tolerance and graceful degradation, are discussed in Section 7.1.3, as these are crucial in building a stable application.

7.1.1 Monolithic versus microservices

An increasing number of software systems are migrating from traditional monolithic architectures to service-based architectures as application complexity rises and requirements for scaleability and flexibility increases. Often connected to the growing adaptation of DevOps practices in systems and software engineering, service-based architectures, such as microservices, allows for larger teams of engineers, designers and developers, to more easily divide responsibility and manage concurrent development, which has shown reduce time-to-market[10].

A monolithic software system is built and managed as a single unit[56] (see Figure 7.1), though the system typically still consist of multiple internal components. An example could be a software application comprised of a user interface, a server-side handler application and a database. In a monolithic architecture each individual sub-system are centralized and managed from one place through a top layer of software logic encompassing the entire system. The main benefits from this approach is simple deployment and development as the system can be handled as one unit. Additionally, communication is simplified greatly compared to a more distributed approach as most requests/responses will be determined entirely internally between sub-systems.

Though this architecture has long been regarded the traditional and most used in software development, technological progress with regards to cloud deployment as well as a general increase in application complexity, has lead many leading technology companies (E.g. Netflix, Google and Amazon)[56], to migrate mainly to a service-based approach such as microservices. The weaknesses of the monolithic architecture becomes increasingly visible as an application scales up and the natural tight coupling of a centralized codebase, results in many cases in an error-prone and rigid system structure, that is hard to maintain and develop.[56]

In a microservices architecture each sub-system is regarded as an independent service running isolated processes[35]. Communication between services are typically facilitated through lightweight mechanisms such as HTTP restful frameworks[56]. No encompassing top-layer logic exist, so even central management of interconnections between services are handled as a completely isolated and separate service. This structure initially adds many layers of complexity but it results in a modularized environment with natural benefits such as independent failure of components and the ability to update services individually. It also allows for engineers to develop entirely isolated on each service without the risk of introducing errors in the system, as long as the interfacing between services is strictly maintained. An example could be in an application comprised of a user interface, a HTTP REST API and a database.

If managed as microservices, a team of database specialists could manipulate data queries and optimize performance without issuing updates to either the frontend or REST API, as long as the returned data format remains the same. The same would be viable for the frontend team with regards to updating visual appearance and the API team for updating route logic. Naturally, this flexibility and modularity is greatly beneficial in larger applications where many teams collaborate concurrently.

The difference between monolithic and microservices architectures are illustrated below on Figure 7.1.

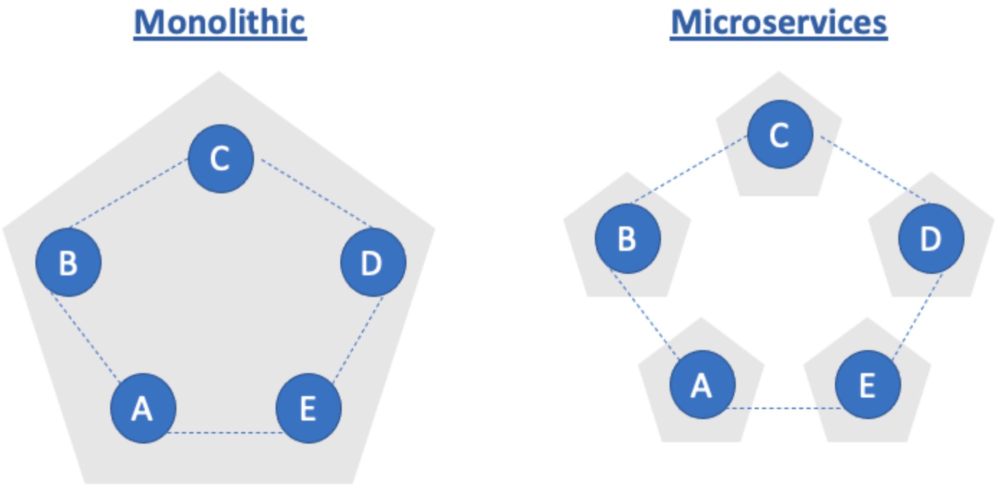


Figure 7.1: Illustration of monolithic and microservices architectures

As detailed in the conceptualization and design chapter (see Chapter 5) and the requirement specification (Section 6.3), the solution envisioned in this project, consist of a large scale application environment consisting of multiple subsystems. Based on the general composition of components and their interfaces to each other is detailed and illustrated in Section 5.4 and the project ambition to establish a software foundation fit for continuous development and contribution from many teams, a microservices architecture is evaluated to be superior to a monolithic approach in this project.

Designing the solution as a microservices architecture can be done in many ways but based on derivations from the conceptualization and design chapter, the individual microservices in this project solution is:

1. Mobile application (Educado Mobile)
2. Web application (Educado Web)
3. HTTP REST API
4. File storage (AWS S3)
5. Database (Mongo DB)

The communication between the systems is facilitated through HTTP requests/responses and managed by the HTTP REST API as a central module. This microservices structure is illustrated bellow on Figure 7.2.

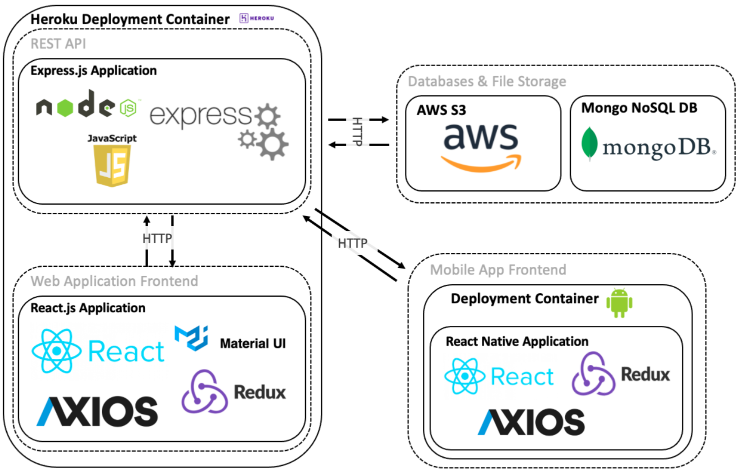


Figure 7.2: Solution microservices architecture

As mentioned, developing a system in microservices architecture adds multiple layers of complexity as each module is built as an isolated service and therefore must interface and communicate with the other distributed components through HTTP calls. Additionally, deployment and application environment sees increased complexity as each service must be deployed individually. One of the most crucial effects of microservices in a distributed context, is that the notion of a global clock is removed, and hereby concurrency, time and synchronization issues arises as components are communicating. The next section addresses these issues and discuss how it is solved in this project by using asynchronous HTTP calls in client-to-server communication.

7.1.2 Concurrency, time & synchronization

Using microservices architecture as described in Section 7.1.1, results in no global notion of time. This introduces the need for asynchronous communication between services. The asynchronous communication is required to diverge from traditional client/server communication which is synchronous. As discussed in the previous section and illustrated on (Figure 7.2), the two frontend applications (mobile and web) will communicate via HTTP with the Express.js REST API. Additionally the REST API will also communicate with respectively the MongoDB database and the AWS S3 File Storage via HTTP requests/responses.

As an example, when a user interacts with Educado Web and creates a new digital learning course, the application sends a HTTP request to the REST API. The request will most likely have attached local data points from user input in the frontend system (such as title, cover image or description as detailed in Chapter 5) as JSON body/params. This must be send asynchronously as the application should wait for the HTTP response before triggering reload or redirect but concurrently continue all other logic.

To realize this functionality requirement an engine for handling asynchronous communication in JavaScript applications is needed. The most common framework of this kind is Ajax (Asynchronous JavaScript XML), which has been widely adapted as the standard engine for asynchronous web communication[19], and is also used in this project.

A comprehensive explanation of the Ajax engine is beyond scope of this project but the remainder of this section, details the most important concepts of the engine as per the use in the project. Initially, Figure 7.3 illustrates the difference between a traditional web model and one using the Ajax engine.

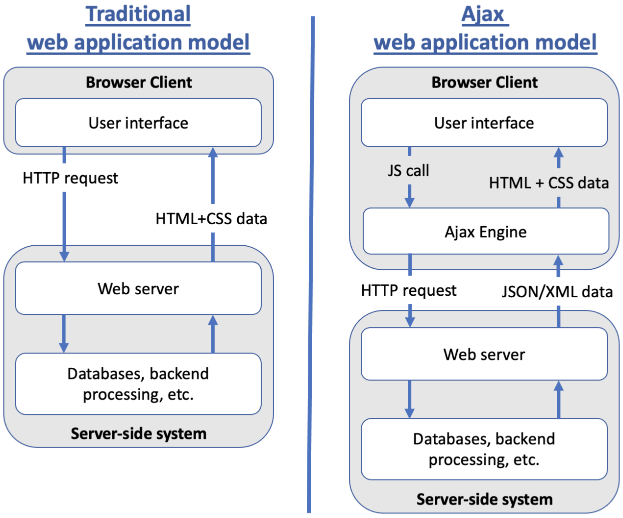


Figure 7.3: Traditional web application versus Ajax application[19]

The difference between a traditional (synchronous) and Ajax (asynchronous) web communication, is similar to that of total and partial order in the context of distributed computing. As illustrated on Figure 7.4, synchronous communication is a linear process over time alternating between user activity on the client-side and server-side processing. This process involves two transactions of data - (i) some user activity, for example the press of a button with an "on click" handler, will trigger a request to the server and (ii) after processing the request, for example deserializing values and subsequently storing in database, the server will issue the response back to the client. As no activity happens on the client side while the request is processed at the server, the process is synchronous.

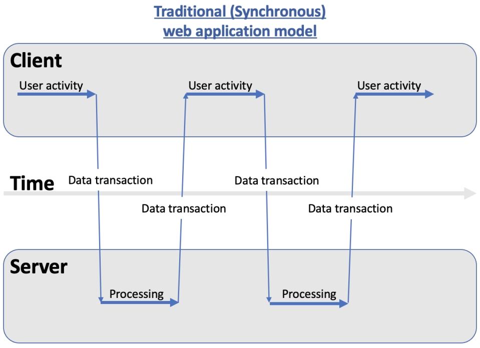


Figure 7.4: Synchronized communication model[19]

Contrary, in the case of asynchronous communication via Ajax, the client side has an internal layer of client-side processing and is able to maintain levels of activity whilst waiting for response from server, as illustrated on Figure 7.5.

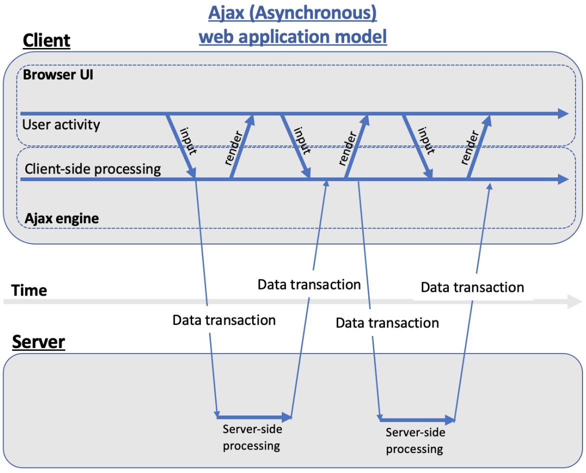


Figure 7.5: Asynchronous communication model[19]

Asynchronous communication will make the basis of all HTTP communication between microservices in this project. In Chapter 8, concrete examples of how these Ajax calls are implemented is presented.

7.1.3 Fault tolerance & graceful degradation

When deploying software systems into production where users rely on functionality to be available when requested, handling errors and tolerating faults becomes an increasingly important factor. Designing a system of distributed web components to handle errors gracefully requires both programmatic, networking and architectural considerations.

As explained in Section 6.4, the entire codebase of the solution - Web frontend (React.js), Mobile frontend (React Native.js) and REST API (Express.js) - is coded in JavaScript. Having only one programming language unifies programmatic conventions, eases maintainability and speeds up development. Additionally, it also allows developers to establish a unified approach to handling programmatic errors and faults that may arise either during development or production testing. JavaScript has built in functionality for catching errors referred to as 'try-catch' expressions. A 'try-catch' expression consist of two parts: (i) a 'try' statement to that executes a block of code with one or multiple function calls and (ii) a 'catch' statement that takes an error as input and subsequently hosts logic for handling that error (see Listing 7.1).

```
try {
  sendAjaxRequest(JSON,"http:restApiLocation:port/path");
  } catch(err) {
    log(err);
    }
```

Listing 7.1: JavaScript 'try-catch' error handling

This is a general programmatic flow that is used often in the codebase for this project. As the pseudo code snippet above exemplifies, an often used example of its application in the project, is when introducing code that executes Ajax requests. By sending the request inside a 'try' statement, potential errors are subsequently caught from the response and logged.

In terms of networking and architecture, graceful handling of errors and faults is obtained by the microservices structure (see Section 7.1.1). By separating responsibilities into individual services the microservices architecture allows the system to continue running even if one component fails. An example could be if the database fails and becomes unavailable, the effect would not be that the entire application crashes, but instead that elements rendered from the database would simply not be updatable and the user could be presented with an error message, telling that the database is currently unavailable.

7.2 State management in Single Page Applications

When working with state and state management within an application, it in broad terms means that the application can return different things or 'States' instead of just returning one static output. Since the learning application is meant to be interactive to enable things such as the smartphone app user being able to click on different symbols to the web application, the user needs to be able to upload and reorder learning materials as described in Section 6.3. This alongside the applications being reactive to said state changes so that the course creator can see the actual course they have created, is what makes working with state essential.

This is in turn a complimentary way in which to build a Single-Page Application (SPA). Within a SPA, the web app dynamically redraws on the same page with data from the server, in this case the Heroku instance described in Section 7.3.1. Traditionally, in a Multi-Page Application (MPA), whenever new information needs to be redrawn it would load a new web page, however with the SPA methodology, the 'state' of the application merely changes. However SPA applications can lead to rising complexity for development.

7.2.1 Rising complexity of SPAs

As a consequence of the SPA development methodology, the complexity of the state management can become very complex. Consider a web application with three pages, however each of these pages in turn each have interactable functionalities. Not only must the switch between each page or state be correct, so does each function within the application and the state update correctly for each state. To exemplify both the concept behind a Single-Page application and this state within state feature creep see figure Figure 7.6

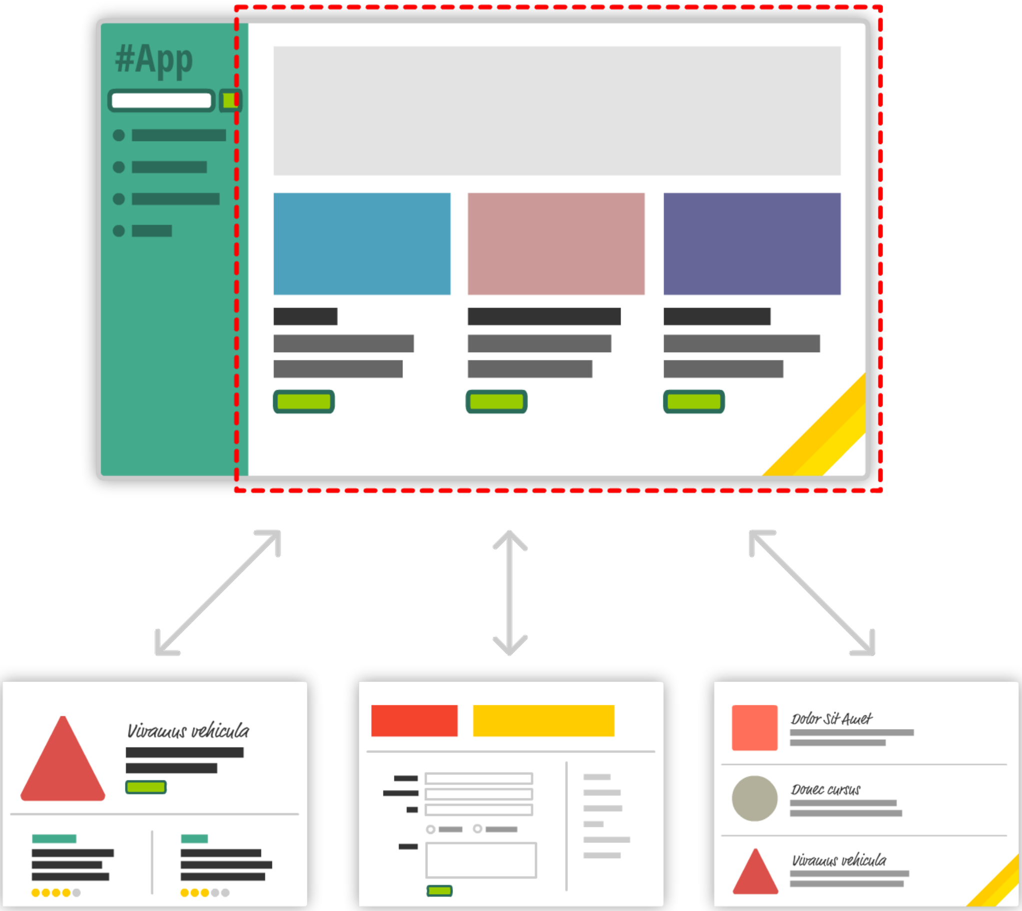
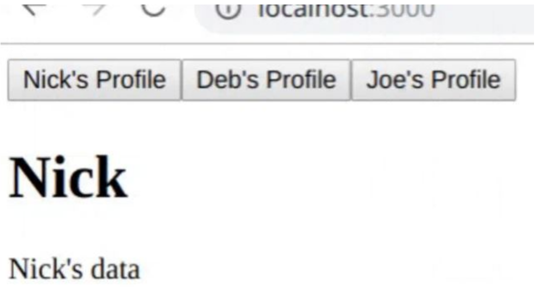


Figure 7.6: A Single-Page Application visualized [15]

Within this Single-Page approach some potential issues can arise. Firstly, race conditions can come into play and cause errors. Race conditions occurs when a software program depends on the timing of one or more processes to function correctly [52].

For example within a React environment, if a user is attempting to fetch data in rapid succession from a server with a random delay, then the wrong data could appear. This could happen due to the state of data and username being desynced, effectively showing the wrong data for the user. In the following example both a user and their data are retrieved. However with a random delay arbitrarily inserted within a margin of time when retrieving data to simulate either delays in traffic or different sizes of the data for each profile. In the first example things go well and Nick's data is retrieved for the user Nick.

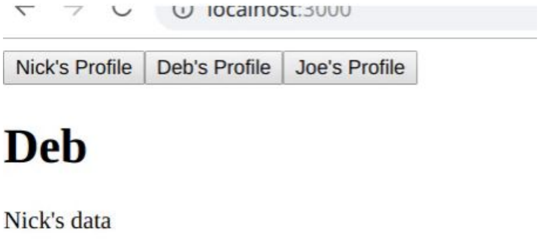


Nick

Nick's data

Figure 7.7: State when data and profile are correctly aligned

However in the second example many users are clicked in rapid succession and due to the random delays the final data updated in the state is Nick's, but the user is actually on Deb's page.



Deb

Nick's data

Figure 7.8: State when race condition error occurs and wrong data is received from person

A coded version of this example alongside a gif of the users input can be found in[44].

This showcases the importance of working with race conditions within this signal page application, as improper handling could lead to these issues. Furthermore, since the different states could manifest in both full 'pages' changing as well as just parts having to update an additional layer of complexity is added. Specifically now both the surrounding state must be correct as well as the specific parameters needed to update.

For example, if a user is to upload a video, the page must firstly load the components, then the upload functionality and correctly update the file once the upload is complete. All of these steps have to be handled correctly within the state management and updates called for correct parts at correct times. Additionally, the changes can also be saved locally if for example the user is writing text within a text input field it makes little sense to update on the backend for each character input, due to the high amounts of traffic needed. Consequently, either a pre-determined autosave or manual save or similar can be implemented only causing a full state update after each save, whilst input is temporarily saved in a local state. This local state does not effect the true 'global' state saved within the backend, but again adds another layer of complexity in state management. In order to manage state within a React web application Redux is used.

7.2.2 Redux

"Redux is a predictable state container for JavaScript apps. It helps you write applications that behave consistently, run in different environments (client, server, and native), and are easy to test. On top of that, it provides a great developer experience, such as live code editing combined with a time traveling debugger." [43].

Essentially redux is the main tool used for state management within a React environment that works for both The Web Application and the Smartphone Application. Redux works by having one central data (State) store and crucially only one store for all state management. Within this store are things such as authentication state, user input etc. The data within the central store is then used by the different components of an application. For example a change to the authentication state would be important to act on, by for example allowing a user to login given the right credentials and consequently update the UI.

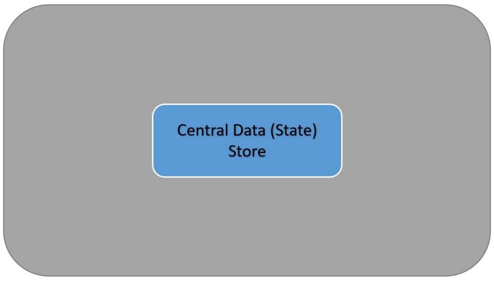


Figure 7.9: The central data store

This works by the components subscribing to the store for the different parameters they use to act upon changes accordingly. Again, for example the authentication state of the application. In other words the components subscribes to the relevant data or states within the store to see if changes that affect them occur so the components can act upon these changes.

Since the data or state within the store changes, one could assume the components then changes the data within the store. However this is never the case. A component never directly changes the state within the store. Effectively this gives a one way flow of data wherein the subscriptions provide data to the components, but the components do not directly change the data within the store.

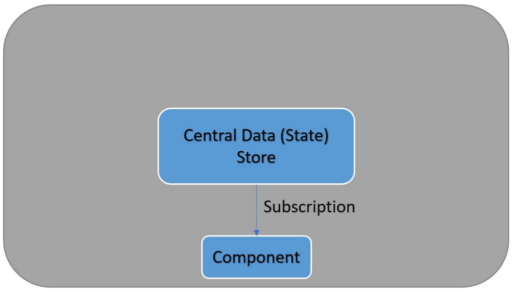


Figure 7.10: The subscription relationship from components to the central store

Instead this is done trough something called a reducerSection 8.3. A reducer or reducer function is responsible for changing or 'Mutating' (within redux terminology) the store data. Reducer functions take an input and transform it into an output which in the case of an reducer in this context changes the store data.

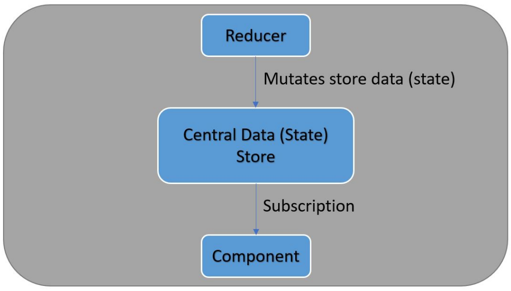


Figure 7.11: The reducer functions relationship to change state

To take activate this reducer function and in turn change the data store, a component dispatches an action. This could be if the user presses login onto a website, the component would upon the button click then dispatch this action of the button click. Another example being if the user has pressed upload and selected an image, the component would then dispatch the action of upload and the image to be uploaded as well.

The action is a simple JavaScript object which describes the type of operation the reducer is to perform. This in turn is passed along to the reducer which reads the action to find out what type of operation or function it has to perform. So the action itself does not perform anything beyond describing what state change it is supposed to perform, and the reducer function performs the actual function and reduces it into the change it inputs into the central data store.

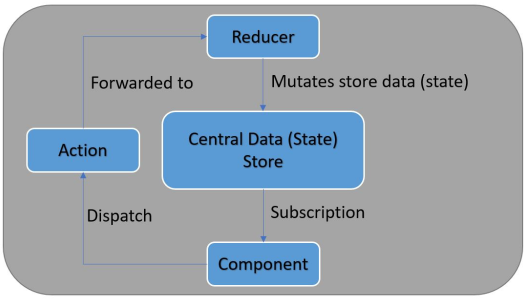


Figure 7.12: The components dispatching actions into the reducer, ultimately changing state

The implementation of redux is showcased in implementation under Section 8.3, wherein code examples from this project are found. During the project development a new way of managing state in react based applications called the Context api was created.[42] However, the project was already to far ahead in its programming phase to consider switching into the context api, though its existence is acknowledged. For further development considerations should be made into switching to a Context api platform instead of Redux. Finally, it should be noted that all of these actions that end up changing the state is usually coupled into the different micro services that in turn work with the backend.

7.3 Databases & File Storage

All of these previous microservices have to connect trough somewhere and be able store data such as files or handle user authentication. For the ladder the need for a database for authentication is used. Furthermore a database could store everything text based including links to course materials. These course materials have to be stored somewhere, as a general purpose server setup or dedicated file storage system. These elements have to be interconnected somehow, with dedicated processing power to enable all of these services.

7.3.1 Cloud solutions

To connect all of the backend structure and enable both the web and smartphone applications to receive data, a processing solution is needed. Furthermore the goal is for this projects prototype to run smoothly for both the Danish team and the Brazilian teams for both PSP2 and PSP1 to use within their respective projects Section 4.1. To accomplish this either a local deployment or a server deployment is needed. Due to the lower loads needed alongside the simplicity of setup a server deployment is chosen.

Deploying applications such as this can be done via the big three companies of AWS, Google, Microsoft [18] among others. As the goal of the deployment is to simplify the process as much as possible, two main contenders are found: AWS amplify and Heroku. Both feature a simple git based deployment with the machine the app is loaded onto being configurable with for example security keys.

Heroku is chosen due to its relative simplicity alongside quality of documentation. In future development of the project including large scale deployment, this should be revisited both in terms of performance and costs. However, for the prototype Heroku will perfectly allow fulfillment of the requirements specification. Notable for both is the simplicity of the migration process between one service to another, meaning to project is not locked firmly into one at this early stage of development [34]. No matter which cloud solution is chosen, the files used for the course must be stored somewhere.

7.3.2 AWS S3 File storage

As with the processing both a local or cloud based structure can be used, again cloud based is chosen at this point due to it being relatively hassle free. Moreover thought is given towards the future of the project as files would need to be stored as close to the user as possible to avoid issues due to high ping or losses due to the sheer length of the connection. Since the goal is to provide education to waste pickers in Brasilia Brazil, its only sensible to allow the files to be located closer. Via a cloud solution it can be easily tested in Denmark whilst the production file storage server was in Brazil.

For the storage, bigger tech companies such as AWS, Google, Microsoft, DigitalOcean among others provide solutions [18]. Within the project in this stage only CRUD on files alongside a way of grabbing them from the server is needed. For this purpose most of the offerings could be chosen, however, due to previous experience from the developer team, S3 storage from AWS is chosen. S3 allows for an implementation that fulfills the requirements as shown in Chapter 6. Moreover both Heroku and AWS amplify runs on AWS servers leading to a somewhat more common approach.

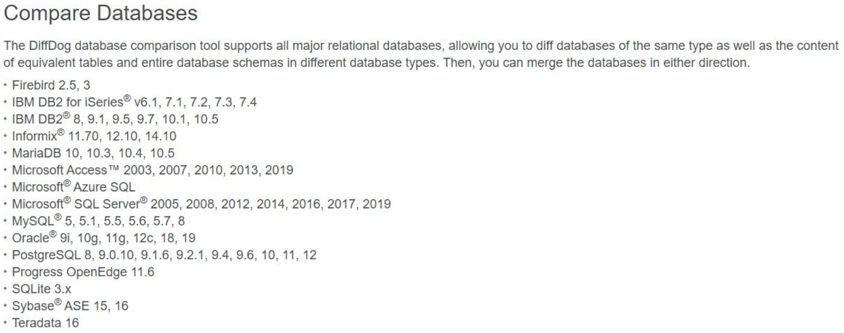
Two significant features of the AWS s3 file storage are the built in REST API and the ability to generate presigned-urls [9]. Firstly, the REAT API simplifies the code needed to contact it and perform CRUD [16]. Furthermore it allows the use of a methodology to send and receive communication securely trough key-based authentication.

Secondly, it enables the generation of presigned-urls, effectively making it so big files do not have to be downloaded all at once when a file is in the active state. For example, the page does not have to load until it has finished downloading a 20 minute long video, but rather enables buffered viewing trough this presigned-url. This methodology is further elaborated upon with code examples in Section 8.6. Generating the presigned-url requires and exact unique identifier for each file stored somewhere. With this requirement alongside the need for a list of authentication, a database implementation is sensible.

7.3.3 SQL vs. NoSQL

For the database implementation, the goal is to fulfill the requirements, effectively connecting many pieces and enabling core features Chapter 6. Since the prototype needs future expansion, for example the need for smart-caching as described in Section 2.2.2 or testing as described in Section 4.3, the implementation must also remain flexible.

Many database structures exists as shown just within this comparison tool showcases, with many more available:



Compare Databases

The DiffDog database comparison tool supports all major relational databases, allowing you to diff databases of the same type as well as the content of equivalent tables and entire database schemas in different database types. Then, you can merge the databases in either direction.

- Firebird 2.5, 3
- IBM DB2 for iSeries® v6.1, 7.1, 7.2, 7.3, 7.4
- IBM DB2® 8, 9.1, 9.5, 9.7, 10.1, 10.5
- Informix® 11.70, 12.10, 14.10
- MariaDB 10, 10.3, 10.4, 10.5
- Microsoft Access™ 2003, 2007, 2010, 2013, 2019
- Microsoft® Azure SQL
- Microsoft® SQL Server® 2005, 2008, 2012, 2014, 2016, 2017, 2019
- MySQL® 5, 5.1, 5.5, 5.6, 5.7, 8
- Oracle® 9i, 10g, 11g, 12c, 18, 19
- PostgreSQL 8, 9.0.10, 9.1.6, 9.2.1, 9.4, 9.6, 10, 11, 12
- Progress OpenEdge 11.6
- SQLite 3.x
- Sybase® ASE 15, 16
- Teradata 16

Figure 7.13: A list of supported types for this comparison tool

[3]

To simplify the selection process, both authors have experience within an SQL environment, making it simpler to work with. As with both the cloud storage and processing, the need for a quick and simple implementation is needed to hasten development. Bearing this alongside the flexibility in mind, mongoDB is chosen which uses NoSQL[31].

As described in an article in keycdn [4], mongoDB is fast and easy to use whilst the data structure is very flexible. Some limitations such as SQL not being used as the direct query language are almost mentioned, but due to the relatively simple (CRUD) [16] operations to be performed on the database, this is not an issue.

In this chapter the composition of a distributed web system in Section 7.1 was examined and a microservices system model (see Figure 7.2) proposed. Additionally, Section 7.2 explored the use of React Redux to manage dynamic updates of local/-global data values and Section 7.3 detailed integration of MongoDB and AWS s3 to handle file and database storage. The following chapter documents the design and implementation of the proposed system.

Chapter 8

Design and implementation

In this chapter, the process of designing, developing and implementing the solution as a software system is documented. The chapter starts by providing a general overview of the solution in Section 8.1 and is subsequently divided into six sections representing the user journey described in Section 5.3.3. In each of the sections, the system functionality is detailed and programmatic examples from both backend and frontend systems are included.

8.1 System overview

As documented in Section 7.1 and illustrated on Figure 7.2, the solution is designed as a microservices architecture and includes five individual services: (i) a HTTP REST API, (ii) a database, (iii) a file storage server, (iv) a web application and (v) a mobile application. Each service is independent but interfaces to other services via. specified channels of communication. As detailed in Section 7.3 the database and file storage systems are managed through respective user interfaces and does not have their separate codebase in the project structure. Instead the communication to/from these systems are handled by the intermediate module - the Express.js REST API. As a result of this structure, the project has three separate code bases for the reminding three services. The directory tree structure for each of these systems are covered in this section alongside explanations as to why the systems are structured this way.

8.1.1 REST API

As detailed in Section 6.4 and Section 7.1.1, the REST API acts an an intermediate service facilitating communication between frontend and backend systems. The API is coded in Express.js and consist of the following directory tree.

```
config/
```

```
    keys.js
    dev.js
    prod.js
  models/
    User.js
    Course.js
    Section.js
    Component.js
  routes/
    authRoutes.js
    fileRoutes.js
    dbRoutes.js
    testRequests.http
  index.js
  package.json
```

The three important top level directories are: (i) config, (ii) models and (iii) routes.

Config contains the necessary keys for both production and development environments. This includes Google Authentication Keys (see Section 8.2), MongoDB URI key Section 7.3.3, session storage key and the API key for AWS s3 communication Section 7.3. The keys.js file handles logic switching between production and development and ensures that the proper keys are used so communication between services remains secure and stable. (See Listing 8.1)

```javascript
// Figure out what set of credentials to return...
if (process.env.NODE_ENV === 'production') {
  // We are in production. Return the production set of keys
  module.exports = require('./prod');
} else {
  // We are in development. Return the development keys
  module.exports = require('./dev');
}
```

Listing 8.1: Keys.js in config folder of REST API

Models contains the logic for defining database objects in the MongoDB database.

Routes contains the routing logic for all HTTP GET/POST calls.

8.1.2 Educado Web

The web application is structured accordingly to Reacts standard[49] and consist of the following application structure.

```
public/
src/
    components/
    consts/
    hooks/
    pages/
        Login/
        Home/
        CreateCourse/
        EditCourse/
        ...
    store/
        actions/
        reducers/
    App.js
    index.js
package.json
```

The most important directories in the structure are: (i) public and (ii) src.

Public contains static files in the application such as logos, sound effects and icons.

Src contains the main react code separated into multiple important structures that reflect the frontends appearance. Reusable components such as video player, image handlers or titles are gathered in the components/ directory, while each page in the application is placed in the pages/ directory. Hooks/ contains reusable function calls and consts/ contains constants such as color themes. Finally, the store/ folder is the central place for Redux logic, as explained in Section 7.2

Focus during development and on the chosen design is utilization of Reacts reusable components structure. This is what enables the object oriented inheritance between courses, sections and components in the application, as also detailed in Section 8.3. This structure creates maintainable and simplistic frontend design, which is a necessity for a successful application in production.

8.1.3 Educado Mobile

The mobile application is written in React Native and the structure is similar to that of the web application detailed above. Exactly this similarity in the programmatic base between the two distinct frontend systems is a main reason why the React framework is chosen. It eases maintainability and simplifies development drastically. The structure of the mobile application is illustrated below.

```
 public/
 components/
 consts/
 hooks/
 page/
     Home/
     Courses/
     ActiveCourse/
     ActiveSection/
 App.js
 package.json
```

8.2 Authenticating users and logging in

The initial step of the user journey (see Section 5.3.3) is when a user visits the web application through a browser and is presented with a login screen. A user login functions as the top level of the Object Oriented hierarchy, so that content can be attached and restricted to/from users. Authentication in this project is achieved through OAuth, as a means to outsource the security concerns of storing passwords.[37] Many third-parties provide OAuth integration API's but since our Brazilian collaborators all uses Gmail, it is decided to use Google OAuth to simplify signup. This authentication process is the first step on the user journey as illustrated on Figure 8.10.

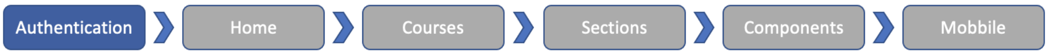
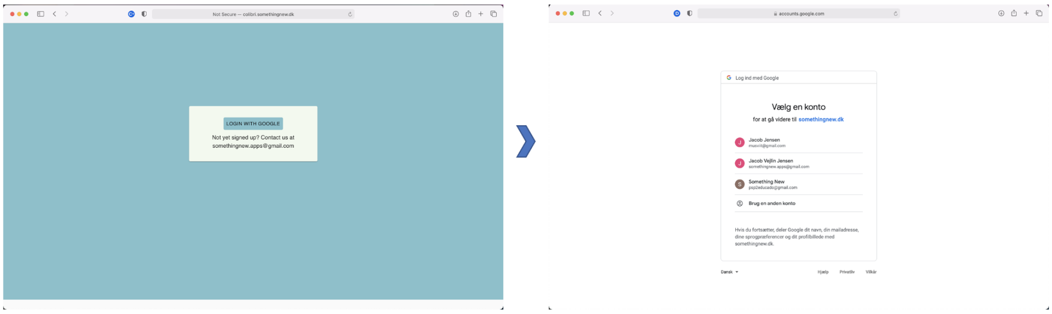


Figure 8.1: User journey step 1: authentication

OAuth is an open standard for authentication that has been widely adopted and for which companies such as Amazon, Google and Facebook all expose open API's for. It is beyond the scope of this project to fully comprehend the authentication principles in an OAuth flow, but generally it is a method for developers to build web applications with secure access logins by outsourcing password handling and storage to a third-party. As mentioned above, many established technology companies provide such third-party integration to other software developers as a way for users to 'login with service'.

The implementation of Google OAuth in this project is developed with the authentication middleware framework for Node.js 'Passport.js'[21]. This framework provides lightweight unobtrusive integration into an Express.js API by allowing developers to choose from a library of maintained authentication strategies, including Google OAuth and therefore supports the system structure in this project. When a user visits the domain 'colibri.somethingnew.dk' in their browser, they will be presented with an initial login screen, as illustrated on the left side of Figure 8.2 below. When pressing the 'login with Google' button they will be redirected to Googles authentication sub-domains as shown on the right side of the figure. Here the user will be able to choose one of their registered Google accounts and issue a request to the Educado Web application to login with that account.



Not yet signed up? Contact us at somethingnew.apps@gmail.com

Vælg en konto

for at gå videre til somethingnew.dk

Jacob Jensen
musviit@gmail.com

Jacob Vejlin Jensen
somethingnew.apps@gmail.com

Something New
psp2educado@gmail.com

Hvis du fortsætter, deler Google dit navn, din mailadresse, dine sprogpræferencer og dit profilbillede med somethingnew.dk.

Figure 8.2: (left): intitial login screen, (right): Google OAuth redirect

The request will then be received by the Express.js REST API (see Figure 7.2) and activate the Passport.js authentication logic. As seen on lines 2-6 in the code snippet on Listing 8.2, the first step of the logic is to configure the Google Strategy with client ID, client secret, callback url and a boolean value for activation of proxy. Subsequently the authentication procedure itself receives some parameters from the Google OAuth request send by the user, including their google Profile info, as seen in line 7. The algorithm then checks if an email registered on the received profile exist in the MongoDB user database (which would indicate that the given user is registered on the platform). If the user exist and has an Google ID attached as value to the user in the database the authentication procedure is finished and the existing user is returned (line 25). If the users Gmail exist in the database but a Google ID has yet to be registered, it indicates that the user is registered but it is the first time this user logs into the system. Here, a new User object is created and saved to the database (line 28-31) and the new user is returned. Finally, if the users email is not registered in the database the authentication procedure will return with an unauthorized error and abort login (line 36).

```
1   // Google OAuth Strategy 1: Login
2 passport.use('google-restricted',new GoogleStrategy({
3     clientID: keys.googleClientID,
4     clientSecret: keys.googleClientSecret,
    callbackURL: '/auth/google/callback',
    proxy: true
}, async (accessToken,refreshToken, profile, done) => {
    // Find user with email of the one clicking
    let existingUser;
    let index;

    for (i=0;i<profile.emails.length;i++) {
        const tempUser =
        await User.findOne({email: profile.emails[i].value});

        if (tempUser) {
            existingUser = tempUser;
            index = i;
        };
    }

    // If such a user exist
    if (existingUser) {
        // If that user ALREADY has a Google ID, finish with that user
        if (existingUser.googleID) {
            done(null,existingUser);
        } else { // Else remove user, add new with ID and email
            const rem =
            await User.remove({email: profile.emails[index].value});
            const user = await new User({
                googleID: profile.id,
                email: profile.emails[index].value,
            }).save();
            done(null,user); // Finish with NEW user
        }
    } else {
        // Only here, if user is NOT registered in Database
        done(null,false) // Return with "unauthorized error"
    }
}));
```

Listing 8.2: JavaScript logging in

This concludes the authentication implementation in the project. When a registered user logs in to the application through Google OAuth, the user is directed to the Educado Web home page. The details of both frontend and backend software for this page and underlying processes is covered in the next section.

8.3 Listing courses on home screen

With the user having successfully logged into the application, the goal is to present a home screen with a list of the different courses that specific user is managing. This is the second step of the user journey.

Authentication
Home
Courses
Sections
Components
Mobbile

Figure 8.3: User journey step 2: Home

Before getting any courses from the backend however the general layout of this page is rendered with a top bar and a navigation bar. Inside of the top bar is the option to log out as well as expand the navigation bar. The navigation bar features both a 'Home' and 'Create Course' option.

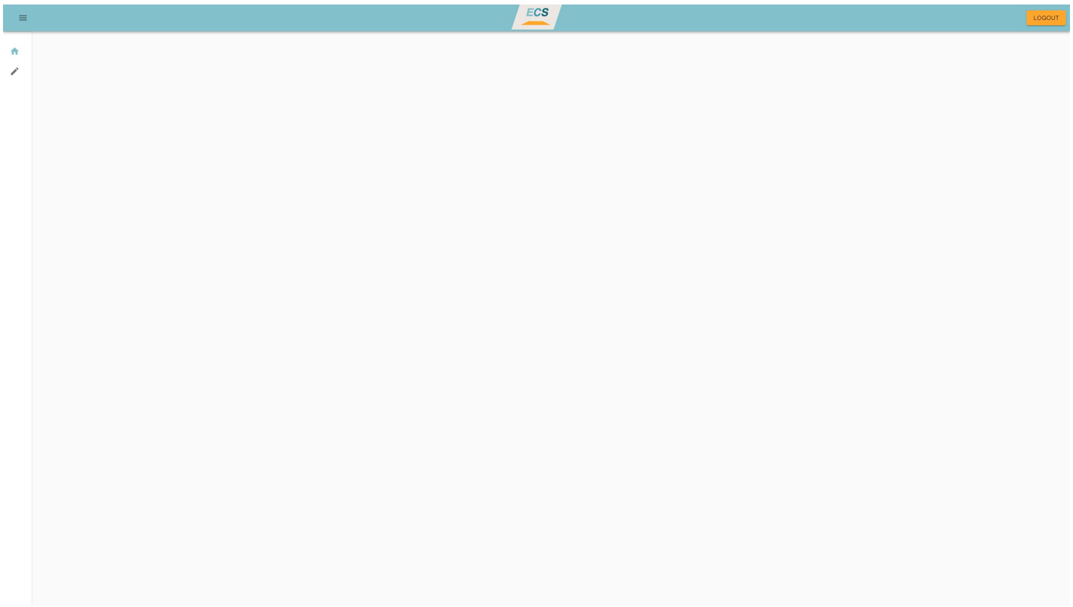


Figure 8.4: Default home screen

The top navigation bar is based on a Material-UI component [28]. Material-UI is a list of pre-made elements such as different buttons, top bars, etc. Many of the UI elements within the web application are made using Material-UI elements. One of the main features of these elements, beyond being quick to implement is the simple implementation of common styling. [29]

Whilst the top bar is relatively simple, with a fixed height and width set to match the screen width, the navigation bar is more complex as it can contract and expand. This alongside a good example of the styling can be seen here:

```
const Navbar = (props) => {
    // ...
    const {width} = useWindowDimensions();

    // Dynamic styles test
    const dynStyles = makeStyles((theme)=>({
        titleClosed: {
            position: 'absolute',
            width: `calc((${width}px) - ${drawerClosedWidth}px)`,
            textAlign: 'center',
            left: drawerClosedWidth,
        },
        titleOpen: {
            position: 'absolute',
            width: `calc(${width}px - ${drawerOpenWidth}px)`,
            textAlign: 'center',
            left: '0px',
        },

    }))
```

Listing 8.3: JavaScript Navbar component

Within this the navbar has two different sizes depending on if it has been opened or not. The drawerOpenWidth and drawerClosedWidth are simply static at 240px and 70px respectively. It should also be mentioned that sidebars are called 'drawers' within Material-UI, hence the terminology within the code. Both bars render using the following:

```
return (
    <div className={classes.root}>
        <CssBaseline />
        <AppBar className={clsx(classes.appBar,
        drawerState && classes.appBarShift)} position="absolute">
            <Toolbar className={classes.toolbar}>
                <RenderDrawerButton />
                <div className={classes.logoContainer}>
                    <img src="/ecs-logo.png"
                    className={classes.logo}></img>
                </div>
                <Button className={classes.barLogoutButton}
                variant="contained"
                href="/api/logout">Logout</Button>
            </Toolbar>
        </AppBar>

        <Drawer
        variant="permanent"
        open={drawerState}
        classes={{paper: clsx(classes.drawerClosed,drawerState
        && classes.drawerOpen)}}
        >
            <div className={classes.appBarSpacer} />
            <Divider />
            <MenuList click={handleDrawerClose}></MenuList>
        </Drawer>
```

Listing 8.4: JavaScript Top and Sidebar components

Firstly, the top bar is created using the app bar component from Material-UI, then the things to be rendered are done from left to right. Starting with the 'RenderDrawerButton' the button to expand or contract the sidebar is rendered, then the logo in the center and finally the logout button on the right. Secondly the sidebar or 'drawer' in the Material-UI connotation which then looks to see if it is in an open or closed state. The 'MenuList' is again a Material-UI element[30] that allows for a dynamic list of menu options on to witch things such as 'OnClickListeners' can be attached.

Having rendered the top and sidebars, the courses managed by the user of the application must be gathered. This is done via the 'componentDidMount' function which is done whenever the selected JavaScript component is rendered. In this specific case it is needed whenever the user accesses the home screen, therefore the following is called:

```
componentDidMount() {
    this.props.getAllCourses();
}
```

Listing 8.5: The componentDidMount function

As described within the state management of the technical analysis, a JavaScript component can dispatch an action if it wants to change the state of the application. Section 7.2 Seeing as the goal is to get all courses managed by the user and render them onto the homepage this constitute a state change. The action being dispatched being the 'this.props.getAllCourses' which as seen in the system tree

Section 8.1 is within a specific folder named 'actions' and since the action is to do with the courses it is within 'Course.js' and is as follows:

```
export const GET_ALL_COURSES = 'GET_ALL_COURSES';

export const getAllCourses = () => {

    return async (dispatch) => {
        const res = await axios.get('/api/course/getall');
        dispatch({type: GET_ALL_COURSES, payload: res.data})
    }
}
}
```

Listing 8.6: The get all courses action

Since the courses need to be gathered from the database, the action needs to call a function within the api, namely the '/api/course/getall' to actually receive the courses. This will in turn allow the courses to be gathered from the database and then finally the state can be updated trough the reducer. First, the api call is made:

```
app.get('/api/course/getall',requireLogin,async (req,res) => {
    const list = await Course.find({_user: req.user.id});
    res.send(list);
})
```

Listing 8.7: The get all courses api

So the application send a get call which requires the user to be logged in asynchronously as described in Section 8.2 wherein a list is made with the courses corresponding to the active user id and this list is then sent back to the application. This means the web app now has the users courses returned as a list as shown in the following model:

```
const courseSchema = new Schema({
    title: String,
    description: String,
    _user: {type: Schema.Types.ObjectId,ref: 'User'},
    dateCreated: Date,
    dateUpdated: Date,
    coverImg: String,
    sections: [{type: Schema.Types.ObjectId,ref: 'Component'}],
});
```

Listing 8.8: Course Schema

Effectively, the title, description, dateCreated, dateUpdated, cover image, and sections ids are now within the web app. However, the application is still not ready to change state, as recalled from the state technical analysis on redux, this is done via a reducer and not directly from an api call. Section 7.2 The reducer in of itself is very simple:

```
case GET_ALL_COURSES:
    return {
        ...state,
        userCourses: action.payload
    }
```

Listing 8.9: Course Schema

The reducer is done with a switch case, and only the exact case for 'GET ALL COURSES' is shown here. It takes the current state of the application and then sets the userCourses to be user courses returned from the api call. This means that Within the store, the state for the userCourses is updated, enabling the courses to be rendered on the home screen, which is done as follows:

```
{this.props.course.userCourses.map((course,index) => {
return (
  <div className={classes.root}>
    <Container className={classes.container} maxWidth="xl">
      <Grid container spacing={3}>
        {this.props.course.userCourses.map((course,index) => {
          return (
            <Grid id={index} item xs={12} md={6} lg={4} xl={3}
            key={index} >
              <Button className={classes.button}
              onClick={event => this.handleClick(event,course._id)} >
                <Paper data-value={course._id}
                data-sections={course.sections}
                className={classes.paper} >
                  <img className={classes.img}
                  src="/logo192.png" data-value={course._id} alt=""/>
                  <Divider></Divider>
                  <Typography variant='h5'
                  data-value={course._id}>{course.title}</Typography>
                </Paper>
              </Button>
            };
```

Listing 8.10: Courses Render on Home Screen

Firstly the entire list of courses is set inside of a Material-UI container [26]. The container is simply a space within the page where the courses can render. Then the list of course(s) is put within a React grid component, which dynamically adapts a grid to the screen size [27]. The grid item sizes are how many items that can be in a line with each other before a new line is made. The sized xs. md, lg, xl are determined earlier depending on the screen resolution via simple constants. Each course is in turn wrapped in an button to allow a user to click directly on each grid item. A logo is then added, however this is set as a static image, which will be updated to use the same function as described in Section 8.5.

Finally, below the course logo the title for the respective course id is written. Since the user is allowed more than one course, a loop must be made to go over the exact amount of courses sent from the get all courses api call.

With this state update complete the user now has access to their own courses if any based on their id gotten from the authentication process Section 8.2. Now the user can move on to creating the courses using the sidebar.

8.4 Creating new courses

The third step of the user journey is that a user intends to create a new digital learning course. When creating a course, the application is not limited to make 'read requests' to the REST API as seen on the home page but also take user input and make a HTTP POST request.

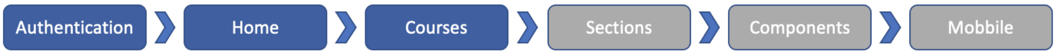


Figure 8.5: User journey step 3: courses

Initially, when accessing the create new course functionality through the navigation bar, the user is presented with the following view

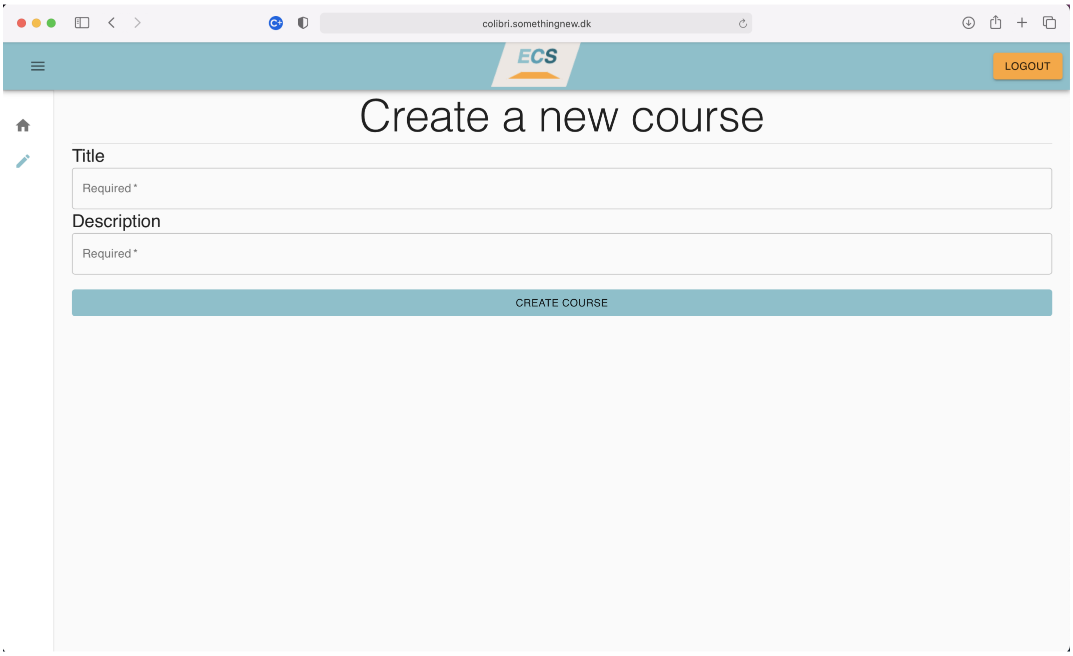


Figure 8.6: Create course screen with user input and button

After entering a title and a short description for the course, the user clicks on the 'create course' button which then activates a handler function. This function is responsible for accepting the user input saved in local state and activate a Redux action, as detailed in Section 7.2.2. This action function responsible for creating a new course, is shown below on Listing 8.11. The function starts by creating a new JSON object with the input parameters, and subsequently returns an asynchronous dispatch function with an Ajax call. The Ajax call (line 7), is an HTTP POST request with the course object attached in the body. The POST request indicates that the asynchronous call includes data that will be written to a database or other storage mechanism and is used by the REST API to determine actions based on incoming requests.

```
1      export const createCourse = (title,description) => {
2      const course = {
3          title: title,
4          description: description
5      }
6      return async (dispatch) => {
7          const res = await axios.post('/api/course/create',course);
8          dispatch({ type: CREATE_COURSE, payload: res.data});
9      }
10 };
```

Listing 8.11: Action function for creating a new course object

The request is then directed to the REST API at route path '/api/course/create' through a proxy middleware service. As seen on line 1 in Listing 8.12 the REST API listens for HTTP POST requests on the specified path. Before executing the asynchronous processing the 'requireLogin' method is executed to ensure that the request comes from an authenticated user (see authentication section). After authenticating the user a new course db object is created with the title and description from the request, the user id and the date registrations for created and updated dates. It also creates an empty array that will contain the sections belonging to the course, which is detailed in the next section. Finally, a try catch (see error chapter) is used to write the new course to the database and on success return the created course and on failure return error 422 and the error message.

```
1      app.post('/api/course/create',requireLogin,async (req,res) => {
2          const {title,description} = req.body;
3  
4          const course = new Course({
5              title: title,
6              description: description,
7              _user: req.user.id,
8              dateCreated: Date.now(),
9              dateUpdated: Date.now(),
10             sections: [],
11         })
12 
13         try {
14             await course.save();
15             res.send(course);
16         } catch (err) {
17             res.status(422).send(err);
18         }
19     });
```

Listing 8.12: Creating a new course object in via REST API and persist in MongoDB

Now a course can be created by a user with a title and description and saved to the database. Next in the user journey is to edit this created course by adding and manipulating nested sections.

8.5 Adding nested sections

Adding these sections is done similarly to the creation of courses, this time however only with a title. This means that the creation is very similar and while the REST api calls are called something else, they fulfill almost the same functionality as seen in Chapter 6. In turn this is done with an api call to: '/api/section/create' Working with these nested sections are the fourth part of the user journey.

Authentication
Home
Courses
Sections
Components
Mobbile

Figure 8.7: User journey step 4: sections

Whilst creating a course and section are very comparable, there is one notable difference. Due to the one to many relationship between a course and its many sections, the course must somehow be linked to its many sections. As seen in the course schema on Listing 8.9, one part of a course is an array of sections. This array features the ids of the sections linked to the course.

```
    sections: [{type: Schema.Types.ObjectId,ref: 'Component'}],
```

Listing 8.13: Sections array within course Schema

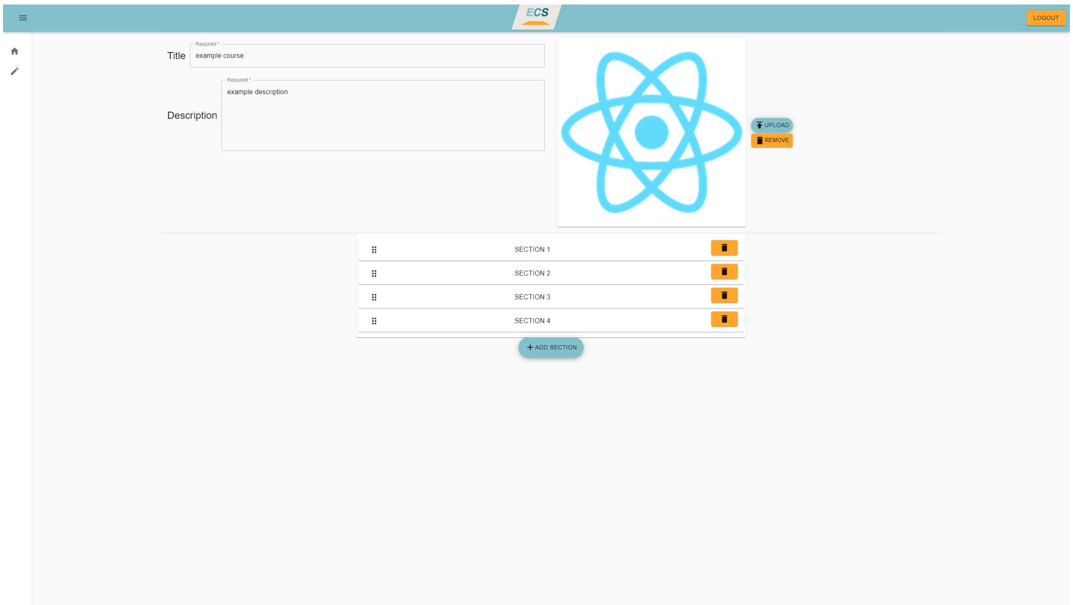
This in turn means that whenever a new section is created, the id created must also be sent to the course and into the array. To accomplish this, a variable for the activate course is used, the course which the sections are to be added to is known. Therefore it is a relatively simple task of taking the id returned from the initial api call and insert it into the 'sections' array.

By creating a section and inserting it into this array, the sections are then rendered on the page in that order using a list and a library called 'beautiful dnd':

```
<DragDropContext
    onDragEnd={this.onDragEnd}
>
    <SectionBucket sectionsList={this.props.course.courseSections} >
    </SectionBucket>
</DragDropContext>
```

Listing 8.14: Sections array within drag and drop

This means that each section is mapped on a list with drag and drop functionality, allowing the order of the sections to be changed simply by dragging and dropping them into new places. As described within the technical analysis on Section 7.2 this becomes very important in state management. Whilst drag and drop per default works on local state, the goal is to save the change in the global state, so that the changes are permanent. To do so, every time two sections changes places on the list, an api call to '/api/course/update/sectionsorder'. Due to the order of the sections being determined within the course schema's section array, the sections are then rendered in order of the array:



Title: example course

Description: example description

Figure 8.8: An example course with sections

The array is updated with the placements of the two sections within the array swapping. This can be seen below with a further video example here:
https://www.youtube.com/watch?v=y7W6dsS-0Ss [11].

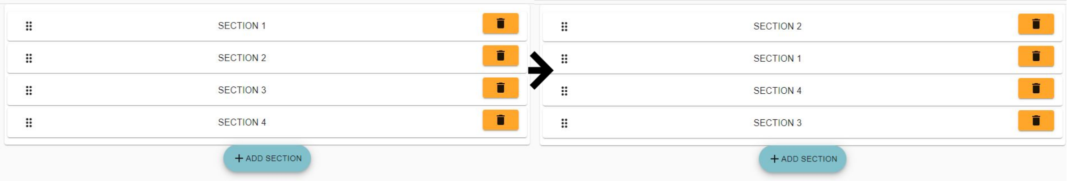


Figure 8.9: Sections before and after moving them

Just as on the home screen Section 8.3, each section is clickable to redirect onto that sections edit page. In accordance with the single page application principle as described in Section 7.2, the edit section page is able to edit all sections, but requires an active one to be set. This is done whilst clicking on said section trough this simple click handler:

```
const handleClickSection = async (event,section_id) => {
    setRedirectId(section_id);
    await props.editSection(section_id);
}
```

Listing 8.15: Active section handler

With the sections in order, a final note on the edit course page is the ability to upload a cover image. This cover image can be seen in Figure 8.8, using the react logo. However, whilst it was static in Section 8.3, it is now dynamic and the user is able to upload a cover image and have it update on the page. This is important due to it being the first time api calls are not just made to the database, but also the file storage server. These file server calls are similar to the ones needed for each non-text component, and are discussed within the next section.

8.6 Editing components in sections

Having made the sections, they are now editable and as the active course page shows the sections, so does the active section page allow editing of components. This is the fifth stager of the user journey and the final one within the web editor.

Authentication
Home
Courses
Sections
Components
Mobbile

Figure 8.10: User journey step 5: Edit components

As the section order is saved within the course schema as shown in Listing 8.9, so is the components saved within the active sections schema. Furthermore the drag and drop wrapping that updates the order and the list displaying it, is the same as in the section order, with minor tweaks to styling. This results in a new page, dedicated to adding / deleting and updating the component types described in Section 7.2, namely text, image audio and video. A sample of the frontend with each type is shown here:

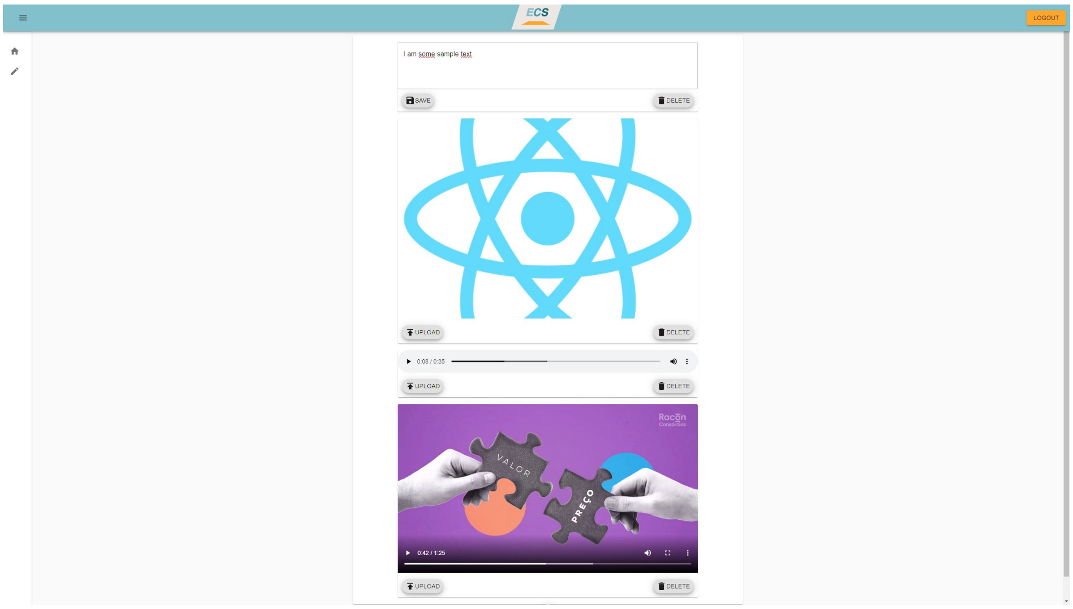

I am some sample text

Figure 8.11: Edit Section screen

Whilst the drag and drop methodology is the same, the way in which different files are handled are not. As mentioned in Section 7.3, when working with image, audio and video it cannot be stored directly in the database and the file server is used instead. As described in section Section 7.2, this constitute a state change, and therefore the redux cycle described in Section 7.2.2 and exemplified in Section 8.3 is used. To facilitate these differences, when creating a component the action being dispatched is the following:

```
export const CREATE_COMPONENT = 'CREATE_COMPONENT';

export const createComponent = (type, section_id) => {
    const obj = {
        type: type,
        section_id: section_id
    }
    return async (dispatch) => {
        const res = await axios.post('/api/component/create',obj);
        return dispatch({ type: CREATE_COMPONENT, payload: res.data});
    }
}
```

Listing 8.16: Create Component action

Whilst the component id is stored within the active section, just as the section was stored within the active course, a type is also set which dictates the type of component and in turn what type of content it is to use. This type is manually set by the user via a popup when creating a new component as seen here:

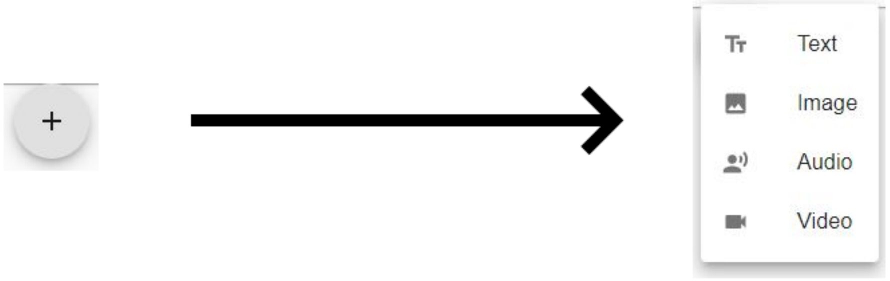


Figure 8.12: The add component input before and after pressing the +

This type creation in the backend is simply just a different event handler depending on where the user clicks:

```
const handleCreateText = async (event) => {
  await props.createComponent("TEXT", props.course.activeSection._id);
  props.trigger();
}

const handleCreateImage = async (event) => {
  await props.createComponent("IMAGE", props.course.activeSection._id);
  props.trigger();
}...
```

Listing 8.17: Create different types of component

Having defined the type, whenever the state is updated and components are changed by uploading an image for example, the application then checks each component in a map function to see which type they are and in turn how it is to be rendered:

```
<List innerRef={provided.innerRef} {...provided.droppableProps}
type="dense">
{props.componentsList.map((component,index) => {
    let componentToRender;
    switch (component.type) {
        case "TEXT":
            componentToRender = <TextComponent trigger={props.trigger}
            id={component._id}></TextComponent>;
            break;

        case "IMAGE":
            componentToRender = <ImageComponent trigger={props.trigger}
            id={component._id}></ImageComponent>;
            break;
            etc..
```

Listing 8.18: JavaScript Navbar component

Effectively depending on the type of component it will be rendered in different ways, giving audio an audio playback option, videos a video player etc. These different components are all made into a components folder similar to the navbar, but specifically for the different components as seen in the file tree Section 8.1.

Additionally, as described earlier, different types of components are managed differently. Text input is simple, as it works the same as the title of a section or course and is stored to and gotten directly from the database, having a similar cycle as described in Section 8.3. However, for image / audio / video it is different as the files must be stored on the file server.

When contacting the file server, the system uses for example '/upload-s3' as shown here:

```
app.post('/upload-s3', mulreq.single('file'),
requireLogin, async (req, res) => {
  aws.config.update({
  region: 'eu-central-1'
  })

  if (!req.file){
    return res.status(500).send({msg: "File not found"})
  }
  const myFile = req.file;

  const {component_id} = req.query;

  const s3 = new aws.S3();

  const params = {
      Bucket: keys.s3Bucket,
      limit: 100000000,
      Key: Date.now() + '-' + req.file.originalname,
      Body: myFile.buffer
  };

  s3.upload(params, async (err, result) => {
      if(err) {
        console.log("Error", err);
      } else {
        (await Component.findOneAndUpdate({_id: component_id},
        {file: stored.Key})).save;
        res.send({link: stored.key});
      }
  });
});
```

Listing 8.19: S3 upload function

Within this function a file as selected by the user when pressing upload is sent trough, alongside the login data from the application. The S3 bucket, uses specific amazon keys [8] that the Heroku server has stored, meaning only uploads from the Heroku server (The web application from the users point of view) is allowed to upload the bucket.

The file is then sent to the S3 server using parameters specific to the S3 api, such as file size limits and the bucket, which is stored in a key folder [6]. A Key for the file is generated to be unique, which is done by appending the current time and date to the front and the file is put into the buffer 'myFile.buffer'.

Finally, the file is uploaded to the s3 server and they key is updated to the component, so it can be called from the S3 bucket. This process is the same for image, audio and video, however when retrieving it, there is a key difference.

For images, they can simply be retrieved from the key and then displayed on the page, however for videos of up to 100mb as required in Chapter 6, this would lead to extremely slow load times, therefore presigned-urls are used. Presigned-urls are a simple way of giving authorized access via an URL to the component.Section 7.3.2 This means for both audio and video, the playback methods simply have to playback from an URL. This can be seen in the audio component for example:

```
return (
      <Card className={classes.root}>
        <audio controls width="720" key={this.state.presignedUrl}
        className={classes.media}>
          <source type="audio/mp3" src={this.state.presignedUrl} />
      </audio>
```

Listing 8.20: Audio component sourcing from presigned url

The presigned-url is gotten via a simple call to the s3 server with the key for the file as shown here:

```
app.post('/api/eml/get-presigned-url-file', async (req, res) => {
  aws.config.update({
  region: 'eu-central-1'
  })

  const {link} = req.body;

  const s3 = new aws.S3();

  const params = {
      Bucket: keys.s3Bucket,
      Key: link,
      Expires: 600
  };

  const signedUrl = s3.getSignedUrl('getObject',params);

  res.send(signedUrl);
});
};
```

Listing 8.21: S3 presigned-url function

the 'link' is simply the file placement within the s3 server previously referred to as the key in Section 8.3, but due to it creating an URL it is referred to as a link here. The S3 parameters are made in accordance with the AWS api [7] and simply returns the signed URL which expiries after 10 minutes. This means if a page is left open for 10 minutes with video playback it would have to be refreshed in order to get a new presigned-url.

Trough this mix of database and S3 connections all components can be managed in order to fulfill the requirements presented in the requirements specification Section 6.3, and only the mobile application remains to display the courses.

8.7 Presenting a course in Educado Mobile

Finally, the digital course must be presented in the android app Educado Mobile, as the last step on the user journey.

Authentication
Home
Courses
Sections
Components
Mobbile

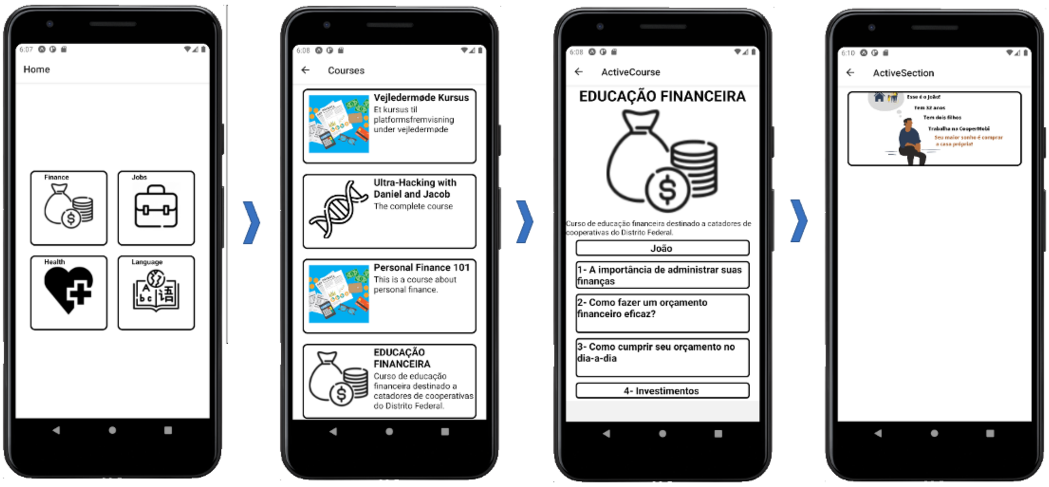
Figure 8.13: User journey step 6: mobile

The mobile application developed in this project only contains the core functionality to read courses and has not focused on styling of elements. The structure of the application is kept as simple as possible and consist of just four screens, that can be accessed through a stack navigator as seen below on Listing 8.22.

```
  const AppNavigator = createStackNavigator(
 {
  Home: HomeScreen,
  Courses: CoursesScreen,
  ActiveCourse: ActiveCourseScreen,
  ActiveSection: ActiveSectionScreen,
 },
 {
  initialRouteName: 'Home',
 }
);
```

Listing 8.22: Mobile app stack navigator

A user is able to navigate progressively through the application, gradually breaking down the content from course, to section and finally to component elements, as seen on Figure 8.14.



Vejledermøde Kursus
Et kursus til platformsfremvisning under vejledermøde

Ultra-Hacking with Daniel and Jacob
The complete course

Personal Finance 101
This is a course about personal finance.

EDUCAÇÃO FINANCEIRA
Curso de educação financeira destinado a catadores de cooperativas do Distrito Federal.

EDUCAÇÃO FINANCEIRA

Curso de educação financeira destinado a catadores de cooperativas do Distrito Federal.

João

1- A importância de administrar suas finanças

2- Como fazer um orçamento financeiro eficaz?

3- Como cumprir seu orçamento no dia-a-dia

4- Investimentos

Figure 8.14: Showcase of the four mobile application views

The application utilizes the same React and Ajax structures as documented in Section 7.2.2 and Section 7.1.2 but does not make use of Redux, as the added complexity of global state management is deemed unnecessary. As an example, consider the code for Listing 8.23, where an asynchronous Ajax call is made through the handler function in the componentDidMount lifecycle method (line 6). When components are returned the local state is updated with an array containing all components. A simple if based logic is then used to determine the type of the components, again similar to the logic used when creating the components in the web application (see Section 8.6), as seen on lines 14-25. The component itself then simply returns the list of content after type specification in a scrollable view (lines 33-35).

```
 1 class ActiveSection extends Component {
 2 state = {
 3   components: []
 4 };
 5 async componentDidMount() {
 6   const componentsTemp =
 7   await...
 8   getComponents(this.props.navigation.state.params.section.components);
 9   this.setState({
10     ...this.state,
11     components: componentsTemp,
12   });
13 };
14 render() {
15   let ListContent;
16   if (this.state.components !== []) {
17       ListContent = this.state.components.map((component,index) => {
18       switch (component.type) {
19         case 'TEXT':
20           return(<TextElement key={index}
21           comp={component}></TextElement>);
22         case 'VIDEO':
23           return(<VideoElement key={index}
24           comp={component}></VideoElement>);
25         case 'IMAGE':
26           return(<ImageElement key={index}
27           comp={component}></ImageElement>);
28         default:
29           break;
30       };
31       });
32     } else {
33       ListContent = <Text>Waiting...</Text>;
34     };
```

```
    return (
        <View style={styles.container}>

            <ScrollView>
                {ListContent}
            </ScrollView>
        </View>
          );
  };
};
```

Listing 8.23: Rendering of components in moible application

This concludes the design and implementation chapter which leads into testing.

Chapter 9

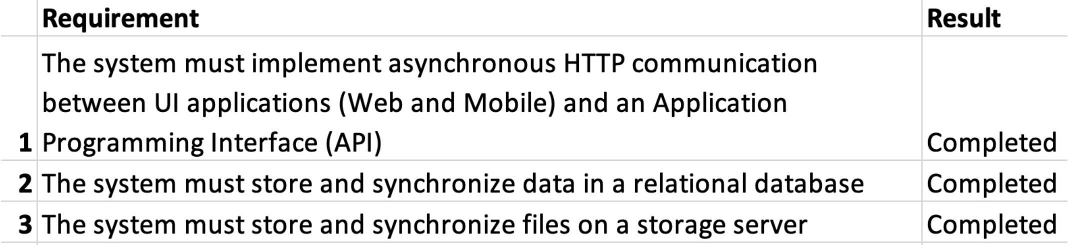
Testing

The test methodology applied in this project is conducted as an acceptance test [48] of the requirements specified in Section 6.3. The aim of the project has been to develop a distributed web system that functions as a fundament for a digital learning platform, that both enables the creation of digitized learning material and presentation of content on a mobile application.

As also described in the requirement specification, the individual requirements for the system is divided into four categories due to the complexity and functional dissimilarity between system services. The four categories are: (i) General, (ii) API and storage systems, (iii) Web frontend and (iv) Mobile frontend. Evaluating wether a requirement is accepted/rejected is done based on the documentation in Chapter 7 and Chapter 8, which means that each requirement must be connected to a reference, so that its validity can be inspected. As described in Chapter 6, the ambitions of this project is to create a system that constitutes a functional basis and enables integration of additional components/functionality. The following sections present the requirements for each of the four groupings and lists their evaluation and references.

9.1 General

The results from acceptance testing for the requirements to the general system are seen below on Figure 9.1.



	Requirement	Result
1	The system must implement asynchronous HTTP communication between UI applications (Web and Mobile) and an Application Programming Interface (API)	Completed
2	The system must store and synchronize data in a relational database	Completed
3	The system must store and synchronize files on a storage server	Completed

Figure 9.1: Interface model

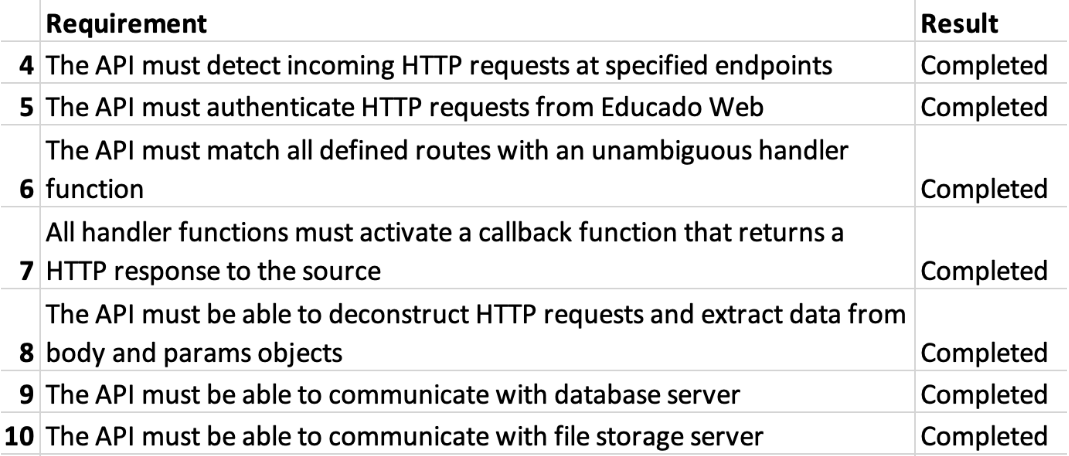
Req. 1 As illustrated on the system architecture Figure 7.2 is based on microservices (see Section 7.1.1) and implements asynchronous HTTP communication between individual services. An example of asynchronous communication between the REST API and the web frontend services can be seen in Section 8.3.

Req. 2 As described in Section 7.3 the system implements a NoSQL based MongoDB database to store and synchronize relational data. See Listing 8.9 to see structure of a relational database component.

Req. 3 As described in Section 7.3 the system implements an AWS s3 file storage server to store and synchronize file objects such as video, audio and images. See Listing 8.19 for an example of the upload functionality to the s3 file server.

9.2 API and storage systems

The results from acceptance testing for the requirements to the API and storage systems are seen below on Figure 9.2.



	Requirement	Result
4	The API must detect incoming HTTP requests at specified endpoints	Completed
5	The API must authenticate HTTP requests from Educado Web	Completed
6	The API must match all defined routes with an unambiguous handler function	Completed
7	All handler functions must activate a callback function that returns a HTTP response to the source	Completed
8	The API must be able to deconstruct HTTP requests and extract data from body and params objects	Completed
9	The API must be able to communicate with database server	Completed
10	The API must be able to communicate with file storage server	Completed

Figure 9.2: Interface model

Req. 4 As detailed in Section 8.4 and exemplified on Listing 8.12 the REST API is able to detect incoming HTTP requests at a specified endpoint such as '/api/-course/create', for the creation of a new course object.

Req. 5 Section 8.2 details how authentication is implemented in the system through the Google OAuth framework. See Listing 8.2 for the specific logic that handles incoming login requests and ensures that users of the web application are authenticated. Note also that all subsequent HTTP requests coming from the web application and detected at endpoints at the API are forced to loop through a 'requireLogin()' function, as for example seen on Listing 8.12.

Req 6. As detailed in the system overview of the REST API in Section 8.1.1, the '/routes' directory contains all handler functions to specified endpoints. An example of such function can be seen on Listing 8.12.

Req 7. To ensure reliability and increase fault tolerance, the JavaScript try/catch statements (see Section 7.1.3) are implemented in the HTTP callback functions. See lines 13-18 on Listing 8.12 for an example of this algorithmic structure which is reused on all HTTP responses and hereby satisfy the requirement.

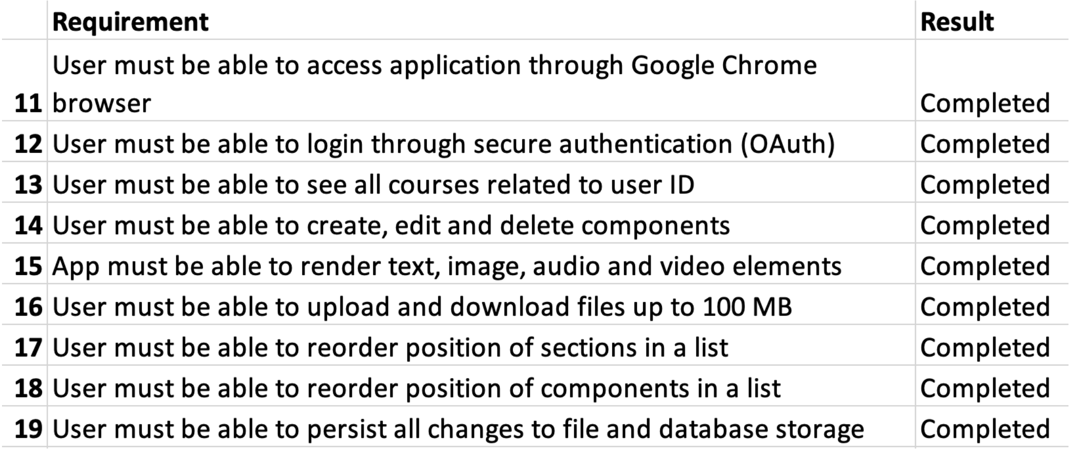
Req 8. Implemented as per example in lines 1-2 in Listing 8.12 where parameters 'title' and 'description' are extracted from the HTTP post request.

Req. 9 Communication with the MongoDB database is described in Section 7.3. An example of communication with the database from the REST API is the request endpoint at 'api/course/getall', which returns all courses for a given user (see Listing 8.7). In line 2 in the code snippet, the API makes an asynchronous (indicated by the JS 'async/await' syntax) call to the MongoDB by activating the find() function from the created Course database scheme (see Listing 8.9). The function returns a list of courses attached to the user ID which is then returned to the user.

Req. 10 Communication with the AWS s3 file storage server is described in Section 7.3. For an example of the API logic for uploading files, see Listing 8.19, where a file object is uploaded to the server and the upload path written to the proper component object - and then finally returned as a pre-signed URL to the user.

9.3 Web frontend

The results from acceptance testing for the requirements to the web frontend system is seen below on Figure 9.3.



	Requirement	Result
11	User must be able to access application through Google Chrome browser	Completed
12	User must be able to login through secure authentication (OAuth)	Completed
13	User must be able to see all courses related to user ID	Completed
14	User must be able to create, edit and delete components	Completed
15	App must be able to render text, image, audio and video elements	Completed
16	User must be able to upload and download files up to 100 MB	Completed
17	User must be able to reorder position of sections in a list	Completed
18	User must be able to reorder position of components in a list	Completed
19	User must be able to persist all changes to file and database storage	Completed

Figure 9.3: Interface model

Req. 11 The application is tested in Safari, Google Chrome and Microsoft Edge browsers and no faulty behavior experienced. See illustrations throughout Chapter 8 for images of application running in browser.

Req. 12 The user is able to login through Google OAuth as documented in Section 8.2.

Req. 13 The user is able to see all courses related to user ID as described in Section 8.3.

Req. 14 User is able to create, edit and delete all component types (Text, Audio, Image and Video) as described in Section 8.6.

Req. 15 Application is able to render all component types (Text, Audio, Image and Video) as shown on Figure 8.12 and described in Section 8.6.

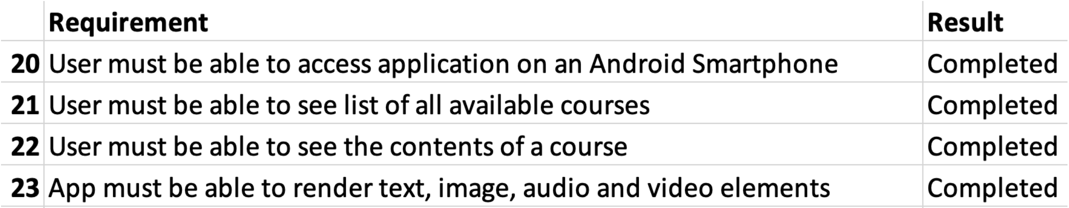
Req. 16 As shown on Listing 8.19 on line 17, the upload function to the AWS s3 file server is limited to 100MB in the params config object.

Req. 17 and req. 18 As described in Section 8.5, sections and components can be reordered via the same programmatic patterns.

Req. 19 Through various saving mechanisms (depending on nested level) user changes to course objects are persisted in the MongoDB and AWS s3 file storage server as detailed in Section 7.3.

9.4 Mobile frontend

The results from acceptance testing for the requirements to the mobile frontend system is seen below on Figure 9.4.



	Requirement	Result
20	User must be able to access application on an Android Smartphone	Completed
21	User must be able to see list of all available courses	Completed
22	User must be able to see the contents of a course	Completed
23	App must be able to render text, image, audio and video elements	Completed

Figure 9.4: Interface model

Req. 20 As illustrated on Figure 8.14, the application is accessible in the Android Studio Mobile emulator. This is regarded as satisfactory for the requirement and initial testing on physical devices through the Android Play Store indicate successful rendering as well.

Req. 21 As illustrated on the second screen on Figure 8.14, the application screen 'Courses' is able to list all courses.

Req. 22 As illustrated on screen three and four on Figure 8.14, the application is able to render both course related and section related contents. Listing 8.23 shows the programmatic logic for rendering components based on their listed type.

Req. 23 As seen on Listing 8.23, the application is able to differentiate between component types and render them accordingly.

9.5 Subsidiary conclusion

As described in the sections above, all 23 requirements established in Chapter 6 are completed and accepted. This indicates the that the project has achieved what it set out to achieve and can be regarded as successful in that matter.

Chapter 10

Conclusion

The initial vision for this project was to increase opportunities of digital learning for individuals in lower socioeconomic communities. In extension of previous research[12][41], the situation of Brazilian waste pickers transitioning into jobs at local sorting facilities are examined through both field and desk research, as documented in Chapter 2. From analyzing the results, it was clear that this group of people face impactful financial issues and that there is a lack of digital education in line with their learning requirements. This leads to the project vision of creating a digital learning platform that allows local stakeholders to create user-friendly educational content for mobile devices and the formulation of the following problem statement:

> How can a digital learning solution be built to satisfy the learning needs of individuals employed at the Brazilian sorting facilities?

Subsequently to presenting the methodological basis of the project in Chapter 4 the system is conceptualized in Chapter 5. Based on descriptions, wireframes and system models, 23 mainly functional requirements are presented in Chapter 6. The requirements clearly indicates a focus on functionality over optimization which reflects in the remainder of the report and developing. Additionally, this functional focus leads to a demarcation of technologies used in the project with aim to make decisions that eases development and maintenance costs, while providing the necessary flexibility to extend the product into a large distributed system with user-friendly interfaces. One of the focal points here, is the decision to keep the entire programmatic base of the project within JavaScript - both frontend, backend and middleware systems, as detailed in Section 6.4.

In Chapter 7 the technical approach to developing a system of distributed web components is discussed. This includes microservices architecture, issues of concurrency, time and synchronization, complexity of Single Page applications and dynamic data management, as well as solutions for file storage and relational database structures. Based on these considerations, Chapter 8 documents the design and implementation of the system consisting of: (i) a HTTP REST API, (i) a MongoDB NoSQL Database, (iii) an AWS File storage server, (iv) a React based web application and (v) a React Native based mobile application. Combined these individual distributed services create a system where users are able to login through the web application, create learning material from scratch (supporting text, audio, image and video) and finally for users to inspect the courses on an Android smartphone.

As described in the testing chapter (Chapter 9), an acceptance test of the requirements specified in Section 6.3, shows that all 23 requirements have been achieved. Though this is definitely an indication of the projects technical success, the scope and limitations as described in Section 6.1 means that the product is still not ready to be launched on the App Store and reach the target group in production. Hereby, to fully satisfy the ambitions set by the problem statement to built a system that satisfies the learning needs of individuals employed at the Brazilian sorting facilities, further development is needed. This includes many challenges - such as increasing usability, integrating workplaces, making high quality content, increasing stability and adding functionality - which is discussed in the following reflection.

Chapter 11

Reflection

The result from this project is a version 1.0 prototype of a system capable of creating simple digital learning material via a web application and present it on Android phones. But, the road to a production ready system is still far and sees many crucial challenges and considerations, such as:

- Financial viability of the solution
    - How can the product me monetized? (Funding, ads, subscriptions etc.)
- Development of additional functionality
    - Styling of mobile application
    - Progression tracking through courses
    - Certification system to acknowledge completion of courses
    - Smart caching and download function
- Development of high quality learning material adhering to the 'locals teaching locals' philosophy
- Production deployment setup
    - Should system architecture be revised to fit production needs?
    - Security in the applications?
    - Optimized API requests?

These are just a few of the concerns necessary to ready the product for production launch which indicates that much work is still to be done to actually help elevate digital learning opportunities in low socioeconomic groups. Therefore, it is decided that the product ownership of the developed system is directly transferred to an engineering startup company founded by the authors of this report.

The idea is for the company to address the issues related to readying the product for deployment to production by initially seeking funding from various NGO's and funds to cover working hours spend. In collaboration with our project partners at UnB a general overview of the product planning is made, which is illustrated on Figure 11.1.

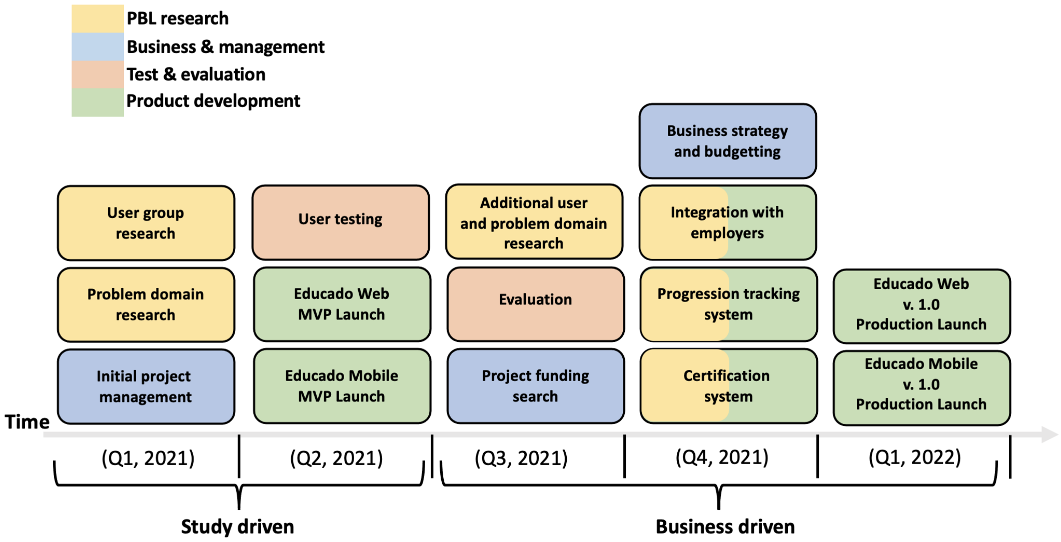


Figure 11.1: Model indicating the continual prospects for the Educado platform

The ambition with this plan and the transferring of product ownership is to avoid this becoming another student project which progress is restarted next semester. Instead, by following this model, new student projects can be proposed in the fall semester 2021 and help contribute to extend the development of the platform. Student groups from various engineering degrees but also from human and social sciences can be a crucial part of building the production ready system. Some examples of possible student projects are:

- How should be UX/UI styling of the mobile application be?
- How to implement the progression tracking system in a motivating way for the target group?
- How can the smart caching system be optimized and integrated?
- ...

The strongest way to solve these issues is to have people from different backgrounds and academic fields collaborate and cultivate each others strength, and hereby cross-discipline and collaborations will continue to be a priority for the project. Additionally, the methodology used in this project is largely expected to be applicable in the context of future development too - both in terms of collaboration and PBL, but also in terms of the cyclical 'idea, prototype, materialization' used in development.

Bibliography

[1] Kahn Academy. *Kahn Acedemy*. https://www.khanacademy.org/. [Visited month. day, year]. 13. 05. 2021.

[2] Kahn Acedemy. *Downloads*. https://www.khanacademy.org/downloads. [Visited month. day, year]. 13. 05. 2021.

[3] Altova. *Database Comparison Tool*. https://bit.ly/34ioufr. [Visited 26-05-2021]. 2021.

[4] Arsenault and Cody. *The Pros and Cons of 8 Popular Databases - KeyCDN*. https://www.keycdn.com/blog/popular-databases. [Visited 26-05-2021]. 2021.

[5] Articulate. *rise*. http://rise.articulate.com/. Visited 12/04-21. 2021.

[6] AWS. *Amazon S3 objects overview*. https://docs.aws.amazon.com/AmazonS3/latest/userguide/UsingObjects.html. [Visited 26-05-2021]. 2021.

[7] AWS. *Amazon S3 REST API Introduction*. https://docs.aws.amazon.com/AmazonS3/latest/API/Welcome.html. [Visited 26-05-2021]. 2021.

[8] AWS. *Managing access keys for IAM users*. https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_access-keys.html. [Visited month. day, year]. 2021.

[9] AWS. *Sharing an object with a presigned URL*. https://docs.aws.amazon.com/AmazonS3/latest/userguide/ShareObjectPreSignedURL.html. [Visited 26-05-2021]. 2021.

[10] Armin Balalaie, Abbas Heydarnoori, and Pooyan Jamshidi. *Microservices Architecture Enables DevOps: An Experience Report on Migration to a Cloud-Native Architecture*. Proceedings of the 1st International Workshop on Cloud Adoption and Migration, September 2015. -. 2015.

[11] Daniel Britze. *Educado Creator Studio Walktrough*. https://www.youtube.com/watch?v=y7W6dsS-0Ss. [Visited 26-05-2021]. 2021.

[12] Daniel Britze and Robert Nedergaard Nielsen. *Mobile Education Platform*. AAU Student Report. 2019.

[13] Tim Brown. *Design Thinking*. http://www.brizabueno.com/wp-content/uploads/2011/07/HBR-on-Design-Thinking.pdf. Visited 12/04/2021. 2008.

[14] Peter Checkland and Jim Scholes. *Peter Checkland*. https://en.wikipedia.org/wiki/Peter_Checkland. 1990.

[15] Kirupa Chinnathambi. *Creating a Single-Page App in React using React Router*. https://www.kirupa.com/react/creating_single_page_app_react_using_react_router.htm. [Visited 26-05-2021]. 2021.

[16] Codecademy. *What is CRUD?* https://www.codecademy.com/articles/what-is-crud. [Visited 26-05-2021]. 2021.

[17] Interaction Design Foundation. *What is Design Thinking?* https://www.interaction-design.org/literature/topics/design-thinking. Visited 12/04-21. -.

[18] G2. *Amazon Simple Storage Service (S3) Alternatives and Competitors*. https://www.g2.com/products/amazon-simple-storage-service-s3/competitors/alternatives. [Visited 26-05-2021]. 2021.

[19] Jesse James Garrett. *Ajax: A New Approach to Web Applications*. https://www.scriptol.fr/ajax/ajax_adaptive_path.pdf. Visited 12/04-21. -.

[20] Nielsen Norman Group. -. https://www.nngroup.com. Visited 12/04-21. -.

[21] Jared Hanson. *Passport*. http://www.passportjs.org. Visited 12/04-21. 2021.

[22] Jesper Kildebogaard. *Her er rapporterne, der dumpede Politiets skandaleramte Polsag-system*. https://www.version2.dk/artikel/her-er-rapporterne-der-dumpede-politiets-skandaleramte-polsag-system-49997. [Visited 23-05-2021]. 2013.

[23] Craig Larman. *APPLYING UML AND PATTERNS: An introduction to Object-Oriented Analysis and Design and the Unified Process*. Ed. by. -, 1997.

[24] learningequality.org. *KOLIBRI - AN ADAPTABLE PRODUCT ECOSYSTEM FOR OFFLINE-FIRST TEACHING AND LEARNING*. https://learningequality.org/kolibri/. [Visited month. day, year]. 13. 05. 2021.

[25] Cathy Li and Farah Lalani. *The COVID-19 pandemic has changed education forever. This is how*. https://www.weforum.org/agenda/2020/04/coronavirus-education-global-covid19-online-digital-learning/. [Visited 26-05-2021]. 2021.

[26] Material-UI. *Container*. https://material-ui.com/components/container/. [Visited 26-05-2021]. 2021.

[27] Material-UI. *Grid*. https://material-ui.com/components/grid/. [Visited 26-05-2021]. 2021.

[28] Material-UI. *MATERIAL-UI*. https://material-ui.com/. [Visited 26-05-2021]. 2021.

[29] Material-UI. *material-ui/styles*. https://material-ui.com/styles/basics/. [Visited 26-05-2021]. 2021.

[30] Material-UI. *MenuList API*. https://material-ui.com/api/menu-list/. [Visited 26-05-2021]. 2021.

[31] MongoDB. *What is NoSQL?* https://www.mongodb.com/nosql-explained. [Visited 26-05-2021]. 2021.

[32] Moodle. *What is Moodle™?* https://bit.ly/3bogIEI. [Visited month. day, year]. 13. 05. 2021.

[33] Thomas Mulvad. *EduCatador (Early Access)*. https://play.google.com/store/apps/details?id=com.aau.educatador&fbclid=IwAR1J5AdfMzWagUv1kW9fG70ZMldpvS2vYyMX
[Visited 26-05-2021]. 2021.

[34] Name. *A Migration Overview: Moving from Heroku to AWS*. https://blog.grio.com/2020/03/a-migration-overview-moving-from-heroku-to-aws.html. [Visited 26-05-2021]. 2021.

[35] Dmitry Namiot and Manfred Sneps-Sneppe. *On Micro-services Architecture*. International Journal of Open Information Technologies ISSN: 2307-8162 vol. 2, no. 9, 2014. -. 2014.

[36] Daniel Britze Jacob Vejlin Jensen Mikkel Steen Hansen Robert Nedergaard Nielsen. *Internet of Things Smart Home*. AAU Student Report. 2018.

[37] Oauth. *OAuth 2.0*. https://oauth.net/2/. Visited 12/04-21. 2021.

[38] Rasmus L. Olsen. "Struktureret system udvikling". In: (2018), pp. 1–50.

[39] James O'Malley. *Stop acting surprised that refugees have smartphones*. https://www.independent.co.uk/voices/comment/surprised-syrian-refugees-have-smartphones-well-sorry-break-you-you-re-idiot-10489719.html. [Visited 26-05-2021]. 2015.

[40] Jenny Preece, Helen Sharp, and Yvonne Rogers. *Interaction Design: beyond human-computer interaction*. Ed. by. Wiley, 2015.

[41] Ina Vedige Brøchner Rasmussen, Anja Bang Mejer, and Emma Bredahl Mortensen. *Mobile education Platform: Finance management for waste pickers*. AAU Student Report. 2020.

[42] React. *Context*. https://reactjs.org/docs/context.html. [Visited 26-05-2021]. 2021.

[43] Redux. *Getting Started with Redux*. https://redux.js.org/introduction/getting-started. [Visited 26-05-2021]. 2021.

[44] Nick Scialli. *Avoiding Race Conditions when Fetching Data with React Hooks*. `https://medium.com/hackernoon/avoiding-race-conditions-when-fetching-data-with-react-hooks-220d6fd0f663`. [Visited 26-05-2021]. 2021.

[45] Green Shoots. *GREEN SHOOTS*. `https://www.greenshootsedu.co.za/`. [Visited 26-05-2021]. 2021.

[46] Laura Silver. *Smartphone Ownership Is Growing Rapidly Around the World, but Not Always Equally*. `https://pewrsr.ch/3ocyikl`. [Visited month. day, year]. 5. Februrary. 2019.

[47] Eric Sink. *Requirements*. `https://ericsink.com/articles/Requirements.html`. [Visited 23-05-2021]. 2007.

[48] softwaretestingfundamentals.com. *Acceptance testing*. `https://softwaretestingfundamentals.com/acceptance-testing/`. Visited 12/04-21. -.

[49] Facebook Open Source. *Create React App*. `https://create-react-app.dev`. Visited 12/04-21. 2021.

[50] Maarten van Steen and Andrew S. Tanenbaum. *Distributed Systems (Third edition)*. Ed. by. Maarten van Steen, 2020.

[51] Digital Surgeons. *How Design Thinking Helped Create the Light Bulb and Tesla*. `https://www.digitalsurgeons.com/thoughts/design-thinking/how-design-thinking-helped-create-the-light-bulb-and-tesla/`. [Visited 26-05-2021]. 2017.

[52] Techterms. *Race Condition*. `https://techterms.com/definition/race_condition`. [Visited 26-05-2021]. 2021.

[53] Udemy. -. `https://www.udemy.com/`. [Visited month. day, year]. 13. 05. 2021.

[54] Aalborg University. *THE AALBORG MODEL FOR PROBLEM BASED LEARNING*. `https://www.en.aau.dk/about-aau/aalborg-model-problem-based-learning/`. [Visited 26-05-2021]. 2021.

[55] Unrefugees.org. *Syria Refugee Crisis Explained*. `https://bit.ly/3vn4fJj`. [Visited 26-05-2021]. 2021.

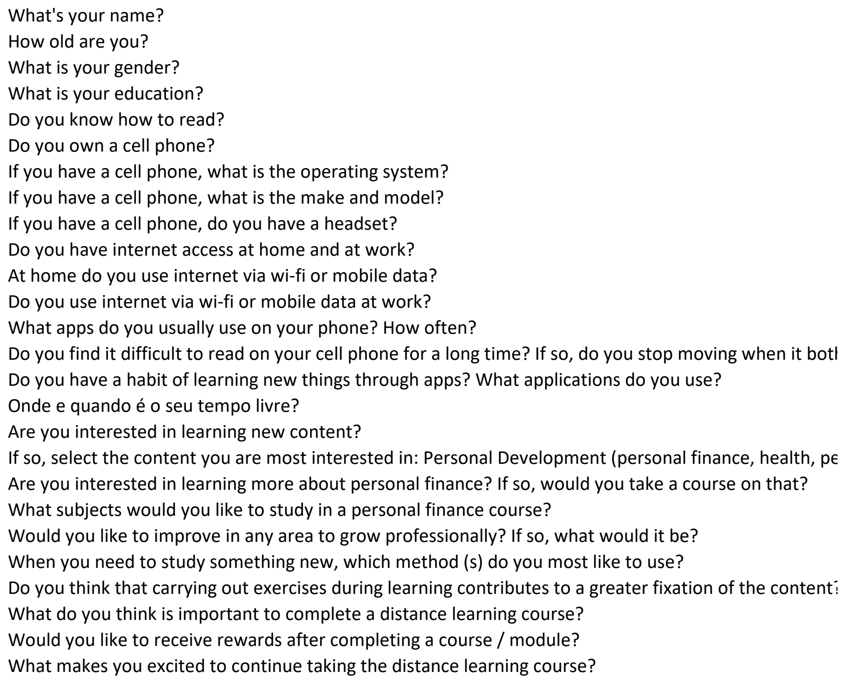
[56] Anton Vasylenko. *Microservices vs Monolith: which architecture is the best choice for your business?* `https://www.n-ix.com/microservices-vs-monolith-which-architecture-best-choice-your-business/`. Visited 12/04-21. -.

[57] Wikipedia. *Peter Checkland*. `https://en.wikipedia.org/wiki/Peter_Checkland`. Visited 12/04-21. 2021.

Appendix A

Figures

A.1 Interview guide from PSP2



What's your name?
How old are you?
What is your gender?
What is your education?
Do you know how to read?
Do you own a cell phone?
If you have a cell phone, what is the operating system?
If you have a cell phone, what is the make and model?
If you have a cell phone, do you have a headset?
Do you have internet access at home and at work?
At home do you use internet via wi-fi or mobile data?
Do you use internet via wi-fi or mobile data at work?
What apps do you usually use on your phone? How often?
Do you find it difficult to read on your cell phone for a long time? If so, do you stop moving when it botl
Do you have a habit of learning new things through apps? What applications do you use?
Onde e quando é o seu tempo livre?
Are you interested in learning new content?
If so, select the content you are most interested in: Personal Development (personal finance, health, pe
Are you interested in learning more about personal finance? If so, would you take a course on that?
What subjects would you like to study in a personal finance course?
Would you like to improve in any area to grow professionally? If so, what would it be?
When you need to study something new, which method (s) do you most like to use?
Do you think that carrying out exercises during learning contributes to a greater fixation of the content?
What do you think is important to complete a distance learning course?
Would you like to receive rewards after completing a course / module?
What makes you excited to continue taking the distance learning course?

Appendix B

Documents

B.1 PAEE/ALE 2021 published paper

Sustainability-focused international PBL project: Rethinking digital education for individuals of low socioeconomic status

Daniel Britze[1], Mateus H. Torres[2], Jacob V. Jensen[1], Luiza C. Q. Melo[2], Thiago B. O. Lima[2], Jens M. Pedersen[3], João M. da Silva[2], Simone B. S. Monteiro[2] and Paulo C. R. Gomes[2]

[1] Department of Electronic Systems, Faculty of It and Design, Aalborg University, Aalborg, Denmark
[2] Production Engineering, Faculty of Technology, University of Brasilia, Brasilia, Brazil
[3] Master´s Degree Program in Cybersecurity, Aalborg University, Copenhagen, Denmark

Email: dbritz17@student.aau.dk, mateusconsigo@gmail.com, jvje17@student.aau.dk, luizacardosoqm@gmail.com, thiagolimapro1@gmail.com, jens@es.aau.dk, joaomello@unb.br, simone_simao@yahoo.com.br, simone_simao@yahoo.com.br

Abstract

Providing access to education for individuals of lower socio-economic status is a significant way to reduce poverty, as it empowers them to grow as professionals and as individuals. Although there is an increasing sense of urgency to promote these changes, notably motivated by the Sustainable Development Goals (SDGs) set by the UN 2030 Agenda, there are still few successful ways to solve this problem on a large-scale. As digital technology develops and affordability increases, new ways to share quality educational content are created. In an exploratory case study, with a qualitative approach, this paper presents the development of a digital application focused on providing quality educational content directed to vulnerable groups that lack access to formal learning experiences, specifically focused on waste pickers in Brasilia, Brazil. The main data collection methods used to make the decisions through the development process were observation, surveys, and interviews. Within the framework of Problem Based Learning (PBL) an international cross-disciplinary collaboration among different universities, the project, denominated "Mobile Education", involved: (i) the design and implementation of a system consisting of both a web and mobile application; (ii) the research of a viable business model to provide long-term sustainability for the project; (iii) and the creation of a pilot course of financial education for the preliminary target group, i.e., waste pickers from Brasilia, Brazil. Conclusively, the project aims to positively impact social transformation for individuals who work at the Waste Sorting facilities and lack financial knowledge. The Mobile Education project resulted in a functional version of the app (Web and Mobile) as well as the delimitation of a viable business model to keep it providing digital equality in Brazilian education.

Keywords: Digital Education; Problem Based Learning (PBL); Active Learning; Lower socio-economic backgrounds; Computer Engineering; Production Engineering.

1 Introduction

In 2018, the second largest dump in the world, the largest in Latin America, located in Brasilia, Brazil, ended its activities (ISWA, 2019). Consequently, the approximately 2500 waste pickers who depended on recycling waste from the dumpsite, found themselves without a source of income. With the goal of solving this problem, the government created sorting facilities so that waste pickers could still earn income by continuing to contribute to the local recycling chain (Barbosa, 2018; Campos et al., 2018, p. 239). Having provided them a better and prepared place to waste collection, this change represented a relevant transition of working conditions (Campos et al., 2018, p. 226). However, this move of the waste pickers to sorting facilities, came with the side effect of income reduction for the waste pickers. This happened due to the amount of waste available in the sorting facilities being smaller than in the dumpsite (Campos et al., 2018, p. 238; Britze, 2019). In this process, government entities, researchers and waste picker leaderships diagnosed that, in addition to their low incomes, the lack of financial education of this population was a bottleneck for the waste pickers to get their basic needs (Cruvinel et al., 2019).

Aware of this challenging scenario, an international, cross-disciplinary collaboration between Aalborg University (AAU) and University of Brasilia (UnB), the Mobile Education Project (MEP) of the EPIC SDG Challenge (SDGC) initiative, started in 2019 with the goal of providing the support to the waste pickers on their educational gaps through digital learning. Since the conception of the SDGC initiative, students have been working within a Problem Based Learning (PBL) approach, with different teams from both universities working on solving the problem, as a result many premises were validated.

This article presents the development of the most recent MEPs, in which a digital solution is designed and implemented in the context of educating individuals of low socioeconomic status, especially contributing to the Sustainable Development Goals #4, #8, #10 and #17 from the 2030 Agenda (Assembly, 2015). The long-term goal is to increase social mobility, enabling the waste pickers, and in future, other lower socio-economic groups, to seek better and higher paying jobs, in addition to the increases in their competences. Usability, maintainability, and financial viability of the product are prioritized, increasing the long-term sustainability of the solution.

The structure of the article reflects the Problem Based Learning (PBL) framework applied throughout the project, and thus consists of the following chapters: (i) Introduction; (ii) Problem examination and analysis; (iii) Methodology; (iv) Product Development and Implementation; and (v) Conclusion.

2 Problem examination and analysis

The project focus is defined by analysing the specific use case in detail based on the initial premises from the problem. In this section, the problem analysis is comprised of three main subsections: Problem Identification; Problem Examination; and Existing Solutions. Therefore, the problem identification, examination and analysis contain the literature review within a problem based learning approach, in addition to both the results of field research of the problem, and research about existing solutions.

2.1 Problem Identification

The Mobile Education Project focuses on a societal problem: the gaps in the education of waste pickers. The absence of educational access is an issue for vulnerable groups in Brazil, who usually need to enter the workforce early, hampering their learning experiences (Sonia M. Dias, 2011). Specifically, in the case of waste pickers in the Centre-West of Brazil, where Brasilia is located, this scenario is evident by the average rate of illiteracy among adult waste pickers at 17.4% (IPEA, 2016, p. 27). Additionally, 89.2% of the waste pickers have not completed basic education (IPEA, 2016, p. 27) and, therefore, have an extra challenge finding a better job (IPEA, 2016, p. 178). Even though the income they earn from the waste collection is low (Ferraz, C., 2021), especially after the closure of the dumpsite, the way they manage their income can also influence in their financial health. According to the coordinator of one of the waste sorting centres, Cleusimar, interviewed by our team, "it is not uncommon for several waste pickers to run out of money well before the end of the month, even if some waste pickers in a similar situation do not go through the same problem", indicating that, even though their income is low beforehand, the way it is managed has significant impact in their finances.

Therefore, the acquisition of financial education became evident as a central need for the waste pickers. Before providing education that empowers them to change their lives for the future, it is essential that they receive the education necessary to stabilize their current financial situations with better management of their present available resources. Consequently, creating a framework for digital learning content focusing on administration of personal finances is defined as the focus of the Mobile Education project.

2.2 Problem Examination

The determination to solve this complex societal and educational problem through a digital learning method raises two main different challenges: (i) designing and implementing the digital platform, in technical and usability aspects; (ii) deciding the educational and learning approach to be implemented. The challenges related to the technical development of the platform are specifically addressed in the topic 2.3, which deals with existing solutions. Thus, in this problem examination, the focus is the usability aspects of (i) and all aspects of (ii).

In this context, 25 waste pickers and 5 cooperative leaders were interviewed by the research team as part of the investigation process. Throughout the interview process, the goal is to acquire information about their educational backgrounds, their level of technological access, their acquaintance with digital education, their preferred learning methods and approaches, and their greatest difficulties in managing their finances. The information about the waste pickers' educational level revealed that, generally, most of the waste pickers, 65.3%, have not completed elementary school (IPEA, 2016, p. 27) and have difficulties reading, writing, and interpreting.

Subsequently, the importance of adapting the course structure to provide a suitable learning experience is apparent. The interviews indicated that their level of technological access would be appropriate, since most of them, 88%, have phones and internet access. Accordingly, 80% of the group surveyed formed by the waste pickers and their leaders considered a learning mobile application a proper alternative to improve their educational access, since, in their perspective, mobile courses would fit their routines better than courses with scheduled time or in-person.

The decisions of attractive teaching methods and approaches have been pointed by the leaders of the cooperatives as potential points of contention, due to waste pickers' resistance to education in general, occasioned by their past experiences with in-person courses, which were, as described by Campos (2018, p. 235 & p. 237), part of the transition period of the dumpsite closure. Whilst considering this possible issue, the leaders reinforced the importance of using simple communication and contents to keep the waste pickers interest. After the interviews, the requirement for the course is defined with focus on developing the waste pickers' financial management skills through a mobile application with basic financial contents in a simple and didactic way.

2.3 Existing Solutions

Before designing and implementing a solution, research is done into similar products to incorporate the findings into the design and implementation process, ensuring that this project's outcome is singular and novel whilst also gaining insights into potential development paths. Broadening the category of this project into digital learning, a list of most successful applications is analysed, and the main characteristics highlighted (Table 1).

Name	Kahn Academy	Rise Articulate	Udemy	Kolibri
Description	A course provider focused on mathematics, with recent expansions into science, arts economics etc.	A user-friendly course creation / course completion tool.	A simple course-based learning environment supporting text, images, audio, and video.	A platform used to share already downloaded courses from other platforms between phones whilst offline.
Features	Heavily focused on video for learning. Quizzes to show progress. Gamification of learning. Downloading of videos on their app.	User focused approach, with a great User experience (UX). High quality course creation tools, making it simpler to create engaging courses.	Course creator with a big backlog of other courses. Downloadable content. Support for assignments beyond just quizzes.	High quality offline sharing tool. Open source – can be implemented and modified.
Issues for use case	Downloading videos manually. Unable to create new courses in Kahn, as it is only done by them and their partners.	Resource-demanding and often complex content elements. Rigid integration with mobile applications. High price.	Most courses are behind a paywall, making licensing expensive. Downloading videos manually.	Courses cannot be created on this platform.

Table 1: Main references of existing solutions

Whilst many digital learning platforms are available, none are optimal for solving this use case, highlighting the need for a tailor-made platform to better suit the needs of the waste pickers. The last cycles of the MEP partnership had also produced a functional application based on the model proposed by Mejer et al. (2020) for financial education issues of waste pickers. Although the platform was successful in many aspects, the focus of it was to be a financial management tool, missing any kind of learning content inside the App.

As a result of this analysis, feature extraction of the different existing platforms is done, gaining an overview of both the core features of a digital learning application and the specialized features that are well suited to the waste pickers' use case. An example of feature extracted is the Kolibri platforms' use of caching, allowing courses to be cloned from one smartphone to another (Kolibri, 2021). The usefulness of this feature becomes apparent based on the research provided in the article by (Britze & Nielsen, 2019) wherein it is established that the courses would only be downloadable whenever the individual user is connected to Wi-Fi. By implementing the same course cloning structure, it could help alleviate this potential roadblock to entry. With these important notes taken from existing solutions, a summary of conclusions from the problem identification and examination is made.

2.4 Subsidiary conclusions

The problem analysis revealed a clear deficit of education tools for both the waste pickers and other groups with lower socioeconomic status. However, beyond this scarcity of educational options, the waste pickers expressed a high willingness to learn, especially via their smartphones. While their perception on the educational mobile application was by a large majority positive, it's fundamental to understand the specific needs for adapting the tool to the waste pickers learning capabilities, by, for example, incorporating text-to-speech and video materials, enabling those with little to no literacy to also benefit from courses (Britze & Nielsen, 2019, p.19; Mejer et al., 2020, p. 43).

Having established these case specific requirements, the examination of existing solutions led to two different inferences. Firstly, it was found that in person classroom-based teaching had proved somewhat ineffective. Secondly, the currently available products in the market do not meet the learning needs of the target group, highlighting the necessity for a tailor-made solution. With these conclusions reached, the methodology for implementing the solutions for this project is defined.

3 Methodology

After establishing key premises about the problem, the methodologies used to solve it are selected and tailored to fit the project. Since the aim of the research is to solve a problem through Problem Based Learning (PBL), the methodologies for implementing PBL in both universities are discussed. Due to the international nature within a larger cross-disciplinary project, project management and teamwork across fields, universities and groups are examined, showcasing the methodology for cooperative PBL based teamwork within larger projects. Therefore, this section is comprised of three main subsections: Problem Based Learning Methodology; Research Methodology; and International Cross-disciplinary Teamwork methodology.

3.1 Problem based learning methodology.

Problem Based Learning (PBL) is a student-centred, collaborative, non-traditional approach to education in which students learn about a subject through the experience of solving an open-ended problem as the main responsible in the learning process (Adderly, 1975; Prince & Felder, 2006; Alfaro, Apaza, Luna-Urquizo, & Rivera, 2019). Specifically, in the engineering area, as part of the applied sciences, PBL appears as a strategic mechanism in training students to apply their knowledge in real problems (Cano, López, & Rebollar, 2008; Habash & Suurtamm, 2010; Tran & Nathan, 2010). This approach guides the students to solutions that require extensive use of soft and interdisciplinary skills (Taajamaa, Kirjavainen, Repokari, Sjöman, Utriainen, & Salakoski, 2013). The PBL experience generates a development for students that goes beyond the understanding of hard skills, generating future professionals with good communication and professional skills, both of them continuously tending to be more required in the labour market. (Deshpande & Huang, 2011).

In this context of active learning, a partnership between the University of Brasilia (UnB) and the Aalborg University (AAU), both learning spaces that empowers the PBL approach, emerged. At UnB, the PBL has been part of the strategy since its conception, which aimed to focus education on real problems as a way to form citizens aware of the challenges of the country and the world (Monteiro et al., 2017). Similarly, since the founding of Aalborg University, all programmes are problem Based learning centred (Aalborg University, 2021). By having problem based learning so closely integrated within the university, effectively covering half of the curriculum, international interdisciplinary projects focused on concrete solutions, like the one referred in this article, become possible.

3.2 Research Methodology

This article is made as an Exploratory Case Study with qualitative research approach. Starting from the case of the Mobile Education project as the reference experience, it aims to develop hypotheses and propositions in the context of providing education to people of lower socio-economic status. It also explores the results and, finally, outlines possibilities for a future educational paradigm in the context of international collaboration between universities using PBL, especially in the engineering scenario.

The data collected to establish the instructional design and the platform premises was obtained by interviews focused on qualitative questions, applied to 25 waste pickers in their work facilities and 5 cooperative leaders. The questionnaire consists of questions related to interests, feedback, and some other questions to a set of general profile of a waste picker.

3.3 International cross-disciplinary teamwork methodology

The project is comprised of students from two different universities, in two different countries. In Denmark, two computer engineering students from AAU utilized the scope of creating the mobile education app as their Computer Engineering Bachelors Project (CEBP) and were leading an initiative to make feasible a business model to ensure long-term sustainability for the solution. In Brazil, 3 groups also used the creation of this application as project scopes for the main part of their activities in 3 different PBL based courses of the industrial engineering bachelor of UnB: "Production System Project 1" (PSP 1), "Production System Project 2" (PSP 2), "Production System Project 3" (PSP 3) and "Production System Project 5" (PSP 5), each with a singular focus in the development process.

To certify the success of the expected deliverables related to the mobile application, some risks had to be acknowledged and an action plan should be made to diminish the impact of them. The international cross-disciplinary teamwork was a premise of this project, but it was also the main source of risks related to communication, cultural differences, schedule, and others (Table 2).

Risks	**Action Plan**
Limited dedicated time and schedule from students	Early planning of important deliverables and respective delivery dates
Different project expectations from students and professors	Previous expectation alignment between professors and main stakeholders of each country
Language Barriers between the 2 different countries	Establishing selection criteria for the students participating of the international groups, based on the sufficient knowledge of the selected language for communication between teams
Difficulties in communication between team members due to physical distance	Structured methodology for online communication, reporting and documents sharing
Challenges in the alignment between the teams	Structured methodology for integration between projects and their respective stakeholders.

Table 2: Tasks from each team

Having analysed the risks, a tailor-made integration and management methodology for the work involving the different teams is made, focusing on minimizing the risks. To apply this methodology, four steps are considered: (i) Division of Responsibilities; (ii) General Alignments; (iii) Setting Project Owners (POs); (iv) Consolidation of Integration Methodology:

Step (i), Division of Responsibilities: Software design and implementation was a central scope as a way to achieve a Minimal Viable Product (MVP), and pilot educational content must be created to allow a sufficient pilot implementation. Moreover, to ensure the long-term economic sustainability of the project, a business model must be created. Thus, the courses were assigned to each of these necessities (Table 3): CEBP project was defined as responsible for the software design and implementation, PSP2 for researching the problem through interviews and creating pilot educational content, PSP1 for the research of business model viability in Brazil, PSP5 with guaranteeing the quality of all results from PSP2 and PSP3 managing the integration between different teams and stakeholders.

Team	Country	Scope
App Developers and Product Owners	Denmark	- Design the app functionalities - Implement and deploy the application - Define and align the expected results from the other teams
PSP 1	Brazil	- Search for improvements and adaptations of the app's Business Model - Checking for possible financial incomes that fits the Business Model
PSP 2	Brazil	- Interview local target market to gather their interests, feedback and general profile - Design the course structure for the app - Detail steps to make a pilot financial course
PSP 3	Brazil	- Manage the schedule and time from the project related demands of professors and teams - Control and supply of deliverables and collected data from each team - Report general status of the project to the teams.
PSP 5	Brazil	- Use of academic research and quality assessment tools to find improvements in the course structure and content created by the PSP 2 team.

Table 3: Tasks from each team

Step (ii), General Alignments: Based on the project responsibilities within Table 3, the research team made alignment meetings with each different Brazilian professor, responsible for the courses, and set the goals for each of the future teams working in the projects related to the international collaboration (Figure 1). The Danish team, working in the CEBP, prescinded alignment since the bachelors' project had beforehand the adequate flexibility to fit the MEP needs.

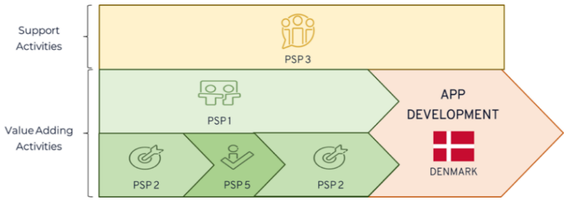


Figure 1: Value chain of the portfolio of projects

The cooperation and management of deliverables from different teams was a requirement for the project success. The value chain representation was one of the key tools to providing the tailoring of the integration, since, as C.P. Killen & C. Kjaer (2012) described, visual and graphical representation can provide benefits by supporting communication and strategic portfolio decision making.

Step (iii), Setting Project Owners (POs): Besides being responsible for the product design and implementation, the students writing their CEBP from AAU also led the initiative of making a viable business (start-up) based on the digital learning solution. Therefore, they were declared the POs of the entire development.

Step (iv), Consolidation of Integration Methodology: For the transition between the preparation and the practice of the process, an effective integration and management system needed to be developed and communicated (Figure 2).

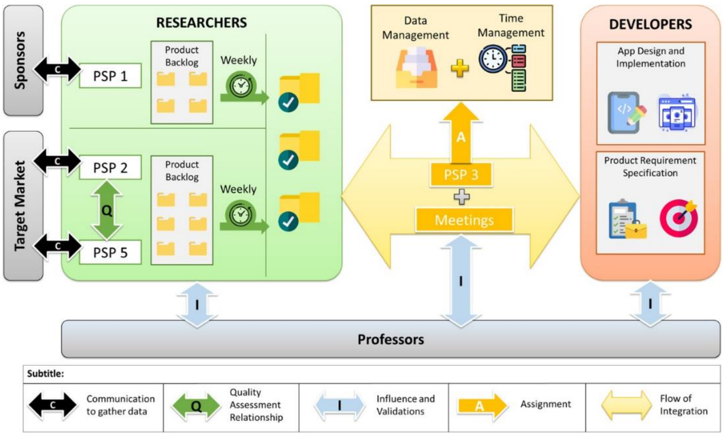


Figure 2: Integration between teams

Therefore, both the Problem Based Learning Methodology, the Research Methodology and the International Cross-disciplinary Teamwork methodology were defined, and the project, involving different teams, monitors / tutors, product owners, and professors is carried out.

4 Product development and financial strategy

Based on analytical results from user and problem domain research as described in Chapter 2, it is proposed to design and implement a digital learning platform as a distributed web system. The system has two user interfaces – (i) Educado Web, for creating and administering content as well as statistical insights for employers and (ii) Educado Mobile, for smartphone based active learning. During this student project, the ambitions are to develop the Minimal Viable Product (MVP) including both interfaces and connecting them both to a synchronized cloud-based backend handling database and file storage functionality. The general architecture of the system as well as an overview of the applied technologies are systematized (figure 3).

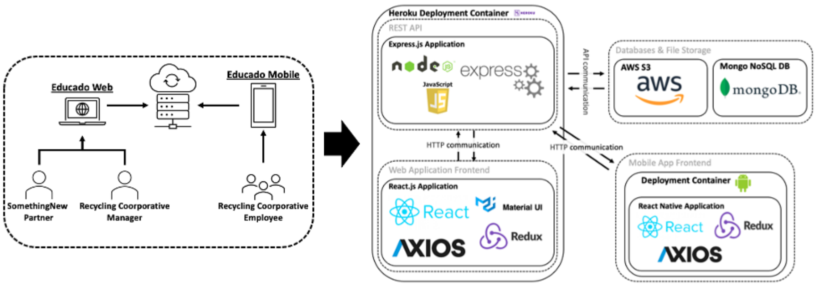


Figure 3: System design

The system is developed accordingly to engineering principles from Object-Oriented Programming, Structured System Development and Micro-Services software architecture. Design and programmatic decisions are made with prioritization of system maintainability, flexibility, and cost-optimisation. The entire codebase, both backend and frontend sub-systems, consists of JavaScript code and open-source libraries, to keep maintainability costs to a minimum while also creating a flexible framework for further development, as this is important to ensure long-term sustainable use of the product.

4.1 Financial strategy for long-term sustainability as a business

For the solution to achieve long-term sustainable impact, it is necessary to create a viable financial strategy that can support continuous development, project management and maintainability costs. The study driven part of the project is finalized in Q2 of 2021 (Figure 4), whereafter the product ownership is transferred to the engineering start-up company SomethingNew, founded by the AAU participants of the project. Though the project converts to a business-driven development, the cross-disciplinary, user-centred and PBL based frameworks presented in this article are integrated into the project management – and future student projects, are expected to progressively contribute significantly to the project (Figure 4).

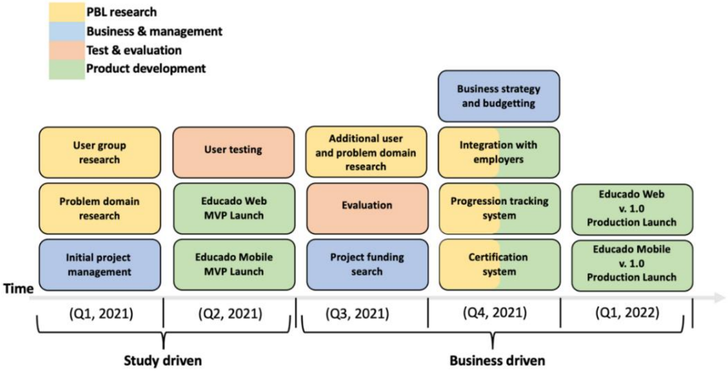


Figure 4: Project planning

The specific financial strategy of SomethingNew is out of scope of this article, but the expense coverage directly associated with this product is expected to come from both private and public funding. Organisations dedicated to UN's sustainability development goals are expected to support the project through strategic partnerships.

4.2 Course implementation

The creation of an instructional course is accomplished via the learning design methodology, as described by Filatro (2008), which involves four steps: plan, design, execute, measure. This project, carried out by the PSP2 team, is focused on the parts of planning, and designing, which will create the basis for the execution and measuring parts to be carried out in future projects. For the current initiative, a pilot version of the course was developed as a gate between the design step and the future execution step.

The planning step starts by conducting an analysis of the real needs of the users, who, in this case, are the waste pickers. As mentioned in the problem examination, section 2.2, their main educational gap was found to be in financial education, and their basic learning requirements were also defined. Therefore, this section of the article is focused on the following step, which includes the results of the design part.

The course design started with the definition of the teaching sequence, chosen as the following: (a) How to manage your finances and to make your budget (how to plan, register and control the budget), (b) Spending less and saving more (how to stick to the plan, how to avoid making debts, how to save for your future), (c) Investing for the future (where to invest for each goal). Then, aiming at obtaining a problem based learning course, learning situations were developed, which are embodied by a character, who will join the student through the entire course, simulating real-life problems. This strategy not only makes the student closer, by creating identification with the character, but also retains the student during the whole learning process by instigating him/her to figure out how to solve the problems proposed. Finally, for the choice of means and resources, the most important factor is the use of simple words and interspersed procedures with the resources included in the platform, such as short videos, audios segments, simple texts, and clear images.

The design of pilot practical exercises is also a part of the design step and was carried out and integrated in the Educado platform for real world testing with the target group. Therefore, the complete and functional MVP was achieved, as a pilot product for implementation with waste pickers, demonstrating the effectiveness of the MEP international PBL partnership in reaching the established objectives.

5 Conclusion

The platform developed in this project in the context of the Brazil-Denmark partnership matches the waste pickers needs to solve the limitations they face in their learning processes. The Mobile Education app can empower them not only to further evolve their financial lives in their current situation, but also is able to develop new competences to allow them to find other jobs, thus enhancing their conditions for social mobility.

By extrapolating the results of the present work, the future SomethingNew product can create a similar propitious learning environment for different groups in lower socioeconomic situations. Therefore, obtaining a funding model for the philanthropic business initiative that can receive support from various organizations aiming to help different specific groups of people in vulnerability status.

This partnership between Brazil and Denmark evidenced important ways of concrete achievement coming from international cross-disciplinary entrepreneurship in the University context. The Danish team had the computer science skills and the idea of transforming the Mobile Education project in a sustainable business part of SomethingNew, while the Brazilians had closer the business planning and management skills, and closer contact to the waste pickers, a key target group.

As it was effectively applied in this scenario of complexity, involving an international collaboration environment, the methodology for conducting the project proved to be flexible and adaptable for its purposes, elevating the learning potential for the student and creating more value to the stakeholders. Even though the methodology facilitated the job done by the groups involved, there is still a lot of risks related to variable motivation and performance from students, which could eventually become a bottleneck to the whole process if an individual team does not accomplish their assignments, putting at risk the other related projects. The early alignments of expectations and showcase of expected results were vital to the success of the project. Moreover, following the PBL standards, professors and monitors created a free development environment for students, stimulating self-sufficiency for the teams to resolve the problems on their own.

The students reported that the cross-disciplinary approach combined with the interdependency between teams was a key factor of engagement, having provided more motivation than in their previous experiences in the standard PBL methodology. Whilst ratifying the students' perception, professors reported that the methodology applied brought a substantial gain in both the students' attention to quality and their willingness to ask for advice during the semester. The social sustainability-related theme also appeared in the reports as a relevant aspect of the engagement. Both students and professors stated that the high level of interdependence between teams generated some challenges not usually found in the conventional PBL modality, such as managing diverse schedules, communication standards, and types of workflows between different teams. However, they pointed out that their level of learning has grown at a similar rate to the challenge. The authors are aware that there is a balance between on the one hand being motivated through the dependencies among teams, and on the other hand that each group should be able to complete their projects successfully even if other teams do not, therefore each teams' deliverables contribute significantly to the project without compromising other teams in case of potential issues.

The main limitation of this PBL international experience were the restriction due to the pandemic scenario, which prevented face-to-face meetings, adding extra uncertainty to the projects since most data collecting methods relied in interviews and in location observation. Another difficulty experienced was adjusting and fitting the project requirements and schedule with the variable different requirements and schedules of the courses in both universities.

For the next semester of the Mobile Education initiative, the goal is to obtain partnerships with a financial support equal to the running costs and future developments of the platform. Additionally, there is the key aim of utilizing the UnB-AAU partnership to conduct more projects and research, mainly focused on creating new courses in the platform and testing the quality of the application to ensure the users like using it as well as finding value in the educational content provided.

6 References

Aalborg University., (2021). The Aalborg model for problem based learning. Retrieved 30 April 2021, from https://www.en.aau.dk/about-aau/aalborg-model-problem-based-learning/

Adderly, K. (1975). Project method in higher education. London: Society for Research into Higher Education. Research into higher education monographs, 24.

Alfaro, L., Apaza, E., Luna-Urquizo, J., & Rivera, C. (2019). Identification of Learning Styles and Automatic Assignment of Projects in an Adaptive e-Learning Environment using Project Based Learning. In International Journal of Advanced Computer Science and Applications (IJACSA), 10(11), 2019. doi:10.14569/IJACSA.2019.0101191

Assembly, U. N. G. (2015). Transforming our World: The 2030 Agenda for Sustainable Development.

Barbosa, V. (2018, January 20). Brasil diz adeus ao maior lixão da América Latina, em Brasília. Exame. Retrieved April 13, 2021, from https://exame.com/brasil/brasil-diz-adeus-ao-maior-lixao-da-america-latina-em-brasilia/

Britze, D., Nielsen, R. N.. Mobile education platform: Smart caching learning materials. Aalborg Universitet project library, pages 1–13, 77, 91–95, 2019. https://projekter.aau.dk/projekter/da/studentthesis/mobile-education-platform(aebca7a9-9b25-4081-89ea-271477091e93).html.

C.P. Killen & C. Kjaer, (2012). Understanding project interdependencies: The role of visual representation, culture and process, International Journal of Project Management 30 (2012) 554–566.

Campos, H. K., Lemos, A. L., Pimenta, A. W., Dourado, A. P., Abreu, M. D., Melo, M. R., & Gomes, P. C. (2018). Case Report: How We Closed the Second Largest Dumpsite in the World. International Solid Waste Association-ISWA, Kuala Lumpur–Malaysia, 65.

Cano, J. L., López, I. L., & Rebollar, R. (2008). Learning project managment through working with real clients. The International journal of engineering education, 24(6), 1199-1209

Cruvinel, V. R. N., Marques, C. P., Cardoso, V., Novaes, M. R. C. G., Araújo, W. N., Angulo-Tuesta, A., ... & da Silva, E. N. (2019). Health conditions and occupational risks in a novel group: waste pickers in the largest open garbage dump in Latin America. BMC public health, 19(1), 1-15.

Deshpande, A. A., & Huang, S. H. (2011). Simulation games in engineering education: A state-of-the-art review. Computer applications in engineering education, 19(3), 399-410.

Ferrez, C. (2021). Waste Pickers Responsible for 90% of Brazil's Recycling At Greater Risk of Coronavirus #ApoieUmCatador - RioOnWatch. Retrieved 30 April 2021, from https://www.rioonwatch.org/?p=59928

Filatro, A. C. (2008). Learning design como fundamentação teórico-prática para o design instrucional contextualizado (Doctoral dissertation, Universidade de São Paulo).

Habash, R. W. Y. & Suurtamm, C. (2010). Engaging High School and Engineering Students: A Multifaceted Outreach Program Based on a Mechatronics Platform. In IEEE Transactions on Education, 136-143 doi: 10.1109/TE.2009.2025659

IPEA (2016). Catadores de Materiais Recicláveis, Um Encontro Nacional. https://www.ipea.gov.br/portal/index.php?option=com_content&view=article&id=27461

ISWA (2019). Climate Benefits Due to Dumpsite Closure: Three Case Studies. https://www.iswa.org/knowledge-base/climate-benefits-to-dumpsite-closure-three-case-studies/?v=19d3326f3137

Kolibri., (2021). Retrieved 30 April 2021, from https://learningequality.org/kolibri/

Mejer, A. B.; Mortensen, E. B.. Rasmussen, I. V. B. (2020). Mobile education Platform: Finance management for waste pickers. Aalborg Universitet project library. https://projekter.aau.dk/ projekter/da/studentthesis/.

Monteiro, S. B. S., Reis, A. C. B., Silva, J. M. da, & Souza, J. C. F. (2017). A Project-based Learning curricular approach in a Production Engineering Program. Production, 27(spe). https://dx.doi.org/10.1590/0103-6513.226116

Prince, M. J., & Felder, R. M. (2006). Inductive teaching and learning methods: Definitions, comparisons, and research bases. Journal of engineering education, 95(2), 123-138. doi:10.1002/j.2168-9830.2006.tb00884.x

Sonia M. Dias (2011). Statistics on Waste Pickers in Brazil, in WIEGO (Women in Informal Employment Globalizing and Organizing) Statistical Brief No 2, May, 2011

Taajamaa, V., Kirjavainen, S., Repokari, L., Sjöman, H., Utriainen, T., & Salakoski, T. (2013). Dancing with ambiguity design thinking in interdisciplinary engineering education. In IEEE Tsinghua International Design Management Symposium, Shenzhen, pp. 353-360, doi:10.1109/TIDMS.2013.6981258

Tran, N., & Nathan, M. J. (2010). An investigation of the relationship between pre-college engineering studies and student achievement in science and mathematics. Journal of Engineering Education, 99(2), 143-157.