

Princess Sumaya University for Technology

King Hussein School for Computing Sciences

EduCare  
A Smart Unified Educational Platform

**Prepared By:**  
Mohammad Ibrahim Abu-Amara

**Supervised By:**

Dr. Ragdah Hriez

Project Submitted in partial fulfillment for the degree of Bachelor of Science in Computer Science

Second Semester-2024

page left intentionally blank

# 

# Declaration of Originality

This document has been written entirely by the undersigned team members of the project. The source of every quoted text is clearly cited and there is no ambiguity in where the quoted text begins and ends. The source of any illustration, image or table that is not the work of the team members is also clearly cited. We are aware that using non-original text or material or paraphrasing or modifying it without proper citation is a violation of the university’s regulations and is subject to legal actions.

Names and Signatures of team members:

**Mohammad Ibrahim Abu-Amara**

# 

# Acknowledgments

Thank you to all the parents, students, teachers, and anyone who believes that education is an important step in the development of human life and believe that we as students can help in the development as a first line army.

# Summary

As students, we believe that the educational system is a core factor in any development, whether it is economical, social, or personal, in addition to influencing other systems like the healthcare system. Thus, to provide more value, this project's mission is to provide a platform for people to engage, develop, and learn at the highest standards. With the approach of being unified, open for everyone, and innovative for the next generation, along with connecting the platform with other platforms to join forces to tackle bigger problems and provide educational resources for anyone.

The results of this project so far are astonishing, with technical advancement, ways to integrate and use artificial intelligence responsibly, and finding ways to have a better platform for students and teachers and anyone engaged in the educational system, as well as addressing issues that have been making it harder for other platforms to make progress in the industry. All of this with making sure the platform stays on course to follow first-class principles to provide the best educational experience.

# List of Abbreviations

List the abbreviations you have used in your project if there are any and what they stand for.

**ML:** Machine Learning.

**K-12:** Kindergarten to 12th grade.

**CRUD:** Create, Read, Update, Delete.

**LMS:** Learning Management System.

**DBMS**: Database Management System.

# **Table of Contents**

[**Declaration of Originality 3**](#_czghsjivmd2i)

[**Acknowledgments 4**](#_vn3mtz83qjbp)

[**Summary 5**](#_ms4zbs7szdk8)

[**List of Abbreviations 6**](#_1fob9te)

[**Table of Contents 6**](#_3znysh7)

[**Table of Figures 11**](#_l2d748wkwu8u)

[**Table of Tables 13**](#_5nnmq2fqgehy)

[**Chapter 1:  
Introduction 15**](#_1t3h5sf)

[1.1 Overview 15](#_4d34og8)

[1.2 Problem Statement 16](#_2s8eyo1)

[1.2.1 Description of the Problem 16](#_m0o3pg1ll59x)

[1.2.2 Outcomes of the Project 17](#_1rgutaenr5ju)

[1.2.3 Targeted Audience 18](#_tvowqaerl2m2)

[1.3 Related Work 19](#_17dp8vu)

[1.4 Document Outline 21](#_n505m216k60c)

[**Chapter 2:  
Project Plan 22**](#_3rdcrjn)

[2.1 Project Deliverables 22](#_av7lcxd9mw77)

[2.2 Project Tasks 23](#_z7m57wvrdjwt)

[2.3 Roles and Responsibilities 25](#_bbidzzkveqkr)

[2.4 Risk Assessment 26](#_1c73ymyvfdi7)

[2.4.1 Risk Identification 26](#_wo7a6zhfh7ch)

[2.4.2 Risk Response 27](#_otavjuzalsyu)

[2.5 Cost Estimation 28](#_m9v3f3erezua)

[2.6 Project Management Tools 29](#_kt33ofdf9olw)

[Google Workspace 29](#_ek9fmjb9vqke)

[GitHub 29](#_muzyqfeogx3d)

[Draw.io 29](#_8prrctk35jms)

[**Chapter 3:  
Requirements Specification 30**](#_18e519nj8b0s)

[3.1 Stakeholders 30](#_lnxbz9)

[3.2 Platform Requirements 33](#_35nkun2)

[3.2.1 Software Platform Requirements 33](#_r6ybhdcqand0)

[3.2.2 Hardware Platform Requirements 34](#_3kk8l545ccz4)

[3.3 Functional Requirements 35](#_1ksv4uv)

[3.3.1 User Identity Requirements 38](#_tutxqpvgc8ol)

[U\_FR\_0 Authenticate a user 38](#_dmrbvkkb5ky9)

[U\_FR\_1 Edit user profile 39](#_ierkaukret3u)

[3.3.2 Course Requirements 40](#_lag68tsc7fu7)

[C\_FR\_0 Create a course 40](#_8bnqbm9u6yy)

[C\_FR\_1 Read a course details 41](#_m8k1x2fuzoga)

[C\_FR\_2 Edit course details 42](#_7z5yk7kymwgc)

[C\_FR\_3 Delete course 43](#_4khlmyvi00kz)

[3.3.3 Task Requirements 44](#_jh6y20f0trhx)

[T\_FR\_0 Create a task 44](#_zgr7z9rkxu4v)

[T\_FR\_1 Read a task 45](#_17dh0o6aar3b)

[T\_FR\_2 Update a task 46](#_57in7jz91tmg)

[T\_FR\_3 Delete a task 46](#_g0lk6jofasje)

[3.3.4 Institution Requirements 47](#_bsobd0kl2juf)

[I\_FR\_0 Create an Institution 47](#_tly73ce8subs)

[I\_FR\_1 Read an Institution 48](#_6dz6nqv3g3km)

[I\_FR\_2 Update an Institution 48](#_alcb55ts326c)

[I\_FR\_3 Delete an Institution 49](#_1s94sdbg2zw6)

[3.3.5 Resources Requirements 50](#_5ssa1l1mlotm)

[R\_FR\_0 Create a Resource 50](#_k6gedna6hmv5)

[R\_FR\_1 Read a Resource 51](#_ucz20cxe9hlw)

[R\_FR\_2 Update a Resource 51](#_vnzaxsd6stos)

[R\_FR\_3 Delete a Resource 52](#_ljihu69m32iu)

[3.3.6 Notification Requirements 53](#_ieimwnp3gisi)

[N\_FR\_0 Send a Notification to a user 53](#_v9lmhtogmjjn)

[3.3.7 Policy Engine Requirements 54](#_nbuvsgp9bno9)

[P\_FR\_0 Create/Modify Permissions between two entities 54](#_we41irg2haqx)

[P\_FR\_1 Validate Permissions 55](#_fwpemf5u57ml)

[P\_FR\_2 Populate join data from permissions 56](#_kbibe5bcxyk8)

[3.3.8 Search Requirements 57](#_vw0vsmnmrm6x)

[S\_FR\_0 Search for information 57](#_2uvh75b2lltb)

[3.3.9 Generator Requirements 58](#_v95jmnqrv8ic)

[G\_FR\_0 Generate Entity 58](#_6db8uny209a8)

[3.3.10 Association Requirement 59](#_vzcklf4gip29)

[A\_FR\_4 Get entities connected to the other entities 59](#_pouqgpzg9pvr)

[3.4 Non-Functional Requirements 60](#_44sinio)

[3.4.1 Security 60](#_jucbd57uadmw)

[3.4.2 Reliability, Availability, and Fault-Tolerance 60](#_xxluht5ncsco)

[3.4.3 Scalability and Performance 61](#_py9s3h6ehzcv)

[3.4.4 Usability, Portability, Ease of use, and Accessibility 61](#_8oax70t9hi26)

[3.4.5 Code quality and Maintainability 61](#_bvwbz230shku)

[3.4.6 Customization and Localization 61](#_u2z7i7iscd9j)

[3.5 Other Requirements 62](#_2jxsxqh)

[3.5.1 Data formats 62](#_s5zpvj63verv)

[3.5.2 Mandatory User Data 63](#_rqlx6vrwqe81)

[3.5.3 Mandatory Course Data 64](#_wc5rubxj3df6)

[3.5.4 Mandatory Task Data 65](#_h6oaw8nejro3)

[N3.5.5 Mandatory Institution Data 67](#_yjkfshx964i9)

[3.5.6 Mandatory Resource Data 69](#_q1gnlfeaehlv)

[3.5.7 Mandatory Notification Data 70](#_rr5ijxf18tuh)

[3.5.8 Mandatory Permissions and Roles 71](#_a0gv1vsry5gy)

[3.5.9 Mandatory Generator Data 73](#_jjev1fkizq04)

[3.5.10 Mandatory Metadata data 74](#_cj9seh7si3gn)

[3.5.11 Auto Generated ID Data 75](#_kz5wo2bvzdml)

[**Chapter 4:  
System Design 76**](#_z337ya)

[4.1 Logical Model Design 76](#_dfj2mhw5ttpu)

[Structured Approach: 77](#_6pxrxz9niwfo)

[Functional Decomposition Diagram (FDD). 77](#_wvfgyhg92q0n)

[User FDD 78](#_qee1owqhn5li)

[Course FDD 79](#_ypbv4be2l9qd)

[Task FDD 80](#_yczlbmwdrka6)

[Institution FDD 81](#_j4afsx4kynn3)

[Resource FDD 82](#_y7p0egb1duti)

[Notification FDD 83](#_uqxy0i3eqjxm)

[Policy Engine FDD 84](#_m6byysjirlmr)

[Search FDD 85](#_hzvgycxqytjq)

[Generator FDD 86](#_2fxpknlrhbn1)

[Association FDD 87](#_wyc6jv8eoszu)

[System Context Diagram (Context DFD). 88](#_iw0gltq9biwq)

[User System Context Diagram 88](#_vxm863dpsfs3)

[Course System Context Diagram 89](#_ojmxpchb1ec5)

[Task System Context Diagram 90](#_jbhe54ljiax7)

[Institution System Context Diagram 91](#_2evwxl47gcy8)

[Resource System Context Diagram 92](#_xbbfv1ga99oc)

[Notification Context DFD 93](#_idc1plv1nrw6)

[Policy Engine System Context Diagram 94](#_ivr66sydykmq)

[Search System Context Diagram 95](#_fcen4fnqqiom)

[Generator System Context Diagram 96](#_nmemvlhndq89)

[Association System Context Diagram 97](#_quiswxgih8my)

[Entity Relationship Diagrams (ERD) 98](#_ywc58zbersxx)

[4.2 Physical Model Design 99](#_dz1byxzgy2v8)

[User Interface Design 99](#_fmv7rm35t50x)

[Login Page Interface 99](#_iuis7md9wyty)

[Landing Page Interface 100](#_1r45ffd84xpb)

[Course Interface 101](#_mmdrymnwijpu)

[Database Design 102](#_2q5yt1o6zlj7)

[User Database 103](#_fjspujhxoxv3)

[Course Database 103](#_o5lmrrtodqyz)

[Task Database 104](#_y71q89dps91r)

[Institution Database 104](#_ozf9jppsfqif)

[Resource Database 105](#_hhf8tud3pc6w)

[Notification Database 105](#_kzg3y9bqo7hd)

[Custom Metadata Database 105](#_nqwyfe6q6jex)

[Generator Database 106](#_igaelknr8yio)

[Policy Engine Database 106](#_mjyn2ulujhht)

[Association service database 107](#_b5evweap4rp2)

[**Chapter 5: Implementation 108**](#_30chfrras1zn)

[5.1 Programming Languages, Tools, API 108](#_xodsebp86j73)

[5.1.1 Front-End 108](#_29qb9lotoilj)

[React 108](#_pcilcv2tzjq8)

[Chakra-ui 109](#_llrqqt7aek1p)

[5.1.1 Back-End 110](#_onhs3rmm3sly)

[JavaScript 110](#_2kd6498ofhpu)

[TypeScript 110](#_myi2yav8ldtw)

[NestJs 110](#_8hlveow4ja7q)

[Fastify 110](#_v2cq4ubn6kqn)

[MongoDB 111](#_4refo3zgj1x1)

[5.2 Overview 111](#_gxwxa08wuj57)

[5.2.1 Front-End 111](#_jomdtuqimw78)

[● Index 111](#_7jix70vhu5o1)

[● SignIn 111](#_tacrr3kbz6cn)

[● Register 112](#_4umsgvxpxum4)

[● Course 112](#_rj7bhkot0o29)

[● Profile 112](#_owrnzvm3x9vs)

[● Track Course 112](#_5ydtjhx582aq)

[5.2.2 Backend 113](#_j1csy7rio5m6)

[1. Modules 113](#_y3eqisckfhr5)

[● Fastify 113](#_yhdz1v96janr)

[● Multer 113](#_6kgga1e9iv68)

[● Mongoose 113](#_iam7g6ok5xqh)

[● JWT 113](#_p6x46zv5k9x1)

[● RxJs 114](#_8bg1huevky8d)

[● Jest 114](#_5pkkjwmmzfpi)

[● Bcrypt 114](#_4y255byq23jj)

[● Backend System Architecture 114](#_qqc5c4e0l594)

[5.3 Main functionality 117](#_o2h6tl4nepyc)

[5.3.1 User Login 117](#_u4ln2iob3cxd)

[5.3.2 User Registration 119](#_i9cd27fxnsj)

[5.3.3 Profile 122](#_sfh0379wr1xi)

[5.3.4 Creating Course 126](#_41p7fz8aefm)

[5.3.5 Creating Tasks 128](#_jj32rh3en391)

[5.3.6 Uploading Assets 130](#_5r5puhh7g2vf)

[5.4 Features 133](#_acbin2dsvfzc)

[Table XXXX: Features Table 133](#_k7my3ym81ggd)

[Newly Implemented Features 134](#_1xwwic1uu73g)

[**Chapter 6: Testing 135**](#_iu49zdfm8mj7)

[6.1 Testing Approach 135](#_jxhi8y51eh2n)

[6.1.1 Functional Testing 136](#_ngnnem1jstqc)

[Unit Testing 136](#_jrljuoia1y2q)

[Integration Testing 136](#_hlrnxpoif4co)

[System Testing 136](#_xnyv0upind8)

[6.2 Testing Tools 137](#_zh4ru923cy23)

[6.3 Testing Features 137](#_7cj0dtnzv39x)

[Table XXXX: Tested Features 138](#_fbh79fd90c7m)

[6.4 Testing Samples 139](#_pou14frvjtw5)

[6.4 Testing Samples 139](#_r17bqzu8av3b)

[Authentication 139](#_hb70fm4soaa1)

[Table XXXX: Tested Samples For User Authentication 139](#_h59ga91h37j2)

[Edit Profile 140](#_xjby40v7zbf3)

[Table XXXX: Test Cases For User Profile Edit 140](#_x2hlgimm30d8)

[Create Resources 141](#_ncvyj3yoktf)

[Table XXXX: Test Cases for Resource Creation 141](#_vj52ebjt45mq)

[**Chapter 7: Conclusions and Future Work 143**](#_25920vxugc23)

[Growth areas 143](#_rekmr8d5arxs)

[Powerful Search Functionality 143](#_6h4siz9fpdky)

[Intelligent Content Organization 143](#_l3t4ayawyo9k)

[Supporting Student Self-assessment 143](#_lyefysf73krs)

[Robust Security Measures 144](#_4vpp8ud2ec1m)

[**References 145**](#_49x2ik5)

[**Appendix 146**](#_49x2ik5)

[1.3 Educational Platforms 146](#_vnvss645sh4t)

[Moodle 146](#_pznu6o4ugwz9)

[Teach ‘n go 146](#_arxzwjlfshfz)

[Canvas 146](#_qqqqi4tp8ype)

[Google Classroom 147](#_w4i6fom1btar)

[3.3.7 Policy Engine service 148](#_t98k7h4k3lma)

[Process 149](#_ls1ldq606jbn)

[Implementation Details 149](#_ms388cxm47xq)

[ID Generation for each entity 149](#_o5leauxzom7f)

[Control Plane 149](#_gp3fqgmwp0ke)

[Data Plane 150](#_1znp1tnp2ze1)

[Properties of the Policy Engine and the Entities IDs 150](#_emgjlinl1lvt)

[3.3.10 Association service 151](#_q00r0drfwnby)

# Table of Figures

[Figure 0: User FDD](#3h72rrc21lq0)

[Figure 1: Course FDD](#j4wl2nj4wo6t)

[Figure 2: Task FDD](#hi91rtbvn76z)

[Figure 3: Institution FDD](#c159jertsme)

[Figure 4: Resource FDD](#otiut1twzlwz)

[Figure 5: Notification FDD](#59tl5zd9stzp)

[Figure 6: Policy Engine FDD](#umaonmyn14x1)

[Figure 7: Search FDD](#jm2twm5jzinp)

[Figure 8: Generator FDD](#ng6l8ld0xgww)

[Figure 9: Association FDD](#df7tj2ltq82f)

[Figure 10: User System Context Diagram](#r1yayjgn8go3)

[Figure 11: Course System Context Diagram](#cmg0gb4g11py)

[Figure 12: Task System Context Diagram](#xi06mw214e70)

[Figure 13: Institution System Context Diagram](#kix.7uskhsadks9)

[Figure 14: Resource System Context Diagram](#d4v0rf5kklng)

[Figure 15: Notification System Context Diagram](#dymno2phr4yq)

[Figure 16: Policy Engine System Context Diagram](#mhijmpegpn6i)

[Figure 17: Search System Context Diagram](#28hp9i9jomza)

[Figure 18: Generator System Context Diagram](#s4ez9kuzle)

[Figure 19: Association System Context Diagram](#1zz7825cqvm4)

[Figure 20: EduCare ER Diagram with relationships](#xeqg6tavaarg)

[Figure 21: EduCare Login User Interface](#8slvn9dcid44)

[Figure 22: EduCare Landing Page Interface](#vvcn8za1guel)

[Figure 23: EduCare Course User Interface](#d2nadifiuvjd)

[Figure 24: User table database design](#rj5c6ddmw7vl)

[Figure 25: Course table database design](#1wmg293ncdzj)

[Figure 26: Task table database design](#kix.4egxa7t7s835)

[Figure 27: Institution table database design](#kix.dyum1a1mybu8)

[Figure 28: Resource table database design](#kix.nukyliemhk1d)

[Figure 29: Notification table database design](#kix.m0s6xpv68b5f)

[Figure 30: Custom Metadata table database design](#kix.yciu5a2a6xfa)

[Figure 31: Generator table database design](#kix.rflviltl3etu)

[Figure 32: Generator table database design](#kix.vlm8wg2fy66m)

[Figure 33: Association service table database design and process](#kix.1683n3y95niu)

# 

# Table of Tables

[Table 0: Comparison of current educational platform](#c1beevci4lm)

[Table 1: Document Outline](#u4gaf7xsj1nc)

[Table 2: Project deliverables and output.](#tf2cb0bl6isr)

[Table 3: Project tasks and their definitions.](#je2hm4d2436i)

[Table 4: Project roles and responsibilities](#h9nljgypyv28)

[Table 5: Project tasks and risk identified with each task and its impact](#ll5kpxlo3sbs)

[Table 6: Risk response to the risk assessment associated with the task.](#1wnh8qbo982v)

[Table 7: Cost comparison for cloud providers for the platform architecture and requirements.](#l7348914qn8x)

[Table 8: Stakeholders definitions, impact, and goals.](#hcwwgdepej5r)

[Table 9: Stakeholders matrix analysis, which describe the interest and influence each stakeholder has,…](#jmbvzmstgsc8)

[Table 10: EduCare Functional Requirements](#14txbgptv55s)

[Table 11: User authentication functional requirement](#tvoeqbraim0z)

[Table 12: User profile modification functional requirement](#49a2zc1w2b59)

[Table 13: Course creation functional requirement](#vratozl9mdo3)

[Table 14: Reading Course data functional requirement](#vpnwxwxvcnzi)

[Table 15: Course data modification functional requirement](#erjuxlrdr5wz)

[Table 16: Course deletion functional requirement](#8ukfvkc58yov)

[Table 17: Task creation functional requirement](#8phspewcdero)

[Table 18: Reading Task data functional requirement](#ixhk6da07vkq)

[Table 19: Task modification functional requirement](#ximbxtlvnmtp)

[Table 20: Task deletion functional requirement](#ftwb04ug526a)

[Table 21: Institution creation functional requirement](#8kdf9758ecl)

[Table 22: Reading Institution data functional requirement](#ctrykvwjumtl)

[Table 23: Institution modification functional requirement](#s4w2562ija9d)

[Table 24: Institution deletion functional requirement](#apoo291iak75)

[Table 25: Resource creation functional requirement](#ekxqju9p6pkd)

[Table 26: Reading Resource data functional requirement](#coe5ihrvt11v)

[Table 27: Resource modification functional requirement](#gnaoymh0w1pc)

[Table 28: Resource deletion functional requirement](#f6eha2pi6ts3)

[Table 29: Sending Notification functional requirement](#o338qkewf850)

[Table 30: Policy Engine Permissions manipulation functional requirement](#axcavotoopfm)

[Table 31: Policy Engine Permissions validation functional requirement](#jnkkr92t0gzd)

[Table 32: Policy Engine and Query Engine streaming functional requirement](#p88nkx6jvcp3)

[Table 33: Search Engine functional requirement](#q58pu1l2qd0)

[Table 34: Generator for entities functional requirement](#7p8l7trrewme)

[Table 35: Query Engine functional requirement](#n9l30vwn44gq)

[Table 36: Data Dictionary for course data model](#23dqs0x7vn51)

[Table 37: Data Dictionary for Task data model](#5rcb5jth8ii5)

[Table 38: Data Dictionary for Institution data model](#7bct6sbezx3y)

[Table 39: Data Dictionary for Resource data model](#f79ebivjyhiy)

[Table 40: Data Dictionary for Notification data model](#d1p4jfb02sdw)

[Table 41: Data Dictionary for Permissions data model](#wfdif4wloxfk)

[Table 42: Data Dictionary for Generator data model](#jy6b75kww0yi)

[Table 43: Data Dictionary for Metadata data model](#zgnhqznj2c2y)

# Chapter 1: Introduction



## 1.1 Overview

EduCare is a unified educational platform that provides the necessary modern tools to be able to collaborate, organize, and search for knowledge with ease. Automating various types of tasks using generative AI and making education more accessible for everyone to find and make use of the vast amount of information and knowledge using an intent and contextual-based search engine.

The platform is going to be a public website for anyone to create courses, and events, upload resources, create study plans, ask and answer questions, and much more. Together with creating private organizations, inviting only the ones that belong to those organizations using a combination of permissions and admin management and privileges.

Giving users and organizations the ability to create hierarchical groups to structure and organize users into different categories (Book Clubs for virginia woolf from a library, department of computer science in a university, admins responsible for registration in a school, students in a calculus course that is taught by volunteer teachers, …, etc) and giving each group in the hierarchy the needed tools and permissions to be able to do their work efficiently; utilizing channels between different levels in the hierarchy to share and join forces using these groups and tools. In addition, the platform will have integration with other public education and collaboration tools to give users the choice and power of the way they like to communicate, collaborate, and get their tasks done. Along with other smart features to make users' life easier, to mention a few, an option to version old entities (Courses, Documents, …, etc) within a timeline so users can go back in time to a certain view of an entity, recommendation for a schedule for given tasks (exam schedule for a university, study schedule for students, …, etc), exams and assignments cheat and plagiarism detectors using custom public integrated solutions, joining courses between universities, organizations, the public community with ease (one-click collaboration), searching the inside of some resources such as searching inside a book that was uploaded to a course, and far more.

## 1.2 Problem Statement

### **1.2.1 Description of the Problem**

Currently, educational platforms are composed of multiple disconnected tools. Each university has its own internal system, which they spend resources developing and maintaining. Other service providers came to fill in the gaps, and left fatal flaws due to poor communication with the industry; creating more distance between each system and impacting the users that needed to learn these tools[[1]](https://educationaltechnologyjournal.springeropen.com/articles/10.1186/s41239-021-00262-1)[[2]](https://ftp.iza.org/pp56.pdf).

Schools (K-12) are not yet fully embracing online education[[3]](https://files.eric.ed.gov/fulltext/EJ1333732.pdf), despite the fact that online education is becoming popular among higher education, and school students will have to get used to online education as they get older; in higher education or if they want to continue something besides school as an extracurricular activity. Additionally, some parents and teachers may be concerned about the quality of online education and reluctant to change.

Moreover, the current system is not conducive to innovation and change. It is difficult to integrate new tools and features, and this makes it difficult to improve the educational experience for students.

The internet has made information more accessible, but it can also be overwhelming. Students and teachers need to be able to effectively search for and evaluate information in order to learn effectively.

From a technical side, owning these systems means risking user experience and their data, i.e.: network issues, hardware resources management, …, etc. Successively spending more dedication to handle these risks instead of focusing on simplifying the educational side of things.

AI is advancing rapidly in many areas, but the education industry is lagging behind. There are many potential AI solutions for educational problems, but resources are being spent elsewhere, so the gap will likely continue to grow.

### 

### 1.2.2 Outcomes of the Project

A one-stop shop for education that streamlines the learning process by automating tasks, securing data, and making it easy for users to find and share information.

Here are some of the key features of this platform:

* Collaboration: Users can easily collaborate with others on projects, assignments, and discussions.
* Search: Users can easily find the information they need, regardless of where it is stored.
* Automation: Tasks can be automated, freeing up users' time to focus on more important things.
* Security: Data is secure and protected from unauthorized access.
* Usability: The platform is easy to use and navigate.
* Hierarchical groups: allowing users to organize themselves into logical groups based on their roles, interests, or other criteria.
* Integration with other tools: giving users the flexibility to choose the tools that work best for them.

This platform is a valuable tool for students, teachers, and organizations of all sizes. It can help to improve the learning experience, increase productivity, and reduce costs.

The project also aims to integrate AI to modernize education and push the boundaries of what's possible. It also aims to make the platform customizable for institutions so that they can focus on teaching and learning without having to worry about managing the platform.

Here are some of the ways that AI can be used to modernize education:

* Personalized learning: AI can be used to create personalized learning plans for each student, based on their individual needs and interests.
* Adaptive learning: AI can be used to adapt the learning experience to each student's pace and understanding.
* Intelligent tutoring: AI can be used to create intelligent tutors that can provide students with real-time feedback and support.
* Assessment: AI can be used to automate assessments, freeing up teachers' time to focus on teaching.

### 

### 1.2.3 Targeted Audience

The educational industry as a whole, thoroughly, Institutions (Schools or K-12, University and Higher Education, Training Centers, …, etc) and their students/instructors are the main audiences here, together with anyone who is looking for a way to organize their studies, search for educational resources, and connect with institutions and/or other students.

## 

## 1.3 Related Work

Here’s a list of some other software that might share somewhat of the experience that EduCare offers, comparison will be done in the following areas:

* **Course and Student Management**: The basis for any educational system, making creating and managing courses as simple as possible, this has been researched the longest in the industry, hence, it has the most development.
* **Groups and Permissions Management**: Repetitive tasks can be done using one-click features, and permissions are managed on the fly for large groups that are naturally represented as a hierarchical structure.
* **AI capabilities and integration**: The idea of making use of current services and technologies, especially AI services that make tedious tasks easier and more interesting, online educational systems lack the most, and they have the least amount of development.
* **Searching for Resources:** The functionality of searching internal or external resources securely and using human language rather than a keyword-based search.
* **Social learning and content creation**: The ability for users to utilize the platform to collaborate with each other, e.g.: asking questions, posting reviews, creating their own version of the course, …, etc.
* **White labeling and Customization**:Giving organizations control over customizing URLs for their pages; unattachment to the platform provider. On top of visual representation for organizations and their own branding.
* **Gamification and Certification**: Creating engaging and appealing challenges and experiences for the student's learning experiences adding a healthy amount of competition, such as leaderboards, extra points for some challenges, and badges. Certification generation from Universities as a safe digitally signed electronic certification.
* **Managed Cloud-based**: Offering a fully managed service for organizations and users so they can focus on education and other important tasks, rather than focusing on managing the platform.
* **Kids Safety features**: K-12 safe platform by default and making sure material is checked and safe. Along with parental control for users who are under a certain age.

| **Area of Comparison** | **EduCare** | **Moodle** | **Teach ‘n go** | **Canvas** | **Google Classroom** |
| --- | --- | --- | --- | --- | --- |
| **Course and Student Management** | Yes | Yes | Yes | Yes | Yes |
| **Groups and Permissions Management** | Yes | Yes | NO | Yes | NO |
| **AI capabilities and integrations** | Yes | Yes | NO | NO | Yes |
| **Internal Resource Search** | Yes | NO | NO | Yes | Yes |
| **Social learning and Content creation** | Yes | NO | NO | Yes | Yes |
| **White labeling and Customization** | Yes | Yes | NO | Yes | Yes |
| **Gamification and Certification** | Yes | NO | NO | NO | NO |
| **Managed Cloud-based** | Yes | NO | Yes | Yes | Yes |
| **Kids Safety features** | Yes | NO | NO | NO | Yes |

Table 0: Comparison of current educational platform

As you can see from the table above, our software is made to be modern and more engaging to the students in the learning process, by providing them what they need to succeed. Besides having convenient tools for teachers to stay connected with their students, as well as managing the platform and being scalable for future technologies and advancements.

To learn more about the mentioned platforms head over to the [Appendix section 1.3](#_vnvss645sh4t)

## 1.4 Document Outline

Table below shows the documentation outlines.

| **Chapter** | **Title of Chapter** | **Description** |
| --- | --- | --- |
| 1 | Introduction | Overall idea of the project and what kind of problem it solves and related products. |
| 2 | Project Plan | Project deliverables and the way it’s planned, project risks, costs, and used tools to manage the project. |
| 3 | Requirement Specification | Discusses the functional and non-functional requirements for the project from an in-depth technical point of view. |
| 4 | System Design | Provides system design, high level and low level of the system. |

Table 1: Document Outline

# Chapter 2: Project Plan

## 2.1 Project Deliverables

A list of what we will be delivering with detailed description for each:

| Description | Name |
| --- | --- |
| The source code for the platform and its resources, as well as some code documentation describing the code itself. | Source code |
| A fully detailed documentation about the software and the system design beside an explanatory documentation for the source code. | Documentation |
| Defining the minimum functional and non-functional requirements for the platform, as well as how it works and why. | System requirement |
| Diagrams that describe system flow, use cases and design of our software and its specification, following software engineering principles to showcase requirements and system designs. | Diagrams |
| ER model, relation scheme, and database tables for users, courses, tasks and custom metadata database as well as other related data assets. | Database design |

Table 2: Project deliverables and output.

## 

## 2.2 Project Tasks

| **Analysis** | | | |
| --- | --- | --- | --- |
| Task ID | Name | Description | Duration |
| T1 | Ideas brainstorming | Brainstorming ideas, figure out a problem and its required solution and the possibility to imply in real life. | 5 Days |
| T2 | EduCare Finalization | Finalizing the features of EduCare and deciding what to include. | 4 Days |
| T3 | Stakeholder Interviews | Interviewing Stakeholders (project owners) to get multiple feedbacks and opinions, beside adding most needed features based on their opinions. | 3 Days |
| **Design** | | | |
| T5 | EduCare UI/UX | Design UI/UX prototype that shows and explain project pages | 3 Days |
| T6 | Database design | Database design regarding of the stored data, and its functionality | 3 Days |
| T7 | Relationship with other entities in ER diagrams/ Database design | Identify the relationship between different entities. ex: Add part of relationship between user and entities, Add have relationship between institution and other entities | 7 Days |
| T8 | Design Data-flow diagram | Add multiple data flow diagrams to explain the system flow design and different parts such as Notification DFD, Association Context DFD, Policy Engine DFD …etc. | 10 Days |
| T9 | Design Review | Review the whole system design and get feedback from the instructor and act upon it. | 1 Day |

Table 3: Project tasks and their definitions.

## 

## 2.3 Roles and Responsibilities

| **Team member** | **Responsibility** |
| --- | --- |
| Mohammed Ibrahim Abu-Amara | UI/UX, Frontend, BackEnd, APIs, Database ,Documentation |
| Baha Ayaad | BackEnd, APIs, Documentation, Integration |
| Mohannad Atmeh | BackEnd, APIs, Documentation, Frontend |

Table 4: Project roles and responsibilities

## 

## 2.4 Risk Assessment

### 2.4.1 Risk Identification

| Risk ID | Task ID | Task Name | Risk | Probability | Impact |
| --- | --- | --- | --- | --- | --- |
| Ri1 | T1 | Define The Project’s Scope | Unclear definition of the project’s scope. | Low | Missing some of the important requirements and capabilities, and instead implementing unneeded requirements and being too ambitious. |
| Ri2 | T2 | Design ERD | Inaccurate ERD. | Medium | Can lead to data integrity, poor performance and limited scalability as well as not accomplishing the functional requirements the right way. |
| Ri3 | T3 | Usability Assurance | Hard for users to learn how to use the platform. | Medium | Disables the user from taking full advantage of the platform’s tools, yet could also lead to an increase in the percentage of User Abandonment, and affect the unified model of the platform. |
| Ri4 | T4 | Availability Assurance | Network Issues | Low | Disable users to access the platform or facing issues getting the data. |
| Ri5 | T5 | User Validation Assurance | Using the platform in an  illegal or inappropriate way by some users | Medium-High | Could lead to ban the platform with sanctions in worst cases, deflection of the platform’s goals in best cases. |

Table 5: Project tasks and risk identified with each task and its impact

## 

### 2.4.2 Risk Response

## 

| Risk ID | Risk | Response |
| --- | --- | --- |
| Ri1 | Unclear definition of the project’s scope | Organize brainstorming meetings aimed at redefining the project's scope, identifying conflicting ideas, and establishing a cohesive vision. |
| Ri2 | Inaccurate ERD | Arrange a meeting with the objective of ensuring a well-designed Entity-Relationship Diagram (ERD) prior to the implementation phase, and make necessary updates as required, in order to minimize the likelihood of risks occurring. |
| Ri3 | Hard for users to learn how to use the platform | Create a brief tutorial for new users upon their initial login to the platform, supplemented by a public link containing instructional videos that provide step-by-step guidance on utilizing each tool. |
| Ri4 | Network Issues | Show a message to user that the network is down eg. Error 500 |
| Ri5 | Using the platform in an  illegal or inappropriate way by some users | By incorporating a report button, users will have the ability to report any instances of illegal or inappropriate behavior. EduCare will take immediate action by categorizing the reports and promptly banning any user found to be engaging in such behavior on the platform. |

Table 6: Risk response to the risk assessment associated with the task.

## 

## 

## 

## 2.5 Cost Estimation

Since the development of EduCare platform doesn't require any hardware and utilizes a serverless (pay on demand for what you use) architecture along with the appropriate data stores and API development techniques, the free tier from any cloud provider is more than enough.

The platform considers this as a huge optimization for both users and developers, in addition to making sure, as a start-up, money is only spent when there is a guaranteed return or a small risk associated with it. Moreover, a huge amount of money will be spent on the execution of the AI models that will be deployed; however, using pre-trained models with the correct API structure can come up with a serverless API for AI models.

Below is the analysis for the cost estimations from different cloud providers and how traffic from users will result in an almost guaranteed profit for the platform which will lead to more development and research; resulting in a better platform overall.

| Cloud Provider | Serverless Cost | Domain Name Cost | Data Stores options and costs | AI Models deployment costs |
| --- | --- | --- | --- | --- |
| Amazon Web Services[[4]](https://aws.amazon.com/free/) | $ for every GB-second used (excluding the first 1 million per month) | Starts from 9$ per year and can go up to hundred of dollars per year | Starts from $ per million writes and $ per million reads | Starts from $ per hour of an execution |
| Google Cloud Platform[[5]](https://cloud.google.com/free) | $ per invocation (excluding the first 2 million per month) | Starts from 7$ and can go up to 20$ per year | Serverless Data stores start from $ per million invocation. | Starts from $ per hour of an execution. |
| Microsoft Azure[[6]](https://azure.microsoft.com/en-us/pricing/free-services) | $ for every GB-second used (excluding the first 400,000 per month) | 11.99$ per year | Starts from $ per GB-month | Starts from $ per hour of an execution. |

Table 7: Cost comparison for cloud providers for the platform architecture and requirements.

## 2.6 Project Management Tools

### Google Workspace

Used for real time collaboration and creating documents and other documentation related assets.

### GitHub

Used to track progress for the project using , version control different resources and assets for the project such as diagrams, and for collaborations.

### Draw.io

Used to design and draw diagrams for the project, as well as integration with Google Workspace and GitHub to manage different assets.

**Figma**

Used for UI/UX prototyping and inspiration for the platform design.

# 

# Chapter 3: Requirements Specification

## 3.1 Stakeholders

| **Stakeholder** | **Description** | **Key Interests** | **Impact and Influence** | **Priority** |
| --- | --- | --- | --- | --- |
| Teachers in an Organization. | A user who is part of an institution or an organization focused on creating content for students to acquire knowledge. | Mainly Engaging with students, teachers, institutions, and the platform. Creating and adding material. Managing their students and resources. | Organizational control over course flow and material. Resource permission control. Organizational student management. | High |
| Students in an Organization | A user who is part of an institution or an organization focused on learning and utilizing the platform to acquire knowledge and follow organizational goals for students. | Mainly Engaging with students, teachers, institutions, and the platform.  Making use of resources. Managing their resources. | Control over consumption of course material and engagement and communication (by definition, the number of students will always be greater than teachers) | High |
| Institution Users (not students or teachers) | A user who acts on behalf of an institution or an organization. | Manage permissions. Customize the Organization. Engage with institution users. | Representing an organization. Managing students, teachers, resources, and permissions. | High |
| Teachers not in an organization | A user who is trying to help users and students across the platform by sharing their knowledge and resources, publicly or privately. | Creating personal courses, uploading public resources, communicating with public users and students | Making use of platform resources. Engaging with users. Control over resources and material. | Medium |
| Normal Users (Students not in an organization) | A user who is using the platform for their own personal goals and makes use of educational resources. | Engaging with students, teachers, and institutions. Making use of resources. | Make use of public resources. Engaging with other users. | Medium |

Table 8: Stakeholders definitions, impact, and goals.

## 

Table 9: Stakeholders matrix analysis, which describe the interest and influence each stakeholder has, as well as how they relate to the project plan.

## 3.2 Platform Requirements

### 3.2.1 Software Platform Requirements

EduCare is going to start as a Web-based application, so for users, the required is their browser of choice (Chrome, Firefox, …, etc).

For developers, developing a web-based application, from the client side (front-end), means interacting with web-based frameworks and languages (HTML, CSS, JavaScript, …, etc).

However, for developers developing the server side (back-end); handling business logic and data manipulation, our plan is to utilize the services-oriented architecture to give more flexibility for developers and more nonfunctional requirements for the platform that fits its goals.

Developing small services that have ownership for specific data models also gives isolation and more secure transitions of data across the platform, as well as giving scale in the sense of developing more features and growing the number of requests for a data model. It also gives the abstraction (black box) of the operations behind that service in an interface so developers can use it without thinking of how it works.

Additionally, some services are responsible for the control side of things (Control Plane) and others are responsible for the data side of things (Data Plane), this provides another level of isolation to the system and provide a more secure way for the internal system to control and have ownership over different parts, utilizing the single responsibility principle as well as separation of concern on a large scale and the least privilege principle to make sure users and developers don't have privileges that give them more control than they need.

Nevertheless, services architecture sometimes adds another level of complexity and over-engineering so managing the balance between how small the service is very essential so developers don't spend effort and time to find perfection for the system when a good enough solution might work just fine during a timeline and the flexibility of the system allows developers to do so.

This is where serverless architecture comes into play as well, since most of our services are only doing CRUD operations, this can be managed by a single serverless function for each service, and services that have more than CRUD operations or has the potential to grow, a management solution between a microservice and another serverless function will be done; such that the serverless functions are favorable since they're more managed and simpler to develop.

### 

### 3.2.2 Hardware Platform Requirements

For clients, all they need is a device that has a browser (Laptop, mobile, tablet, …, etc). This means institutions that already have computer labs can easily integrate with the system without any special type of hardware or migration (unless it is a data migration or some optional customization).

For Developers, deploying a service on a server might require a special type of hardware, however, this is not scalable for a service-oriented architecture, each service might need different hardware causing confusion for developers; removing a dimension of flexibility and portability for each service, together with not getting the optimized money-value for each hardware; there might be services that have very expensive hardware but it is not used very often. So the solution is to use containers along with a container orchestration tool, hence, using a cloud-based container solution that would manage most/all of the infrastructure (AWS EKS, GCP GKE, …, etc). As for the development side of things, a laptop or a virtual machine that could run general-purpose languages (Java, C/C++, Go, …, etc) would work perfectly for any developer. Moreover, serverless functions are managed by the cloud providers by default, meaning developers can focus only on the development.

## 

## 3.3 Functional Requirements

| **ID** | **Name** | **Concept** | **Priority** |
| --- | --- | --- | --- |
| [U\_FR\_0](#_cpli9b2wikpe) | Authenticate user | Authenticate a user to the platform | Essential |
| [U\_FR\_1](#_ierkaukret3u) | Modify Profile Details | Edit user information | Essential |
| [C\_FR\_0](#_8bnqbm9u6yy) | Create Course | Add a course to the platform | Essential |
| [C\_FR\_1](#_98xc1xmscukx) | Read Course Details | Read course details to view | Essential |
| [C\_FR\_2](#_7z5yk7kymwgc) | Modify Course Details | Update course details | Essential |
| [C\_FR\_3](#_4khlmyvi00kz) | Delete Course | Delete course | Essential |
| [T\_FR\_0](#_zgr7z9rkxu4v) | Create Task | Create task | Essential |
| [T\_FR\_1](#_17dh0o6aar3b) | Read Task | Read task details | Essential |
| [T\_FR\_2](#_57in7jz91tmg) | Modify Task | Update task details | Essential |
| [T\_FR\_3](#_g0lk6jofasje) | Delete Task | Remove task | Essential |
| [I\_FR\_0](#_tly73ce8subs) | Create Institution | Add institution to the platform | Essential |
| [I\_FR\_1](#_6dz6nqv3g3km) | Read Institution Details | Read institution details | Essential |
| [I\_FR\_2](#_alcb55ts326c) | Modify Institution Details | Modify institution details | Essential |
| [I\_FR\_3](#_1s94sdbg2zw6) | Delete Institution | Remove institution | Essential |
| [R\_FR\_0](#_k6gedna6hmv5) | Create Resource | Add resource to the platform | Essential |
| [R\_FR\_1](#_ucz20cxe9hlw) | Read Resource | Read resource details | Essential |
| [R\_FR\_2](#_vnzaxsd6stos) | Modify Resource | Modify resource details | Essential |
| [R\_FR\_3](#_ljihu69m32iu) | Delete Resource | Remove resource | Essential |
| [N\_FR\_0](#_v9lmhtogmjjn) | Send Notification | Send a Notification for the user. | Essential |
| [P\_FR\_0](#_we41irg2haqx) | Create/Modify Permissions | Create/Modify permissions between any two entities | Essential |
| [P\_FR\_1](#_fwpemf5u57ml) | Validate Permissions | Validate entities permissions | Essential |
| [P\_FR\_2](#_kbibe5bcxyk8) | Populate join data from permissions | Stream process permissions to answer different join data queries | Essential |
| [S\_FR\_0](#_2uvh75b2lltb) | Search for information (Search Engine) | Search a resource or information in the platform | Essential |
| [G\_FR\_0](#_6db8uny209a8) | Generate Entity | Generate questions based on a resource or something similar | Essential |
| [A\_FR\_4](#_vzcklf4gip29) | Get entities connected to the other entities (Query Engine) | Get (users, resources, …, etc) entities that are connected to the other entities (users enrolled in a course, resources for the institution, …, etc) | Essential |

Table 10: EduCare Functional Requirements

### 3.3.1 User Identity Requirements

The user identity functional requirements are responsible for any user operation, this includes any CRUD operations for the user. This requirement is mainly for authentication purposes. Authorization and Accounting is decoupled from the user identity requirement; to provide more isolation between the different data models in the system, e.g. the user model and the permissions model.

#### U\_FR\_0 Authenticate a user

| Function | Authenticating a user into the system |
| --- | --- |
| Description | Adding or logging in a user into the system. This will be done using identity providers rather than storing emails, passwords, and phone numbers (for the OTPs) and other such states; to not risk security issues in the platform and storing private information. But rather just having a user ID and a way to communicate (email address or phone number) and some basic details that they want to be shared with the platform. More in section [3.5.2 Mandatory User Data](#_rqlx6vrwqe81).  If the provider isn't available, passwordless sign-in will be used (sending an email to log in each time and sending an OTP to the phone number). |
| Input | Account from {Google, Microsoft, or supported identity providers} or a valid email address to send the sign-in link to (passwordless login) and a valid phone number to send the OTP to. |
| Output | An error message if the input is invalid (according to the constraints), otherwise redirected to the platform |
| Constraints | Valid email address or a valid phone number or a valid account by one of the supported identity providers |
| Process | A user will open the website to see that they aren't signed in to the platform, so they will try to authenticate, they see a sign in using {Google, Microsoft, …} account so they try it and then they are authenticated.  If they don't have one of those accounts, they will provide an email address and a phone number, for which an email will be sent with a link to sign in, and then an OTP will be sent out if the user existed before in the platform and have their two-factor authentication turned on. |
| Priority | Essential |

Table 11: User authentication functional requirement

#### **U\_FR\_**1 **Edit user profile**

| Function | Modify user profile data |
| --- | --- |
| Description | A user profile describes a user to the public or to a subset of users in the platform, it can contain information such as (but not limited to): a profile picture, name (first, middle, last, username …), gender, phone number, birthday, email (primary, secondary). More on these fields in section [3.5.2 Mandatory User Data](#_rqlx6vrwqe81).  This functionality allows users to modify profile data, which includes modifying the data itself or the privacy of such data. |
| Input | Modified fields and their new value. |
| Output | An error message if any of the input is invalid (according to the constraints), otherwise a success message to the valid modification (modification is done successfully) |
| Constrains | The information which is modified has to be logically and semantically correct, meaning, emails are valid, gender is valid, phone numbers are valid, and so on [3.5.2 Mandatory User Data](#_rqlx6vrwqe81). |
| Process | A user will open their profile, click the modify button, then modify the fields they want to modify, and click save, the valid field will be saved, and the invalid fields will show an error message showing why it is invalid and won't be updated. |
| Priority | Essential |

Table 12: User profile modification functional requirement

### 3.3.2 Course Requirements

A course is a unit of teaching that is associated with material (media, books, .., etc) owned by one or more users and is taught to zero or more users.

#### C\_FR\_0 Create a course

| Function | Create a course |
| --- | --- |
| Description | Add a course to the platform, this is the initial step of any course on the platform. Moreover, a user who creates a course initially owns it. |
| Input | Course information, this includes but is not limited to the course name, description, course logo, and more in the [3.5.3 Mandatory Course Data section](#_wc5rubxj3df6). |
| Output | An error if the course information was not valid (according to constraints), a success and a course page otherwise. |
| Constraints | Valid mandatory and optional fields [3.5.3 Mandatory Course Data section](#_wc5rubxj3df6). |
| Process | A user initiates course creation, fills up a form for the required fields, and a course is added to the platform. |
| Priority | Essential |

Table 13: Course creation functional requirement

#### 

#### C\_FR\_1 Read a course details

| Function | Read the course information |
| --- | --- |
| Description | A user would like to know some information about the course (name, description, …, etc). This functionality allows users, who are authorized to do so, to read this information. |
| Input | Course ID and User ID |
| Output | Course information if the user was authorized to read the course information and the course ID existed. Otherwise, an error indicating the type of invalidation that was done. |
| Constraints | Valid User ID, valid Course ID. |
| Process | Given the User ID and a Course ID, the course will be checked if it existed in the platform and then user permission will be checked to see if this action is valid, then the requested information will be returned. |
| Priority | Essential |

Table 14: Reading Course data functional requirement

#### 

#### C\_FR\_2 Edit course details

| Function | Modify course information |
| --- | --- |
| Description | An owner or an editor for the course would like to modify the course information, this includes the mandatory fields and the optional ones as well. |
| Input | Course ID and Editor ID (and Editor is a user who is an owner or an editor for the course that has a subset of admin permissions that the owner has), and modified fields. |
| Output | Successful modification for the course information if the editor was authorized to modify the course information, the course ID existed, and the modification was valid. Otherwise, an error indicating the type of invalidation that was done. |
| Constrains | Valid User ID, valid Course ID, and valid modification of the fields (name should be a string that follows the name constraints more in the [3.5.3 Mandatory Course Data](#_wc5rubxj3df6)). |
| Process | Given the Editor ID and a Course ID, the course will be checked if it existed in the platform and then editor permission will be checked to see if this modification is authorized. Then the modification will be checked to see if it's valid or not, if all is green, a success will return to the user, otherwise an error will be returned. |
| Priority | Essential |

Table 15: Course data modification functional requirement

#### C\_FR\_3 Delete course

| Function | Remove course from the platform |
| --- | --- |
| Description | An owner of the course would like to remove the course from the platform. |
| Input | Course ID and Owner ID. |
| Output | Successful removal for the course if the owner was authorized to remove the course, the course ID existed. Otherwise, an error indicating the type of invalidation that was done. |
| Constraints | Valid Owner ID, valid Course ID. |
| Process | Given the Owner ID and a Course ID, the course will be checked if it existed in the platform and user permission will be checked to see if this removal is authorized, then removal of the course will be done, otherwise the course stays as is. |
| Priority | Essential |

Table 16: Course deletion functional requirement

### 

### 3.3.3 Task Requirements

A task is an event (an exam, a quiz, an assignment, a literal event, a reminder, a course lecture that needs to be watched, and more). It is responsible for anything that is a piece of work assigned to one or more users and owned by one or more users and has some deadline associated with it.

#### T\_FR\_0 Create a task

| Function | Create a task in the platform |
| --- | --- |
| Description | This functional requirement is responsible for creating the task and initiating its state from the [3.5.4 Mandatory Task Data](#_h6oaw8nejro3) section, the initial user that creates the task is its owner by default. |
| Input | The mandatory task fields and data. |
| Output | A new task if the state was valid according to the constraints, an error showing what is invalid otherwise. |
| constrains | A valid fields according to the [3.5.4 Mandatory Task Data](#_h6oaw8nejro3) |
| Process | A task is created by the user with its field, then a check will be done to see if that task can be created by that user, then the task is added to the platform if all is valid, otherwise an error is returned. |
| Priority | Essential |

Table 17: Task creation functional requirement

#### 

#### T\_FR\_1 Read a task

| Function | Read a task information |
| --- | --- |
| Description | This functional requirement is responsible for reading the task and its state. |
| Input | User ID, Task ID. |
| Output | The task information and state, an error showing what is invalid otherwise. |
| constrains | A valid User ID, a valid Task ID, and permission to read that Task ID using that User ID. |
| Process | The Task ID will be checked if it exists in the platform, and then the permission of the User ID will be checked, if all is successful, the information for that Task ID will be returned to the user. |
| Priority | Essential |

Table 18: Reading Task data functional requirement

#### 

#### T\_FR\_2 Update a task

| Function | Modify a task state |
| --- | --- |
| Description | This functional requirement is responsible for modifying the task and its state. This includes modification from both the owners (modifying information such as name) and the users that are associated with the task (changing the state to done for example). |
| Input | User ID, Task ID, and modification fields. |
| Output | A success if the modification is valid according to the constraints, an error showing what is invalid otherwise. |
| constrains | A valid User ID, a valid Task ID, and permission to modify the fields. |
| Process | The Task ID will be checked if it exists in the platform, then the permission of the User ID will be checked for modification, and then the modification will be checked if it is valid, if all is successful the modification takes place, otherwise an error will be returned. |
| Priority | Essential |

Table 19: Task modification functional requirement

#### T\_FR\_3 Delete a task

| Function | Delete a task |
| --- | --- |
| Description | This functional requirement is responsible for removing the task and its state from the platform. |
| Input | Owner ID, Task ID. |
| Output | A success if the removal is valid according to the constraints, an error showing what is invalid otherwise. |
| constrains | A valid Owner ID, a valid Task ID, and permission to remove the task. |
| Process | The Task ID will be checked if it exists in the platform, and then the permission of the Owner ID will be checked for removal. |
| Priority | Essential |

Table 20: Task deletion functional requirement

### 3.3.4 Institution Requirements

An Institution is an organization or a grouping of users that share some interest, such as universities, boot camps, schools, student clubs, large gatherings of people …, etc.

An institution can be divided into more levels (hierarchical structure); meaning a group can be a parent or a child of other groups and so on, this way the platform manages different permissions and can broadcast information or entities (Courses, Tasks, …, etc) down their chain or in some path in the hierarchy.

#### I\_FR\_0 Create an Institution

| Function | Create an Institution in the platform |
| --- | --- |
| Description | This functional requirement is responsible for creating the institution and initiating its state from the [3.5.5 Mandatory Institution Data](#_yjkfshx964i9) section; the initial user that creates the institution is its owner by default. |
| Input | The mandatory institution fields and data. |
| Output | A new institution if the state was valid according to the constraints, an error showing what is invalid otherwise. |
| constrains | Valid fields according to the [3.5.5 Mandatory Institution Data](#_yjkfshx964i9) |
| Process | An institution is created by the user with its field, then a check will be done to see if that institution can be created by the user, then the institution is added to the platform if all is valid, otherwise an error is returned. |
| Priority | Essential |

Table 21: Institution creation functional requirement

#### I\_FR\_1 Read an Institution

| Function | Read an Institution information |
| --- | --- |
| Description | This functional requirement is responsible for reading the institution and its state. |
| Input | User ID, Institution ID. |
| Output | The institution information and state, an error showing what is invalid otherwise. |
| constrains | A valid user ID, a valid Institution ID, and permission to read that Institution ID using that User ID. |
| Process | The Institution ID will be checked if it exists in the platform, and then the permission of the User ID will be checked, if all is successful, the information will be returned to the user, otherwise an error is returned. |
| Priority | Essential |

Table 22: Reading Institution data functional requirement

#### I\_FR\_2 Update an Institution

| Function | Modify an Institution |
| --- | --- |
| Description | This functional requirement is responsible for modifying the institution and its state. |
| Input | Editor ID, Institution ID, and modification fields. |
| Output | A success if the modification is valid according to the constraints, an error showing what is invalid otherwise. |
| constrains | A valid Editor ID, a valid Institution ID, and permission to modify the fields that Institution ID using that Editor ID. |
| Process | The Institution ID will be checked if it exists in the platform, then the permission of the Editor ID will be checked for modification, and then the modification will be checked if it is valid. |
| Priority | Essential |

Table 23: Institution modification functional requirement

#### I\_FR\_3 Delete an Institution

| Function | Delete an Institution |
| --- | --- |
| Description | This functional requirement is responsible for removing the institution and its state from the platform. |
| Input | Owner ID, Task ID. |
| Output | A success if the removal is valid according to the constraints, an error showing what is invalid otherwise. |
| constrains | A valid owner ID, a valid Institution ID, and permission to remove the institution. |
| Process | The Institution ID will be checked if it exists in the platform, and then the permission of the owner ID will be checked for removal. |
| Priority | Essential |

Table 24: Institution deletion functional requirement

### 

### 3.3.5 Resources Requirements

A resource is any asset in the platform, this includes but is not limited to: Media (pictures, videos, animations, …, etc), Documents (Books, PDFs, Sheets, …, etc), Plugins, …, etc.

Moreover, versioning for resources will be done to make sure that users view old versions and compare them or make use of them even when new types of the same asset are available.

#### R\_FR\_0 Create a Resource

| Function | Create a Resource in the platform |
| --- | --- |
| Description | This functional requirement is responsible for creating the Resource and initiating its state from the [3.5.6 Mandatory Resource Data](#_q1gnlfeaehlv) section. |
| Input | The mandatory resource fields and data, Owner ID |
| Output | A new resource if the state was valid according to the constraints, an error showing what is invalid otherwise. |
| constrains | A valid Owner ID, a valid name and fields according to the [3.5.4 Mandatory Resource Data](#_q1gnlfeaehlv) |
| Process | A resource is created by the user with its field, then a check will be done to see if that resource can be created by the user, then the resource is added to the platform if all is valid, otherwise an error will return. |
| Priority | Essential |

Table 25: Resource creation functional requirement

#### R\_FR\_1 Read a Resource

| Function | Read a resource information |
| --- | --- |
| Description | This functional requirement is responsible for reading the resource and its state. |
| Input | User ID, Resource ID. |
| Output | The resource information and state, an error showing what is invalid otherwise. |
| constrains | A valid user ID, a valid resource ID, and permission to read that resource ID using that user ID. |
| Process | The resource ID will be checked if it exists in the platform, and then the permission of the user ID will be checked, if all is successful, the information for that Resource ID will be returned to the user, otherwise an error will be returned. |
| Priority | Essential |

Table 26: Reading Resource data functional requirement

#### R\_FR\_2 Update a Resource

| Function | Modify Resource state |
| --- | --- |
| Description | This functional requirement is responsible for modifying the Resource and its state, such as updating a document to a new version. |
| Input | Editor ID, Resource ID, and modification fields. |
| Output | A success if the modification is valid according to the constraints, an error showing what is invalid otherwise. |
| constrains | A valid Editor ID, a valid Resource ID, and permission to modify the fields that Resource ID using that Editor ID. |
| Process | The Resource ID will be checked if it exists in the platform, then the permission of the Editor ID will be checked for modification, and then the modification will be checked if it is valid. |
| Priority | Essential |

Table 27: Resource modification functional requirement

#### 

#### R\_FR\_3 Delete a Resource

| Function | Delete a Resource |
| --- | --- |
| Description | This functional requirement is responsible for removing the resource and its state from the platform. |
| Input | Owner ID, Resource ID. |
| Output | A success if the removal is valid according to the constraints, an error showing what is invalid otherwise. |
| constrains | A valid Owner ID, a valid Resource ID, and permission to remove the resource. |
| Process | The Resource ID will be checked if it exists in the platform, and then the permission of the Owner ID will be checked for removal. |
| Priority | Essential |

Table 28: Resource deletion functional requirement

### 

### 

### 3.3.6 Notification Requirements

A notification is a way to notify a user about something according to some trigger, this includes but is not limited to website notification, email notification, SMS notification, and more. Notice that notifications are immutable; once they're sent out to users, they can't change.

#### N\_FR\_0 Send a Notification to a user

| Function | Send a Notification on the platform. |
| --- | --- |
| Description | This functional requirement is responsible for sending the Notification and its state to the user. |
| Input | The mandatory notification fields and data according to [3.5.7 Mandatory Notification Data](#_rr5ijxf18tuh), User ID |
| Output | A new notification will be sent to the user if the state was valid according to the constraints, an error showing what is invalid otherwise. |
| constrains | A valid User ID, a valid name and fields according to the [3.5.7 Mandatory Notification Data](#_rr5ijxf18tuh) |
| Process | A notification is created with its field according to the trigger, then a check will be done to see if that notification can be created and the User ID exists, then the notification will be sent to the user(s) according to the trigger and its type (email, website notification, …, etc) if all is valid, otherwise the notification won't be sent out (this can be done before triggering the notification). |
| Priority | Essential |

Table 29: Sending Notification functional requirement

#### 

### 

### 3.3.7 Policy Engine Requirements

"A policy engine is a way to control who can do what on a platform. This helps to keep the platform secure and efficient. Security, Efficiency, and ensure that a platform complies with regulations"

A Policy Engine is a way to give users permissions to different parts called entities (Institutions, Users, Resources, Courses, Tasks, …, etc) of the platform according to [3.5.8 Mandatory Permissions and Roles](#_a0gv1vsry5gy), this way we maintain a secure platform with zero-trust architecture (validating requests in and out of the platform) with an efficient way to authorize different operations at different stages of the platform.

Moreover, this provides flexibility between different parts of the system, e.g.: institutions can own courses, courses can own resources, …, etc.

More on Policy Engine in Appendix [3.3.7 Policy Engine service](#_t98k7h4k3lma)

#### P\_FR\_0 Create/Modify Permissions between two entities

| Function | Create/Modify Permissions between two entities |
| --- | --- |
| Description | This functional requirement is responsible for linking two different entities and their state (permission) together. |
| Input | EntityA ID, EntityB ID, User ID, Editor ID |
| Output | A success if the modification is valid according to the constraints, an error saying what is invalid otherwise. |
| constrains | A valid User ID, a valid EntityA ID, and a valid EntityB ID, valid permission combinations and fields according to the [3.5.8 Mandatory Permissions and Roles](#_a0gv1vsry5gy). |
| Process | The User ID, EntityA, and EntityB will be checked if it exists in the platform, then the permission of the Editor ID will be checked for modification, then the modification will be checked if it is valid according to [3.5.8 Mandatory Permissions and Roles](#_a0gv1vsry5gy). |
| Priority | Essential |

Table 30: Policy Engine Permissions manipulation functional requirement

#### 

#### P\_FR\_1 Validate Permissions

| Function | Validate Permissions between two entities |
| --- | --- |
| Description | This functional requirement is responsible for validating some permissions between two entities. |
| Input | EntityA ID, EntityB ID, Permissions to Check |
| Output | A true if the permissions are valid, a false otherwise. |
| constrains | valid permission combinations and fields according to the [3.5.8 Mandatory Permissions and Roles](#_a0gv1vsry5gy). |
| Process | The entities IDs will be concatenated, then the permissions bits will be checked for validation according to [3.5.8 Mandatory Permissions and Roles](#_a0gv1vsry5gy). |
| Priority | Essential |

Table 31: Policy Engine Permissions validation functional requirement

#### 

#### P\_FR\_2 Populate join data from permissions

This is a pure internal functionality that will serve query questions like, what users are registered in course A, or how many admins are there in institution B, or what tasks are linked with resource C, and so on.

| Function | Populate join data from permissions |
| --- | --- |
| Description | This functional requirement is responsible for stream processing permissions to be able to populate data into different other data destinations, this is more described in details in [Mandatory Permissions and Roles](#_a0gv1vsry5gy). |
| Input | EntityA ID, EntityB ID |
| Output | populated data based on the Entity types and the queries of the platform. |
| constrains | valid IDs. |
| Process | The entities IDs will be pulled from the stream; after they are put into the Permissions and Roles data store, the type of entity will be pulled as well, then depending on the queries from the platform, different data stores will be populated. |
| Priority | Essential |

Table 32: Policy Engine and Query Engine streaming functional requirement

### 

### 3.3.8 Search Requirements

A search is a way to find information (Media, Text, Documents, Public Information, …, etc) in the platform, this includes public search where entities (Institutions, Resources, Tasks, Courses, …, etc) or results are visible to everyone, and enterprise/private search which includes search in specific entities that aren't public; meaning the platform will index any entity that can be indexed (a public entity by default is indexed unless it is specified otherwise, a private entity by default isn't indexed unless it is specified otherwise).

The privacy of the results is handled by the policy engine automatically (users that do not have permission to an entity won't see it in the result). Moreover, a neural search will be used to understand the semantics of both the search query.

#### S\_FR\_0 Search for information

| Function | Search for information |
| --- | --- |
| Description | This functional requirement is responsible for searching for different entities and their state in a private and public manner. |
| Input | User ID, Search Query |
| Output | A success if the modification is valid according to the constraints, an error saying what is invalid otherwise. |
| constrains | A valid User ID, a valid Search Query ID, a valid permission to view search results according to the [3.5.8 Mandatory Permissions and Roles](#_a0gv1vsry5gy). |
| Process | The User ID will be checked if it exists in the platform, and then the search query will be fed to the platform; that already has an index of information, a ranked result will be returned with a check that the permission of the User ID is valid to view the results. |
| Priority | Essential |

Table 33: Search Engine functional requirement

### 3.3.9 Generator Requirements

A generator is a way to generate different tasks (creating fair exams, assignments, and quizzes, summaries, diagrams, schedules, Text to Speech, …, etc) using Generative AI.

#### G\_FR\_0 Generate Entity

| Function | Generate Entity |
| --- | --- |
| Description | This functional requirement is responsible for generating different entities based on provided resources and their state in a private and public manner. |
| Input | User ID, List of entities, and Generator Mandatory field according to [3.5.9 Mandatory Generator Data](#_jjev1fkizq04). |
| Output | An entity if the generation is valid according to the constraints, an error saying what is invalid otherwise. |
| constrains | A valid User ID, a valid List of entities, a valid mandatory fields according to [3.5.9 Mandatory Generator Data](#_jjev1fkizq04), and permission to generate using the [3.5.8 Mandatory Permissions and Roles](#_a0gv1vsry5gy). |
| Process | The User ID will be checked if it exists in the platform, then the list of entities will be checked if it is valid and a check that the permission of the User ID is valid to generate using the entities the results, the Generative AI will be engaged to generate the type of entity that was requested. |
| Priority | Essential |

Table 34: Generator for entities functional requirement

### 

### 3.3.10 Association Requirement

This functionality is the interface for the policy engine population, [P\_FR\_2](#_kbibe5bcxyk8), where answers about different queries is given, such as:

* Give me the users enrolled in Course A.
* How many students have finished Task B.
* Give me Resources owned by Course B.
* Tasks that have the type as Exam owned by a User A and Course C.
* and more

More on how it work in Appendix [3.3.10 Association service](#_q00r0drfwnby)

#### A\_FR\_4 Get entities connected to the other entities

| Function | Get entities connected to the other entities |
| --- | --- |
| Description | This functional requirement is responsible for answering complex queries that join different data points, utilizing the Policy Engine. |
| Input | Entity A ID, the query |
| Output | query result |
| constrains | A valid Entity ID, a valid query that is already populated in some data store more on populated entities in [3.5.8 Mandatory Permissions and Roles](#_a0gv1vsry5gy). |
| Process | This functionality just connects populated tables with the requested queries, the consistency of the data will be handled by stream process changes from the permissions data store; that describe the relationship between different entities. |
| Priority | Essential |

Table 35: Query Engine functional requirement

### 

## 

## 

## 3.4 Non-Functional Requirements

Non-Functional Requirements for EduCare are listed in order; meaning when developing a feature, trade-offs have to be taken in priority depending on what nonfunctional requirement it gives, removes, or modifies. Managing trade-offs is very critical for a unified system, and understanding why failure happens and when is key to developing resilient applications.

### 3.4.1 Security

Offering a secure platform isn't a privilege for users, but a must, especially in education. Secure in EduCare is the security for data in rest and in transactions (between services or between client and server), including the AAA framework making sure users are authenticated, data is only accessed by authorized users that have permission to do so, and monitoring user behavior over the network; taking accountability for their actions is key to maintain security in a unified platform.

This does not mean restricting users from certain actions, but rather giving users the ability to control their data and what they share (with the platform or with other users), similar to private resources in institutions that are hidden from the public EducCare search engine.

This is achieved by using state-of-the-art mechanisms to store and transfer data (usually managed by cloud providers), isolating the server side and its operations, and making sure user operations are monitored and legit.

Zero trust architecture[[8]](https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-207.pdf) is going to be used in order to manage the multi-tenancy roles and the vast amount of permission combinations, adding another layer of benefits on the security front of the system.

### 3.4.2 Reliability, Availability, and Fault-Tolerance

EduCare is an Internet application so failure is inevitable; networks can crash, storage can be corrupted by mechanical component issues, and other types of collapsing elements in any computing system.

Thus, the system has to find the balance between preventing failures and embracing it, meaning the application has to be robust and resilient, so when a failure happens, users have minimal or no damage, their data is reserved, and no security compromises happen.

One of reasons service oriented architecture was found is to handle failure and to prevent it, so doing it the right way is the first step in reaching this requirement, using containers and a container orchestration tool also manage a big part of it, and finally developing features and changing the system has to be done with caution ensure safety operations that would not break the system. Moreover, this is where serverless functions have an edge over other types of services; it is completely managed, hence, reliability and such factors are also managed by a cloud provider.

### 3.4.3 Scalability and Performance

The service-oriented architecture and serverless functions by definition follow the scale cube principle, and for a unified educational system that is distant to grow in users and features, scale is very important to handle the amount of requests and performance (the response time) in which any operation is done is key for making users life easier.

Maintaining a stable reasonable performance, and being able to scale EduCare, means keeping users happy, which is also important in an educational system, and it reflects on the other nonfunctional requirements mentioned where it is important for the system to have the secure and accurate information during a network crash while also keeping up with the number of requests.

### 3.4.4 Usability, Portability, Ease of use, and Accessibility

This is purely a client side nonfunctional requirement, and giving users an easy to use interface that is unified and convenient to use across different devices and places is the kickoff of maintaining user satisfaction on a platform. Furthermore, an accessibility platform in education is fundamental to make every user feel included, there are over 2 billion people who are disabled[[9]](https://www.who.int/news-room/fact-sheets/detail/disability-and-health), and giving them the power to get educated is core to society and to them.

### 3.4.5 Code quality and Maintainability

Developers are the first users of any system, and are the first point of contact when modifying or adding a new feature, so giving the ability for the system to have above average code quality is going to make developers lives much easier.

Besides, having a maintainable system means tracking changes and finding issues much easier, scaling features and the whole system effortlessly, and adding more developers to the system easier; easier to understand and get on board.

### 3.4.6 Customization and Localization

A unified system does not mean one size fits all, but a dynamic size that gives users the ability to control many aspects of the system, such as colors, languages, and how certain permissions behave. Additionally, giving power for different communities the ability to integrate or migrate to a system smoothly, and getting users to use it simple and nearly without training.

## 

## 3.5 Other Requirements

### 3.5.1 Data formats

LMS systems have standards in file types and formats support, especially for video and for tracking and recording learning experiences. So EduCare has to make it clear that it is not replacing any standard any time soon, but rather enhancing them; this is due to the fact that many won't be able to migrate without these standards.

Moreover, there might be cases where the optimized solution is to give an interface to users or other developers outside the system, and have one unified data format for the system, to make it easier and more unified in terms of backend data manipulation; the user can use any format they want but the underlying infrastructure and data formats in the system is just parsing that input and converting it to a more unified version. This has many advantages, some of them are having other services consume it in a normal format rather than formatting it each time, integrating with other information extraction services or feeding it to some AI model, and many more data operations where it does not necessarily follow the LMS format.[[10](https://www.ijte.net/index.php/ijte/article/view/36)]

### 

### 3.5.2 Mandatory User Data

User data has to be handled with care, this means the platform should aim to use as little information as possible about its users to maintain their privacy, however, this does not mean any data from users but use the necessary information to be able to provide a functional platform for users and help deliver the best experience for a diverse set of users. Below is a Table showing these data points, what is optional, and some constraints.

| User ID | Mandatory, Unique, Auto-Generated by the platform, and used to identify the user across the platform. |
| --- | --- |
| Email address or phone number | Mandatory, Unique, At least one of the email address or phone number should be provided, Provided by the user, and used for communication between the platform and the user. |
| Name | Mandatory, Provided by the user, used to display the user across the platform, should be a valid name (all alphabets no symbols or numerical values). |
| Birth date | Mandatory, Provided by the user, used to provide an appropriate experience for all age groups, should be a valid date (DAY/MONTH/YEAR or some other format). |
| Gender | Optional, Provided by the user. |
| Photo | Optional, Provided by the user. |
| Additional Email address or phone number | Optional, Provided by the user. |

Table 35: Data Dictionary for user data model

### 

### 3.5.3 Mandatory Course Data

Course data is data that is used to provide the best learning experience for users, from smooth onboarding to taking final exams. Communication between teachers and students is vital as well, so showing a clear path for both to be able to provide feedback and understand the goals of the course.

| Course ID | Mandatory, Unique, Auto-Generated by the platform, and used to identify the course across the platform. |
| --- | --- |
| Owner ID | Mandatory, List of one or more User IDs that own the course, the initial creator of the course is the first owner of the course by default, ownership can be transferred or have multiple ownership, Unique Set of IDs. |
| Name | Mandatory, Provided by the creator of the course, valid name (it can have numbers or symbols) |
| Created Date | Mandatory, Provided and Generated by the platform. |
| Version Number | Mandatory, Provided by the platform, a course can be created from another course or updated; in which the newly created course has a higher version value than the parent course. |
| Editors ID | Mandatory, List of zero or more User IDs that can do some admin functions but not all (not full ownership) such as the removal of the course or some core modification, Unique Set of IDs, can't be owner and editor at the same time (there is no intersection between this list and the Owner ID list), an owner has editor privileges by default but an editor does not have owner privileges. |
| Users ID | Mandatory, List of zero or more User ID that does not have any admin functions, Unique set of IDs, an owner or an editor can be a student as well, but should not be part of this list to avoid redundancy. |
| Type | Optional, selected from a predefined categorical type or provided by the owner. |

Table 36: Data Dictionary for course data model

### 

### 3.5.4 Mandatory Task Data

Tasks are core for interactions between the platform, the students, and the teachers. Thus, it's important to provide an easy-to-use data model for better interactions, especially since tasks have different types but follow the same abstract idea, where there is some job that should be done at some deadline.

| Task ID | Mandatory, Unique, Auto-Generated by the platform, and used to identify the task across the platform. |
| --- | --- |
| Owner ID | Mandatory, List of one or more User IDs that own the task, the initial creator of the course is the first owner of the task by default, ownership can be transferred or have multiple ownership, Unique Set of IDs. |
| Name | Mandatory, Provided by the creator of the task, valid name (it can have numbers or symbols) |
| Created Date | Mandatory, Provided and Generated by the platform. |
| Type | Mandatory, selected from a predefined categorical type (Quiz, Exam, Assignment, Event, Reminder, …, etc). |
| State | Mandatory, one of the following (No Status, Todo, In Progress, In Review, Done) |
| Deadline | Mandatory, valid date where the state of the task can't change afterwards, there can be an infinite date in which the task is open forever. |
| Version Number | Mandatory, Provided by the platform, a task can be created from another task or updated; in which the newly created task has a higher version value than the parent task. |
| Editors ID | Mandatory, List of zero or more User IDs that can do some admin functions but not all (not full ownership) such as the removal of the task or some core modification, Unique Set of IDs, can't be owner and editor at the same time, an owner has editor privileges by default but an editor does not have owner privileges. |
| Users ID | Mandatory, List of zero or more User ID that does not have any admin functions, Unique set of IDs, an owner or an editor can be a student as well, but should not be part of this list to avoid redundancy. |

Table 37: Data Dictionary for Task data model

### 

### N3.5.5 Mandatory Institution Data

An institution is an organization of users that share some goal, meaning it is a grouping of users and other assets in the platform, moreover, these groupings can be hierarchical (Graph) as well (institution inside of institutions), this way each level can have different permission that is automated for it, and different levels have different views. Moreover, this manages communication between the organization and students and everything in between with ease.

| Institution ID | Mandatory, Unique, Auto-Generated by the platform, and used to identify the task across the platform. |
| --- | --- |
| Owner ID | Mandatory, List of one or more User IDs that own the task, the initial creator of the course is the first owner of the task by default, ownership can be transferred or have multiple ownership, Unique Set of IDs. |
| Name | Mandatory, Provided by the creator of the task, valid name (it can have numbers or symbols) |
| Created Date | Mandatory, Provided and Generated by the platform. |
| Parent Institution | Optional, List of zero or more Institution IDs that are parents of the institution in the hierarchical structure (e.g: a computer science department is considered a parent of the computer science specialization), adding an institution shouldn't result in a cycle in the overall structure. |
| Children Institution | Optional, List of zero or more Institution IDs that are children of the institution in the hierarchical structure (e.g: a computer science department is considered a child of the Information Technology department), adding an institution shouldn't result in a cycle in the overall structure. |
| Editors ID | Mandatory, List of zero or more User IDs that can do some admin functions but not all (not full ownership) such as the removal of the task or some core modification, Unique Set of IDs, can't be owner and editor at the same time, an owner has editor privileges by default but an editor does not have owner privileges. |
| Users ID | Mandatory, List of zero or more User ID that does not have any admin functions, Unique set of IDs, an owner or an editor can be a student as well, but should not be part of this list to avoid redundancy. |
| Logo | Optional, Picture provided by an Owner, used to customize the institution. |
| Color Palette | Optional, Colors provided by an Owner, used to customize the institution |
| Information | Optional, data related to the institution such as website, socials, established date, …, etc. Similar to profile data. |

Table 38: Data Dictionary for Institution data model

### 

### 3.5.6 Mandatory Resource Data

A resource is a way to provide more assets to the platform, this includes different types of media, documents, excel files, and more. Furthermore, it can be used to provide a more rich environment for students, and more communication and interaction methods through these assets.

| Resource ID | Mandatory, Unique, Auto-Generated by the platform, and used to identify the resource across the platform. |
| --- | --- |
| Owner ID | Mandatory, List of one or more User IDs that own the resource, the initial creator of the resource is the first owner of the resource by default, ownership can be transferred or have multiple ownership, Unique Set of IDs. |
| Name | Mandatory, Provided by the creator of the resource, valid name (it can have numbers or symbols) |
| Created Date | Mandatory, Provided and Generated by the platform. |
| Size | Mandatory, Provided and Computed by the platform. |
| Version Number | Mandatory, Provided by the platform, a resource can be created from another course or updated; in which the newly created resource has a higher version value than the parent course. |
| Editors ID | Mandatory, List of zero or more User IDs that can do some admin functions but not all (not full ownership) such as the removal of the resource or some core modification, Unique Set of IDs, can't be owner and editor at the same time, an owner has editor privileges by default but an editor does not have owner privileges. |
| Users ID | Mandatory, List of zero or more User ID that does not have any admin functions, Unique set of IDs, an owner or an editor can be a student as well, but should not be part of this list to avoid redundancy. |

Table 39: Data Dictionary for Resource data model

### 3.5.7 Mandatory Notification Data

Notifications are an easy way to give a heads-up to users, it is also a great communication method between different connections in the platform, and a notification should be abstract, simple, easy to understand for users, hence its data field should follow the same principle as well.

| Notification ID | Mandatory, Unique, Auto-Generated by the platform, and used to identify the notification across the platform. |
| --- | --- |
| Owner ID | Mandatory, a User ID that owns the notification, the initial creator of the notification is the first owner of the notification by default. |
| Name | Mandatory, Provided by the creator of the resource, valid name (it can have numbers or symbols) |
| Created Date | Mandatory, Provided and Generated by the platform. |
| Content | Mandatory, Provided by the user, can contain text, images, …, etc. |
| Sent Date | Mandatory, provided by the owner, when to send the notification. |
| User IDs | Mandatory, a List of one or more User IDs to send the notification to. |
| Method | Mandatory, selected from one of the following (Email or phone number, website notification) |

Table 40: Data Dictionary for Notification data model

### 

### 3.5.8 Mandatory Permissions and Roles

This is the core of the privacy and information security in the platform, thus it needs to be as perfect as possible to avoid flaws and have resiliency, but simple enough to maintain a great experience for users and a fast processing time for other internal services that might use it.

The permission between any two entities (say A and B) will be defined by permissions bits (similar to Unix), there will be a default permission set for simple functionality for some public entities, similarly, for private entities, there will be a defined set of permissions by the owner of the entity, e.g. all users will have read access for public entities and read and write/modification access to their own entities.

There will be 8 permission bits, each (from the Least Significant Bit) responsible for the following: Read, Write, Modify Level C, Modify Level B, and Modify Level A, and extra 3 bits are for scalability reasons.

Each level of Modification provides different ways for ownership to manage the entity, with Level A being the most privileged modification level (full ownership) and C being the lowest (normal user).

The data will be saved as a <key, value> pair, this way the key will be the pair of EntityA and EntityB, and the value is the permission bits.

| ID | Mandatory, an ID of Entity A concatenated with an ID of Entity B. |
| --- | --- |
| Permission Bits | Mandatory, 8 bits of permission. |

Table 41: Data Dictionary for Permissions data model

Different Permissions combinations are possible between entities, however, there are some constraints, some of which are:

* The entities can't be both users.
* Entity A, Entity B is not the same as Entity B, Entity A (not transitive)
* When an entity such as an Institution, a Task, a Course, a Resource, or similar is Entity A, this means it becomes part of the Owners/Editors Lists for Entity B; owners and/or editors of Entity A become part of owners/editors of Entity B.
* Modify Level A includes every modification permission (Deletion, Editing core fields, adding or removing other entities, control over the owners and the editors list, …, etc) indicating ownership, where Modify Level B contains only a subset of those permissions indicating editorship, Level C is a user level having a subset of the the editorship permissions.
* Entities without owners can't be paired, having permissions between each other such as search and generators for example doesn't provide any secure value, meaning there is no ownership over some search result or generator results but rather the entities that these entities use, hence there won't be such relationships.
* The default for public entities (Resources, Tasks, Course, Public User Information, Public Institution Information, …, etc) is that they are viewed only unless specified otherwise by one of the owners.
* The policy engine will be composed of a control plane; managing permissions and answering security queries such as whether Entity A has read access over Entity B. And a data plane that will manage populating other tables for joined data for answering data queries such as what courses belong to institution A.
* When an entity is connected with another entity such that none of the two entities is a user, then a disjoint set (union-find) will be used to figure out the permissions of users that are also connected with that entity. Moreover, it is not advised to connect entities that way; to make sure permissions are explicitly defined.

### 

### 3.5.9 Mandatory Generator Data

A generator is one of the creative minds that are in the platform, it can create questions from a set of resources, or some schedule for a user with a set of tasks. So, it needs to be simple enough for users to use.

| Generator ID | Mandatory, Unique, Generated by combining resources ID and hashing them, and used to identify the generator across the platform. |
| --- | --- |
| Owner ID | Mandatory, a list of one or more User IDs that owns the generator result, the initial creator of the generator result is the first owner of the result by default. |
| Name | Mandatory, Provided by the creator, valid name (it can have numbers or symbols) |
| Created Date | Mandatory, Provided and Generated by the platform. |
| Type | Mandatory, Provided by the user, select from one of the following (questions, diagrams, summaries, text2speech, course recommendations) |
| Users IDs | Mandatory, a List of one or more User IDs that can use the results of the generator. |

Table 42: Data Dictionary for Generator data model

### 

### 3.5.10 Mandatory Metadata data

A metadata is just extra data about other data, in our case that is something like white labeling and branding for an institution for example, this is separated from the entity itself to be able to have these different branding spread out to different part of the entity, following our example, the course that is in the institution for example.

| Metadata ID | Mandatory, Unique, Auto-Generated by the platform, and used to identify the metadata across the platform. |
| --- | --- |
| Owner ID | Mandatory, a list of one or more User IDs that owns the metadata, the initial creator of the metadata result is the first owner of the result by default. |
| Websites | Optional, a list of websites (socials, other related websites to the entity, …, etc). |
| Picture | Optional, an image related to the entity. |
| Color palette | Optional, a set of colors related to the entity, these can be from 6 to 10 colors. |

Table 43: Data Dictionary for Metadata data model

### 

### 3.5.11 Auto Generated ID Data

An ID will be used to identify an entity in the platform; that is a user, a resource, an internal data model, and others, thus it needs to be unique by default and it can also be used to identify the type of entity; to ease other operations in the platform e.g. policy engine data population.

The process of generating an ID has been researched for a long time, moreover, randomization has shown to be effective such as UUIDs. Furthermore, the platform will use the 128 bits of the UUIDv4 prefixed with 12 bits describing the type of the entity, resulting in 140 bits. The type will be predefined before prefixing the UUID.

# **Chapter 4**: **System Design**

## 4.1 Logical Model Design

A structured approach is used in our case; EduCare services are serverless functions that are completely stateless, this is to maintain security and performance.

Moreover, our data might be connected in an object oriented way but the data model is completely separated from other application models, the association between these data is returned by the platform query engine [3.3.10](#_vzcklf4gip29) supported by the policy, in which both are also stateless and function oriented (both have there control layer and data layer separated), hence, choosing a structured approach will follow our principles in the non-functional requirements as well as the overall system architecture for the platform which results in decoupling services and maintaining a better codebase for future use cases and scalability.

### 

### Structured Approach:

#### Functional Decomposition Diagram (FDD).

Each of the FDD describes a way to understand how different tasks, and subtasks, are gonna be done in the platform, as well as how the overall structure of the platform is in terms of functions insteads of objects; relationships are represented in the data layer of the platform and the functions are on the control layer (fully stateless and separated from the data).

The documentation will separate each diagram based on the data model first; each control model will control a single data mode. Thus, the intersection between all diagrams results in the full platform FDD.

##### 

##### User FDD

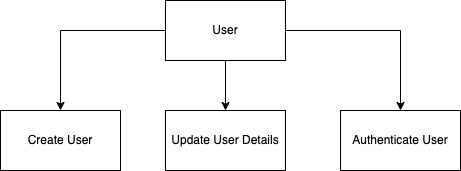


Figure 0: User FDD

##### 

##### Course FDD



Figure 1: Course FDD

##### 

##### Task FDD



Figure 2: Task FDD

##### 

##### Institution FDD

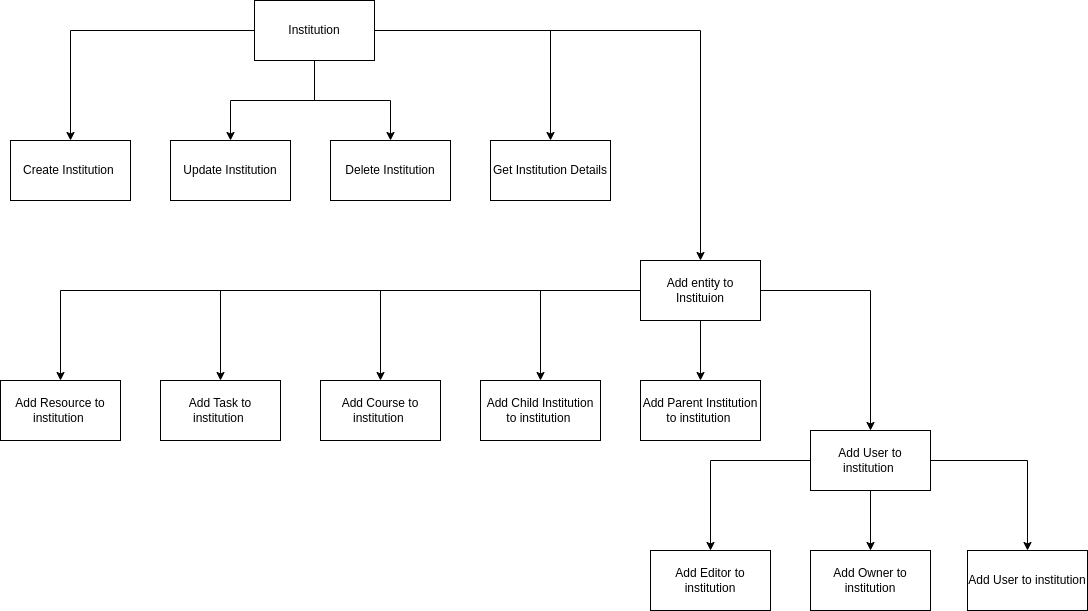


Figure 3: Institution FDD

##### Resource FDD

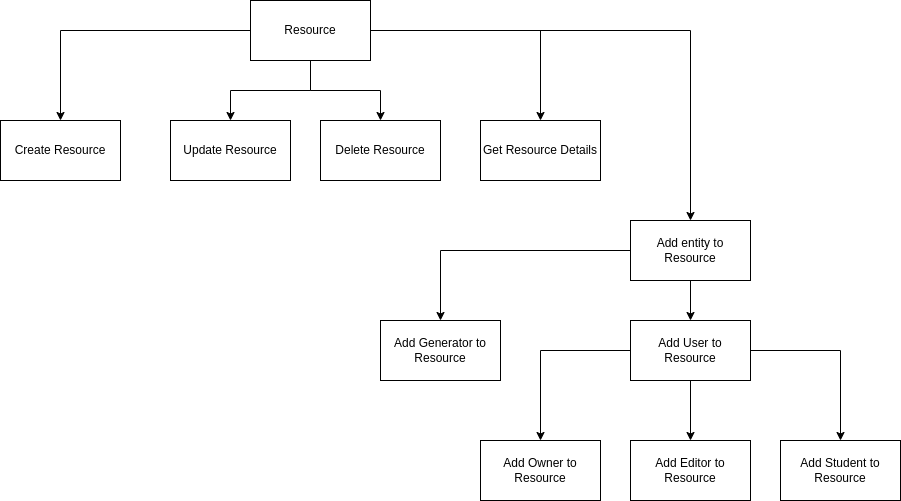


Figure 4: Resource FDD

##### Notification FDD

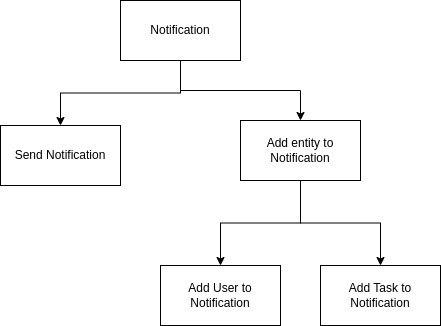


Figure 5: Notification FDD

##### Policy Engine FDD

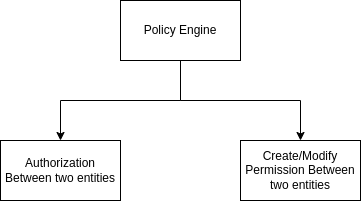


Figure 6: Policy Engine FDD

##### Search FDD

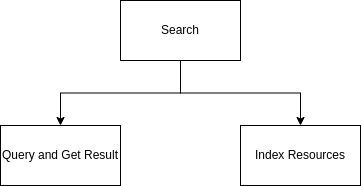


Figure 7: Search FDD

##### Generator FDD

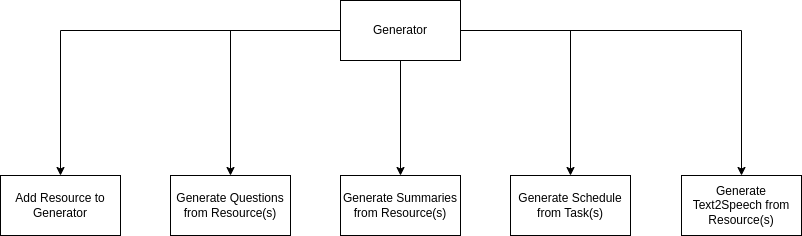


Figure 8: Generator FDD

##### Association FDD

##### 

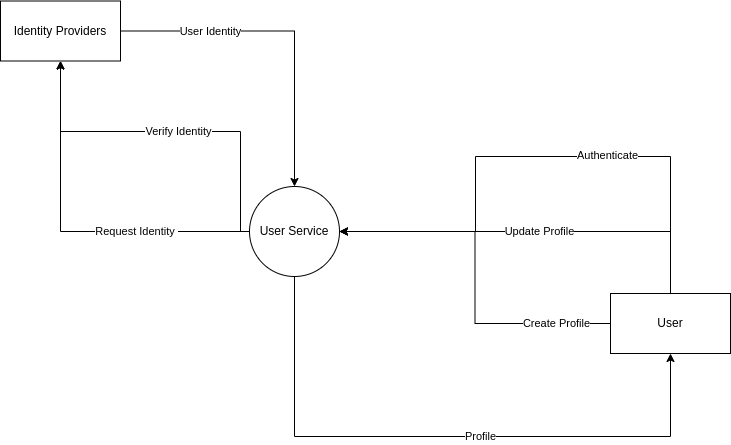
Figure 9: Association FDD

##### 

#### 

#### System Context Diagram (Context DFD).

##### User System Context Diagram

Figure 10: User System Context Diagram

##### 

##### Course System Context Diagram

##### 

Figure 11: Course System Context Diagram

##### 

##### Task System Context Diagram

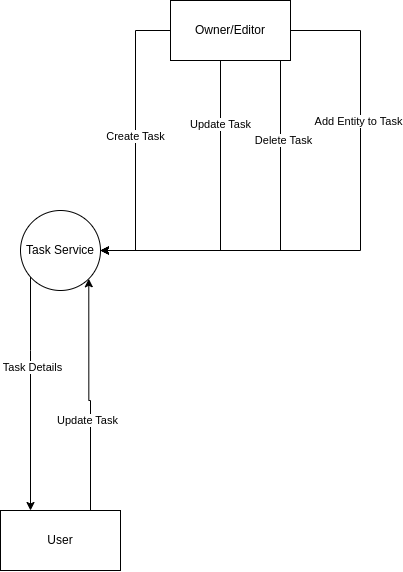


Figure 12: Task System Context Diagram

##### 

##### Institution System Context Diagram

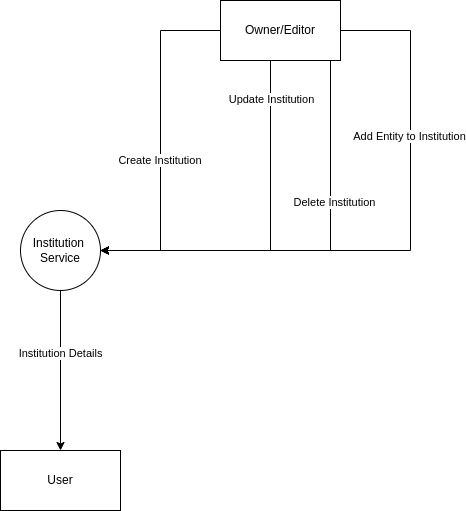


Figure 13: Institution System Context Diagram

##### Resource System Context Diagram

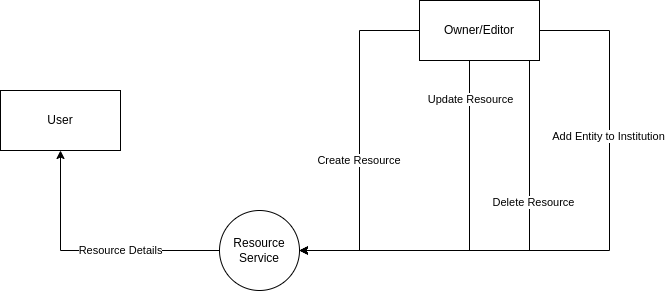


Figure 14: Resource System Context Diagram

##### 

##### Notification Context DFD

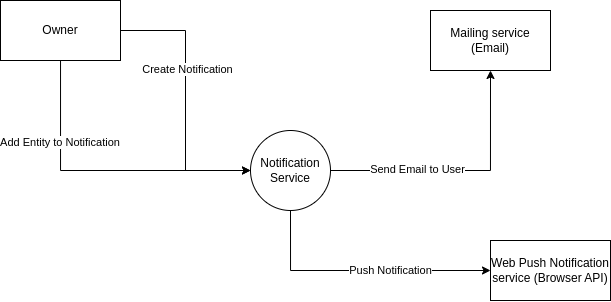


Figure 15: Notification System Context Diagram

##### 

##### Policy Engine System Context Diagram

This is a sample of the system context diagram, the full diagram would not fit the document, but it is the same idea.

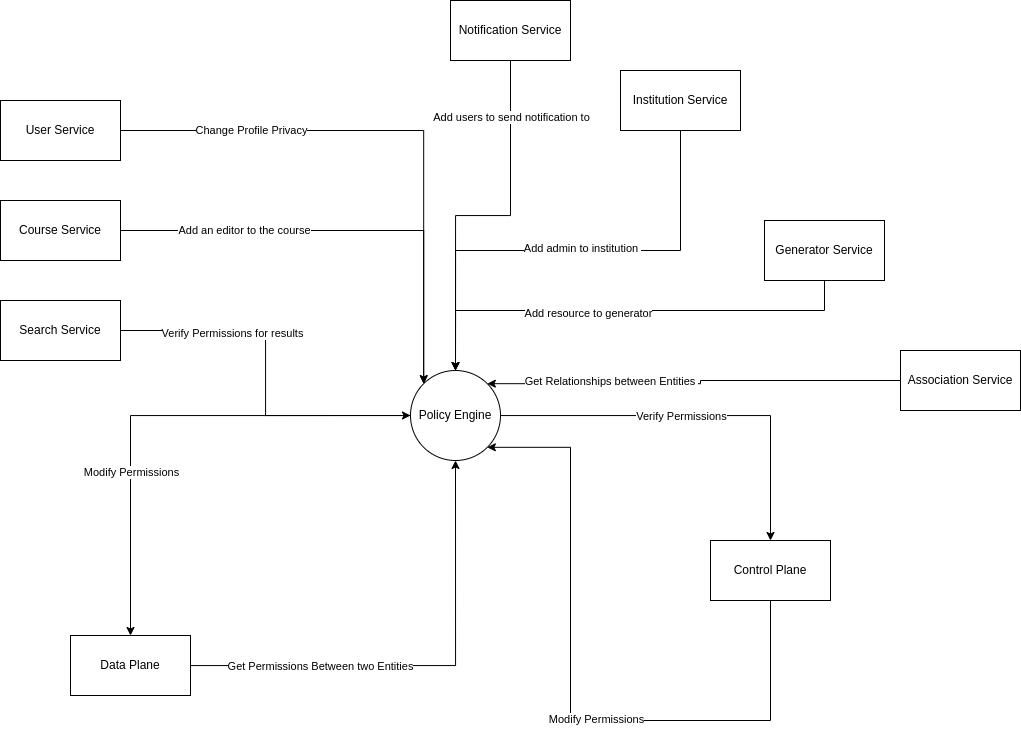


Figure 16: Policy Engine System Context Diagram

##### Search System Context Diagram

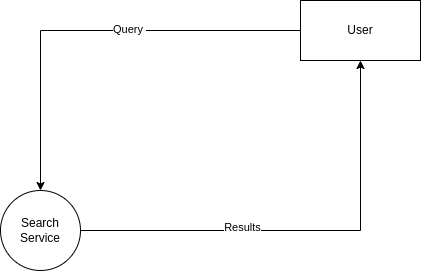


Figure 17: Search System Context Diagram

##### Generator System Context Diagram

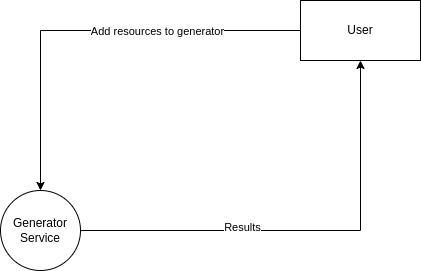


Figure 18: Generator System Context Diagram

##### 

##### Association System Context Diagram

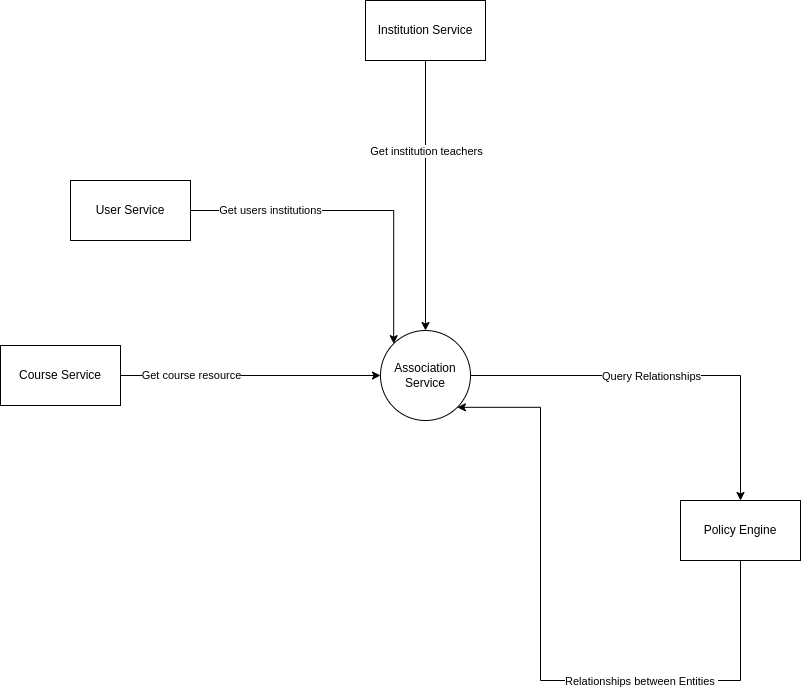


Figure 19: Association System Context Diagram

### Entity Relationship Diagrams (ERD)

Relationships will be mentioned under the ER diagram due to the complexity of the Diagram.

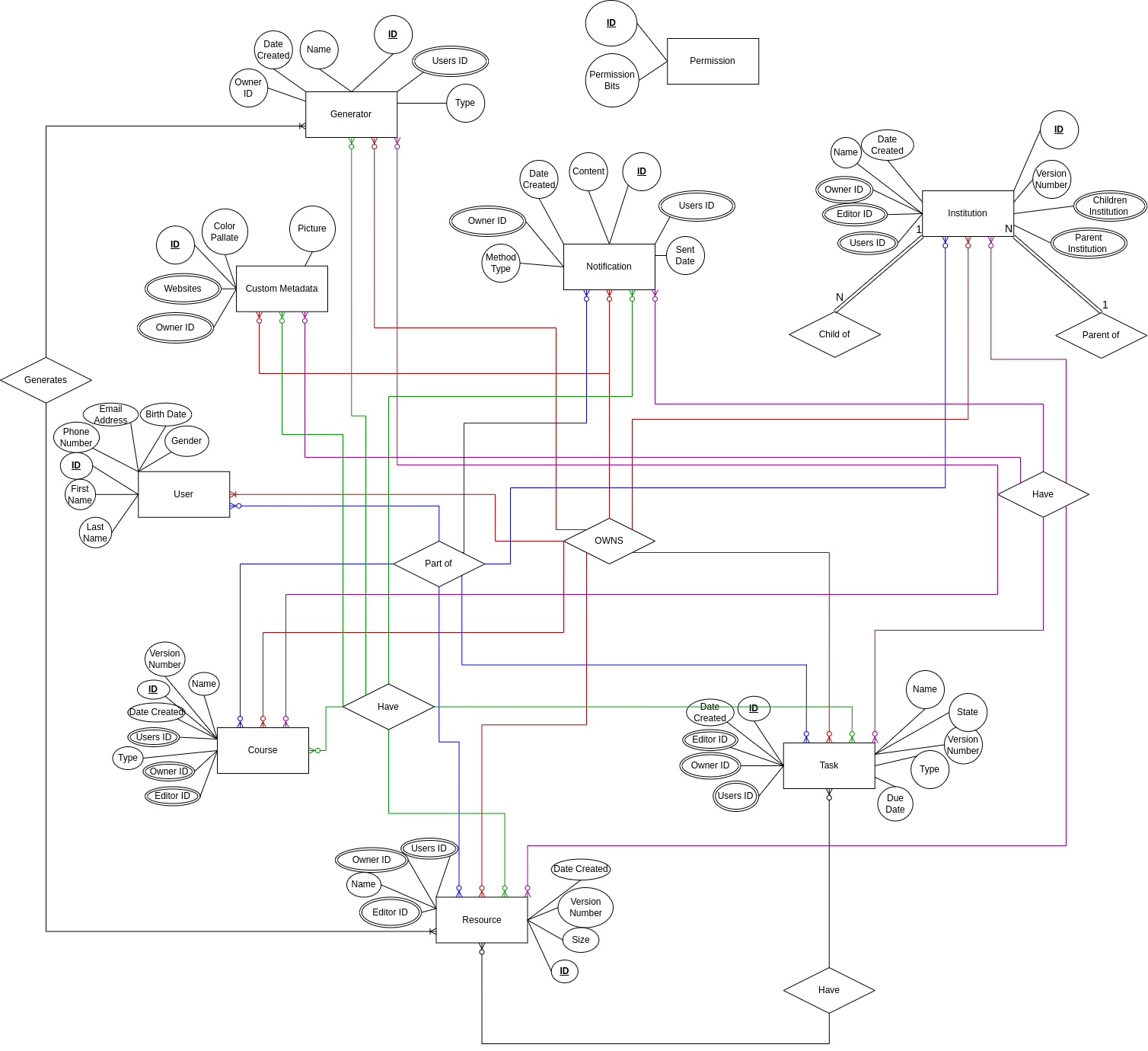


Figure 20: EduCare ER Diagram with relationships

Notice that the search service does not have a database, as the search engine will index the data in its own model and pull the data from the Resource database, and for the association system, it is a query system rather than a data model, hence it is stateless in some form and is not an entity.

## 4.2 Physical Model Design

### User Interface Design

##### Login Page Interface

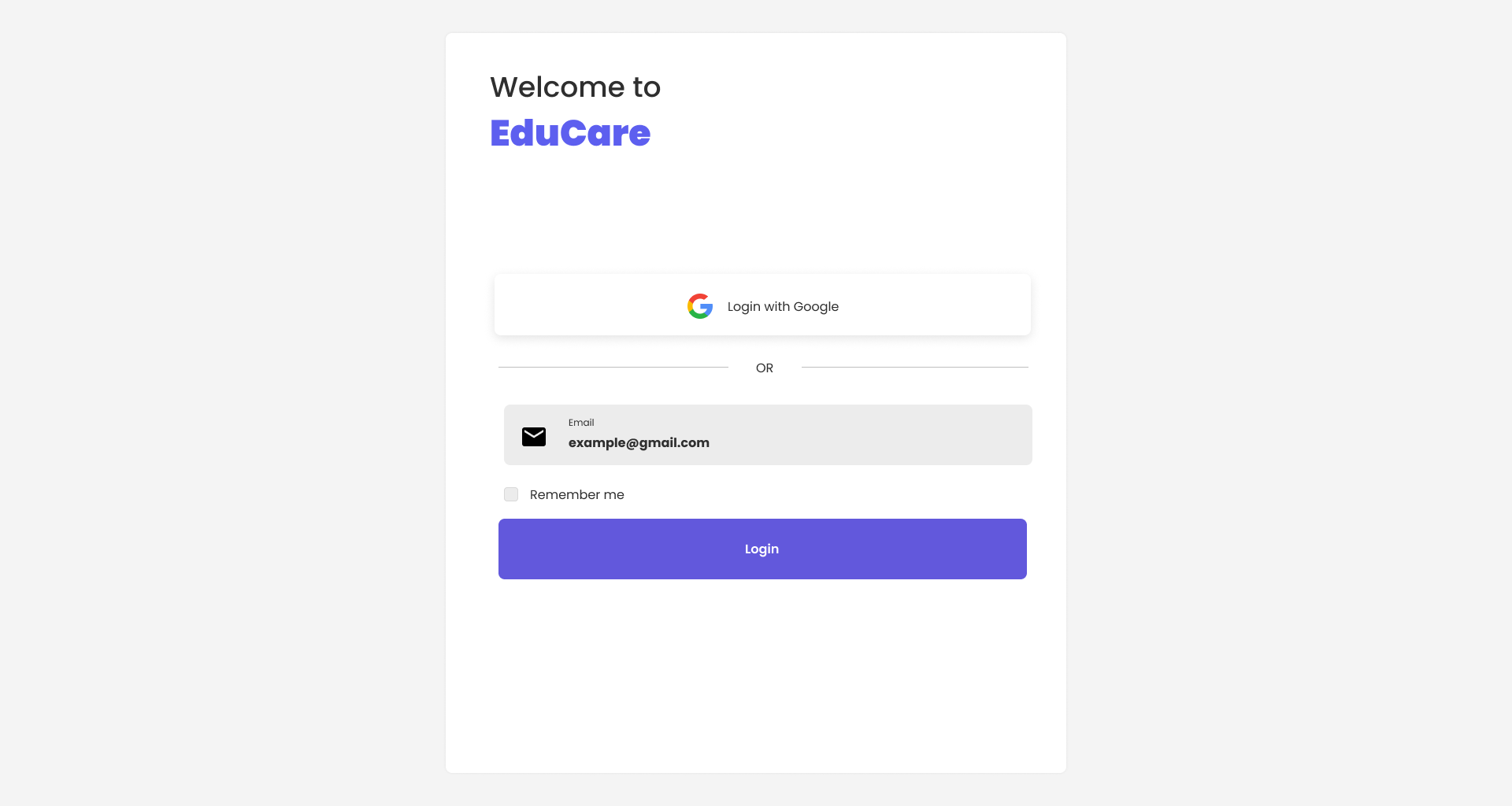
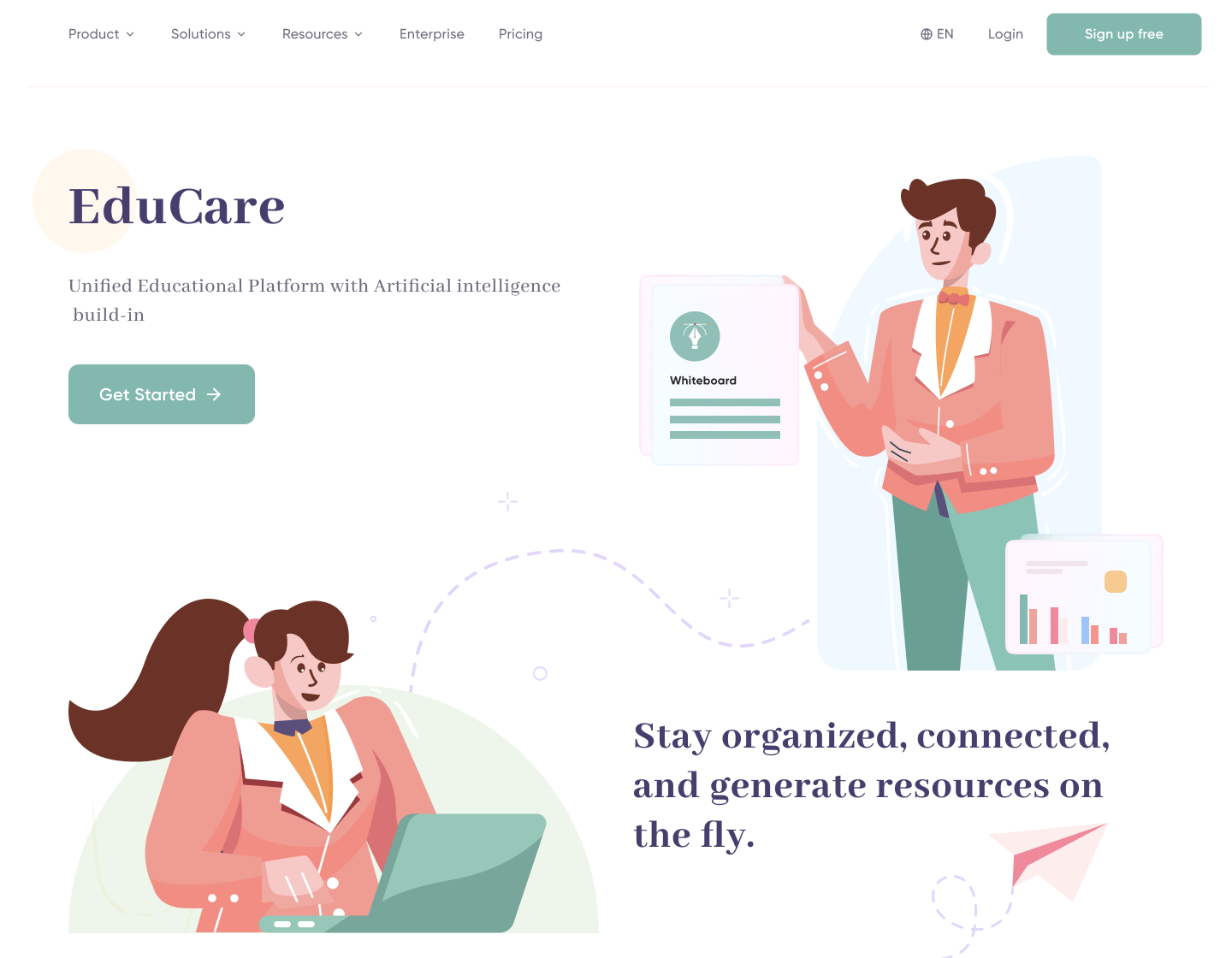


Figure 21: EduCare Login User Interface

##### 

##### Landing Page Interface

Figure 22: EduCare Landing Page Interface

##### 

##### Course Interface

Figure 23: EduCare Course User Interface

### 

### Database Design

Since we are using a service-oriented architecture, databases are very isolated, the data ownership principle states that each service should own its own database without any coupling with other databases, “**don't communicate by sharing memory, share memory by communicating**”, in addition to that, the platform will make use of GraphQL which is a query and data manipulation language for APIs; meaning the platform API will only return what it is asked for (unlike REST APIs, GraphQL would provide a single endpoint to access a combination of resources and data manipulation operations), and most of the database will only have one table in it, making it perfect for the serverless use case.

##### 

##### User Database

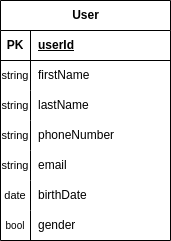


Figure 24: User table database design

##### Course Database

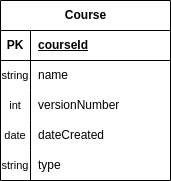


Figure 25: Course table database design

##### 

##### Task Database

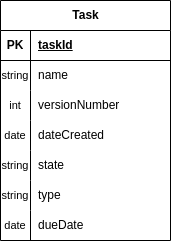


Figure 26: Task table database design

##### Institution Database

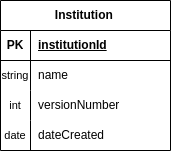


Figure 27: Institution table database design

##### 

##### Resource Database

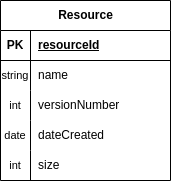


Figure 28: Resource table database design

##### Notification Database

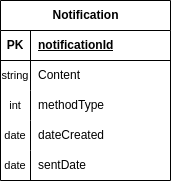


Figure 29: Notification table database design

##### Custom Metadata Database

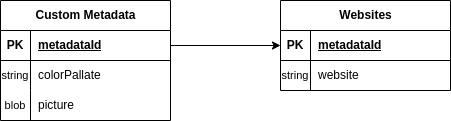


Figure 30: Custom Metadata table database design

##### Generator Database

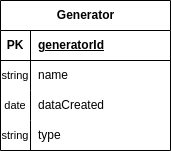


Figure 31: Generator table database design

##### Policy Engine Database



Figure 32: Generator table database design

##### 

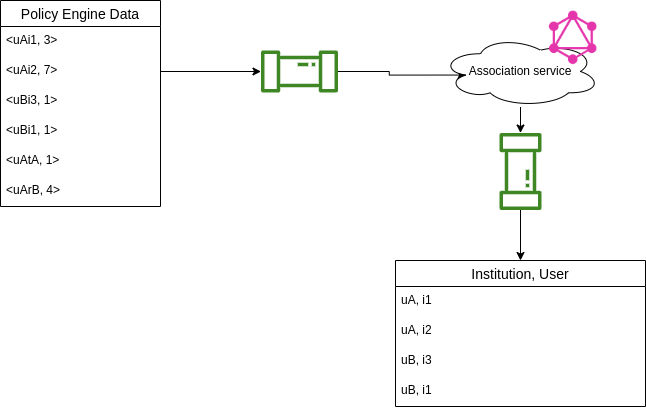
##### Association service database

Notice that the association service does not have a database, but rather materialized views, that may or may not be stored in a data store, and are created from some transformations to the policy engine data. An example for materialized views in table form is “what institutions is user A part of (admin, student, teacher, …, etc)” and it is done in the following way:

Let’s assume the following

* User A ID be uA
* Institution 1, 2, 3 IDs be i1, i2, i3 respectively
* The key value pair be denoted by <key, value>
* A row of data is shown as comma separated.

then the materialized view is built in the following way from the policy engine data:

Figure 33: Association service table database design and process

More on how the service works in [Appendix 3.3.10 Association service](#_q00r0drfwnby)

# 

# Chapter 5: Implementation

In this chapter the Implementation for the backend and the frontend for the website will be discussed in more details. In addition to that, an overview about some of the web pages will be given.

## 5.1 Programming Languages, Tools, API

In this section different aspects of implementation will be discussed, from the programming languages, frameworks, and different implementation decisions. Furthermore, the architecture of the backend will be talked through from a high level view and decisions taken will be more detailed. For the frontend, the frameworks chosen as well as the components built will be talked through in more details.

## 5.1.1 Front-End

This section will discuss the chosen technology and the reason behind choosing them as well as how they were utilized.

#### React

React is an open source library from Facebook by an engineer called Jordan Walke in 2013 as a way to build an interface that is dynamic and responsive, and it aims to provide a single-page web application richer user experience with faster load times and smoother interactions. It is mainly a JavaScript library, but it can be used with TypeScript as well, moreover, JavaScript and TypeScript were used in order to speed up the process of rendering inside the browser since most browsers use JavaScript as the main rendering engine.

The main features of react is having something called the virtual DOM, the regular DOM which exist in every web page as a way to render the HTML elements using a tree structure and a way for the browser to know which element to render first, however, the limitation for this is that the tree needs to be build each time an update happen inside the page, so for example if one character changes inside some profile web page for example, the whole page needs to be reloaded and the DOM tree has to be rebuild, this is where the Virtual DOM comes into place, the virtual DOM make sure to only re-render the necessary elements that changes, this is done very efficiently and is currently used in facebook website as a way to keep the news feed as fresh as possible. Furthermore, this enables the developers to build highly efficient web applications using low memory and processing time. This virtual DOM is usually managed using states, these states can be updated reactively and handled with ease providing an easy way for developers to manage data displayed on the page.

The other main feature for react is that it is a Component-based library, meaning in order to build a web page, you can divide the web page into components and render each component separately, as well as maintain it separately, this improves both performance and making sure the code is clean and maintainable.

#### Chakra-ui

Chakra-ui is an open source library for building React components using best User Experience practice. It also has an internal library for icons, custom CSS designs, as well as strictly following the WAI-ARIA for accessibility. In addition to that, chakra provides a better developer experience for writing react applications.

## 5.1.1 Back-End

This section discusses the technologies chosen and the reason behind them, as well as how they were utilized and used to build different features in the website.

### JavaScript

JavaScript, sometimes called the core technology of the Web, is a lightweight programming language that is considered a just-in-time compiled language due to its uses in browsers and web applications. Its syntax is based on the Java and C languages. Moreover, JavaScript views functions as first-class citizens and considers objects as prototypes meaning they are expressed in as functions rather than classes internally, this all provide a fertile space for web developers to focus on the functionality for the web page rather than managing objects and such things.

### TypeScript

Typescript is a programming language that solves the shortcomings of JavaScript, mainly providing a way to statically type variables and objects, as well as trying to provide better memory management, this is the main reason why Microsoft created TypeScript, to provide a way to create large scale web application without using too much memory or getting out of memory exceptions. Moreover, the management for JavaScript application were tedious in the sense of deployment and custom tooling, in TypeScript, the application can be modern in the sense of deployment and can be used with modern deployment tools, and for custom tooling, mostly it is now provided as libraries in TypeScript, such as exception handling and variable types checking.

### NestJs

For server-side development, NestJs is an open source framework for developing scalable and maintainable applications, it extends the standard runtime environment, NodeJs, and provides more functionality, modularity, and gives an efficient and lightweight way to build services. NestJs main functionality lies in the fact that most repetitive tasks such as creating endpoints, creating data transfer objects, and others are fully automated whether it is through libraries or through the nest command line tool, this provide a better developer experience and a fast development process with focus on business needs instead of focusing on how the program should be written and such things.

### Fastify

Fastify goes hand in hand with nestjs in order to build a web application with the least overhead; so that the requests get handled faster and provide a responsive backend to the user.

### 

### MongoDB

MongoDB is a NoSQL database that is also considered a document-oriented database. It utilizes JSON to build its “documents” which is where data is stored. For our use case, MongoDB is used due to its flexibility in handling data, schemas can change, files can be uploaded into chunks, and most importantly MongoDB is classified as a Availability/Partition tolerance database in the CAP theorem context, which is the most important dimensions in a student assistant tool. On top of that, MongoDB has a full text search feature that lets the document be searched like it is indexed.

## 5.2 Overview

This section showcases some of the core features of the website, in addition to how they were built and why.

### 5.2.1 Front-End

In the frontend section, core components will be discussed from a developer point of view of how the technologies were used to build them, plus the user point of view of how they will be used.

The website is a component-based structure such that each page is represented by one or more components, each component is rendered separately in order to pull the appropriate data for each component separately. Prioritize a component-based architecture for development experience and fault tolerance, allowing for efficient creation and continued functionality despite component issues. Here are the main Pages and their components:

##### Index

This is the root file for React, sometimes called App as well, to start rendering components. It filters out requests requiring authentication, ensuring only authorized access; this is done by having an authentication component wrapping other components that needs to be filtered by authorization.

##### SignIn

The sign in component is responsible for managing the states of taking input from the user, mainly the username and the password, and rendering the login page for the website, in addition to validating input and making sure only users who are authorized can login.

##### Register

The register component is responsible for managing the states of taking the required input from the user such as their first name, last name, username, email, and password, as well as validating the input and rendering the required elements for the user.

* Dashboard

This is considered the homepage of the website, it displays some insights and analytics about the user data as well as routes to other pages in the website. It manages the states of those data as well displaying it in a user friendly way. It is important for the first page to be appealing so the user feels like it is worth using.

##### Course

The course page is responsible for displaying public courses together with showing other public information about users creating courses and such.

##### Profile

This is the main page for displaying, modifying, and deleting any information about the user such as first name, last name, email, password, deleting account and more. This page also shows how much storage the user has used as well as giving the ability to upload assets such as PDFs, Images, and others.

##### Track Course

Tracking the course and its tasks, notes, and assets, as well as showing data about each course and being able to edit it later on. This page also allows users to upload assets, like an image that was taken during the class, and put it in the right course.

There are other main components and pages that the website uses, however, the mentioned pages above are the main ones.

### 

### 5.2.2 Backend

This section discusses how the backend is organized in terms of maintainability, and how different technologies play a part in managing the website and its features.

#### Modules

In JavaScript (and TypeScript), code is organized in directories called modules, in React, these modules are usually the component, and it contains the services that connect to other services and databases.

These modules will be built using some build system, in this project the webpack is the build system, it also contains external and third party dependencies that we use to connect the database for example or to automate certain tasks.

##### Fastify

Fastify is a way to reduce toil and improve performance of request-response based web applications, this module is an open source third party module that is used to build the endpoint of the backend and make it available for authorized users.

##### Multer

Multer is a node.js middleware to handle form-data, in this project it is used for asset uploading, it is library that make sure developer focus on developing applications rather than setting up the backend for file upload, it has core functionality that can produce functionality of any type of file upload with ease.

##### Mongoose

Mongoose is an Object Data Modeling library that helps developers connect to a mongodb database. On top of that, making queries to the database easy and safe. This is all done using a custom configuration or letting mongoose handle the configuration.

##### JWT

JWT is an authentication technology, short for JSON Web Tokens, in node.js and JavaScript in general, the library/module JWT handles the setup of this token as well making sure users can’t directly access user information from the token. This provides an easy way to authenticate users and gives a good developer experience and maintainable code for others to extend to the current functionality.

##### RxJs

RxJs is a reactive programming library for JavaScript, it makes sure that doing asynchronous operations is easy and safe, without manually handling threads or such things. Furthermore, it prevents callback-based coding, which is not maintainable and easy to get wrong.

##### Jest

Jest is a testing library for JavaScript and it provides an easy way to create mocks and create unit tests for the code, it also provides insights about code coverage if needed and can be used with other testing libraries as well.

##### Bcrypt

Bcrypt is a password hashing library and module that also helps with salting and making sure passwords are safe and secure from developers and unauthorized users, it is based on the Blowfish cipher.

#### Backend System Architecture

The backend system of Instrument is based on a set of services, mainly the CRUD service for different data models, in addition to the asset management service which handles any asset manipulation.

The whole backend was created with the intention of providing an asynchronous way to deal with most operations, this way more requests can be processed. Thankfully the frameworks and libraries were chosen for this purpose providing an easy way to implement it.

To dive into more details, the Create, Read, Update, and Delete operation service named instrument-backend is based on the REST API architecture, which stands for Representational State Transfer. It is an architecture that is based on the HTTP protocol and provides the best practices for developers to build their web services and make sure to communicate any data through HTTP in a reasonable manner. Moreover, REST provides a set of rules for designing web-based systems that work well on the internet, hence the HTTP protocol, in our case because the backend and the frontend is separated into different programs, REST will provide a well-defined interface, endpoints, that the frontend will communicate with, this include well defined web paths as well as a well defined data format such as JSON.

On top of that, the instrument-backend acts as the interface between the application and the database. It facilitates CRUD (Create, Read, Update, Delete) operations and allows for complex queries, such as retrieving the number of users enrolled in a particular course, and other complex queries. This is done using the library mongoose and mongodb as the database.

The service itself is also divided into different modules, where each module is based on data ownership, for example all the user related functions are under the same directory, this makes sure the code is readable, consistent, and maintainable. In addition, each module contains a couple of other modules, such as configuration, data transfer objects, schemas, controllers, and more. This way the code is decoupled in a reasonable way and makes sure that developers work with interfaces rather than having to know how each module works.

Moreover the service is protected with the JWT token, a stateless token that gets verified with each request, providing a layer of security before each request gets manipulated.

For the asset management service, it utilizes multer and mongodb to manipulate files, including uploading, viewing, requesting name or other information, as well as downloading assets. This is done using chunk based uploading and downloading, when the file is first uploaded, it will be split into chunks of constant size and each chunk is going to have some certain sequential data from the file as well as metadata that tells information about the chunk, such as an MD5 hash, the chunk number, a file id. Note that the chunks will be saved as Base64 encoded data, in order to provide faster manipulation and a unified way of reading any type of file (Image, audio, PDF, …, etc).

This service is also protected with the JWT token, and can communicate with the instrument-backend service using the REST API service endpoint, providing information back and forth between the two services.

Each service is separated into three main modules and an optional configuration module, the three main modules are:

1. Schema

This module includes the expected state of the data that the service will receive for a certain data model. However, because the application is using mongodb, a schemaless database, some data might be process but not saved, which is acceptable in our case, the most trivial example in this case is the JWT token, since it is a stateless token we don’t save anything from it, but we expect it to have certain required parameters and other optional ones.

1. Service

This module is responsible for communicating with the database in a safe and maintainable way, making sure there exists a layer between the endpoints and the database in a decoupled, safe, and authorized way. The service module can be a single class in TypeScript, and might include other functions that manipulate the data without saving it to the database.

1. Controller

The controller module is considered the interface of the users of the services, it contains the endpoints as well as a way to call the right type of service that is mentioned above. In this module minimal implementation should be done and it should encapsulate any type of information from the user and the user can use it without any knowledge about how the service works.

## 5.3 Main functionality

This section presents some of the core features of the website and how it was built from a code point of view, moreover, it will also discuss some of the limitations of the features and how they can be used by the user.

### 5.3.1 User Login

Figures XXXX show the user login process, the user simply enters their username and password and clicks sign in. Furthermore, the input is validated in both the frontend and the backend, and this validation is done in both ends because some errors can only be caught in the backend such as a wrong password, while the frontend makes sure the input is not empty and valid.

Figure XXXX: Frontend compoment form submissions

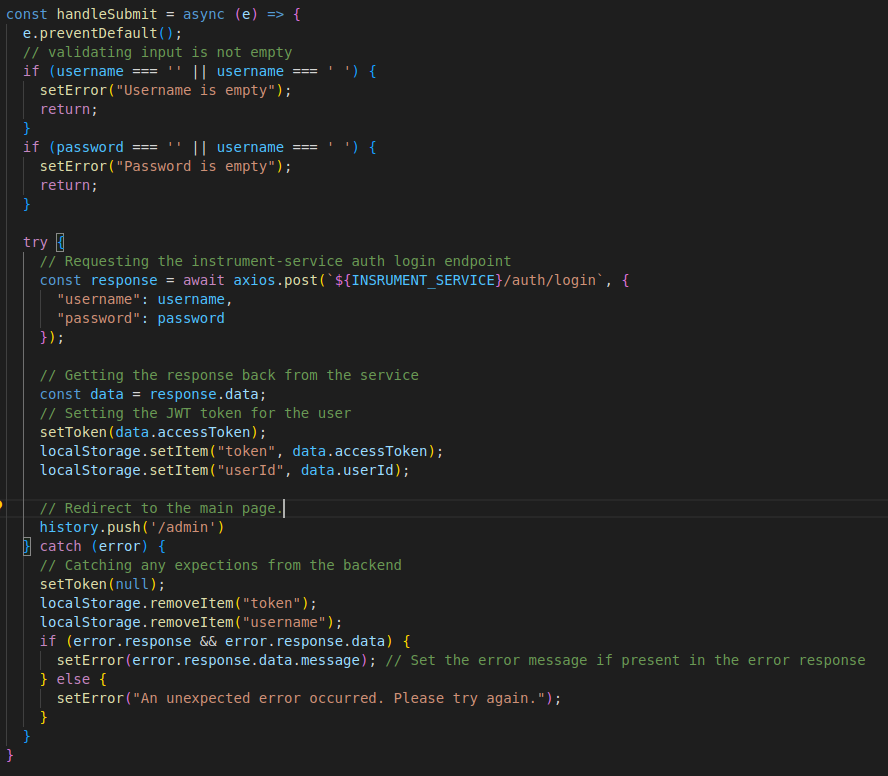


Figure XXXX: backend endpoint

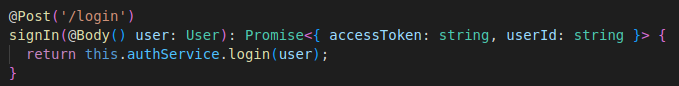
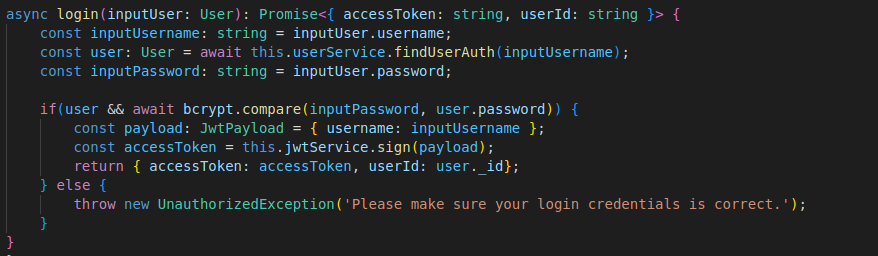


Figure XXXX: backend service that handles password hashing and salting



### 

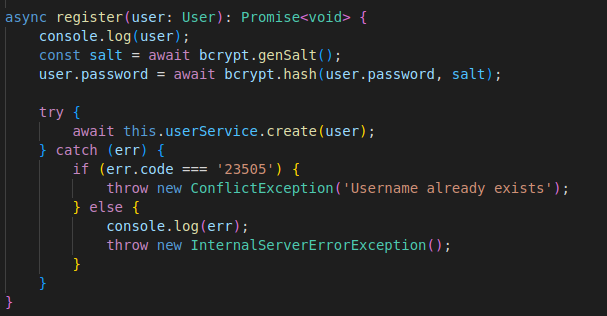
### 5.3.2 User Registration

Figures XXXX show the user registration process, the user simply enters their first name, last name, email, password, and clicks register. Furthermore, the input is validated in both the frontend and the backend, and this validation is done in both ends because some errors can only be caught in the backend such as a wrong password for example, while the frontend makes sure the input is not empty and valid.

Figure XXXX: Frontend compoment form submissions



Figure XXXX: Backend service form submissions



### 

### 5.3.3 Profile

Figures XXXX show some user profile functionality, the user simply enters the profile page and they can see their general information as well as buttons to upload assets, delete accounts, and create courses. Furthermore, the users can modify their information.

Figure XXXX: Frontend compoment form submissions



Figure XXXX: Backend service for user authentication

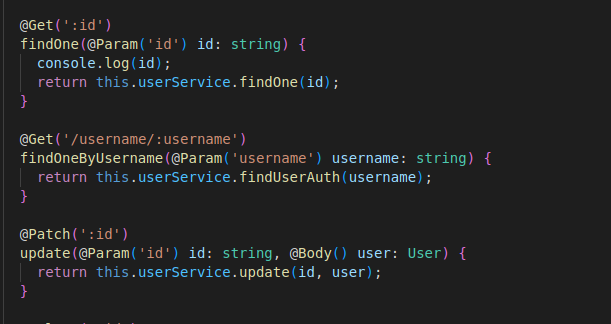


Figure XXXX: Backend service for user operations



### 

### 5.3.4 Creating Course

Figures XXXX show creating courses with the minimal required information, the user simply enters the name, duration, days, if the course is public, and the course description. Moreover, the same component updates the user course list and fetches them when the user opens the page.

Figure XXXX: Frontend compoment form submissions

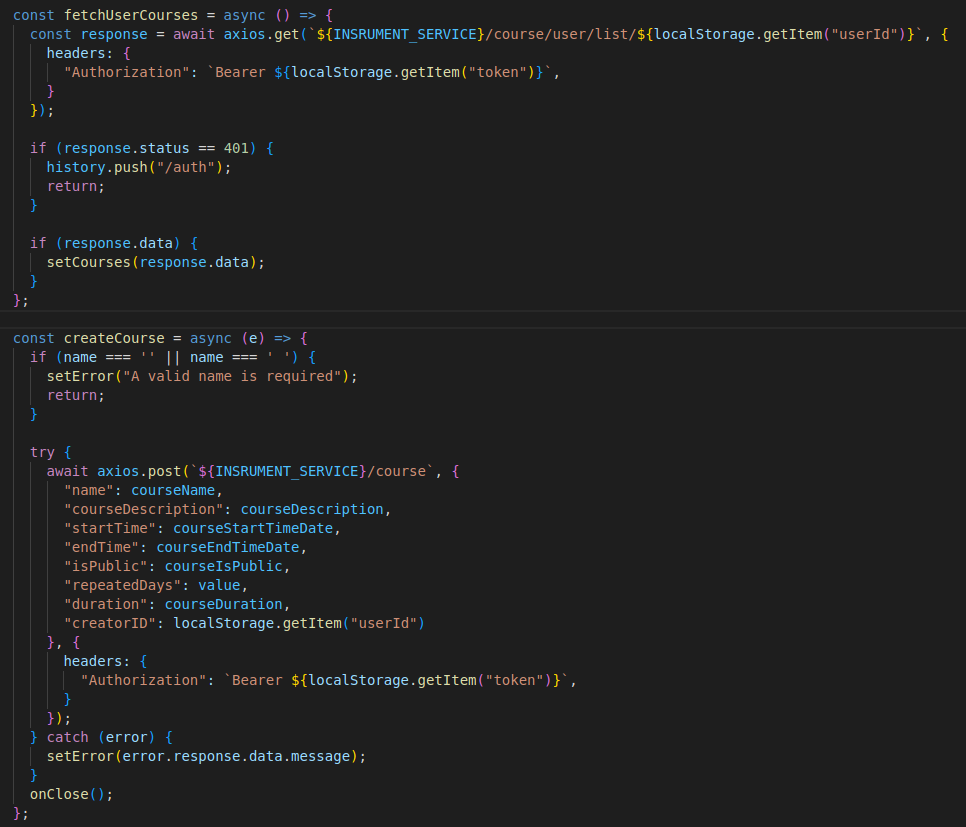
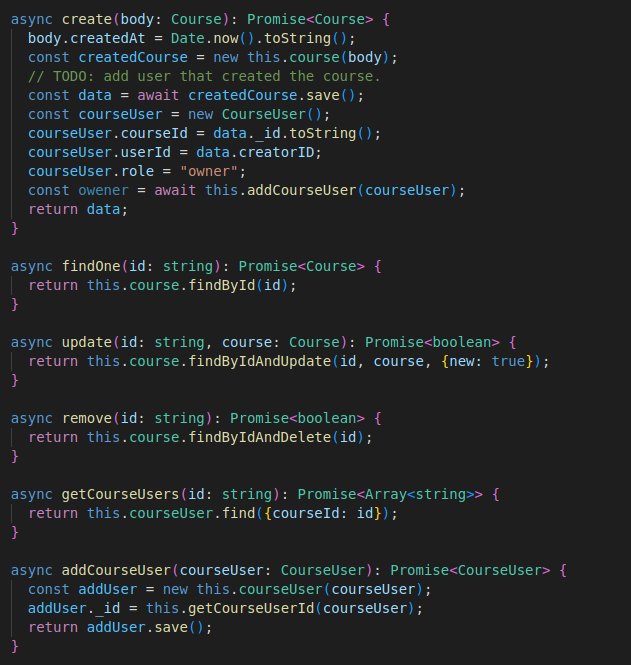


Figure XXXX: Backend service for course operations



### 

### 5.3.5 Creating Tasks

Figures XXXX show creating tasks with the minimal required information, the user simply enters the name, due date, and the description. Moreover, the same component updates the user tasks list and fetches them when the user opens the page.

Figure XXXX: Frontend compoment form submissions

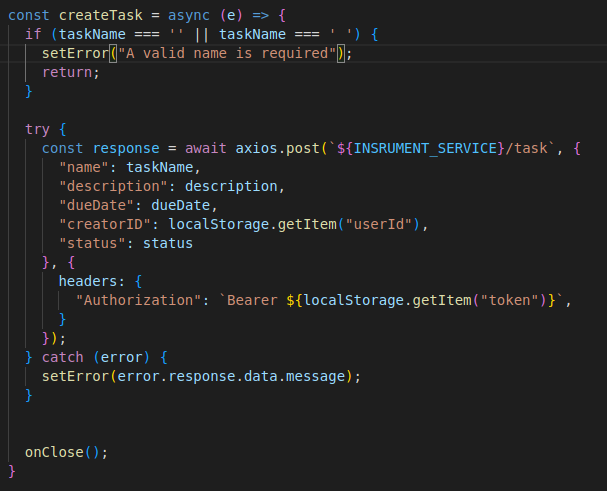
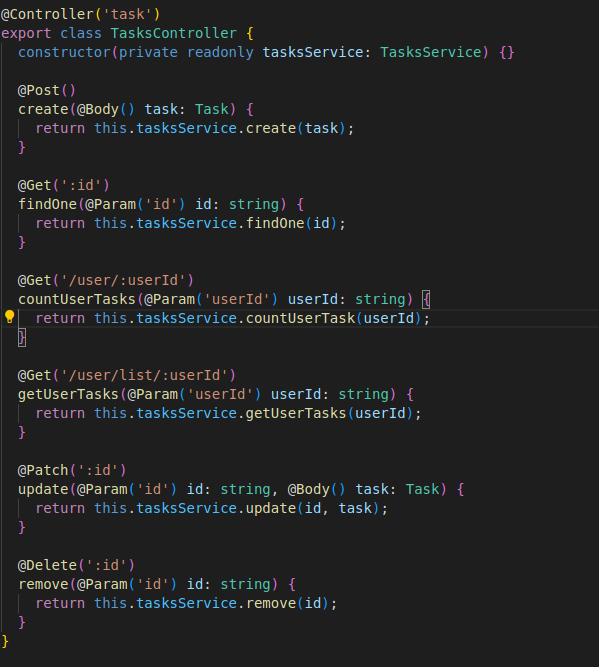


Figure XXXX: Backend controller and endpoints



### 

### 5.3.6 Uploading Assets

Figures XXXX show uploading assets with the minimal required information, the user simply selects the file and uploads it.

Figure XXXX: Asset management service

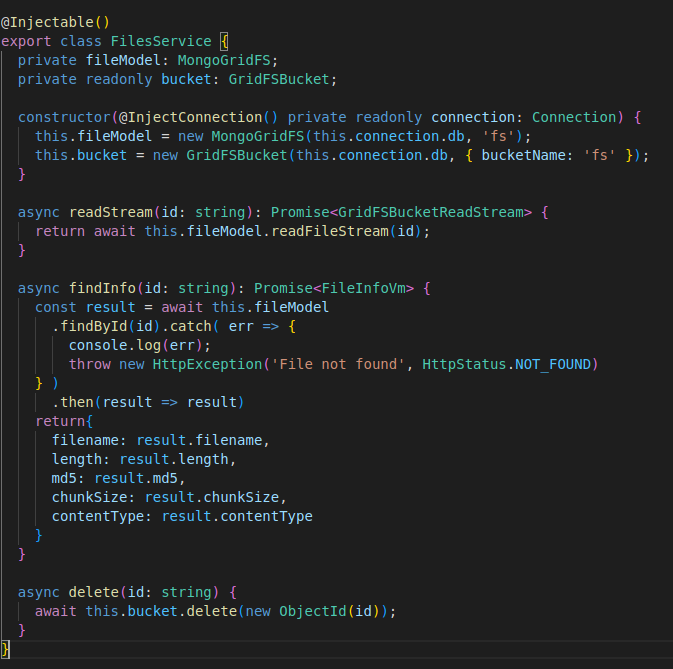


Figure XXXX: Asset management service and file uploading

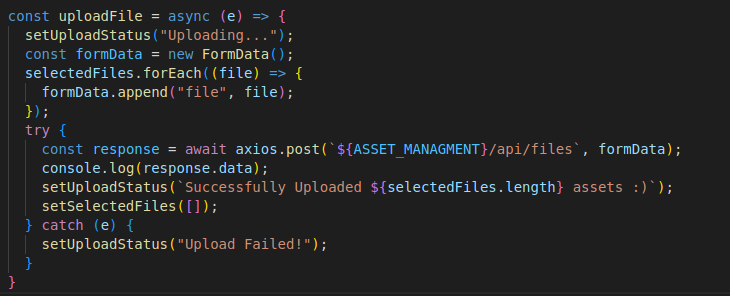


Figure XXXX: Asset management controller and endpoint



## 

## 

## 5.4 Features

Table xxx showcases all the features mentioned in chapter 3 and if they were implemented or not.

###### *Table XXXX: Features Table*

| **Functional Requirement ID** | **Features(s)** | **Implemented or Not** |
| --- | --- | --- |
| U\_FR\_0 | Authentication | Implemented |
| U\_FR\_1 | Edit Profile | Implemented |
| C\_FR\_0 | Create a course | Implemented |
| C\_FR\_1 | Read a course details | Implemented |
| C\_FR\_2 | Edit course details | Implemented |
| C\_FR\_3 | Delete course | Implemented |
| T\_FR\_0 | Create a task | Implemented |
| T\_FR\_1 | Read task | Implemented |
| T\_FR\_2 | Update task | Implemented |
| T\_FR\_3 | Delete Task | Implemented |
| R\_FR\_0 | Create Resources | Implemented |
| R\_FR\_1 | Read Resources | Implemented |
| R\_FR\_2 | Update Resources | Implemented |
| R\_FR\_3 | Delete Resources | Implemented |
| I\_FR\_\* | Institution Features | NOT Implemented |
| N\_FR\_0 | Notify Users | NOT Implemented |
| PR\_FR\_\* | Policy Engine | NOT Implemented |
| S\_FR\_0 | Search Engine | NOT Implemented |
| G\_FR\_0 | Generator | NOT Implemented |
| A\_FR\_4 | Association | NOT Implemented |

##### Newly Implemented Features

The following are the newly implemented features:

* Image course finding
  + This feature lets users upload assets to the platform, and the platform will suggest which course it should be in. This feature mainly uses the information inside each course such as the start date, end date, duration to match each image with each course.
* Dashboard Insights
  + This feature is responsible for displaying different insights and analytics about the user courses, tasks, schedule and other important data.

# Chapter 6: Testing

In this chapter a walkthrough in the testing dimension of the website will take place, including why different features are tested and how they were tested. Furthermore, it will go over some of the tools, frameworks, and technologies used to achieve a high test coverage and prove a certain level of safety for the core features of the website.

## 6.1 Testing Approach

For a website that handles student data, it should make sure that different data manipulation is done in a safe environment to ensure user trust. Moreover, the security of different assets and user information is also a key factor in the success of the website, as well as making sure the website is accessible for different users and different use cases. Testing also helps in figuring out the flaws in the system and might be a good indicator for future problems.

Two different types of testing have been used in the website,

* Black Box Testing, this is done by manually checking and paying attention for errors and different validation, this includes:
  + Use Case testing
  + Accessibility testing
* White Box Testing, it is used to improve usability, security and the overall architecture of the system, and this includes:
  + Unit testing functions, components, and endpoints
  + Security Testing
  + Static and Dynamic Analysis

There are two main approaches to test a system: functional testing and non-functional testing, functional testing will be highlighted in this section..

### 

### 6.1.1 Functional Testing

Functional testing ensures the system meets its designed functionalities in a safe and correct way. Here are the specific types of functional testing performed:

#### Unit Testing

To identify and minimize errors early on, individual modules were thoroughly tested before being integrated with the rest of the system. This ensured each module functioned as designed. Unit tests will be done on the backend, between different functions, services, and endpoints. In the frontends, unit tests will be done for functions, components, and webpages.

#### Integration Testing

Following individual testing, units are integrated and tested together to verify overall functionality and identify any issues arising from interactions between them. An example would be testing whether the backend successfully receives a confirmation from the driver upon order delivery.

#### System Testing

After integration testing, the complete system undergoes system testing to validate its functionality against overall requirements. This comprehensive evaluation ensures the system functions as a cohesive unit and meets user expectations.

## 

## 6.2 Testing Tools

* VSCode: used for writing code as well as debugging using breakpoints and typescript debugger.
* Postman: used for testing API endpoints as well as passing headers and other metadata to the APIs.
* Chrome Dev Tools: used for tracking network requests and responses, in addtion to debugging styling and design issues.
* MongoDB Compass: used a GUI for the database and provided an overall view of the metadata for the database as well as creating indices and editing data.
* Jest: used as a testing framework for JavaScript and TypeScript, moreover, Jest provides a way to mock functions and components and it was used to ensure the correctness of the codebase.

## 6.3 Testing Features

Functional testing procedures ensure that all implemented functionalities of the system are thoroughly evaluated to confirm they operate according to the designed specifications. The following Table XXXX shows the features that were implemented.

###### 

###### *Table XXXX: Tested Features*

| **Feature ID** | **Feature Name** | **Details** | **Priority** | **Tested** |
| --- | --- | --- | --- | --- |
| U\_FR\_0 | Authentication | Implemented | Essential | Yes |
| U\_FR\_1 | Edit Profile | Implemented | Essential | Yes |
| C\_FR\_0 | Create a course | Implemented | Essential | Yes |
| C\_FR\_1 | Read a course details | Implemented | Essential | Yes |
| C\_FR\_2 | Edit course details | Implemented | Essential | Yes |
| C\_FR\_3 | Delete course | Implemented | Essential | Yes |
| T\_FR\_0 | Create a task | Implemented | Essential | Yes |
| T\_FR\_1 | Read task | Implemented | Essential | Yes |
| T\_FR\_2 | Update task | Implemented | Essential | Yes |
| T\_FR\_3 | Delete Task | Implemented | Essential | Yes |
| R\_FR\_0 | Create Resources | Implemented | Essential | Yes |
| R\_FR\_1 | Read Resources | Implemented | Essential | Yes |
| R\_FR\_2 | Update Resources | Implemented | Essential | Yes |
| R\_FR\_3 | Delete Resources | Implemented | Essential | Yes |

## 

## 6.4 Testing Samples

The following examples illustrate test cases designed to verify the system's functionalities against the requirements

### Authentication

This test case covers the authentication of the user to be able to access resources in the website.

###### *Table XXXX: Tested Samples For User Authentication*

| **Test Case ID** | T\_U\_FR\_0 |
| --- | --- |
| **Test Title** | User Login |
| **Related Function** | U\_FR\_0 |
| **Test Description** | This test verifies the system's user sign-in functionality, including:   * Validate user input on both the frontend and backend to prevent invalid data submission and display informative error messages for user correction. * Prevent null and empty values. * Validate the user input and verify if the data is stored in the backend. |
| **Pre-conditions** | * Valid email and password. |
| **Test Steps** | * The user opens the sign in page. * Enter their valid email and password |
| **Test Data Sample** | Email: mrmoon@gmail.com  Password : 1Iy9sSHdBg9kmY7 |
| **Expected Results** | * The login page should be rendered successfully. * If the user entered the correct data they should be redirected to the dashboard. |
| **Post-condition** | User successfully loged in. |
| **Status** | Success. |

### Edit Profile

This test validates the edit profile operation and makes sure the information is processed correctly.

###### *Table XXXX: Test Cases For User Profile Edit*

| **Test Case ID** | T\_U\_FR\_1 |
| --- | --- |
| **Test Title** | User Edit Profile |
| **Related Function** | U\_FR\_1 |
| **Test Description** | This test verifies the system's user profile editing functionality, including:   * Handling invalid input. * Preventing null and empty values |
| **Pre-conditions** | * Clear and concise error message. |
| **Test Steps** | * The user opens the profile page. * Edit their profile operation. * Click Update. |
| **Test Data Sample** | newFirstName: “Moh” |
| **Expected Results** | * The page should provide an error message for any invalid input. * The page should provide a success message when the update reflects in the backend. * The profile page data should be updated. |
| **Post-condition** | Update is done successfully. |
| **Status** | successful. |

### Create Resources

This test validates the creating resources operations, ex: image upload, and makes sure the information is processed correctly.

###### *Table XXXX: Test Cases for Resource Creation*

| **Test Case ID** | T\_R\_FR\_0 |
| --- | --- |
| **Test Title** | Create Resource |
| **Related Function** | R\_FR\_0 |
| **Test Description** | This test verifies the system's ability to create resources such as images, PDFs, and others, including   * Handling differnt types of resources, jpg, png, PDF, PPTX, XLS, …, etc |
| **Pre-conditions** | * The asset must exist locally on the device before uploading to the website, or dragged and dropped in the upload section |
| **Test Steps** | * The user opens the profile page. * Click Upload Asset. * Choose the correct asset to upload * Upload |
| **Test Data Sample** | “Filename”: EduCare.pdf |
| **Expected Results** | * The page should reflect the status of uploading, and it should provide a message of success or failure afterwards. |
| **Post-condition** | Asset Uploaded Successfully. |
| **Status** | successful. |

# Chapter 7: Conclusions and Future Work

This chapter will go over future plans about the project along with current growth and improvement areas in the projects.

The EduCare system is designed to streamline and simplify the way organizations manage their workflow. It offers a user-friendly interface with a suite of intuitive features and tools. These features help teams stay organized and focused by providing a clear overview of tasks and priorities.

## Growth areas

This section discusses different areas that can improve in the website as well as different features that can help students accomplish more with less. Furthermore, provide more features that can attract more user bases to the website.

### Powerful Search Functionality

**Unified Search:** Effortlessly locate information within the EduCare platform. Search across public website content and various internal assets, including documents, presentations, and multimedia files. No more jumping between different search interfaces – EduCare provides a one-stop shop for all your information needs.

### Intelligent Content Organization

**Advanced Filters:** Refine your search results with granular filters. Find specific information faster by narrowing down your search based on content type, creation date, author, or relevant keywords. This saves valuable time and ensures you're focusing on the most pertinent information.

### Supporting Student Self-assessment

**Automated Test Generation:** Breathe new life into student assessment! EduCare streamlines the creation of tests for subjects where question generation is straightforward, such as multiple-choice and true-or-false formats. This frees up valuable educator time while allowing students to test and solidify their knowledge through a variety of practice tests.

### Robust Security Measures

**Two-Factor Authentication:** EduCare prioritizes data security with two-factor authentication (2FA). This adds an extra layer of protection by requiring a secondary verification code after entering your username and password. This significantly reduces the risk of unauthorized access to student records and sensitive information.

# References

1. [The disconnected: COVID-19 and disparities in access to quality …](https://educationaltechnologyjournal.springeropen.com/articles/10.1186/s41239-021-00262-1)
2. [Connecting the Disconnected: Improving Education and Employment Outcomes Among Disadvantaged Youth](https://www.econstor.eu/bitstream/10419/91776/1/pp56.pdf)
3. [The Shift to Online Classes During the Covid-19 Pandemic: Benefits, Challenges, and Required Improvements from the Students’ P](https://files.eric.ed.gov/fulltext/EJ1333732.pdf)
4. [Free Cloud Computing Services - AWS Free Tier](https://aws.amazon.com/free/)
5. [Free Trial and Free Tier | Google Cloud](https://cloud.google.com/free)
6. [Free Services Microsoft Azure](https://azure.microsoft.com/en-us/pricing/free-services)
7. [Serverless Architectures](https://aws.amazon.com/lambda/serverless-architectures-learn-more/#:~:text=What%20is%20a%20serverless%20architecture,management%20is%20done%20by%20AWS)
8. [Rose, S. , Borchert, O. , Mitchell, S. and Connelly, S. (2020), Zero Trust Architecture, Special Publication (NIST SP), National Institute of Standards and Technology, Gaithersburg, MD](https://doi.org/10.6028/NIST.SP.800-207)
9. [Bradley, V. M. (2021). Learning Management System (LMS) use with online instruction. International Journal of Technology in Education (IJTE), 4(1), 68-92. https://doi.org/10.46328/ijte.36](https://doi.org/10.46328/ijte.36)
10. [GraphQL](https://graphql.org/)

# 

# Appendix

## 1.3 Educational Platforms

### Moodle

So, Moodle is an open-source platform used by schools and institutions for online learning. It offers a virtual place where instructors can upload readings, homework, tests, and other course materials. And, students can access these materials whenever they want and from any place. Moodle is a useful tool for both traditional and online learning since it gives teachers and students an effective way to organize their courses and keep on top of their studies. Moreover, Moodle provides a number of tools to keep the connection between students and teachers, including message boards, chat rooms, and discussion forums. [More on Moodle](https://moodle.com)

### 

### Teach ‘n go

An education software called Tech 'n Go Education System offers solutions for students, teachers, and administrators. By providing tools for students to access and manage their educational content. A learning management system is built that enables instructors to design and share course materials, monitor student progress, and give comments. Tech 'n Go also provides tools, including discussion boards and video conferencing. The software also gives educational institutions access to an administrative interface that they may use to administer, track, and monitor student progress as well as conduct data analysis. [More on Teach'n go](https://www.teachngo.com)

### 

### Canvas

Canvas is currently called as the world's #1 teaching and learning software in the LMS industry, it is used by top institutions in the industry. Providing many tools in one place, and having a modern client interface gives canvas a huge advantage for organizations to use, making education more engaging and many other Canvas tools such as Canvas studio for creating content. [More on Canvas](https://www.instructure.com/canvas).

### 

### 

### Google Classroom

Google is known for its services, and how high end it is, classroom is no different, during COVID it was the go to for many organization because of its great user interface and integration with the Google suite of services, it has the core of many teaching operations, and it makes it easy for new users to use it and collaborate. Moreover, it integrates with many Google's technologies and AI to advance education. [More on Google classroom](https://edu.google.com/workspace-for-education/classroom).

## 

## 3.3.7 Policy Engine service

The policy engine is the backbone of the security in EduCare, this means it needs to be resilient and efficient. Moreover, the way the policy engine works is highly important on how the whole platform performs.

The main idea of the policy engine is to answer the questions whether entity A has permissions to do some operations on entity B, along with modifying these permissions when authorized to and needed to. This is achieved by having a key-value pair; where the key is the concatenation of any two entities IDs and the values are permission bits that specify what type of operations are authorized.

That is the high level idea of the policy engine, getting more under the hood, the engine is split into a control plane that answers questions and orders the modification of permissions between entities. The other part is the data plane, where it handles any data related operation such as the actual modification of the permission bits and the data store management for the permissions. This is highly important to make sure the platform remain isolated and secure.

#### Process

* Own a <key, value> data store.
  + The key will be a string concatenation between two entity IDs, i.e.: Given two entities A and B with IDs 325e3d35 and 184d6f98 respectively, they key will be 325e3d35184d6f98
  + Notice that the pairing between A and B is different between B and A by definitions, which provide different permissions combinations and provide an extra level of security since it follows the least privilege principle.
  + Notice that the size of the string will always be the size of the first ID + the size of the second ID. That is why it is important to unify an ID generation method for the platform; the size of the key will be constant for any two entities and the performance will be easier to manage and measure.
* The value will be 8 bits (zeros and ones) that indicate the following permissions (from the Least Significant Bit) respectively: Read, Write, Modify Level C, Modify Level B, Modify Level A, and extra 3 bits for scalability reasons.
  + Notice that the modification levels can be used to know what type of ownership the entity has over some other entity; full ownership or semi-ownership or none.
* To create/modify a permission for two entities, one must first concatenate their IDs, then play with the bits as required, similar to validation for some permission between two entities.

#### Implementation Details

* Using a key-value pair data store or some in-memory is ideal for this application; keys will be hashed and indexed based on that hash and the value will be simple bits that can be manipulated easily.
* Moreover, the combination of a serverless function and Golang will help in concurrency and start-up time for the function.
* Recommended in memory databases: Redis, Scylladb

#### ID Generation for each entity

The ID for any entity in the platform will be defined by a UUIDv4 (128 bits) prefixed with another 12 bits for the type. Using a key-value pair data store to store what each 12 bit stands for thus knowing what each ID is for which entity.

#### Control Plane

The control plane decides how data is managed, routed, and processed, in our case the control in the policy engine is composed of how to manage different permission combinations and answering questions like does Entity A have read access over Entity B in a fast efficient way.

#### Data Plane

The data plane is responsible for the actual moving of data, that is using the policy engine data for other purposes in the platform, such as populating joined data tables to answer different queries such as how many Entities of A does B and C has read access to, this includes stream processing the permission data to other internal service to understand the relationship between these entities.

#### Properties of the Policy Engine and the Entities IDs

There are 3 properties that are established by the Policy Engine and the ID for each entity, also called the VIC model:

* **Verification**: For any operation to work, first a verification that the operation must be done (from the permissions), this is established by the Policy Engine Control Plane and its <key, value> data store.
* **Identification**: Any entity in the platform is uniquely identified by a UUIDv4 prefixed by 12 bits indicating its type; notice that the type will be used later on in the association service to transform data on the fly.
* **Connection**: The concatenation of any two IDs and setting the appropriate permissions bits is a way of telling the platform that there is a relationship between two entities; more than two entities can have a relation by doing a high level entity connection (connecting the two main entities and creating a graph from them).

## 

## 3.3.10 Association service

The association service is a query engine that make use of relationships in the policy engine and has an interface for other internal services to query it and make fast data joins on the fly, this is possible because of how the Policy Engine works and relationships between entities can be derived from how the permission bits between any two entities.

The main idea of the query engine to get the relationships, and materialize it so other services can query it on the fly without having to wait for the engine, this is done by the fact that the any data store contains a Change Data Capture, CDC, and what is does it basically capture any operation that was done on the data (INSERT, UPDATE, DELETE, …, etc), from this we can get entities and create our own views of joined data, with low-latency between capturing the CDC and transforming the data into the appropriate views.

Notice how one change can trigger the change in many views, this is highly useful in creating multiple views on the fly in a low latency fashion, especially if the whole process of capturing and transforming was done in a serverless way. The maintenance of this service should be trivial as the view is created once, and changed when required only, or creating a new view to replace it to maintain consistency.

The way the view is obtained from the policy engine is simple, the changes in the data store for the policy engine is captured in a message broker in a topic depending on the change and the data, lets call it UAI, the association service read from the appropriate topic in the message broker in this case the UAI, and then the right worker is spinned off to transform the messages; the right worker in this case is a simple serverless program that is directly specified to be triggered when new messages are put into topic UAI (the association service is the control plane for redirecting this triggers).

In the example in [Association service database](#_b5evweap4rp2), The worker reads the messages and the data associated with it, get the user ID from the prefix (all IDs in the platform are generated with a prefix specifying its type, [3.5.11 Auto Generated ID Data](#_kz5wo2bvzdml) for more information), get the institution prefix (All IDs in the platform have a specific length, hence, if the worker got the user prefix the rest is the other entity), validate that it is an institution (check the ID prefix) to make sure it only transform the right messages.

Notice that there are a couple of things in this example that shows the power of the association service

* In a many-to-many relationship, the view will consist of Two IDs only (less data), ID of the first entity and the ID of the second entity. Depending on the query, the rest of the information can be derived from the entity data store(s) through the entity service(s), with GraphQL[[10]](https://graphql.org/) and the serverless architecture[[7]](https://aws.amazon.com/lambda/serverless-architectures-learn-more/#:~:text=What%20is%20a%20serverless%20architecture,management%20is%20done%20by%20AWS) for the services, the rest of the information will be requested easily, efficiently, and concurrently.
* The workers are very flexible, programmable, and cheaper than joins in a normal database management system; since the data is transformed on the fly and with each change rather than storing indices and joining with each query.
* Each worker can have multiple tasks and transformation, since it reads from one or more topics, the messages can also be used to create other views and joins, which also provide more flexibility and efficiency for the association service and the platform as a whole.
* The service itself is the control plane for other workers that materialize relationships into views.