EduBlock

etases

12/10/22

Table of contents

# Acknowledgement

Thanks to the following people for their help and support:

* [FPT Edu Experience Space](https://www.facebook.com/FEExpSpace) for providing an opportunity for us to do a team building activity and to have a chance to learn basic Blockchain technology.
* [Quách Luyl Đa](https://www.facebook.com/luyldaquach), our mentor & supervisor, for his guidance and support. He encouraged us to do our best and gave us a lot of useful advice.
* Fellow members at [ETASES](https://github.com/etases) to work together, share knowledge and experience, and to learn from each other. They did their best to make this project more complete and better.
* You, the reader, for your interest in our project.

A digital version of this report can be found here: https://etases.github.io/EduBlock.Report/

# Definition and Acronyms

| Acronym | Definition |
| --- | --- |
| BR | Business Rule |
| BA | Business Analysis |
| SRS | Software Requirement Specification |
| UC | Use Case |
| UAT | User Acceptance Test |
| API | Application Programming Interface |
| GUI | Graphical User Interface |
| RS | Request Server |
| CN | Chain Node |
| OCR | Optical Character Recognition |
| PM | Project Manager |
| Fabric | Hyperledger Fabric |
| FE | Feature |
| LI | Limitation |
| SI | Software Interface |
| HI | Hardware Interface |
| UI | User Interface |
| CRUD | Create, Read, Update & Delete |

# 1. Project Introduction

## 1.1 Overview

### 1.1.1 Project Information

* Project name: BlockChain application in academic record management to support online University/College admissions
* Project code: EduBlock
* Group name: ETASES
* Software type: Web app

### 1.1.2 Project Team

#### 1.1.2.1 Supervisor

| Full Name | Email | Phone Number | Title |
| --- | --- | --- | --- |
| Quach Luyl Da | daql@fpt.edu.vn | 0976703075 | Lecture |

#### 1.1.2.2 Team Members

| Full Name | Email | Phone Number | Title |
| --- | --- | --- | --- |
| Huynh Quang Tien | TienHQCE150130@fpt.edu.vn | 0976608340 | Leader |
| Le Xuan Tu | TuLXCE150344@fpt.edu.vn | 0939774512 | Member |
| Nguyen Dang Khoa | KhoaNDCE140165@fpt.edu.vn | 0382554293 | Member |
| Cao Hoang Anh Uy | UyCHACE150661@fpt.edu.vn | 0706456981 | Member |
| Nguyen Minh Khoi | KhoiNMCE150103@fpt.edu.vn | 0338616352 | Member |

## 1.2 Background

Keeping track of and managing student records is typically time-consuming and challenging, needing approval from the school and certain permissible circumstances. When transferring transcripts from high school to university, this causes difficulties. When applying to universities based only on their transcripts, students must request permission from the institution in order to be given a temporary transcript, which is both time-consuming and inconvenient. My group then developed the concept for a web application that enables students to monitor their academic progress over the course of their education. In order to make the academic records stand out and add additional features to the web app that improve user experience, we used the Blockchain platform.

## 1.3 Existing Systems

In the project, we use 4 main systems

1. Blockchain and Node system
2. Smart contract
3. Request server and client server
4. OCR

### 1.3.1 Blockchain and Node system

Blockchain, at its heart, is a distributed digital ledger that houses all types of data. The ownership of NFTs, bitcoin transactions, and smart contract definitions can all be recorded on a blockchain. (Andolfatto 2018)

This type of data may be stored in any conventional database, but blockchain is distinct since it is totally decentralized. The blockchain database is held on numerous computers dispersed throughout the network, as opposed to being kept in one place by a centralized administrator. These solitary machines are referred as nodes.

Blockchains can be public or private. Anyone can participate in a public blockchain, which allows them to access, write, or verify data on the blockchain. Because there is no single organization in charge of all the blockchain’s nodes, it is challenging to alter transactions that have been recorded on a public blockchain. (Andolfatto 2018)

A private blockchain, however, is managed by a company or group. Only that company or group has the authority to select users for the system, after which it has the right to modify the blockchain. Similar to an internal data storage system, this private blockchain procedure is spread over numerous nodes for added protection. (Andolfatto 2018)

Pros:

* Transactions are more accurate since a blockchain transaction requires numerous nodes to verify it. This can reduce mistakes.
* Absence of Middlemen Using blockchain, two parties to a transaction can confirm and finish it directly amongst themselves.
* Additional Protection: Theoretically, it is nearly impossible to conduct fraudulent transactions on a decentralized network like blockchain. They would have to hack each node and alter each piece of data in the ledger in order to carry out fraudulent transactions.

Cons:

* Limit on Transactions Per Second: Because Blockchain relies on a broader network to authorize transactions, its speed is constrained.
* Risk of Asset Loss: Some digital assets, like cryptocurrencies in a blockchain wallet, are safeguarded using a cryptographic key. This key needs to be properly guarded. There is presently no means to recover a private cryptographic key that grants access to a digital asset, thus the asset will be lost forever if the owner misplaces it.

### 1.3.2 Smart contract

Smart Contract is a computer program, or a transaction protocol designed to automatically carry out, manage, and record legally significant events and activities in accordance with the conditions of a contract or agreement. (“Front Matter” 2019)

Simply said, a stored program is run on the blockchain under predetermined conditions, ensuring that users receive immediate results free from the influence of middlemen.

Smart Contracts are written as code on the blockchain and work with a straightforward “if/when… then…” statement. A network of computers that will act when a verification condition is satisfied. (Lamb 2018)

Only parties who have been given permission may access the results; the transaction cannot be changed at will. If so, they must develop guidelines for dealing, trade standards, planning for all potential deviations, and dispute resolution. The signing of the contract will be more successful as a result of the participants’ satisfaction. (Lamb 2018)

Pros:

* When compared to using conventional contracts, smart contracts sometimes help save time and money.
* High security means that data is virtually completely safe.
* Because a smart contract is a collection of programmable code, developers can quickly alter and construct it into numerous contracts to accommodate various kinds of services and goods.
* Furthermore, because Smart Contracts are decentralized systems, they cannot be hampered by outside parties. This contributes to greater operational efficiency, reduced operating expenses, and increased transparency.

Cons:

* The information in the contract is essentially unchangeable because it is kept on a ledger. If the parties decide to alter some of the terms of the contract, this is advantageous but also disadvantageous.
* It is difficult for hackers to attack since the system is very secure, unaffected by intermediaries, and data protection is guaranteed, but this also implies that it will be challenging to fix.

### 1.3.3 Request server and client server

A computer network paradigm with a client and a server as its two major parts. The server serves as the location for storing resources, setting up service applications, and handling client requests in this approach. The client is responsible for submitting the server’s request.

Provides the network with the ability to concentrate applications and operations into one or more specific file services. Additionally enables simultaneous usage of the same resource by users, independent of their location.

Pros:

* Ensure data integrity in case of errors.
* Expanding your network is simple.
* Possesses the capacity to withstand network overload.

### 1.3.4 OCR

Optical Character Recognition is referred to as OCR. This specialized software program reads text from picture files. This technology is used to transfer and enter data. It is referred to as a digital scanning instrument that specializes in identifying characters, handwriting, or written letters. (Vihar Kurama 2022)

A printed or handwritten page that is scanned using OCR is saved in the TIF file format. This graphic underneath the display is clear and easy to read. However, a sequence of images with either white or black dots will be present, depending on the computer. To check whether the markers match, the technology now examines each line of the image. (Vihar Kurama 2022)

Pros:

* OCR can scan and interpret words on a screen, which will be helpful to the blind and visually impaired. People with eyesight impairments can readily understand things from here.
* Fast data entry: OCR ensures functionality to increase productivity and office job efficiency. The demand to scan papers is growing because the majority of people work in offices. By doing this, users will benefit from time savings and accurate, timely data updates.

Cons:

* On the basis of clear photos, the majority of OCR software and applications can only reliably recognize roughly complete of documents.
* OCR is challenging to identify because the accessible images include backgrounds and text that are similar in colour.

## 1.4 Business Opportunity

Although many high schools in Vietnam still keep paper records for post-graduation and enrolment in college or university, many now employ an online system to track students’ academic progress and inform parents of any latest outcomes. Because they must do it for both the paper records and the internet system, teachers find it challenging to update the information on their students. There is also a minimum level of transparency for students who wish to verify their information at any time because the internet system is centralized and only administrators and teachers have access to it. As a result, a system is required to help teachers and students manage student records in an easier, quicker, and more effective manner.

## 1.5 Software Product Vision

With the use of this system, students may simply keep track of changes to their grades in their academic records, reducing teacher grade entry errors. In order to gain rapid admission to graduate programs and colleges, students can also more conveniently retrieve their transcripts. High security and restrictions on data editing also assist in limiting the issue of phony points that are inaccurate representations of reality. Additionally, it eliminates challenges with entering grades into instructors’ school records because doing so will be quicker and easier with the aid of the system.

## 1.6 Project Scope & Limitations

A record management system will always be the best in terms of security and purity because it must, of course, assure data security. When interacting with and storing the data, always keep it intact and unaltered. Additionally, it must be user-friendly, with an interface that is clear and unambiguous and avoids misinterpretations of the translation or the information on the screen. Without the responsible user’s consent, data editing procedures cannot be carried out at will.

### 1.6.1 Major Features

FE-01: Using the blockchain platform, store data.

FE-02: Transcripts can be updated by converting photos to alphanumeric data.

FE-03: May replace paper school records entirely (electronic school records but have the nature of paper school records).

FE-04: Enhancing the effort teachers put into entering grades.

FE-05: Utilization dependability for admissions parties.

### 1.6.2 Limitations & Exclusions

LI-1: It is impossible to synchronize student counts between institutions due to the dispersed nature of the data.

LI-2: The only time to use the system is at the end of the year because it only saves the semester’s overall grade (can be expanded later)

LI-3: There is no option to switch schools (due to not processing student codes synchronously)

# 2. Project Management Plan

## 2.1 Overview

### 2.1.1 WBS & Estimation

| ID | WBS Item | Complexity | Estimated Effort (man-days) |
| --- | --- | --- | --- |
| 1 | Analysis |  | 20 |
| 1.1 | SRS | Medium | 5 |
| 1.2 | UML | Simple | 5 |
| 1.3 | Technology Research | Medium | 10 |
| 2 | Design |  | 20 |
| 2.1 | Architecture Design | Medium | 5 |
| 2.2 | Database Design | Medium | 5 |
| 2.3 | Detailed Design | Medium | 10 |
| 3 | Development |  | 80 |
| 3.1 | Backend | Medium | 30 |
| 3.2 | Frontend | Medium | 30 |
| 3.3 | Blockchain | Complex | 10 |
| 3.4 | OCR | Complex | 10 |
| 4 | Testing |  | 15 |
| 4.1 | Unit Testing | Simple | 3 |
| 4.2 | Integration Testing | Medium | 5 |
| 4.3 | System Testing | Medium | 7 |
| 5 | Documentation |  | 20 |
| 5.1 | Final Report | Medium | 10 |
| 5.2 | Final Product | Medium | 10 |
| # | Total |  | 155 |

### 2.1.2 Project Objectives

* Timeliness: 70%
* Allocated Effort (man-days): 90 days
* Quality Coverage:
  + Reviewing: 100%
  + Unit Test: 95%
  + Integration Test: 95%
  + System Test: 98%
  + User Acceptance Test: 80%

### 2.1.3 Project Risks

| Risk | Impact | Probability | Mitigation |
| --- | --- | --- | --- |
| The team members are not familiar with the technologies used in the project | High | High | The team members will be trained before the project starts |
| The team members can not work together | High | Medium | Prepare a team building activity before the project starts |
| There is not enough time to research the technologies used in the project | High | Low | Find a mentor to help the team members / Find a workaround or similar technology that can be used in the project |

## 2.2 Management Approach

### 2.2.1 Project Process

|  |  |  |
| --- | --- | --- |
| |  | | --- | | (a) Agile |      |  | | --- | | (b) Scrum |   Figure 2.1: Software Development Process Model |

Since parts of the requirements are not clear, we will use a process model of the Agile family called Scrum. Using this model, we can understand requirements, adapt changes quickly and deliver target products in a short time to test & review them for next iterations. The project will be divided into 4 iterations (sprints). Each sprint will be a month long and include 2 phases: planning and development. The planning phase will be used to estimate tasks from the product backlog, create a sprint backlog and assign tasks to the team members to develop in the development phase. At the end of the sprint, the team will review the sprint, discuss the problems and changes during the sprint and plan for the next sprint.

### 2.2.2 Quality Management

* Reviewing: There will be a meeting every week for the team and the supervisor to review the status of the tasks in the current sprint. The team will show the progress of the tasks and discuss the problems and changes during the sprint. The supervisor will take notes and give advices to the team.
* Defect Prevention: Each change of the code should be reviewed by the code master or the team leader before applying to the main project. They will check for bugs, code smells, anti-patterns and other problems.
* Unit Testing: Each module of the system should be tested before applying to the main project. The team can test the module by themselves or use a testing framework / software to test the module.
* Integration Testing: The team will test the system after each sprint to make sure that the system is working properly. There will be an automatic testing software that includes test scenarios for the system.
* System Testing: The team will test the completed system based on the requirements and the test scenarios. The team can test the system by themselves or use a testing framework / software to test the system.

### 2.2.3 Training Plan

| Area | Participants | When, Duration | Waiver Criteria |
| --- | --- | --- | --- |
| Blockchain | Project Team | 20/07/2022, 2 weeks | Mandatory |
| Hyperledger Fabric | Project Team | 01/10/2022, 2 weeks | Mandatory for Backend Team |
| Java, Javalin | Project Team | 01/09/2022, 1 weeks | Mandatory |

## 2.3 Master Schedule

| Deliverable | Due | Deliverable Scope |
| --- | --- | --- |
| Project Plan | 05/09/2022 |  |
| SRS | 15/09/2022 |  |
| Design 1 | 15/09/2022 | Architecture Design, Database |
| Design 2 | 30/09/2022 | Detailed Design |
| Backend | 30/09/2022 | Code, Test, Deployment |
| Frontend | 30/11/2022 | Code, Test, Deployment |
| Extra Services | 30/11/2022 | Blockchain, OCR |
| Final Report | 10/12/2022 |  |
| Final Product | 15/12/2022 |  |

## 2.4 Project Organization

### 2.4.1 Team Structure

|  |
| --- |
| Figure 2.2: Team Structure |

### 2.4.2 Roles

| Role | Responsibilities |
| --- | --- |
| Project Manager | Supervise the project, Manage the team, Manage the schedule, Manage deliverables |
| Analysis Leader | Analyze the requirements, Create SRS, Manage the analysis team |
| Analysis Member | Analyze the requirements, Create SRS |
| Design Leader | Design the system, Create system design document, Manage the design team |
| Design Member | Design the system, Create system design document |
| Backend Leader | Develop the backend, Manage the backend team |
| Backend Member | Develop the backend |
| Frontend Leader | Develop the frontend, Manage the frontend team |
| Frontend Member | Develop the frontend |
| Test Leader | Test the system, Create test report, Manage the test team |
| Test Member | Test the system, Create test report |

## 2.5 Project Communication

### 2.5.1 Communication Plan

| Item | Target | Purpose | When | Type |
| --- | --- | --- | --- | --- |
| Discord | Project Team | Review meeting & Status report | Monday, Friday | Voice, Remote |
| Google Meeting | Project Team & Supervisor | Review meeting, Sprint revision & Closeup | Wednesday | Voice, Remote |
| Messenger | Project Team & Supervisor | Meeting planning, Q&A & Status report | Everyday | Text, Remote |
| FU Library | Project Team | Pair programming & Code review | Planned | Offline |

### 2.5.2 External Interfaces

| Function | Contact Person | Contact Address | Responsibility |
| --- | --- | --- | --- |
| Supervisor | Quach Luyl Da | daql@fpt.edu.vn | Review deliverables, Provide document template, Check project progress, Give advices to project team |

## 2.6 Configuration Management

### 2.6.1 Tools & Infrastructures

#### 2.6.1.1 Common

| Type | Tool |
| --- | --- |
| Version Control | Git, GitHub |
| UML | PlantUML, Graphviz |
| Deployment | Docker |
| Project Management | Quarto, GitHub Projects |

### 2.6.2 Backend

| Type | Tool |
| --- | --- |
| Programming Language | Java |
| Library | Javalin, Fabric SDKs, HSCore, Guava, Genson, EvalEx, JWT |
| Compiler | JDK, Lombok |
| UI | TinyLog, JLine, Fabric CLI |
| DBMS | H2, Hibernate, Minifabric |
| IDE / Editor | IntelliJ IDEA, VSCode |

### 2.6.3 Frontend

| Type | Tool |
| --- | --- |
| Programming Language | TypeScript |
| Library | React |
| Compiler | Node |
| UI | React |
| IDE / Editor | VSCode |

### 2.6.4 Document Management

We use Quarto to build documents from Markdown files and use GitHub to manage the files and their changes. A participant will create a new branch to edit the files, create pull requests and wait for the project manager to review the changes and merge to the main branch. Then, it’ll be built in three outputs: a website using GitHub Pages for visualization, A PDF document & A MS-Word document.

### 2.6.5 Source Code Management

We manage the source code by using GitHub. Endpoints of the project will be upload into separated repositories. Once the code is changed, the participant will create a new branch, create a relevant pull request, and wait for code owners to review and merge to the main branch.

# 3. Software Requirement Specification

## 3.1 Overall Description

### 3.1.1 Product Overview

This is the software requirement specification for the project “EduBlock”. EduBlock is an web-application that will help the school to manage their student’s records, more specifically, the students and teachers can reduce paper’s work to manage their records. Lately, the school has been using paper to manage their student’s record, which is not efficient and not environmentally. EduBlock will help the school to manage their student’s records in a more efficient way, although there are some other third-party apps that can help schools to manage student’s records nowadays, but it is not really efficient and safe. Our application uses blockchain technology to make sure the data is safe and secure. Every step of the process that needs to be worked with records will be tracked by EduBlock, so the school can easily track the data changes and make sure the data is not being tampered.

### 3.1.2 Business Rules

| **ID** | **Rules Description** |
| --- | --- |
| BR-1 | The application will be used by the students, teachers, staff and admin. |
| BR-2 | Only Staff have permission to manage classroom including create new class, edit class, assign or remove teacher from the class, assign student to class, remove student from class, edit student information. |
| BR-3 | Only Admin have permission to create new account(s). |
| BR-4 | Only Teacher who teach the subject can edit the grade of the student. |
| BR-5 | Student can only view their class, profile and academic record. |
| BR-6 | In Blockchain, the data is immutable, so the data cannot be changed once it is created. Because of this, the data can just be appended, can’t be edited or deleted, this will help ensure the student’s record is safe and secure. |
| BR-7 | A node if wants to join the network, it must have other nodes permission or the node must be approved by the admin. |
| BR-8 | In a private blockchain, every node knows each other, which node own the data. Data is shared between nodes so the data can be recovered if one node is down. |
| BR-9 | Other nodes can only read the data, they cannot change the data. |
| BR-10 | Third-party’s member can only view the academic record and statistics of the students by using verified keys. |

## 3.2 User Requirements

* The Academic record management web-app has five active actors: Student, Teacher, Staff, Administrator and Third-party’s member (i.e., parents, etc.).
* Admin can create accounts for each role such as staff, student, teacher.
* Students can view their academic record.
* Teachers can manage their class and their student’s academic record.
* Staff can manage the classroom and view academic records of students, assign or delete teachers from the class, assign students to class, create new classes and edit student information.
* Third-party’s member can view the academic record and statistic of the students by using verified key.

### 3.2.1 System Actors

| **ID** | **Actor** | **Description** |
| --- | --- | --- |
| 1 | ADMIN | Admin is the person who has the highest authority in the system. Admin can manage the account of the other actors. |
| 2 | STAFF | Staff is the person who has the authority to manage the classroom. Staff can assign or remove teachers from the class, assign students to class and remove students from class. |
| 3 | TEACHER | Teacher is the person who has the authority to manage their class. Teachers can view their student’s academic record; subject teachers can send requests to edit student grade. |
| 4 | STUDENT | Student is the person who has the authority to view their academic record. |
| 5 | THIRD PARTY | Third party is the person who has the authority to view the academic record and statistics of the students by using verified a key. |

### 3.2.2 Use cases

#### 3.2.2.1 Admin

|  |
| --- |
| Use case diagram for Admin |

##### 3.2.2.1.1 UC-1 Admin Login

* **Description:** Admin can login with their username and password.
* **Actors:** Admin.
* **Preconditions:** Admin has an account.
* **Post Conditions:** Admin can access the system.
* **Flow of Events:**
  + Admin goes to EduBlock.
  + Admin enters their username and password.
  + System verifies the username and password.
  + System displays the dashboard.
* **Exceptions:**
  + If the username or password is incorrect, the system will display an error message.

##### 3.2.2.1.2 UC-2 Admin view list of accounts

* **Description:** Admin can view list of all accounts.
* **Actors:** Admin
* **Preconditions:** Admin is logged in.
* **Post Conditions:** System shows list of all accounts.
* **Flow of Events:**
  + Admin goes to EduBlock.
  + Admin login with username and password.
  + Admin click on “Account”.
  + System shows list of all accounts.
* **Alternate Flow:**
  + System displays notification “No account found” if there is no account.
* **Exception:**
  + System displays notification “No account found” if there is no account.

##### 3.2.2.1.3 UC-3 Admin view account details

* **Description:** Admin can view account details.
* **Actors:** Admin
* **Preconditions:** Admin is logged in.
* **Post Conditions:** System show account details.
* **Flow of Events:**
  + Admin goes to EduBlock.
  + Admin login with username and password.
  + Admin click on “Account”.
  + Admin click on “Details” (human icon) on actions column.
  + System shows account details.
* **Alternate Flow:**
  + System displays notification “No account found” if there is no account.
* **Exception:**
  + System displays notification “No account found” if there is no account.

##### 3.2.2.1.4 UC-4 Admin create (multiple) account

* **Description:** Admin can create (multiple) accounts for each role such as staff, student, teacher.
* **Actors:** Admin
* **Preconditions:** Admin is logged in.
* **Post Conditions:** System create (multiple) accounts.
* **Flow of Events:**
  + Admin goes to EduBlock.
  + Admin login with username and password.
  + Admin click on “Account”.
  + Admin click on “Create” button.
  + Admin fill in the form.
  + Admin click on “Create” button.
  + System create account.
* **Alternate Flow:**
  + Admin can create multiple accounts by clicking on “Add Account” button.
  + Admin fill in the form.
  + Admin click on “Create” button.
  + System creates accounts.
* **Exception:**
  + System displays notification if the form is not filled correctly.

##### 3.2.2.1.5 UC-5 Admin search account

* **Description:** Admin can search accounts by text, username, email, id, first name and last name.
* **Actors:** Admin
* **Preconditions:** Admin is logged in.
* **Post Conditions:** System shows list of accounts that match the search criteria.
* **Flow of Events:**
  + Admin goes to EduBlock.
  + Admin login with username and password.
  + Admin click on “Account”.
  + Admin click on “Search” button.
  + Admin input text to search account.
  + System shows list of accounts that match the search criteria.
* **Alternate Flow:**
  + Admin add search criteria by clicking on Search in” field.
  + Admin choose which criteria to search and combine with search by text.
  + system show list of accounts that match the search criteria.
* **Exception:**
  + System displays notification “No account found” if there is no account that matches the search criteria.

##### 3.2.2.1.6 UC-6 Admin update their profile

* **Description:** Admin can update their profile’s information.
* **Actors:** Admin
* **Preconditions:** Admin is logged in.
* **Post Conditions:** System update admin’s profile.
* **Flow of Events:**
  + Admin goes to EduBlock.
  + Admin login with username and password.
  + Admin click on their avatar above the logout button at bottom left.
  + Admin click on “Update” button at top right corner.
  + A form will appear, admin then input their new information.
  + Admin click “Submit”.
  + System update admin’s information and show success message.
* **Alternate Flow:**
  + System displays notification if the form is not filled correctly.
* **Exception:**
  + System displays notification if the form is not filled correctly.

##### 3.2.2.1.7 UC-7 Admin change password of other account

* **Description:** Admin can change password of other account.
* **Actors:** Admin
* **Preconditions:** Admin is logged in.
* **Post Conditions:** System change password of target account.
* **Flow of Events:**
  + Admin goes to EduBlock.
  + Admin login with username and password.
  + Admin click on “Account”.
  + System shows list of all account.
  + Admin click on “Change Password” (lock icon) on actions column.
  + Admin input new password.
  + Admin click “Submit”.
  + System changes password of the target account and shows success message.
* **Alternate Flow:**
  + System displays notification if the password is invalid.
* **Exception:**
  + System displays notification if the password is invalid.

##### 3.2.2.1.8 UC-8 Admin get grade report and get classification report in a year

* **Description:** Admin get grade report or classification report in a year.
* **Actors:** Admin
* **Preconditions:**
  + Staff is logged in.
  + Class exists.
* **Post Conditions:** System export report to admin’s computer.
* **Flow of Events:**
  + Admin goes to EduBlock.
  + Admin login with username and password.
  + At dashboard Admin choose year and grade to get report.
  + Admin click “Get Report” button.
  + System ready to export report to admin’s computer.
  + Admin choose where to save export and click “Save”.
* **Alternate Flow:**
  + Admin click “Get Classification Report” to get classification report.
  + System ready to export classification report to Admin’s computer.
  + Admin choose where to save export and click “Save”.
* **Exception:**
  + Button will be disabled if there is nothing to report.

##### 3.2.2.1.9 UC-9 Admin get report of a class

* **Description:** Admin get report of a class.
* **Actors:** Admin
* **Preconditions:**
  + Staff is logged in.
  + Class exists.
* **Post Conditions:** System export report to admin’s computer.
* **Flow of Events:**
  + Admin goes to EduBlock.
  + Admin login with username and password.
  + Admin choose “Classroom” on sidebar.
  + System shows list of all classes.
  + Admin click on “Details” button on actions column to view detail of a class.
  + On details page Admin click “Get classroom report” button.
  + System ready to export report to admin’s computer.
  + Admin choose where to save export and click “Save”.
* **Exception:**
  + If class don’t exist Admin will not able to get report.

##### 3.2.2.1.10 UC-10 Admin view classrooms list

* **Description:** Admin can view list of all classrooms.
* **Actors:** Admin
* **Preconditions:** Admin is logged in.
* **Post Conditions:** System show list of all classrooms.
* **Flow of Events:**
  + Admin goes to EduBlock.
  + Admin login with username and password.
  + Admin click on “Classroom” on sidebar.
  + System shows list of all classrooms.
* **Alternate Flow:**
  + System displays notification “No classroom found” if there is no classroom.
* **Exception:**
  + System displays notification “No classroom found” if there is no classroom.

##### 3.2.2.1.11 UC-11 Admin view classroom details

* **Description:** Admin can view details of a classroom.
* **Actors:** Admin
* **Preconditions:**
  + Admin is logged in.
  + Classroom exists.
* **Post Conditions:** System show details of a classroom.
* **Flow of Events:**
  + Admin goes to EduBlock.
  + Admin login with username and password.
  + Admin click on “Classroom” on sidebar.
  + System shows list of all classrooms.
  + Admin click on “Details” button on actions column to view detail of a classroom.
  + System shows details of a classroom.
* **Alternate Flow:**
  + System displays notification “Classroom not found” if classroom don’t exist.
* **Exception:**
  + System displays notification “Classroom not found” if classroom don’t exist.

##### 3.2.2.1.12 UC-12 Admin Create statistic key

* **Description:** Admin can create statistic key for third party’s member.
* **Actors:** Admin
* **Preconditions:** Admin is logged in.
* **Post Conditions:** System create statistic key.
* **Flow of Events:**
  + Admin goes to EduBlock.
  + Admin login with username and password.
  + Admin click on “Manage stats key list” on sidebar.
  + Admin choose grade and year.
  + Admin then click on “Create new Statistic key” button.
  + System creates statistic key and show it on the list.
* **Alternate Flow:**
  + System displays notification if the key can’t be created.
* **Exception:**
  + System displays notification if the key can’t be created.

#### 3.2.2.2 Staff

|  |
| --- |
| Use case diagram for Staff |

##### 3.2.2.2.1 UC-13 Staff Login

* **Description:** Staff can login with their username and password.
* **Actors:** Staff.
* **Preconditions:** Staff has an account.
* **Post Conditions:** Staff can access the system.
* **Flow of Events:**
  + Staff goes to EduBlock.
  + Staff enters their username and password.
  + System verifies the username and password.
  + System redirect Staff to dashboard.
* **Exceptions:**
  + If the username or password is incorrect, the system will display an error message.

##### 3.2.2.2.2 UC-14 Staff view list of accounts

* **Description:** Staff can view list of all accounts.
* **Actors:** Staff
* **Preconditions:** Staff is logged in.
* **Post Conditions:** System show list of all accounts.
* **Flow of Events:**
  + Staff goes to EduBlock.
  + Staff login with username and password.
  + Staff click on “Account”.
  + System shows list of all accounts.
* **Alternate Flow:**
  + System displays notification There’s nothing to show” if there is no account.
* **Exception:**
  + System displays notification “No account found” if there is no account.

##### 3.2.2.2.3 UC-15 Staff view account details

* **Description:** Staff can view account details.
* **Actors:** Staff
* **Preconditions:** Staff is logged in.
* **Post Conditions:** System show account details.
* **Flow of Events:**
  + Staff goes to EduBlock.
  + Staff login with username and password.
  + Staff click on “Account”.
  + Staff click on “Details” (human icon) on actions column.
  + System shows account details.
* **Alternate Flow:**
  + System displays notification “No account found” if there is no account.
* **Exception:**
  + System displays notification “No account found” if there is no account.

##### 3.2.2.2.4 UC-16 Staff search account

* **Description:** Staff can search account by text, username, email, id, first name and last name.
* **Actors:** Staff
* **Preconditions:** Staff is logged in.
* **Post Conditions:** System show list of accounts that match the search criteria.
* **Flow of Events:**
  + Staff goes to EduBlock.
  + Staff login with username and password.
  + Staff click on “Account”.
  + Staff click on “Search” button.
  + Staff input text to search account.
  + System shows list of accounts that match the search criteria.
* **Alternate Flow:**
  + Admin add search criteria by clicking on Search in” field.
  + Admin choose which criteria to search and combine with search by text.
  + system show list of accounts that match the search criteria.
* **Exception:**
  + System displays notification “No account found” if there is no account that match the search criteria.

##### 3.2.2.2.5 UC-17 Staff view class list

* **Description:** Staff can view list of all classes.
* **Actors:** Staff
* **Preconditions:** Staff is logged in.
* **Post Conditions:** System show list of all classes.
* **Flow of Events:**
  + Staff goes to EduBlock.
  + Staff login with username and password.
  + Staff click on “Classroom”.
  + System shows list of all classes.
* **Alternate Flow:**
  + System displays notification “No class found” if there is no class.
* **Exception:**
  + System displays notification “No class found” if there is no class.

##### 3.2.2.2.6 UC-18 Staff create new class

* **Description:** Staff create a new class.
* **Actors:** Staff
* **Preconditions:** Staff is logged in.
* **Post Conditions:** System create a new class.
* **Flow of Events:**
  + Staff goes to EduBlock.
  + Staff login with username and password.
  + Staff click on “Classroom”.
  + Staff click on “Create” button.
  + A form will appear, staff then input class’s information and choose homeroom teacher.
  + Staff click “Create classroom”.
  + System creates a new class and show success message.
* **Alternate Flow:**
  + System displays error notification if the form is not filled correctly.
* **Exception:**
  + System displays error notification if the form is not filled correctly.

##### 3.2.2.2.7 UC-19 Staff view class details

* **Description:** Staff view details of a class.
* **Actors:** Staff
* **Preconditions:** Staff is logged in.
* **Post Conditions:** System show class details.
* **Flow of Events:**
  + Staff goes to EduBlock.
  + Staff login with username and password.
  + Staff click on “Classroom”.
  + Staff click on “Details” (details icon) on actions column.
  + System shows class details.
* **Alternate Flow:**
  + System displays notification “No class found” if there is no class.
* **Exception:**
  + System displays notification “No class found” if there is no class.

##### 3.2.2.2.8 UC-20 Staff edit class

* **Description:** Staff edit class’s information.
* **Actors:** Staff
* **Preconditions:** Staff is logged in.
* **Post Conditions:** System edit class’s information.
* **Flow of Events:**
  + Staff goes to EduBlock.
  + Staff login with username and password.
  + Staff click on “Classroom”.
  + Staff click on “Details” (icon) on actions column.
  + System shows class details.
  + Staff click “Update details”.
  + A form will appear, staff then edit class’s information.
  + System edits class’s information and show success message.
* **Alternate Flow:**
  + System displays error notification if the form is not filled correctly.
* **Exception:**
  + System displays error notification if the form is not filled correctly.

##### 3.2.2.2.9 UC-21 Staff view student of a class

* **Description:** Staff view list of students in a class.
* **Actors:** Staff
* **Preconditions:**
  + Staff is logged in.
  + Class exists.
* **Post Conditions:** System show list of students in a class.
* **Flow of Events:**
  + Staff goes to EduBlock.
  + Staff login with username and password.
  + Staff click on “Classroom”.
  + Staff click on “Details” (icon) on actions column.
  + System shows class details.
  + Staff click “Students”.
  + System shows list of students of the class.
* **Alternate Flow:**
  + System displays notification “No student found” if there is no student in the class.
* **Exception:**
  + System displays notification “No student found” if there is no student in the class.

##### 3.2.2.2.10 UC-22 Staff add students to a class

* **Description:** Staff add students to a class.
* **Actors:** Staff
* **Preconditions:**
  + Staff is logged in.
  + Class exists.
* **Post Conditions:** System add students to a class.
* **Flow of Events:**
  + Staff goes to EduBlock.
  + Staff login with username and password.
  + Staff click on “Classroom”.
  + Staff click on “Details” (icon) on actions column.
  + System shows class details.
  + Staff click “Students”.
  + System shows list of students of the class.
  + Staff click “Add students”.
  + A form will appear, staff then choose students to add to the class.
  + Staff click “Add student” to add more students.
  + Staff click “Confirm”.
  + System adds student to the class and show success message.
* **Alternate Flow:**
  + In add student form Staff click “Add student” to add more student.
  + Staff choose students to add to the class.
  + Staff click “Confirm”.
  + System add students to the class and show success message.
* **Exception:**
  + System displays error notification if the form is not filled correctly.
  + System displays error notification if the student is already in the class.
  + System displays error notification if no student is selected.

##### 3.2.2.2.11 UC-23 Staff view student details

* **Description:** Staff view details of a student.
* **Actors:** Staff
* **Preconditions:**
  + Staff is logged in.
  + Student exists.
  + Student is in a class.
* **Post Conditions:** System show student details.
* **Flow of Events:**
  + Staff goes to EduBlock.
  + Staff login with username and password.
  + Staff click on “Classroom”.
  + Staff click on “Details” (icon) on actions column.
  + System shows class details.
  + Staff click “Students”.
  + System shows list of students of the class.
  + Staff click “Details” (icon) on actions column.
  + System shows student details.
* **Alternate Flow:**
  + System displays notification “No student found” if there is no student in the class.
* **Exception:**
  + System displays notification “No student found” if there is no student in the class.

##### 3.2.2.2.12 UC-24 Staff edit student information

* **Description:** Staff edit student’s information.
* **Actors:** Staff
* **Preconditions:**
  + Staff is logged in.
  + Student exists.
  + Student is in a class.
* **Post Conditions:** System edit student’s information.
* **Flow of Events:**
  + Staff goes to EduBlock.
  + Staff login with username and password.
  + Staff click on “Classroom”.
  + Staff click on “Details” (icon) on actions column.
  + System shows class details.
  + Staff click “Students”.
  + System shows list of students of the class.
  + Staff click “Details” (icon) on actions column.
  + System shows student details.
  + Staff click “Update” button.
  + A form will appear, staff then edit student’s information.
  + System edits student’s information and show success message.
* **Alternate Flow:**
  + Staff click “Reset” button in case they want to reset the form.
* **Exception:**
  + System displays error notification if the form is not filled correctly.

##### 3.2.2.2.13 UC-25 Staff remove student from a class

* **Description:** Staff remove student from a class.
* **Actors:** Staff
* **Preconditions:**
  + Staff is logged in.
  + Class exists.
  + Student exists.
  + Student is in a class.
* **Post Conditions:** System remove student from the class.
* **Flow of Events:**
  + Staff goes to EduBlock.
  + Staff login with username and password.
  + Staff click on “Classroom”.
  + Staff click on “Details” (icon) on actions column.
  + System shows class details.
  + Staff click “Students”.
  + System shows list of students of the class.
  + Staff click “Remove” (icon) on actions column.
  + System remove student from the class and show success message.
* **Alternate Flow:**
  + System displays error notification if the student is not in the class.
* **Exception:**
  + System displays error notification if the student is not in the class.

##### 3.2.2.2.14 UC-26 Staff view teacher of a class

* **Description:** Staff view list of teachers in a class.
* **Actors:** Staff
* **Preconditions:**
  + Staff is logged in.
  + Class exists.
* **Post Conditions:** System show list of teachers in a class.
* **Flow of Events:**
  + Staff goes to EduBlock.
  + Staff login with username and password.
  + Staff click on “Classroom”.
  + Staff click on “Details” (icon) on actions column.
  + System shows class details.
  + Staff click “Teachers”.
  + System shows list of teachers of the class.
* **Alternate Flow:**
  + System displays notification “No teacher found” if there is no teacher in the class.
* **Exception:**
  + System displays notification “No teacher found” if there is no teacher in the class.

##### 3.2.2.2.15 UC-27 Staff assign teacher(s) to a class

* **Description:** Staff assign teacher(s) to a class.
* **Actors:** Staff
* **Preconditions:**
  + Staff is logged in.
  + Class exists.
* **Post Conditions:** System assign teacher(s) to a class.
* **Flow of Events:**
  + Staff goes to EduBlock.
  + Staff login with username and password.
  + Staff click on “Classroom”.
  + Staff click on “Details” (icon) on actions column.
  + System shows class details.
  + Staff click “Teachers”.
  + System shows list of teachers of the class.
  + Staff click “Add teachers”.
  + A form will appear, staff then choose teacher and the subject they will teach.
  + Staff click “Confirm”.
  + System add teacher to the class and show success message.
* **Alternate Flow:**
  + Staff click “Add teacher” to add more teachers.
  + Staff choose teacher and subject.
  + Staff click “Confirm”.
  + System adds teachers to the class and show success message.
* **Exception:**
  + System displays error notification if the form is not filled correctly.
  + System displays error notification if the teacher is already in the class.
  + System displays error notification if no teacher is selected.
  + System displays error notification if no subject is selected.

##### 3.2.2.2.16 UC-28 Staff remove teacher from a class

* **Description:** Staff remove teacher from a class.
* **Actors:** Staff
* **Preconditions:**
  + Staff is logged in.
  + Class exists.
  + Teacher exists.
  + Teacher is in a class.
* **Post Conditions:** System remove teacher from the class.
* **Flow of Events:**
  + Staff goes to EduBlock.
  + Staff login with username and password.
  + Staff click on “Classroom”.
  + Staff click on “Details” (icon) on actions column.
  + System shows class details.
  + Staff click “Teachers”.
  + System shows list of teachers of the class.
  + Staff click “Remove” (icon) on actions column.
  + System remove teacher from the class and show success message.
* **Alternate Flow:**
  + System displays error notification if the teacher is not in the class.
* **Exception:**
  + System displays error notification if the teacher is not in the class.

##### 3.2.2.2.17 UC-29 Staff edit their profile

* **Description:** Staff edit their profile.
* **Actors:** Staff
* **Preconditions:**
  + Staff is logged in.
* **Post Conditions:** System edit staff’s profile.
* **Flow of Events:**
  + Staff goes to EduBlock.
  + Staff login with username and password.
  + Staff click on their avatar at bottom left of screen.
  + System display staff’s profile.
  + Staff click “Update” button.
  + A form will appear, staff then edit their information.
  + Staff click “Submit” button.
  + System edits staff’s information and show success message.
* **Alternate Flow:**
  + Staff click Change Password button to change their password.
  + A form will appear, staff then edit their password.
  + Staff click “Submit” button.
  + System edits staff’s password and show success message.
* **Exception:**
  + System displays error notification if the form is not filled correctly.

##### 3.2.2.2.18 UC-30 Staff print student record

* **Description:** Staff print student’s records.
* **Actors:** Staff
* **Preconditions:**
  + Staff is logged in.
  + Student exists.
* **Post Conditions:** System will save a file for staff to print student’s records.
* **Flow of Events:**
  + Staff goes to EduBlock.
  + Teacher login with username and password.
  + On classroom list Staff click on “Details” (icon) on actions column.
  + System show class details.
  + Staff click “Students”.
  + System shows list of students of the class.
  + Staff choose a student and click “Details” (icon) on actions column.
  + System shows student details.
  + Staff click “Print Record” button at Record session of student profile.
  + System shows print preview of student’s records.
  + Staff click “Save” button.
  + System saves student’s records to teacher’s computer.
  + Staff using printer to print student’s records.
* **Alternate Flow:**
  + On save, staff click “Cancel” button to cancel.

##### 3.2.2.2.19 UC-31 Staff get grade report and get classification report in a year

* **Description:** Staff get grade report or classification report in a year.
* **Actors:** Staff
* **Preconditions:**
  + Staff is logged in.
  + Class exists.
* **Post Conditions:** System export report to staff’s computer.
* **Flow of Events:**
  + Staff goes to EduBlock.
  + Staff login with username and password.
  + At dashboard Staff choose year and grade to get report.
  + Staff click “Get Report” button.
  + System ready to export report to staff’s computer.
  + Staff choose where to save export and click “Save”.
* **Alternate Flow:**
  + Staff click “Get Classification Report” to get classification report.
  + System ready to export classification report to staff’s computer.
  + Staff choose where to save export and click “Save”.
* **Exception:**
  + Button will be disabled if there is nothing to report.

##### 3.2.2.2.20 UC-32 Staff create statistic key

* **Description:** Staff create statistic key for third party’s member.
* **Actors:** Staff
* **Preconditions:**
  + Staff is logged in.
* **Post Conditions:** System create a statistic key.
* **Flow of Events:**
  + Staff goes to EduBlock.
  + Staff login with username and password.
  + Staff click on “Manage stats key list” on sidebar.
  + Staff choose grade and year.
  + Staff then click on “Create new Statistic key” button.
  + System creates statistic key and show it on the list.
* **Alternate Flow:**
  + System displays notification if the key can’t be created.
* **Exception:**
  + System displays notification if the key can’t be created.

#### 3.2.2.3 Teacher

|  |
| --- |
| Use case diagram for Teacher |

##### 3.2.2.3.1 UC-33 Teacher Login

* **Description:** Teacher login to EduBlock.
* **Actors:** Teacher
* **Preconditions:**
  + Teacher has an account.
* **Post Conditions:** System login teacher to EduBlock.
* **Flow of Events:**
  + Teacher goes to EduBlock.
  + Teacher goes to login page.
  + Teacher enters username and password.
  + Teacher clicks “Login” button.
  + System authorizes and login teacher to EduBlock.
* **Alternate Flow:**
  + System displays error notification if the username or password is incorrect.
* **Exception:**
  + System displays error notification if the username or password is incorrect.

##### 3.2.2.3.2 UC-34 Teacher view their profile

* **Description:** Teacher view their profile.
* **Actors:** Teacher
* **Preconditions:**
  + Teacher is logged in.
* **Post Conditions:** System display teacher’s profile.
* **Flow of Events:**
  + Teacher goes to EduBlock.
  + Teacher login with username and password.
  + Teacher clicks on their avatar at bottom left of screen.
  + System display teacher’s profile.

##### 3.2.2.3.3 UC-35 Teacher change their password

* **Description:** Teacher change their password.
* **Actors:** Teacher
* **Post Conditions:** System successfully change their password.
* **Flow of Events:**
  + Teacher goes to EduBlock.
  + Teacher login with username and password.
  + Teacher clicks on their avatar at bottom left of screen.
  + System display teacher’s profile.
  + Teacher clicks “Change Password” button.
  + System display form to change password.
  + Teacher enters old password, new password and confirm new password.
  + Teacher clicks “Submit” button.
  + System changes teacher’s password and show success message.
* **Alternate Flow:**
  + Teacher clicks on “X” button to cancel.
* **Exception:**
  + System displays error notification if the password is invalid.

##### 3.2.2.3.4 UC-36 Teacher view their classes

* **Description:** Teacher view their classes.
* **Actors:** Teacher
* **Preconditions:**
  + Teacher is logged in.
* **Post Conditions:** System display teacher’s classes.
* **Flow of Events:**
  + Teacher goes to EduBlock.
  + Teacher login with username and password.
  + On login success, system will redirect teach to dashboard where their classes will be display.
* **Alternate Flow:**
  + System displays notification “No class found” if there is no class.
* **Exception:**
  + System displays notification “No class found” if there is no class.

##### 3.2.2.3.5 UC-37 Teacher view class details

* **Description:** Teacher view class details.
* **Actors:** Teacher
* **Preconditions:**
  + Teacher is logged in.
  + Class exists.
  + Teacher is assigned to the class.
  + Teacher is in the class.
* **Post Conditions:** System display class details.
* **Flow of Events:**
  + Teacher goes to EduBlock.
  + Teacher login with username and password.
  + At dashboard Teacher choose a class and click on “Details” (icon) on actions column.
  + System shows class details.
* **Alternate Flow:**
  + System displays error notification if the class is not found.
* **Exception:**
  + System displays error notification if the class is not found.

##### 3.2.2.3.6 UC-38 Teacher view students in a class

* **Description:** Teacher view list of students in a class.
* **Actors:** Teacher
* **Preconditions:**
  + Teacher is logged in.
  + Class exists.
  + Teacher is assigned to the class.
  + Teacher is in the class.
* **Post Conditions:** System show list of students in a class.
* **Flow of Events:**
  + Teacher goes to EduBlock.
  + Teacher login with username and password.
  + On dashboard Teacher click on “Details” (icon) on actions column.
  + System shows class details.
  + Teacher clicks “Students”.
  + System shows list of students of the class.
* **Alternate Flow:**
  + System displays notification “No student found” if there is no student in the class.
* **Exception:**
  + System displays notification “No student found” if there is no student in the class.

##### 3.2.2.3.7 UC-39 Teacher view teachers in the class

* **Description:** Teacher view list of teachers who teach in the class.
* **Actors:** Teacher
* **Preconditions:**
  + Teacher is logged in.
  + Class exists.
  + Teacher is assigned to the class.
  + Teacher is in the class.
* **Post Conditions:** System show list of teachers in the class.
* **Flow of Events:**
  + Teacher goes to EduBlock.
  + Teacher login with username and password.
  + On dashboard Teacher click on “Details” (icon) on actions column.
  + System shows class details.
  + Teacher clicks “Teachers”.
  + System shows list of teachers of the class.
* **Alternate Flow:**
  + System displays notification “No teacher found” if there is no teacher in the class.
* **Exception:**
  + System displays notification “No teacher found” if there is no teacher in the class.

##### 3.2.2.3.8 UC-40 Teacher view Student details

* **Description:** Teacher view student details.
* **Actors:** Teacher
* **Preconditions:**
  + Teacher is logged in.
  + Student exists.
  + Teacher is assigned to the class.
  + Teacher is in the class.
  + Student is in the class.
* **Post Conditions:** System show student details.
* **Flow of Events:**
  + Teacher goes to EduBlock.
  + Teacher login with username and password.
  + On dashboard Teacher click on “Details” (icon) on actions column.
  + System shows class details.
  + Teacher clicks “Students”.
  + System shows list of students of the class.
  + Teacher clicks “Details” (icon) on actions column.
  + System shows student details.
* **Alternate Flow:**
  + System displays “Student not found” if no student in class.
* **Exception:**
  + System displays “Student not found” if no student in class.

##### 3.2.2.3.9 UC-41 Teacher print student’s records

* **Description:** Teacher export student’s records.
* **Actors:** Teacher
* **Preconditions:**
  + Teacher is logged in.
  + Student exists.
  + Teacher is assigned to the class.
  + Teacher is in the class.
  + Student is in the class.
* **Post Conditions:** System save student’s records to teacher’s computer.
* **Flow of Events:**
  + Teacher goes to EduBlock.
  + Teacher login with username and password.
  + On dashboard (*classroom list*) Teacher click on “Details” (icon) on actions column.
  + System shows class details.
  + Teacher clicks “Students”.
  + System shows list of students of the class.
  + Teacher clicks “Details” (icon) on actions column.
  + System shows student details.
  + Teacher scrolls down to “Record” section.
  + Teacher chooses class to print.
  + Teacher clicks “Print Record” button at Record session of student profile.
  + System shows print preview of student’s records.
  + Teacher clicks “Save” button.
  + System saves student’s records to teacher’s computer.
  + Teacher then uses printer to print saved file.
* **Alternate Flow:**
  + On save, teacher click “Cancel” to cancel save process.

##### 3.2.2.3.10 UC-42 Subject teacher change student’s score of their subject

* **Description:** Subject teacher change student’s score.
* **Actors:** Teacher
* **Preconditions:**
  + Teacher is logged in.
  + Student exists.
  + Teacher is assigned to the class.
  + Teacher is in the class.
  + Student is in the class.
* **Post Conditions:** Subject teacher successfully change student’s score of their subject.
* **Flow of Events:**
  + Teacher goes to EduBlock.
  + Teacher login with username and password.
  + On dashboard (*classroom list*) Teacher click on “Details” (icon) on actions column.
  + System shows class details.
  + Teacher clicks “Students”.
  + System shows list of students of the class.
  + Teacher clicks “Details” (icon) on actions column.
  + System shows student details.
  + Teacher clicks “Request Update” (icon) on Action column at Record session of student profile.
  + System show edit score form.
  + Teacher fills the form and click “Request” button.
  + System send request to edit student’s score.
* **Alternate Flow:**
  + System displays error notification if the form is not filled correctly.
* **Exception:**
  + System displays error notification if the form is not filled correctly.

##### 3.2.2.3.11 UC-43 Teacher upload Record using image

* **Description:** Teacher upload Record using image.
* **Actors:** Teacher
* **Preconditions:**
  + Teacher is logged in.
  + Student exists.
  + Teacher is assigned to the class.
  + Teacher is in the class.
  + Student is in the class.
* **Post Conditions:** System analyze and generate Record from the image.
* **Flow of Events:**
  + Teacher goes to EduBlock.
  + Teacher login with username and password.
  + On dashboard (*classroom list*) Teacher click on “Details” (icon) on actions column.
  + System shows class details.
  + Teacher clicks “Students”.
  + System shows list of students of the class.
  + Teacher clicks “Details” (icon) on actions column.
  + System shows student details.
  + Teacher clicks “Upload Record” button at Record session of student profile.
  + System shows upload methods.
  + Teacher chooses upload methods.
  + Teacher upload image.
  + System analyze and generate Record from the image.
  + Teacher clicks “Upload” button.
  + System saves the Record.
* **Alternate Flow:**
  + If system can’t recognize the image, it’s will show notification.
* **Exception:**
  + If system can’t recognize the image, it’s will show notification.

##### 3.2.2.3.12 UC-44 Teacher view list of pending Record’s Request and Approve or Reject

* **Description:** Teacher view list of pending record’s request.
* **Actors:** Teacher
* **Preconditions:**
  + Teacher is logged in.
  + Student exists.
  + Teacher is assigned to the class.
  + Teacher is in the class.
  + Student is in the class.
  + Subject teacher sent request to edit student’s grade.
* **Post Conditions:** System show list of pending record’s request.
* **Flow of Events:**
  + Teacher goes to EduBlock.
  + Teacher login with username and password.
  + Teacher clicks on “Pending Verification” button at sidebar.
  + System show list of pending record’s request.
  + Teacher clicks “Approve” (check icon) on Action column to approve the request.
  + System approves the request.
* **Alternate Flow:**
  + Teacher clicks “Reject” (close icon) on Action column to reject the request.
  + System rejects the request.
* **Exception:**
  + System displays “No pending request” if there is no pending request.

##### 3.2.2.3.13 UC-45 Teacher view history of student’s records changes

* **Description:** Teacher view history of student’s records changes.
* **Actors:** Teacher
* **Preconditions:**
  + Teacher is logged in.
  + Student exists.
  + Teacher is assigned to the class.
  + Teacher is in the class.
  + Student is in the class.
* **Post Conditions:** System show history of student’s records changes.
* **Flow of Events:**
  + Teacher goes to EduBlock.
  + Teacher login with username and password.
  + On dashboard (*classroom list*) Teacher click on “Details” (icon) on actions column.
  + System shows class details.
  + Teacher clicks “Students”.
  + System shows list of students in the class.
  + Teacher click “Details” (icon) on actions column.
  + System shows student details.
  + Teacher scrolls down to “Record” section.
  + Teacher hover on “History” button(icon) at Action columns of Record session.
  + System shows history of student’s records changes.
* **Alternate Flow:**
  + System displays nothing if there is no changes.
* **Exception:**
  + System displays nothing if there is no changes.

##### 3.2.2.3.14 UC-46 Teacher request to change student’s grade

* **Description:** Teacher request to change student’s grade.
* **Actors:** Teacher
* **Preconditions:**
  + Teacher is logged in.
  + Student exists.
  + Teacher is in the class.
  + Student is in the class.
* **Post Conditions:** Teacher successfully requests to change student’s grade.
* **Flow of Events:**
  + Teacher goes to EduBlock.
  + Teacher login with username and password.
  + On dashboard (*classroom list*) Teacher click on “Details” (icon) on actions column.
  + System shows class details.
  + Teacher clicks “Students”.
  + System shows list of students of the class.
  + Teacher clicks “Details” (icon) on actions column.
  + System shows student details.
  + Teacher clicks “Request Update” (icon) on Action column at Record session of student profile.
  + System show edit grade form.
  + Teacher fills the form and click “Request” button.
  + System send request to change student’s grade to subject teacher.
* **Alternate Flow:**
  + System displays error notification if the form is not filled correctly.
* **Exception:**
  + System displays error notification if the form is not filled correctly.

#### 3.2.2.4 Student

|  |
| --- |
| Use case diagram for Student |

##### 3.2.2.4.1 UC-47 Student login

* **Description:** Student can login to EduBlock.
* **Actors:** Student
* **Preconditions:** Student has an account.
* **Post Conditions:** EduBlock bring student to dashboard.
* **Flow of Events:**
  + Student goes to EduBlock.
  + Student click “Login” at homepage.
  + System redirect student to login page.
  + Student enter username and password.
  + System checks the credentials.
  + System bring student to dashboard.
* **Alternate Flow:**
  + System displays error notification if the credentials is not correct.
* **Exception:**
  + System displays error notification if the credentials is not correct.

##### 3.2.2.4.2 UC-48 Student view list of class they are in

* **Description:** Student view list of class they are in.
* **Actors:** Student
* **Preconditions:**
  + Student is logged in.
  + Student is in the class.
* **Post Conditions:** System show list of class they are in.
* **Flow of Events:**
  + Student goes to EduBlock.
  + Student login with username and password.
  + System bring student to dashboard.
  + Classes are listed on the dashboard.
* **Alternate Flow:**
  + System displays “No class” if there is no class.
* **Exception:**
  + System displays “No class” if there is no class.

##### 3.2.2.4.3 UC-49 Student view class details

* **Description:** Student view class details.
* **Actors:** Student
* **Preconditions:**
  + Student is logged in.
  + Student is in the class.
* **Post Conditions:** System show class details.
* **Flow of Events:**
  + Student goes to EduBlock.
  + Student login with username and password.
  + System bring student to dashboard.
  + Classes are listed on the dashboard.
  + Student click “Details” (icon) on actions column.
  + System shows class details.
* **Alternate Flow:**
  + System displays “No class” if there is no class.
* **Exception:**
  + System displays “No class” if there is no class.

##### 3.2.2.4.4 UC-50 Student view teachers in the class

* **Description:** Student view teachers in the class.
* **Actors:** Student
* **Preconditions:**
  + Student is logged in.
  + Student is in the class.
* **Post Conditions:** System show list of teachers in the class.
* **Flow of Events:**
  + Student goes to EduBlock.
  + Student login with username and password.
  + System bring student to dashboard.
  + Classes are listed on the dashboard.
  + Student click “Details” (icon) on actions column.
  + System shows class details.
  + Student click “Teachers”.
  + System shows list of teachers in the class.
* **Alternate Flow:**
  + System displays “No teacher” if there is no teacher.
* **Exception:**
  + System displays “No teacher” if there is no teacher.

##### 3.2.2.4.5 UC-51 Student view their profile

* **Description:** Student view their profile.
* **Actors:** Student
* **Preconditions:**
  + Student is logged in.
  + Student is in the class.
* **Post Conditions:** Student is able to view their profile.
* **Flow of Events:**
  + Student goes to EduBlock.
  + Student login with username and password.
  + System bring student to dashboard.
  + Student click on their avatar at bottom left corner.
  + System shows student details.

##### 3.2.2.4.6 UC-52 Student view their academic records

* **Description:** Student view their academic records.
* **Actors:** Student
* **Preconditions:**
  + Student is logged in.
  + Student is in the class.
* **Post Conditions:** System show their academic records.
* **Flow of Events:**
  + Student goes to EduBlock.
  + Student login with username and password.
  + System bring student to dashboard.
  + Student click on their avatar at bottom left corner.
  + System shows student details.
  + Student scroll down to “Record” section.
  + Student view their academic records.
* **Alternate Flow:**
  + At dashboard, student click on “View My Record” (icon) on actions column.
  + System shows their academic records.
  + Student view their academic records.
* **Exception:**
  + System displays “No record” if there is no record.

##### 3.2.2.4.7 UC-53 Student send request to ask for re-check their academic records

* **Description:** Student send request to ask for re-check their academic records.
* **Actors:** Student
* **Preconditions:**
  + Student is logged in.
* **Post Conditions:** System send request of student.
* **Flow of Events:**
  + Student goes to EduBlock.
  + Student login with username and password.
  + System bring student to dashboard.
  + Student click on their avatar at bottom left corner.
  + System shows student details.
  + Student scroll down to “Record” section.
  + Student click “Request record change” icon on action column.
  + System show form to send request.
  + Student fill the form and click “Submit”.
  + System sends request of student.
* **Alternate Flow:**
  + System display notification if the form is not filled correctly.
* **Exception:**
  + System display notification if the form is not filled correctly.

##### 3.2.2.4.8 UC-54 Student create key for parent to view their academic profile and records

* **Description:** Student create key for parent to view their academic profile and records.
* **Actors:** Student
* **Preconditions:**
  + Student is logged in.
* **Post Conditions:** System create a key for student.
* **Flow of Events:**
  + Student goes to EduBlock.
  + Student login with username and password.
  + System bring student to dashboard.
  + Student click on manage verified key.
  + Student click “Create key” button.
  + System creates a key for student.
  + The key is displayed on the screen.
  + Student copy the key and send it to parent.
  + Parent use the key to view student’s academic profile and records.
* **Alternate Flow:**
  + On entering the key, system displays “Invalid key” if the key is invalid.
* **Exception:**
  + On entering the key, system displays “Invalid key” if the key is invalid.

##### 3.2.2.4.9 UC-55 Student print their academic records

* **Description:** Student print their academic records.
* **Actors:** Student
* **Preconditions:**
  + Student is logged in.
* **Post Conditions:** System save the academic records file to student’s computer.
* **Flow of Events:**
  + Student goes to EduBlock.
  + Student login with username and password.
  + System bring student to dashboard (classroom list).
  + At dashboard Student click on “View my record icon” on actions column corresponding to the class.
  + System shows student details.
  + Student scroll down to “Record” section.
  + Student click “Print Record” button at bottom of the record table.
  + System shows print preview.
  + System saves the academic records file to student’s computer.
  + Student then use printer to print the file.
* **Alternate Flow:**
  + System will cancel the save file process if the student click “Cancel” button.
* **Exception:**
  + System will cancel the save file process if the student click “Cancel” button.

##### 3.2.2.4.10 UC-56 Student view history of their academic record’s changes

* **Description:** Student view history of their academic record’s changes.
* **Actors:** Student
* **Preconditions:**
  + Student is logged in.
* **Post Conditions:** System show history of their academic record’s changes.
* **Flow of Events:**
  + Student goes to EduBlock.
  + Student login with username and password.
  + At dashboard Student click on “View my record icon” on actions column corresponding to the class.
  + System shows student details.
  + Student scroll down to “Record” section.
  + Student hover on “History” icon on action column.
  + System show history of their academic record’s changes.
* **Alternate Flow:**
  + System displays nothing if there are no changes.
* **Exception:**
  + System displays nothing if there are no changes.

#### 3.2.2.5 Third Party

|  |
| --- |
| Use case diagram for Third Party |

##### 3.2.2.5.1 UC-57 Third party view student’s academic profile and records

* **Description:** Third party view student’s academic profile and records by using verified key that student created without logging in.
* **Actors:** Third party
* **Preconditions:**
  + Third party has the key.
* **Post Conditions:** System show student’s academic profile and records for third party’s member.
* **Flow of Events:**
  + Third party goes to EduBlock.
  + Third party enter the key.
  + System shows student’s academic profile and records for third party’s member.
* **Alternate Flow:**
  + System displays “Invalid key” if the key is invalid.
* **Exception:**
  + System displays “Invalid key” if the key is invalid.

##### 3.2.2.5.2 UC-58 Third party’s member view statistics of a grade in a year

* **Description:** Third party’s member view statistics of a grade in a year by using verified statistic key given by admin or staff.
* **Actors:** Third party’s member
* **Preconditions:**
  + Staff or admin created the key.
  + Third party’s member has the key.
* **Post Conditions:** System show statistics of a grade in a year for third party’s member.
* **Flow of Events:**
  + Third party’s member goes to EduBlock.
  + Third party’s member go to “Statistics key verification” page.
  + Third party’s member enter the key.
  + System shows statistics of a grade in a year for third party’s member.
* **Alternate Flow:**
  + System displays “Invalid key” if the key is invalid.
* **Exception:**
  + System displays “Invalid key” if the key is invalid.

## 3.3 Functional Requirements

### 3.3.1 System Functional Overview

The system is designed to provide a platform for school to manage their student’s record, information with high security, fast and private.

### 3.3.2 Features

#### 3.3.2.1 **Account Features**

* **User Login**
  + Use cases: UC-1, UC-13, UC-32, UC-47
  + Description: The system shall allow user to login to EduBlock using their account.
* **Create Account**
  + Use cases: UC-4
  + Description: System shall allow admin to create account for staff, teacher, student, parent.
* **View list of all accounts**
  + Use cases: UC-2, UC-14
  + Description: System shall allow admin and staff to view list of accounts.
* **View account’s detail**
  + Use cases: UC-3, UC-15, UC-23, UC-34, UC-40, UC-51
  + Description: The system shall allow user to view account’s detail.
* **Update profile**
  + Use cases: UC-6, UC-24, UC-29
  + Description: The system shall allow only admin and staff to update account’s profile.
* **Update password**
  + Use cases: UC-7, UC-29, UC-34
  + Description: The system shall allow user to reset their password.

#### 3.3.2.2 **Class Features**

* **View classroom list**
  + Use cases: UC-10, UC-17, UC-36, UC-48
  + Description: The system shall allow user to view list of classes filter by their role.
* **View classroom detail**
  + Use cases: UC-11, UC-19, UC-37, UC-49
  + Description: The system shall allow user to view class detail.
* **View student in class**
  + Use cases: UC-11, UC-21, UC-38
  + Description: The system shall allow user who have the right to view list of students in class.
* **View teacher in class**
  + Use cases: UC-11, UC-26, UC-39, UC-50
  + Description: The system shall allow user who have the right to view list of teachers in class.
* **Create classroom**
  + Use cases: UC-18
  + Description: The system shall allow staff to create new class
* **Update classroom detail**
  + Use cases: UC-20
  + Description: The system shall allow only staff to update class detail.
* **Add student to class**
  + Use cases: UC-22
  + Description: The system shall allow only staff to add student to class.
* **Assign teacher to class**
  + Use cases: UC-27
  + Description: The system shall allow only staff to assign teacher to class.
* **Remove student from class**
  + Use cases: UC-25
  + Description: The system shall allow only staff to remove student from class.
* **Remove teacher from class**
  + Use cases: UC-28
  + Description: The system shall allow only staff to remove teacher from class. #### **Record Features**
* **View student’s record** (view, print)
  + Use cases: UC-23, UC-30, UC-40, UC-41, UC-51, UC-52, UC-55
  + Description: The system shall allow user to view student’s record and print the record.
* **Send request to change or re-check student’s record**
  + Use cases: UC-46, UC-53
  + Description: The system shall allow student and homeroom teacher to send request to re-check or change their record.
* **View list of pending change requests** (view, approve, reject)
  + Use cases: UC-44
  + Description: The system shall allow homeroom teacher to view list of pending change requests and approve or reject the request.
* **View history of record’s changes**
  + Use cases: UC-45, UC-56
  + Description: The system shall allow user to view history of record’s changes.
* **Subject teacher changes their subject score**
  + Use cases: UC-42
  + Description: The system shall allow subject teacher to change their subject score on student’s record.

#### 3.3.2.3 **Student Key Features**

* **Student create verified key**
  + Use cases: UC-51
  + Description: The system shall allow student to create verified key for their parents to use it to view their academic profile and records.
* **Third party’s member view student’s academic profile and records**
  + Use cases: UC-57
  + Description: The system shall allow third party’s member to view student’s academic profile and records by using verified key given by the student.
* **Third party’s member view statistics of a grade in a year**
  + Use cases: UC-58
  + Description: The system shall allow third party’s member to view statistics of a grade in a year by using verified statistic key given by admin or staff.

## 3.4 Non-Functional Requirements

### 3.4.1 External Interfaces

#### 3.4.1.1 **User Interfaces**

* **UI-1:** The system shall provide a user interface for admin manage all accounts.
* **UI-2:** The system shall provide a user interface for staff to manage classes.
* **UI-3:** The system shall provide a user interface for teacher to view classes and manage students, student’s records.
* **UI-4:** The system shall provide a user interface for teacher to view list of requests to change student’s records.
* **UI-5:** The system shall provide a user interface for student to view classes and view their records.
* **UI-6:** The system shall provide a user interface for parent to view their children’s records.
* **UI-7:** The system shall provide a user interface for student to generate private key for their parents to view their records.
* **UI-8:** The system shall permit complete access to the system via a web browser.
* **UI-9:** The web-application shall permit complete navigation.
* **UI-10:** The web-application shall permit complete all functions.

#### 3.4.1.2 **Hardware Interfaces**

* **HI-1:** The web-app shall be able to run on any device that can run a web browser.
* **HI-2:** Graphic card is required to upload student’s academic record using image file.

#### 3.4.1.3 **Software Interfaces**

* **SI-1:** Hyperledger Fabric network.
  + **SI-1.1:** The system shall initialize decentralized network using Mini-fabric smoothly.
  + **SI-1.2:** The network shall install chaincode in all peers smoothly.
* **SI-2:** EduBlock client
* The request server shall communicate with user interface through API to perform following operations:
  + **SI-2.1:** The system shall allow user to login.
  + **SI-2.2:** The system shall allow user to view their profile.
  + **SI-2.3:** The system shall allow user to reset their password.
  + **SI-2.4:** The system shall allow Admin to perform CRUD operations on account.
  + **SI-2.5:** The system shall allow Staff to perform CRU operations on classes.
  + **SI-2.6:** Teacher to send request to change student’s academic record.
  + **SI-2.7:** Teacher to approve or reject request to change student’s academic record.
  + **SI-2.8:** Student to send request to re-check student’s academic record.
  + **SI-2.9:** Student to upload student’s academic record.

### 3.4.2 Quality Attributes

Our application ensures the following quality attributes:

* **Usability:**
  + The application is easy to use and understand.
  + The application is designed to be intuitive.
  + The application is designed to be used by admin, staff, teacher, student, and third-party’s member.
* **Reliability:**
  + The application is designed to be reliable.
  + The application is designed to be used with blockchain technology to ensure data integrity.
* **Performance:** The application is designed to be fast and responsive.
* **Security:** The application is designed to be secure. The application is designed to be used with blockchain technology to ensure data integrity.
* **Maintainability:** The application is designed to be easy to maintain, update, and extend.
* **Portability:** The application is designed to be portable.
* **Scalability:** The application is designed to be scalable and can be extended to support more users and more features.
* **Interoperability:** The application is designed to be interoperable with other applications.
* **Reusability:** The application is designed to be reusable.
* **Testability:** The application is designed to be easy to test.

# 4. Software Design Description

## 4.1 Overall Description

### 4.1.1 Assumptions

* The target platform is a Docker-compatible operating system (Preferably Linux).
* The target web browser is Google Chrome.
* All products are run and operated in the same machine (Monolithic architecture).
* Only administrators can access the running system. Other users can only access the system through the frontend server.
* The blockchain is a private blockchain.
* The blockchain is an optional feature that can be turned on or off.
* The database is a H2 database.
* The mode of the database can be in-memory (for testing), file or remote (for production).

### 4.1.2 Design Constraints

* The backend system is a REST API server.
* The backend system is a Java application.
* The blockchain is a Hyperledger Fabric blockchain.
* There should be an option to turn on or off the blockchain. If the blockchain is turned off, the backend system should still work as a normal REST API server with a local database.

### 4.1.3 Technology Suggestion

* The endpoints of the backend system can be exposed so that a node browser can be developed to search & access the endpoints and get the necessary information.

## 4.2 System Architecture Design

### 4.2.1 Overall Architecture

|  |
| --- |
| Figure 4.1: Overall architecture |

| Component | Description |
| --- | --- |
| Chain Node (CN) | A node of the blockchain. This stores the records and handles the history and transaction requests from the Request Server (Change/View the score, information, etc.) |
| Request Server | The off-chain backend of a CN. This stores the pending requests from the user and is the only way to call a request to the CN. Each Request Server may have a different way to handle user requests (Voting, Direct Request, etc.) |
| Frontend Server | Provide the UX/UI for interacting with the Request Server |

### 4.2.2 System Architecture

### 4.2.3 Package Diagram

|  |
| --- |
| Figure 4.2: Package Diagram of Request Server |

| Package Name | Description |
| --- | --- |
| root | Main classes |
| api | The abstract classes & interfaces |
| entity | The entities of the database |
| handler | The handlers of the endpoints of the REST API server |
| internal | Internal classes used by other packages |
| internal/student | The instances of the Student Updater |
| model | The input / output objects |
| model/input | The input objects for the handlers |
| model/output | The output objects returned from the handlers |
| model/fabric | The models used internally by the student updater |

## 4.3 System Detailed Design

### 4.3.1 Class Specification

|  |
| --- |
| Figure 4.3: Class Diagram of the Request Server |

#### 4.3.1.1 Account

| Field Name | Type | Description |
| --- | --- | --- |
| id | long | The account id |
| username | String | The username |
| hashedPassword | String | The hashed password |
| salt | String | The salt of the password |
| role | String | The role of the account |
| createdAt | Date | The date when the account was created |
| classrooms | List ClassTeacher | The list of references to the classrooms that the account participates if its role is Teacher |
| recordEntries | List RecordEntry | The list of record entries related to the subjects that the account is teaching if its role is Teacher |
| requestedRecordEntries | List RecordEntry | The list of verified record entries that the account requested to changes |
| approvedRecordEntries | List RecordEntry | The list of verified record entries that the account accepted |
| pendingRecordEntries | List PendingRecordEntry | The list of pending record entries related to the subjects that the account is teaching if its role is Teacher |
| requestedPendingRecordEntries | List PendingRecordEntry | The list of pending record entries that the account requested to changes |
| homeClassrooms | List Classroom | The list of classrooms the the account is a homeroom teacher at |

#### 4.3.1.2 Profile

| Field Name | Type | Description |
| --- | --- | --- |
| id | long | The account id |
| account | Account | The reference to the Account object |
| firstName | String | The first name |
| lastName | String | The last name |
| male | boolean | Is the person male? false if she is a female |
| avatar | String | The link to the avatar image |
| birthDate | Date | The date of the birthday |
| address | String | The address |
| phone | String | The phone number |
| email | String | The email |
| updated | boolean | The flag indicates that the profile requires synchronization with the Chain Node |

#### 4.3.1.3 Student

| Field Name | Type | Description |
| --- | --- | --- |
| id | long | The account id |
| account | Account | The reference to the Account object |
| ethnic | String | The ethnic of the student |
| fatherName | String | The name of the father of the student |
| fatherJob | String | The job of the father of the student |
| motherName | String | The name of the mother of the student |
| motherJob | String | The job of the mother of the student |
| guardianName | String | The name of the guardian of the student |
| guardianJob | String | The job of the guardian of the student |
| homeTown | String | The home town of the student |
| classrooms | List ClassStudent | The list of references to the classrooms that the student participates |
| records | List Record | The list of records related to the classrooms that the student participates |
| updaterKey | List UpdaterKey | The list of updater keys of the student. Used to allow outsiders to get information of the student. |

#### 4.3.1.4 Classroom

| Field Name | Type | Description |
| --- | --- | --- |
| id | long | The classroom id |
| name | String | The name of the classroom |
| grade | int | The grade of the classroom |
| year | int | The year of the classroom |
| homeroomTeacher | Account | The reference to the homeroom teacher of the classroom |
| students | List ClassStudent | The list of references to the students that participate in the classroom |
| teachers | List ClassTeacher | The list of references to the teachers that participate in the classroom |
| records | List Record | The list of records related to the classroom |

#### 4.3.1.5 ClassStudent

| Field Name | Type | Description |
| --- | --- | --- |
| id | long | The id of the reference |
| classroom | Classroom | The reference to the classroom |
| student | Student | The reference to the student |

#### 4.3.1.6 ClassTeacher

| Field Name | Type | Description |
| --- | --- | --- |
| id | long | The id of the reference |
| classroom | Classroom | The reference to the classroom |
| teacher | Account | The reference to the teacher |
| subjectId | long | The id of the subject that the teacher teaches |

#### 4.3.1.7 Record

| Field Name | Type | Description |
| --- | --- | --- |
| id | long | The record id |
| classroom | Classroom | The reference to the classroom |
| student | Student | The reference to the student |
| recordEntry | List RecordEntry | The list of verified record entries related to the record |
| pendingRecordEntry | List PendingRecordEntry | The list of pending record entries related to the record |

#### 4.3.1.8 RecordEntry

| Field Name | Type | Description |
| --- | --- | --- |
| id | long | The record entry id |
| subjectId | long | The id of the subject that the record entry is related to |
| firstHalfScore | int | The score of the first semester of the subject |
| secondHalfScore | int | The score of the second semester of the subject |
| finalScore | int | The final score of the subject |
| requestDate | Date | The date when the record entry was requested |
| approvalDate | Date | The date when the record entry was approved |
| updateComplete | boolean | The flag indicates that the record entry was updated to the Chain Node |
| teacher | Account | The reference to the teacher that teaches the subject |
| requester | Account | The reference to the account that requested the record entry |
| approver | Account | The reference to the account that approved the record entry |
| record | Record | The reference to the record that the record entry is related to |

#### 4.3.1.9 PendingRecordEntry

| Field Name | Type | Description |
| --- | --- | --- |
| id | long | The pending record entry id |
| subjectId | long | The id of the subject that the pending record entry is related to |
| firstHalfScore | int | The score of the first semester of the subject |
| secondHalfScore | int | The score of the second semester of the subject |
| finalScore | int | The final score of the subject |
| requestDate | Date | The date when the pending record entry was requested |
| teacher | Account | The reference to the teacher that teaches the subject |
| requester | Account | The reference to the account that requested the pending record entry |
| record | Record | The reference to the record that the pending record entry is related to |

#### 4.3.1.10 UpdaterKey

| Field Name | Type | Description |
| --- | --- | --- |
| id | String | The unique key |
| student | Student | The reference to the student that the key is related to |

#### 4.3.1.11 StatisticKey

| Field Name | Type | Description |
| --- | --- | --- |
| id | String | The unique key |
| year | int | The year that the key is referred to |
| grade | int | The grade that the key is referred to |

### 4.3.2 Sequence Diagram

#### 4.3.2.1 Create Account



#### 4.3.2.2 Get Account



#### 4.3.2.3 Get Account List



#### 4.3.2.4 Login



#### 4.3.2.5 Update Account Password



#### 4.3.2.6 Update Account Profile



#### 4.3.2.7 Create Classroom



#### 4.3.2.8 Add Students To Classroom



#### 4.3.2.9 Remove Students From Classroom



#### 4.3.2.10 Add Teachers To Classroom



#### 4.3.2.11 Remove Teachers From Classroom



#### 4.3.2.12 Update Classroom



#### 4.3.2.13 Get Classroom



#### 4.3.2.14 Get Classroom List



#### 4.3.2.15 Get Students In Classroom



#### 4.3.2.16 Get Teachers In Classroom



#### 4.3.2.17 Get Student Record



#### 4.3.2.18 Get Student Record List



#### 4.3.2.19 Update Student Record



#### 4.3.2.20 Create Request To Update Student Record



#### 4.3.2.21 Get Pending Record Requests



#### 4.3.2.22 Approve Pending Record Request



#### 4.3.2.23 Upload Record To Chain Node



#### 4.3.2.24 Create Statistic Key



#### 4.3.2.25 Get Statistic Key List



#### 4.3.2.26 Delete Statistic Key



#### 4.3.2.27 Get Statistic Data



#### 4.3.2.28 Create Student Key



#### 4.3.2.29 Get Student Key List



#### 4.3.2.30 Delete Student Key



#### 4.3.2.31 Get Student Data From Key



#### 4.3.2.32 Upload Legacy Student Record



## 4.4 Data & Database Design

### 4.4.1 Database Design

|  |
| --- |
| Figure 4.4: Database Design of the Request Server |

#### 4.4.1.1 Account

| Field Name | Type | Size | Unique | Not Null | Flag | Notes |
| --- | --- | --- | --- | --- | --- | --- |
| ID | bigint |  | x | x | PK |  |
| USERNAME | character varying | 255 | x | x |  |  |
| HASHEDPASSWORD | character varying | 255 |  | x |  |  |
| SALT | character varying | 255 |  | x |  |  |
| ROLE | character varying | 255 |  | x |  |  |
| CREATEDAT | timestamp |  |  | x |  |  |

#### 4.4.1.2 Profile

| Field Name | Type | Size | Unique | Not Null | Flag | Notes |
| --- | --- | --- | --- | --- | --- | --- |
| ACCOUNT\_ID | bigint |  | x | x | PK, FK |  |
| ADDRESS | character varying | 255 |  | x |  |  |
| AVATAR | character varying | 255 |  | x |  |  |
| BIRTHDATE | timestamp |  |  | x |  |  |
| EMAIL | character varying | 255 |  | x |  |  |
| FIRSTNAME | character varying | 255 |  | x |  |  |
| LASTNAME | character varying | 255 |  | x |  |  |
| MALE | boolean |  |  | x |  |  |
| PHONE | character varying | 255 |  | x |  |  |
| UPDATED | boolean |  |  | x |  | Used internally by student updater |

#### 4.4.1.3 Student

| Field Name | Type | Size | Unique | Not Null | Flag | Notes |
| --- | --- | --- | --- | --- | --- | --- |
| ACCOUNT\_ID | bigint |  | x | x | PK, FK |  |
| ETHNIC | character varying | 255 |  | x |  |  |
| FATHERJOB | character varying | 255 |  | x |  |  |
| FATHERNAME | character varying | 255 |  | x |  |  |
| GUARDIANJOB | character varying | 255 |  | x |  |  |
| GUARDIANNAME | character varying | 255 |  | x |  |  |
| HOMETOWN | character varying | 255 |  | x |  |  |
| MOTHERJOB | character varying | 255 |  | x |  |  |
| MOTHERNAME | character varying | 255 |  | x |  |  |

#### 4.4.1.4 Classroom

| Field Name | Type | Size | Unique | Not Null | Flag | Notes |
| --- | --- | --- | --- | --- | --- | --- |
| ID | bigint |  | x | x | PK |  |
| NAME | character varying | 255 |  | x |  |  |
| GRADE | character varying | 255 |  | x |  |  |
| HOMEROOMTEACHER\_ID | bigint |  |  | x | FK |  |
| START\_YEAR | integer |  |  | x |  |  |

#### 4.4.1.5 Class Student

| Field Name | Type | Size | Unique | Not Null | Flag | Notes |
| --- | --- | --- | --- | --- | --- | --- |
| ID | bigint |  | x | x | PK |  |
| CLASSROOM\_ID | bigint |  |  | x | FK |  |
| STUDENT\_ID | bigint |  |  | x | FK |  |

#### 4.4.1.6 Class Teacher

| Field Name | Type | Size | Unique | Not Null | Flag | Notes |
| --- | --- | --- | --- | --- | --- | --- |
| ID | bigint |  | x | x | PK |  |
| CLASSROOM\_ID | bigint |  |  | x | FK |  |
| TEACHER\_ID | bigint |  |  | x | FK |  |
| SUBJECTID | bigint |  |  | x |  | Defined in the system’s config |

#### 4.4.1.7 Record

| Field Name | Type | Size | Unique | Not Null | Flag | Notes |
| --- | --- | --- | --- | --- | --- | --- |
| ID | bigint |  | x | x | PK |  |
| CLASSROOM\_ID | bigint |  |  | x | FK |  |
| STUDENT\_ACCOUNT\_ID | bigint |  |  | x | FK |  |

#### 4.4.1.8 Record Entry

| Field Name | Type | Size | Unique | Not Null | Flag | Notes |
| --- | --- | --- | --- | --- | --- | --- |
| ID | bigint |  | x | x | PK |  |
| RECORD\_ID | bigint |  |  | x | FK |  |
| REQUESTER\_ID | bigint |  |  |  | FK |  |
| TEACHER\_ID | bigint |  |  |  | FK |  |
| APPROVER\_ID | bigint |  |  |  | FK |  |
| APPROVALDATE | timestamp |  |  | x |  |  |
| REQUESTDATE | timestamp |  |  | x |  |  |
| FIRSTHALFSCORE | double precision |  |  | x |  |  |
| SECONDHALFSCORE | double precision |  |  | x |  |  |
| FINALSCORE | double precision |  |  | x |  |  |
| SUBJECTID | bigint |  |  | x |  | Defined in the system’s config |
| UPDATECOMPLETE | boolean |  |  | x |  | Used internally by student updater |

#### 4.4.1.9 Pending Record Entry

| Field Name | Type | Size | Unique | Not Null | Flag | Notes |
| --- | --- | --- | --- | --- | --- | --- |
| ID | bigint |  | x | x | PK |  |
| RECORD\_ID | bigint |  |  | x | FK |  |
| REQUESTER\_ID | bigint |  |  | x | FK |  |
| TEACHER\_ID | bigint |  |  | x | FK |  |
| REQUESTDATE | timestamp |  |  | x |  |  |
| FIRSTHALFSCORE | double precision |  |  | x |  |  |
| SECONDHALFSCORE | double precision |  |  | x |  |  |
| FINALSCORE | double precision |  |  | x |  |  |
| SUBJECTID | bigint |  |  | x |  | Defined in the system’s config |

#### 4.4.1.10 Updater Key

| Field Name | Type | Size | Unique | Not Null | Flag | Notes |
| --- | --- | --- | --- | --- | --- | --- |
| ID | character varying | 255 | x | x | PK |  |
| STUDENT\_ACCOUNT\_ID | bigint |  |  | x | FK |  |

#### 4.4.1.11 Statistic Key

| Field Name | Type | Size | Unique | Not Null | Flag | Notes |
| --- | --- | --- | --- | --- | --- | --- |
| ID | character varying | 255 | x | x | PK |  |
| GRADE | integer |  |  | x |  |  |
| START\_YEAR | integer |  |  | x |  |  |

### 4.4.2 Data File Design

| File Name | Type | Notes |
| --- | --- | --- |
| db | Folder | The folder of The H2 Database files |
| updater | Folder | Contains the data files of the local student updater |

# 5. Software Testing Documentation

## 5.1 Overall Description

### 5.1.1 Test Model

We apply the V-model in our project, which is a development of the waterfall model. Testing is carried out concurrently with the software development cycle in the V-model, where a testing phase corresponds to a phase of software development.

### 5.1.2 Testing Levels

About the Testing levels in our project, we apply all those levels including Unit testing, Integration testing, System testing and Acceptance testing.

With Unit testing, we test each small module in the system, each class and function.

With Integration testing is a type of testing in which individual software modules or functions are logically integrated and tested in groups together. For instance, we can test the interoperability of two functions, add 1 item and search for the item to see if they interact well with each other, after successfully creating an item, we can proceed to search for the newly created item. or not.

System testing is the last test phase to determine whether the system is about to deliver satisfying the requirements and goals. It tests the whole functionality and interface of the system. For instance, a database test for a system test is used to see if the data displayed on the system matches the data in the database.

Finally, with Acceptance Test, similar to System Test but usually tested by customers, the purpose is to see if the software meets the customer’s requirements or not.

### 5.1.3 Testing Types

Functional testing is checking if the system is working according to the business requirements and is performed in every level of testing. Non-Functional testing is similar to Functional testing in that both occur in all levels of testing. Non-functional testing is primarily concerned with the software’s other features, such as its security and if data is exposed by straightforward queries in any input field. Structural testing is often considered a type of white box testing. Instead than focusing on the software’s functionality, this method examines what is happening inside the program. Structural testing is also applicable at all testing levels. Changes Testing is done to determine whether or not the program is functioning correctly after bugs have been fixed.

## 5.2 Test Plan

### 5.2.1 Test Stages

[Report\_Test-Stages.xlsx](excel/Report_Test-Stages.xlsx)

### 5.2.2 Resources

#### 5.2.2.1 Human Resources

| Worker/ Doer | Role | Specifice Responsibilities/Comments |
| --- | --- | --- |
| TienHQ | BE-Tester | Test if the Request Server is working properly or not |
| TuLX | FE-Tester | Test if the ChainCode is working properly or not |
| KhoaND | Reporter | Handling reports related work |
| UyCHA | BE-Tester | Test if the UI is working properly or not |
| KhoiNM | FE-Tester | Test if the UI is working properly or not |

#### 5.2.2.2 Environment

None

### 5.2.3 Test Milestones

| Milestone Task | Effort (md) | Start Date | End Date |
| --- | --- | --- | --- |
| ChainCode | 6 | Nov 9, 2022 | Nov 14, 2022 |
| Request Server | 20 | Nov 15, 2022 | Dec 04, 2022 |

### 5.2.4 Deliverables

| No | Deliverables | Due Date |
| --- | --- | --- |
| 1 | Test Design | Nov 11, 2022 |
| 2 | ChainCode Test script | Nov 14, 2022 |
| 3 | Request Server Test script | Dec 15, 2022 |
| 4 | Test results | Dec 16, 2022 |

## 5.3 Test Cases

• Unit Test Cases: [Report\_Unit-Test-Case.xlsx](excel/Report_Unit-Test-Case.xlsx)

• Other Test Cases: [Report\_Test-Case-Document.xlsx](excel/Report_Test-Case-Document.xlsx)

## 5.4 Test Reports

Test Report has been fully integrated in Unit\_Test-Case and Test-Case-Document.

# 6. Release Package & User Guides

## 6.1 Deliverable Package

### 6.1.1 Source Codes & Documents

| Items | Sub-Items | Type | Version |
| --- | --- | --- | --- |
| EduBlock | EduBlock Client | Code | 1.0 |
| Request Server | EduBlock Backend | Code | 3.0 |
| Blockchain Network | Blockchain Chaincode | Code | 1.0 |
| OCR | Record Table Processor | Code | 1.0 |
| Requirement | SRS.docx | Document | 1.0 |
| Deployment | UserGuide.docx | Document | 1.0 |
| Final Report | FinalReport.docx | Document | 1.0 |

### 6.1.2 Known Issues, Limitations & Restrictions

#### 6.1.2.1 Limitations

* Verified records coundn’t be synchronized between nodes.
* Verified keys for third-party can only be used in the node it was created.
* Currently network have ability to provide verified data with blockchain technology, and a node can be used as restore point if database of other nodes missing.

## 6.2 Installation Guides

### 6.2.1 System Requirements

OS: any

CPU: at least 4 cores

RAM: at least 4Gb

NETWORK: required

SOFTWARE: Docker

### 6.2.2 Setup Files

* docker-compose.yml
* .env

### 6.2.3 Installation Instruction

* Update value in the .env as required
* Run command docker-compose up to deploy program
* There are many solution to deploy a network. In this scenario, we deploy using virtual network. Other solution can be found [here](https://hyperledger-fabric.readthedocs.io/en/release-2.5/deployment_guide_overview.html)

## 6.3 User Manual

### 6.3.1 Terms & Definitions

| No. | Term | Definition |
| --- | --- | --- |
| 01 | FT | Feature |
| 02 | R.GUEST | Guest, Third-party |
| 03 | R.ADM | Admin |
| 04 | R.STF | Staff |
| 05 | R.TCH | Teacher |
| 06 | R.STD | Student |
| 07 | R.AUTH | Any authenticated role |
| 08 | R.MAN | Admin or Staff |
| 09 | R.USR | Teacher or Student |
| 10 | R.PRV | Personal |

### 6.3.2 System Requirements

OS: any

CPU: any

RAM: at least 1Gb

NETWORK: required

### 6.3.3 Application Usage

#### 6.3.3.1 Overview

| No. | Feature | R.ADM | R.STF | R.TCH | R.STD | R.GUEST | Note |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 01 | Login | x | x | x | x |  |  |
| 02 | View list of all accounts | x | x |  |  |  |  |
| 03 | View details of an account | x | x |  |  |  |  |
| 04 | Create new accounts | x |  |  |  |  |  |
| 05 | Search account | x | x |  |  |  |  |
| 06 | Update own profile | x | x | x | x |  |  |
| 07 | Change others account password | x |  |  |  |  |  |
| 08 | Get classification report of year and grade | x | x |  |  |  |  |
| 09 | Get report of a classroom | x | x | x |  |  |  |
| 10 | View list of all classrooms | x | x |  |  |  |  |
| 11 | View details of a classroom | x | x | x | x |  |  |
| 12 | Create new classroom |  | x |  |  |  |  |
| 13 | Update details of a classroom |  | x |  |  |  |  |
| 14 | View list of students of a classroom | x | x | x |  |  |  |
| 15 | Assign multiple students to a classroom |  | x |  |  |  |  |
| 16 | View details of a student | x | x | x |  |  |  |
| 17 | Update details of a student |  | x |  |  |  |  |
| 18 | Remove a student from classroom |  | x |  |  |  |  |
| 19 | View list of teachers of a classroom | x | x | x | x |  |  |
| 20 | Assign multiple teachers to a classroom |  | x |  |  |  |  |
| 21 | Remove a teacher from classroom |  | x |  |  |  |  |
| 22 | Print record table of a student | x | x | x | x |  |  |
| 23 | View own profile | x | x | x | x |  |  |
| 24 | Update own password | x | x | x | x |  |  |
| 25 | View teaching classrooms |  |  | x |  |  |  |
| 26 | Request update record of student |  |  | x | x |  |  |
| 27 | Request update record of student by image |  |  | x | x |  |  |
| 28 | Update a record of student |  |  | x |  |  |  |
| 29 | View list of pending record’s update requests |  |  | x |  |  |  |
| 30 | Approve or reject record update request |  |  | x |  |  |  |
| 31 | View update history of a record of student | x | x | x | x |  |  |
| 32 | View list of classroom taking part in |  |  |  | x |  |  |
| 33 | View own records |  |  |  | x |  |  |
| 34 | Create key to view records of a student |  |  |  | x |  |  |
| 35 | View list of keys to view records of a student |  |  |  | x |  |  |
| 36 | Remove a key to view records of a student |  |  |  | x |  |  |
| 37 | Create key to view records of students | x | x |  |  |  |  |
| 38 | View list of keys to view records of students | x | x |  |  |  |  |
| 39 | Remove a key to view records of students | x | x |  |  |  |  |
| 40 | View verified records of a student | x | x | x | x | x |  |
| 41 | View verified records of students | x | x | x | x | x |  |

#### 6.3.3.2 Login

| Step | Description | Effect | Figure |
| --- | --- | --- | --- |
| 01 | Click Login in the homepage | Navigate to login page | [Figure 6.1](#fig-ug-01-step1) |
| 02 | Fill the form with provided credential |  | [Figure 6.2](#fig-ug-01-step2) |
| 03 | Click Login to submit the credential | Navigate to dashboard if credential is valid | [Figure 6.3](#fig-ug-01-step3) |

|  |
| --- |
| Figure 6.1: Step 1 |

|  |
| --- |
| Figure 6.2: Step 2 |

|  |
| --- |
| Figure 6.3: Step 3 |

|  |
| --- |
| Figure 6.4: Result |

#### 6.3.3.3 View list of all accounts

| Step | Description | Effect | Figure |
| --- | --- | --- | --- |
| 01 | Click on Account in the left sidebar | Navigate to the account list page | [Figure 6.5](#fig-ug-02-step1) |

|  |
| --- |
| Figure 6.5: Step 1 |

|  |
| --- |
| Figure 6.6: Result |

#### 6.3.3.4 View details of an account

| Step | Description | Effect | Figure |
| --- | --- | --- | --- |
| 01 | Click on Account in the left sidebar | Navigate to the account list page | [Figure 6.7](#fig-ug-03-step1) |
| 02 | Find the target account |  | [Figure 6.8](#fig-ug-03-step2) |
| 03 | Click on Details in the Actions column of the table | Navigate to the account’s profile page | [Figure 6.9](#fig-ug-03-step3) |

|  |
| --- |
| Figure 6.7: Step 1 |

|  |
| --- |
| Figure 6.8: Step 2 |

|  |
| --- |
| Figure 6.9: Step 3 |

|  |
| --- |
| Figure 6.10: Result |

#### 6.3.3.5 Create new accounts

| Step | Description | Effect | Figure |
| --- | --- | --- | --- |
| 00 | Follow Feature 02 to navigate to the account list page |  |  |
| 01 | Click on Create to open the accounts creation form | A modal with form show up | [Figure 6.11](#fig-ug-04-step1) |
| 02 | Fill the form with desired values |  | [Figure 6.12](#fig-ug-04-step2) |
| 03 | Click on Create Accounts | New accounts added at the end of list | [Figure 6.13](#fig-ug-04-step3) |

|  |
| --- |
| Figure 6.11: Step 1 |

|  |
| --- |
| Figure 6.12: Step 2 |

|  |
| --- |
| Figure 6.13: Step 3 |

|  |
| --- |
| Figure 6.14: Result |

#### 6.3.3.6 Search an account

| Step | Description | Effect | Figure |
| --- | --- | --- | --- |
| 00 | Follow Feature 02 to navigate to the account list page |  |  |
| 01 | Click on Search to show search input | Search input visible | [Figure 6.15](#fig-ug-05-step1) |
| 02 | Fill the input with desired values | List of accounts auto refreshed with search value | [Figure 6.16](#fig-ug-05-step2) |

|  |
| --- |
| Figure 6.15: Step 1 |

|  |
| --- |
| Figure 6.16: Step 2 |

|  |
| --- |
| Figure 6.17: Result |

#### 6.3.3.7 Update own profile

| Step | Description | Effect | Figure |
| --- | --- | --- | --- |
| 01 | Click on Account name in the left sidebar | Navigate to the profile page | [Figure 6.18](#fig-ug-06-step1) |
| 02 | Click on Update profile to open profile update form | A modal with form show up | [Figure 6.19](#fig-ug-06-step2) |
| 03 | Fill the form with desired values |  | [Figure 6.20](#fig-ug-06-step3) |
| 04 | Click on Submit | Update if form values is valid | [Figure 6.21](#fig-ug-06-step4) |

|  |
| --- |
| Figure 6.18: Step 1 |

|  |
| --- |
| Figure 6.19: Step 2 |

|  |
| --- |
| Figure 6.20: Step 3 |

|  |
| --- |
| Figure 6.21: Step 4 |

|  |
| --- |
| Figure 6.22: Result |

#### 6.3.3.8 Change other’s password

| Step | Description | Effect | Figure |
| --- | --- | --- | --- |
| 00 | Follow Feature 02 to navigate to the account list page |  |  |
| 01 | Find the target account |  | [Figure 6.23](#fig-ug-07-step1) |
| 02 | Click on Update password in the Actions column | A modal with form show up | [Figure 6.24](#fig-ug-07-step2) |
| 03 | Fill the form with desired values |  | [Figure 6.25](#fig-ug-07-step3) |
| 04 | Click on Update | Password for target account updated | [Figure 6.26](#fig-ug-07-step4) |

|  |
| --- |
| Figure 6.23: Step 1 |

|  |
| --- |
| Figure 6.24: Step 2 |

|  |
| --- |
| Figure 6.25: Step 3 |

|  |
| --- |
| Figure 6.26: Step 4 |

#### 6.3.3.9 Get classification report of grade and year

| Step | Description | Effect | Figure |
| --- | --- | --- | --- |
| 01 | Click on Dashboard in the left sidebar | Navigate to the dashboard page | [Figure 6.27](#fig-ug-08-step1) |
| 02 | Select grade and year |  | [Figure 6.28](#fig-ug-08-step2) |
| 03 | Click on Get grade report or Get classification report | Report file can be download if exist | [Figure 6.29](#fig-ug-08-step3) |

|  |
| --- |
| Figure 6.27: Step 1 |

|  |
| --- |
| Figure 6.28: Step 2 |

|  |
| --- |
| Figure 6.29: Step 3 |

|  |
| --- |
| Figure 6.30: Result |

#### 6.3.3.10 Get report of a classroom

| Step | Description | Effect | Figure |
| --- | --- | --- | --- |
| 01 | Click on Classroom in the left sidebar | Navigate to the classroom list page | [Figure 6.31](#fig-ug-09-step1) |
| 02 | Find the target classroom |  | [Figure 6.32](#fig-ug-09-step2) |
| 03 | Click on Details in the Actions column | Navigate to the classroom details page | [Figure 6.33](#fig-ug-09-step3) |
| 04 | Click on Get semester report or Get subject report | Report file can be download | [Figure 6.34](#fig-ug-09-step4) |

|  |
| --- |
| Figure 6.31: Step 1 |

|  |
| --- |
| Figure 6.32: Step 2 |

|  |
| --- |
| Figure 6.33: Step 3 |

|  |
| --- |
| Figure 6.34: Step 4 |

|  |
| --- |
| Figure 6.35: Result |

#### 6.3.3.11 View list of all classrooms

| Step | Description | Effect | Figure |
| --- | --- | --- | --- |
| 01 | Click on Classroom in the left sidebar | Navigate to the classroom list page | [Figure 6.36](#fig-ug-10-step1) |

|  |
| --- |
| Figure 6.36: Step 1 |

|  |
| --- |
| Figure 6.37: Result |

#### 6.3.3.12 View details of a classroom

##### 6.3.3.12.1 R.MAN

| Step | Description | Effect | Figure |
| --- | --- | --- | --- |
| 00 | Follow Feature 10 to navigate to the classroom list page |  |  |
| 01 | Find the target classroom |  | [Figure 6.38](#fig-ug-11-step1-rman) |
| 02 | Click on Details in the Actions column | Navigate to the classroom details page | [Figure 6.39](#fig-ug-11-step2-rman) |

|  |
| --- |
| Figure 6.38: Step 1 |

|  |
| --- |
| Figure 6.39: Step 2 |

|  |
| --- |
| Figure 6.40: Result |

##### 6.3.3.12.2 R.USR

| Step | Description | Effect | Figure |
| --- | --- | --- | --- |
| 01 | Click on Dashboard in the left sidebar | Navigate to the dashboard page | [Figure 6.41](#fig-ug-11-step1-rusr) |
| 02 | Find the target classroom |  | [Figure 6.42](#fig-ug-11-step2-rusr) |
| 03 | Click on Details in the Actions column | Navigate to the classroom details page | [Figure 6.43](#fig-ug-11-step3-rusr) |

|  |
| --- |
| Figure 6.41: Step 1 |

|  |
| --- |
| Figure 6.42: Step 2 |

|  |
| --- |
| Figure 6.43: Step 3 |

|  |
| --- |
| Figure 6.44: Result |

#### 6.3.3.13 Create new classroom

| Step | Description | Effect | Figure |
| --- | --- | --- | --- |
| 00 | Follow Feature 10 to navigate to classroom list page |  |  |
| 01 | Click on Create | A modal with form show up | [Figure 6.45](#fig-ug-12-step1) |
| 02 | Fill the form with desired value |  | [Figure 6.46](#fig-ug-12-step2) |
| 03 | Click on Create classroom | New classroom created | [Figure 6.47](#fig-ug-12-step3) |

|  |
| --- |
| Figure 6.45: Step 1 |

|  |
| --- |
| Figure 6.46: Step 2 |

|  |
| --- |
| Figure 6.47: Step 3 |

#### 6.3.3.14 Update details of a classroom

| Step | Description | Effect | Figure |
| --- | --- | --- | --- |
| 00 | Follow Feature 11 to navigate to target classroom details |  |  |
| 01 | Click on Update details | A modal with form show up | [Figure 6.48](#fig-ug-13-step1) |
| 02 | Fill the form with desired value |  | [Figure 6.49](#fig-ug-13-step2) |
| 03 | Click on Update | Classroom details updated | [Figure 6.50](#fig-ug-13-step3) |

|  |
| --- |
| Figure 6.48: Step 1 |

|  |
| --- |
| Figure 6.49: Step 2 |

|  |
| --- |
| Figure 6.50: Step 3 |

#### 6.3.3.15 View list of students of a classroom

| Step | Description | Effect | Figure |
| --- | --- | --- | --- |
| 00 | Follow Feature 11 to navigate to target classroom details |  |  |
| 01 | Click on Students |  | [Figure 6.51](#fig-ug-14-step1) |

|  |
| --- |
| Figure 6.51: Step 1 |

|  |
| --- |
| Figure 6.52: Result |

#### 6.3.3.16 Assign multiple students to classroom

| Step | Description | Effect | Figure |
| --- | --- | --- | --- |
| 00 | Follow Feature 14 to navigate to student list of target classroom |  |  |
| 01 | Click on Add students | A modal with form show up | [Figure 6.53](#fig-ug-15-step1) |
| 02 | Fill the form with desired value |  | [Figure 6.54](#fig-ug-15-step2) |
| 03 | Click on Confirm | New student appear in target classroom | [Figure 6.55](#fig-ug-15-step3) |

|  |
| --- |
| Figure 6.53: Step 1 |

|  |
| --- |
| Figure 6.54: Step 2 |

|  |
| --- |
| Figure 6.55: Step 3 |

#### 6.3.3.17 View details of a student

##### 6.3.3.17.1 R.MAN

| Step | Description | Effect | Figure |
| --- | --- | --- | --- |
| 00 | Follow Feature 02 to navigate to list of accounts |  |  |
| 01 | Find the target student |  | [Figure 6.56](#fig-ug-16-step1-rman) |
| 02 | Click on Details on the Actions column | Navigate to the target student profile | [Figure 6.57](#fig-ug-16-step2-rman) |

|  |
| --- |
| Figure 6.56: Step 1 |

|  |
| --- |
| Figure 6.57: Step 2 |

|  |
| --- |
| Figure 6.58: Result |

##### 6.3.3.17.2 R.TCH

| Step | Description | Effect | Figure |
| --- | --- | --- | --- |
| 00 | Follow Feature 14 to navigate to student list of a classroom |  |  |
| 01 | Find the target student |  | [Figure 6.59](#fig-ug-16-step1-rtch) |
| 02 | Click on Details on the Actions column | Navigate to the target student profile | [Figure 6.60](#fig-ug-16-step2-rtch) |

|  |
| --- |
| Figure 6.59: Step 1 |

|  |
| --- |
| Figure 6.60: Step 2 |

|  |
| --- |
| Figure 6.61: Result |

#### 6.3.3.18 Update details of a student

| Step | Description | Effect | Figure |
| --- | --- | --- | --- |
| 00 | Follow Feature 16 to navigate to student profile |  |  |
| 01 | Click on Update | A modal with form show up | [Figure 6.62](#fig-ug-17-step1) |
| 02 | Fill the form with desired values |  | [Figure 6.63](#fig-ug-17-step2) |
| 03 | Click on Confirm | Student information updated if provided form valid | [Figure 6.64](#fig-ug-17-step3) |

|  |
| --- |
| Figure 6.62: Step 1 |

|  |
| --- |
| Figure 6.63: Step 2 |

|  |
| --- |
| Figure 6.64: Step 3 |

#### 6.3.3.19 Remove a student from classroom

| Step | Description | Effect | Figure |
| --- | --- | --- | --- |
| 00 | Follow Feature 14 to navigate to student list of a classroom |  |  |
| 01 | Find the target student |  | [Figure 6.65](#fig-ug-18-step1) |
| 02 | Click on Remove on the Actions column | Target student removed from classroom | [Figure 6.66](#fig-ug-18-step2) |

|  |
| --- |
| Figure 6.65: Step 1 |

|  |
| --- |
| Figure 6.66: Step 2 |

#### 6.3.3.20 View list of teachers of a classroom

| Step | Description | Effect | Figure |
| --- | --- | --- | --- |
| 00 | Follow Feature 11 to navigate to classroom details |  |  |
| 01 | Click on Teachers |  | [Figure 6.67](#fig-ug-19-step1) |

|  |
| --- |
| Figure 6.67: Step 1 |

|  |
| --- |
| Figure 6.68: Result |

#### 6.3.3.21 Assign multiple teachers to a classroom

| Step | Description | Effect | Figure |
| --- | --- | --- | --- |
| 00 | Follow Feature 19 to navigate to teacher list of a classroom |  |  |
| 01 | Click on Add teachers | A modal with form show up | [Figure 6.69](#fig-ug-20-step1) |
| 02 | Fill the form with desired values |  | [Figure 6.70](#fig-ug-20-step2) |
| 03 | Click on Confirm | New teachers added to target classroom | [Figure 6.71](#fig-ug-20-step3) |

|  |
| --- |
| Figure 6.69: Step 1 |

|  |
| --- |
| Figure 6.70: Step 2 |

|  |
| --- |
| Figure 6.71: Step 3 |

#### 6.3.3.22 Remove a teacher from classroom

| Step | Description | Effect | Figure |
| --- | --- | --- | --- |
| 00 | Follow Feature 19 to navigate to teacher list of a classroom |  |  |
| 01 | Find the target teacher |  | [Figure 6.72](#fig-ug-21-step1) |
| 02 | Click on Remove | Target teacher removed from classroom | [Figure 6.73](#fig-ug-21-step2) |

|  |
| --- |
| Figure 6.72: Step 1 |

|  |
| --- |
| Figure 6.73: Step 2 |

#### 6.3.3.23 Print record table of a student

| Step | Description | Effect | Figure |
| --- | --- | --- | --- |
| 00 | Follow Feature 16 to navigate to student profile |  |  |
| 01 | Select a classroom of the target student |  | [Figure 6.74](#fig-ug-22-step1) |
| 02 | Click on Print record | Print dialog show up | [Figure 6.75](#fig-ug-22-step2) |

|  |
| --- |
| Figure 6.74: Step 1 |

|  |
| --- |
| Figure 6.75: Step 2 |

|  |
| --- |
| Figure 6.76: Result |

#### 6.3.3.24 View own profile

| Step | Description | Effect | Figure |
| --- | --- | --- | --- |
| 01 | Click on Account name in the left sidebar | Navigate to the profile page | [Figure 6.77](#fig-ug-23-step1) |

|  |
| --- |
| Figure 6.77: Step 1 |

|  |
| --- |
| Figure 6.78: Result |

#### 6.3.3.25 Update own password

| Step | Description | Effect | Figure |
| --- | --- | --- | --- |
| 00 | Follow Feature 23 to navigate to profile page |  |  |
| 01 | Click on Change password | A modal with form appear | [Figure 6.79](#fig-ug-24-step1) |
| 02 | Fill the form with desired values |  | [Figure 6.80](#fig-ug-24-step2) |
| 03 | Click on Submit | Password updated | [Figure 6.81](#fig-ug-24-step3) |

|  |
| --- |
| Figure 6.79: Step 1 |

|  |
| --- |
| Figure 6.80: Step 2 |

|  |
| --- |
| Figure 6.81: Step 3 |

#### 6.3.3.26 View teaching classroom

| Step | Description | Effect | Figure |
| --- | --- | --- | --- |
| 01 | Click on Dashboard | Show list of teaching classroom | [Figure 6.82](#fig-ug-25-step1) |

|  |
| --- |
| Figure 6.82: Step 1 |

#### 6.3.3.27 Request record update

##### 6.3.3.27.1 R.TCH

| Step | Description | Effect | Figure |
| --- | --- | --- | --- |
| 00 | Follow Feature 25 to navigate to teaching classroom list |  |  |
| 01 | Click Details in the Actions column | Navigate to classroom details | [Figure 6.83](#fig-ug-26-step1-rtch) |
| 02 | Click Students | Navigate to classroom student list | [Figure 6.84](#fig-ug-26-step2-rtch) |
| 03 | Find the target student |  | [Figure 6.85](#fig-ug-26-step3-rtch) |
| 04 | Click on Details in the Actions column | Navigate to student record of current classroom | [Figure 6.86](#fig-ug-26-step4-rtch) |
| 05 | Find the target record |  | [Figure 6.87](#fig-ug-26-step5-rtch) |
| 06 | Click on Request in the Actions column | A modal with form show up | [Figure 6.88](#fig-ug-26-step6-rtch) |
| 07 | Fill the form with desired values |  | [Figure 6.89](#fig-ug-26-step7-rtch) |
| 08 | Click on Request | Request sent | [Figure 6.90](#fig-ug-26-step8-rtch) |

|  |
| --- |
| Figure 6.83: Step 1 |

|  |
| --- |
| Figure 6.84: Step 2 |

|  |
| --- |
| Figure 6.85: Step 3 |

|  |
| --- |
| Figure 6.86: Step 4 |

|  |
| --- |
| Figure 6.87: Step 5 |

|  |
| --- |
| Figure 6.88: Step 6 |

|  |
| --- |
| Figure 6.89: Step 7 |

|  |
| --- |
| Figure 6.90: Step 8 |

##### 6.3.3.27.2 R.STD

| Step | Description | Effect | Figure |
| --- | --- | --- | --- |
| 00 | Follow Feature 23 to navigate to profile page |  |  |
| 01 | Find the target classroom | Record data switch to target classroom | [Figure 6.91](#fig-ug-26-step1-rstd) |
| 02 | Find the target record |  | [Figure 6.92](#fig-ug-26-step2-rstd) |
| 03 | Click on Request in the Actions column | A modal with form show up | [Figure 6.93](#fig-ug-26-step3-rstd) |
| 04 | Fill the form with desired values |  | [Figure 6.94](#fig-ug-26-step4-rstd) |
| 05 | Click on Request | Request sent | [Figure 6.95](#fig-ug-26-step5-rstd) |

|  |
| --- |
| Figure 6.91: Step 1 |

|  |
| --- |
| Figure 6.92: Step 2 |

|  |
| --- |
| Figure 6.93: Step 3 |

|  |
| --- |
| Figure 6.94: Step 4 |

|  |
| --- |
| Figure 6.95: Step 5 |

#### 6.3.3.28 Request record update by legacy record image

##### 6.3.3.28.1 R.TCH

| Step | Description | Effect | Figure |
| --- | --- | --- | --- |
| 00 | Follow Feature 25 to navigate to teaching classroom list |  |  |
| 01 | Click Details in the Actions column | Navigate to classroom details | [Figure 6.96](#fig-ug-27-step1-rtch) |
| 02 | Click Students | Navigate to classroom student list | [Figure 6.97](#fig-ug-27-step2-rtch) |
| 03 | Find the target student |  | [Figure 6.98](#fig-ug-27-step3-rtch) |
| 04 | Click on Details in the Actions column | Navigate to student record of current classroom | [Figure 6.99](#fig-ug-27-step4-rtch) |
| 05 | Click on Upload next to Record Heading | A modal with form show up | [Figure 6.100](#fig-ug-27-step5-rtch) |
| 06 | Fill the form with desired values |  | [Figure 6.101](#fig-ug-27-step6-rtch) |
| 07 | Click on Submit | Request sent | [Figure 6.102](#fig-ug-27-step7-rtch) |

|  |
| --- |
| Figure 6.96: Step 1 |

|  |
| --- |
| Figure 6.97: Step 2 |

|  |
| --- |
| Figure 6.98: Step 3 |

|  |
| --- |
| Figure 6.99: Step 4 |

|  |
| --- |
| Figure 6.100: Step 5 |

|  |
| --- |
| Figure 6.101: Step 6 |

|  |
| --- |
| Figure 6.102: Step 7 |

##### 6.3.3.28.2 R.STD

| Step | Description | Effect | Figure |
| --- | --- | --- | --- |
| 00 | Follow Feature 23 to navigate to profile page |  |  |
| 01 | Find the target classroom | Record data switch to target classroom | [Figure 6.103](#fig-ug-27-step1-rstd) |
| 02 | Click on Upload next to Record Heading | A modal with form show up | [Figure 6.104](#fig-ug-27-step2-rstd) |
| 03 | Fill the form with desired values |  | [Figure 6.105](#fig-ug-27-step3-rstd) |
| 04 | Click on Submit | Request sent | [Figure 6.106](#fig-ug-27-step4-rstd) |

|  |
| --- |
| Figure 6.103: Step 1 |

|  |
| --- |
| Figure 6.104: Step 2 |

|  |
| --- |
| Figure 6.105: Step 3 |

|  |
| --- |
| Figure 6.106: Step 4 |

#### 6.3.3.29 Update a record of student

| Step | Description | Effect | Figure |
| --- | --- | --- | --- |
| 00 | Follow Feature 25 to navigate to teaching classroom list |  |  |
| 01 | Click Details in the Actions column | Navigate to classroom details | [Figure 6.107](#fig-ug-28-step1) |
| 02 | Click Students | Navigate to classroom student list | [Figure 6.108](#fig-ug-28-step2) |
| 03 | Find the target student |  | [Figure 6.109](#fig-ug-28-step3) |
| 04 | Click on Details in the Actions column | Navigate to student record of current classroom | [Figure 6.110](#fig-ug-28-step4) |
| 05 | Find the target record |  | [Figure 6.111](#fig-ug-28-step5) |
| 06 | Click on Request in the Actions column | A modal with form show up | [Figure 6.112](#fig-ug-28-step6) |
| 07 | Fill the form with desired values |  | [Figure 6.113](#fig-ug-28-step7) |
| 08 | Click on Request | Record updated | [Figure 6.114](#fig-ug-28-step8) |

|  |
| --- |
| Figure 6.107: Step 1 |

|  |
| --- |
| Figure 6.108: Step 2 |

|  |
| --- |
| Figure 6.109: Step 3 |

|  |
| --- |
| Figure 6.110: Step 4 |

|  |
| --- |
| Figure 6.111: Step 5 |

|  |
| --- |
| Figure 6.112: Step 6 |

|  |
| --- |
| Figure 6.113: Step 7 |

|  |
| --- |
| Figure 6.114: Step 8 |

#### 6.3.3.30 View list of pending record update requests

| Step | Description | Effect | Figure |
| --- | --- | --- | --- |
| 01 | Click on Pending request in the Actions column | Navigate to request list | [Figure 6.115](#fig-ug-29-step1) |

|  |
| --- |
| Figure 6.115: Step 1 |

#### 6.3.3.31 Approve or reject record update request

| Step | Description | Effect | Figure |
| --- | --- | --- | --- |
| 00 | Follow Feature 29 to navigate to request list page |  |  |
| 01 | Find the target request |  | [Figure 6.116](#fig-ug-30-step1) |
| 02 | Click Approve or Reject the request | Request status changed | [Figure 6.117](#fig-ug-30-step2) |

|  |
| --- |
| Figure 6.116: Step 1 |

|  |
| --- |
| Figure 6.117: Step 2 |

#### 6.3.3.32 View update history of a record of student

| Step | Description | Effect | Figure |
| --- | --- | --- | --- |
| 00 | Follow Feature 16 to navigate to student profile |  |  |
| 01 | Find the target classroom | Record data switch to target classroom | [Figure 6.118](#fig-ug-31-step1) |
| 02 | Find the target record |  | [Figure 6.119](#fig-ug-31-step2) |
| 03 | Hover History in the Actions column | A temporary card show update history | [Figure 6.120](#fig-ug-31-step3) |

|  |
| --- |
| Figure 6.118: Step 1 |

|  |
| --- |
| Figure 6.119: Step 2 |

|  |
| --- |
| Figure 6.120: Step 3 |

#### 6.3.3.33 View list of classroom taking part in

| Step | Description | Effect | Figure |
| --- | --- | --- | --- |
| 01 | Click on Dashboard | Navigate to classroom list | [Figure 6.121](#fig-ug-32-step1) |

|  |
| --- |
| Figure 6.121: Step 1 |

#### 6.3.3.34 View own records

| Step | Description | Effect | Figure |
| --- | --- | --- | --- |
| 00 | Follow Feature 23 to navigate to own record page |  |  |

#### 6.3.3.35 Create key to view records of a student

| Step | Description | Effect | Figure |
| --- | --- | --- | --- |
| 01 | Click on Manage verified keys | Navigate to verified key list page | [Figure 6.122](#fig-ug-34-step1) |
| 02 | Click on Create new key | new key created | [Figure 6.123](#fig-ug-34-step2) |

|  |
| --- |
| Figure 6.122: Step 1 |

|  |
| --- |
| Figure 6.123: Step 2 |

#### 6.3.3.36 View list of keys to view records of a student

| Step | Description | Effect | Figure |
| --- | --- | --- | --- |
| 01 | Click on Manage verified keys | Navigate to verified key list page | [Figure 6.124](#fig-ug-35-step1) |

|  |
| --- |
| Figure 6.124: Step 1 |

#### 6.3.3.37 Remove a key to view records of a student

| Step | Description | Effect | Figure |
| --- | --- | --- | --- |
| 00 | Follow Feature 35 to navigate to the list of verified key page |  |  |
| 01 | Find the target key |  | [Figure 6.125](#fig-ug-36-step1) |
| 01 | Click on Remove | Target key removed | [Figure 6.126](#fig-ug-36-step2) |

|  |
| --- |
| Figure 6.125: Step 1 |

|  |
| --- |
| Figure 6.126: Step 2 |

#### 6.3.3.38 Create key to view records of students

| Step | Description | Effect | Figure |
| --- | --- | --- | --- |
| 01 | Click on Manage stats key list | Navigate to verified key list page | [Figure 6.127](#fig-ug-37-step1) |
| 02 | Select year and grade for the key |  | [Figure 6.128](#fig-ug-37-step2) |
| 02 | Click on Create new key | new key created | [Figure 6.129](#fig-ug-37-step3) |

|  |
| --- |
| Figure 6.127: Step 1 |

|  |
| --- |
| Figure 6.128: Step 2 |

|  |
| --- |
| Figure 6.129: Step 3 |

#### 6.3.3.39 View list of keys to view records of students

| Step | Description | Effect | Figure |
| --- | --- | --- | --- |
| 01 | Click on Manage stats key list | Navigate to verified key list page | [Figure 6.130](#fig-ug-38-step1) |

|  |
| --- |
| Figure 6.130: Step 1 |

#### 6.3.3.40 Remove a key to view records of students

| Step | Description | Effect | Figure |
| --- | --- | --- | --- |
| 00 | Follow Feature 38 to navigate to the list of verified key page |  |  |
| 01 | Find the target key |  | [Figure 6.131](#fig-ug-39-step1) |
| 01 | Click on Remove | Target key removed | [Figure 6.132](#fig-ug-39-step2) |

|  |
| --- |
| Figure 6.131: Step 1 |

|  |
| --- |
| Figure 6.132: Step 2 |

#### 6.3.3.41 View verified records of a student

| Step | Description | Effect | Figure |
| --- | --- | --- | --- |
| 01 | Press Ctrl + K | A command pallette show up | [Figure 6.133](#fig-ug-40-step1) |
| 02 | Select View verified data | Navigate to verified data page | [Figure 6.134](#fig-ug-40-step2) |
| 02 | Input key | Show verified data of the provided key | [Figure 6.135](#fig-ug-40-step3) |

|  |
| --- |
| Figure 6.133: Step 1 |

|  |
| --- |
| Figure 6.134: Step 2 |

|  |
| --- |
| Figure 6.135: Step 3 |

|  |
| --- |
| Figure 6.136: Result |

#### 6.3.3.42 View verified records of students

| Step | Description | Effect | Figure |
| --- | --- | --- | --- |
| 01 | Press Ctrl + K | A command pallette show up | [Figure 6.137](#fig-ug-41-step1) |
| 02 | Select View verified data list | Navigate to verified data page | [Figure 6.138](#fig-ug-41-step2) |
| 02 | Input key | Show verified data of the provided key | [Figure 6.139](#fig-ug-41-step3) |

|  |
| --- |
| Figure 6.137: Step 1 |

|  |
| --- |
| Figure 6.138: Step 2 |

|  |
| --- |
| Figure 6.139: Step 3 |

|  |
| --- |
| Figure 6.140: Result |

### 6.3.4 Troubleshooting

# Appendix

## References

Andolfatto, David. 2018. “Blockchain: What It Is, What It Does, and Why You Probably Don’t Need One.” *Blockchain* 100 (2): 87–95. <https://doi.org/10.20955/r.2018.87-95>.

“Front Matter.” 2019. In *Smart Contracts*, I–IV. Mohr Siebeck GmbH; Co. KG. <http://www.jstor.org/stable/j.ctvn96h9r.1>.

Lamb, Kirsten. 2018. “Blockchain and Smart Contracts: What the AEC Sector Needs to Know.” *Blockchain&SmartContracts*, no. CDBB\_REP\_003 (July). <https://doi.org/10.17863/CAM.26272>.

Vihar Kurama, Prithiv S &. 2022. “What Is OCR and What Is It Used For?” *OCR*, November. <https://nanonets.com/blog/what-is-ocr>.