جامعة الإمام عبد الرحمن بن فيصل
IMAM ABDULRAHMAN BIN FAISAL UNIVERSITY

Kingdom of Saudi Arabia
Ministry of Higher Education
Imam Abdulrahman Bin Faisal University
College of Computer Sciences & Information Technology

HADER – EXAM ATTENDANCE SYSTEM

A project submitted
in partial fulfillment of the requirements for the degree of
Bachelor of Science in Computer Information Systems

Done by:

NAME	ID
(LEADER) SHAHAD SAUD ALTAMIMI	2170001466
LAMIA ABDULLAH ALHAMMAD	2180001345
DANAH NASHEER ALDOSARI	2180005949
DEEMAH ADEL ALRUSHOOD	2180000066

Supervised By:
Ms. Asiya Abdus Salam

May 2022

DECLARATION

We hereby declare that this project report entitled "HADER | Exam Attendance System" is a record of an original work done by us under the guidance of Supervisor Ms. Asiya Abdus Salam, Lecturer Computer Information Systems at CCSIT, and that no part has been plagiarized without citations, which have been duly acknowledged. We also declare that it has not been previously or concurrently submitted for any other degree or award at any university or any other institution. This project work is submitted in the partial fulfillment of the requirements for the degree of "Bachelor of Science in Computer Information Systems" at Computer Information Systems Department, College of Computer Sciences and Information Technology, Imam Abdulrahman Bin Faisal University.

Project Team:

Group Number: 9F14(41127)		
Team Member's Name	Signature	Date
Shahad Saud AlTamimi	Shahad	25/11/2021
Lamia Abdullah AlHammad	Lamia	25/11/2021
Danah Nasheer AlDosari	Danah	25/11/2021
Deemah Adel AlRushood	Deemah	25/11/2021

Project Supervisor:

Name	Signature	Date
Ms. Asiya Abdus Salam		

ACKNOWLEDGEMENT

To begin, we express gratitude to Allah for the innumerable gifts bestowed upon us. This opportunity would not have been possible without the aid and knowledge provided by the College of Computer Science and Information Technology and its faculty members. Also, we wish to convey our gratitude to our CIS department. Additionally, we would like to show our gratitude to Ms. Asiya, our supervisor, for her direction and counsel during this journey, as well as for supplying us with important project information and guidance throughout the project time. We would like to offer our heartfelt appreciation to the exam unit and academic affairs unit for all the support extended. Finally, we want to express our heartfelt gratitude for all the love and kindness shown to us by our families and friends.

ABSTRACT

In today's modern educational industry, work and time efficiency are required, which can be achieved through the use of technology. Many manual processes are currently being transformed by technology. This rapid pace of innovation is critical in any process that seeks to save time and effort. This project supports the student attendance system at CCSIT by utilizing QR Code technology. Traditionally, an invigilator would pass around an attendance sheet for students to sign alongside their names, and another way would require the invigilator to call out the students' names one by one and register their attendance. The QR Code attendance system is intended to replace the manual attendance process and accelerate its efficacy. In this project, we will propose a system that automates the examination attendance process in CCSIT by using a QR-based mobile application.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1. INTRODUCTION

1. Introduction

All institutions, whether educational or commercial, appreciate the need of maintaining and managing accurate attendance records and reports of students or staff to plan, manage, and operate efficiently. In the majority of academic institutions in developing countries, students' attendance is typically taken using an old file system method of calling students' names and using paper sheets; this approach has been in use for a long period, making it difficult for university's administration to maintain a current record of attendance; additionally, taking attendance with a pen and paper is always going to be slow, inconvenient, and sometimes inaccurate. In the college of computer science and information technology at Imam Abdulrahman Bin Faisal University, an invigilator would traditionally hand around an attendance sheet for students to sign beside their names, or the invigilator would call out the students' names one by one and register their attendance. With these considerations in mind, the purpose of this project has been raised. The project is designed and implemented to address and solve the issues associated with exam attendance recording. A software solution based on a mobile application is the most logical method to implement this solution since people carry their phones nowadays anywhere. In addition, The QR Code solution to take attendance is an effective, more accurate, and convenient approach to taking student's examination attendance. The exam attendance system is designed to replace manual attendance and increase efficiency. As well as enabling students to return safely to study and normal life in the presence of the COVID-19. Furthermore, to facilitate and improve this process in general through the use of modern technology to align with the Kingdom of Saudi Arabia's 2030 vision now The Kingdom was named the "Top Digital Riser" among the G20 nations due to the comprehensive government support for digital transformation in the kingdom, as part of Vision 2030 [1]. The scope of the project is students, invigilators, instructors, and exam unit and academic affairs at the college of computer science and information technology at Imam Abdulrahman Bin Faisal University.

Hader exam attendance system is a native mobile application that operates on both iOS and Android operating systems. It is designed to aid students, invigilators, instructors, and the Exam Unit at Imam Abdulrahman bin Faisal University, more specifically in the college of computer science and information technology in Imam Abdulrahman Bin Faisal University. This smartphone application is extremely useful. When a student enters the examination hall, the student shows the invigilator/ instructors his/her QR code, the invigilator/ instructors scan the student's QR code, and the student's attendance is automatically recorded in the Academic Affairs Unit's databases, by utilizing the phone and QR code, this process was facilitated and automated. In addition, the student can upload their excuses, and the Academic Affairs Unit will approve/decline the excuses.

2. Problem statement

HADER team members are all aware that taking attendance with a pen and paper is always going to be slow, inconvenient, and error-prone. As a result, the team decided to develop a mobile application that streamlines the exam attendance process for both students and invigilators by allowing invigilators to scan students' unique QR codes. The app also acts as a conduit between the CCSIT administration department and students in the event of absence or tardiness to exams.

HADER application will be extremely advantageous for all parties involved in the exam attendance process because the majority of people now own a smartphone. HADER team members view this as a critical approach for the CCSIT department to expedite the attendance process and, more specifically, to return to college and regular life while conducting exams safely in the face of the COVID-19 pandemic.

3. Motivation

"Hader" application is a community service since CCSIT does not have a similar system. This motivates us to take on the task of developing an application that will benefit, if not be required, at every educational institution by reducing the time and effort required to compile, collect, and record attendance information for both students and instructors. HADER team members are now dealing with "COVID-19" and the development of technology, as well as typical human mistakes such as data loss or misplacing. this project would streamline the process of taking students' attendance at exams and providing late excuses for student affairs, which will reduce both time and effort for the instructors supervising the hall.

4. Business Implications

In the College of Computer Science and Information Technology, the Hader application is considered a novel and unique application. This application's adoption will result in a qualitative leap in the college, as it will encompass everything relevant to midterm and final exams. The application was created to assist all parties involved in taking and documenting attendance, including students, teachers, and invigilators, as well as unit assessments. When you merge the college student database with the Hader application, you can count on it to be your first assistant in all things exam-related, as it ensures speed, efficiency, and most importantly, security, thanks to the feature of scanning each student's unique code and displaying his or her image to ensure credibility.

5. Background and Literature Review

This section examines existing systems that are relevant to this system, as well as the functionality of each system. It also presents the relationship between the proposed systems and this system by demonstrating their similarities and differences.

1) Smart Student Attendance

This research was conducted by the Computer Engineering Department K. J. Somaiya Institute of Engineering & Information Technology, University of Mumbai, Maharashtra, India entitled “Smart Student Attendance System Using QR Code” [2]**.** The system generates a barcode on the server-side by instructors and students can scan the generated barcode to mark their attendance. The systems have two modules, one for instructors and the other for students. The instructor’s module is a website, while the student's module is a smartphone app. Instructors can create QR codes, view and modify student attendance, calculate monthly attendance, and send notifications to students. If a student has less than 50% attendance, instructors may also use the mail procedure to notify the student's parents. In a student module, students can scan QR codes to record attendance at each lecture, view their attendance, and view the notifications sent by the instructor.

The literature review [2] answered and convinced the team members in “which user should scan the code?” and “should the team generate a QR code for each exam?”. Literature review [2] mentioned that the QR code is being scanned by instructors and/or invigilators, and each student has a constant QR Code. This will increase the efficiency and decrease the load on HADER

System, and the error rates are decreased since the application will not face situations such as generating a duplicated QR code for a single exam.

2) Planning the Web-Based Attendance Information System

This research was conducted by Ade Setiawan (2019) entitled "Planning the Web-Based Attendance Information System at SMA Negeri 21 Kabupaten Tangerang" [3]. This research looks at absenteeism, which may be confusing since student absence data is often damaged or lost. By creating a QR code for lecture attendance, researchers created a website-based student attendance information system that improved and reduced all previously existed procedures of the attendance system.

The literature review [3] mentioned QR code technology. The QR code technology in this literature review has been chosen to be the technology that will be used in the project since the QR code technology saves more data, costs -effective, and team members are more familiar with QR code.

3) The biometrics-based attendance registration system

The goal of the [4] project is to create a biometrics-based attendance registration system for academic institutions like colleges and schools. The system attempted to combine hardware and software design ideas to create a consumer product that can replace the existing method of attendance tracking. The Internet of Things (IoT) is also used in this project for data transport, storage, and presentation. The hardware part is included in the pod, which is a small gadget that may be distributed between students in the classroom. It has an LCD screen with different functional choices and a fingerprint scanner that could scan, store, and recognize students' fingerprints. The fingerprint ID that has been identified is then saved in memory and prepped for transmission. The information is subsequently transferred to a database on a web server, in which the Database Management System (DBMS) makes the necessary modifications to the attendance records. This database is available for academics to download or examine.

The literature review [4] mentioned taking modifications to the attendance report. This is crucial to HADER Application. The system should be developed and work in a way that it can work in situations like if the scanner of the invigilator doesn't work suddenly, or if the scanner doesn't recognize the QR Code of the student.

4) Face recognition combined with RFID

The authors of [5] used a model that focuses on how to Face recognition combined with Radio Frequency Identification (RFID) detects and counts permitted students while entering and exiting the classroom. The Intelligent Attendance System maintains an accurate record of each enrolled student and significantly reduces the time spent on the formerly time-consuming job. Furthermore, this intelligent system maintains the data of each student enrolled in a certain course in the attendance record and delivers essential information as needed. In the authors' proposal, the constraints of the current manual attendance monitoring system are largely removed by identifying the person's face and verifying it with RFID simultaneously.

The literature review [5] mentioned using RFID technology. The technology in this literature review was not chosen to be the technology for validation of students since it has a high load to be saved on the database and face recognition is an Artificial intelligence aspect that the team should create a model and should train that model with huge data sets of human faces to ensure high efficiency. This training takes time and resources. By that, the team is supporting the QR code technology.

5) Computerized attendance management systems

Traditionally, attendance is taken in one of two ways: by calling out student names or by taking student signatures on paper. Both ways took more time and were unproductive. As a result, a computer-based student attendance management system is essential to aid faculty in maintaining student attendance and presence. This paper explores the different computerized attendance management systems. This paper defines the fundamental challenge of managing student attendance, which is typically handled manually by instructors. Computer Vision is one option for automating the student attendance system. this paper discusses numerous computerized systems that are being constructed using various methodologies these systems are Computerized Attendance Systems, Bluetooth Based Attendance systems, NFC-based Attendance systems, Fingerprint-based Attendance systems, Iris Based Attendance systems, Face Recognition based Attendance systems, Mobile Based Attendance systems, and RFID based Attendance systems. Upon the basis of this review, a novel technique for recording and managing student attendance is introduced for use in a variety of colleges or academic institutes.[6]

The literature review [6] mentioned a computerized attendance system, this helped and motivated the team to find a programming language that has a high percentage of reusable code, a programming language that is cross-functional to ease developments in the future. The team chose Flutter and dart for the development since it supports many operating systems and is featured with a high percentage of reusable code, a single piece of code can be reused in future developments for a desktop solution.

6) Portable Examination Attendance System

This research was conducted by Hafizul Fahri Hanafi, Cassidy Meyer Joe Meyer, Mohd Helmy Abd Wahab, and Herdawatie Abdul Kadir (2009). The research "PEAS: Portable Examination Attendance System" This research is a non-touch Technology that involves a portable hardware attendance system utilizing a smart card and a PDA (portable digital assistant), not a QR scanner, which can be applied in all educational institutions. The research recommends using the system to help change the conventional way of paper attendance and help reduce the risks of attendance losses. The research also describes the five stages involved in developing the system.[7]

The literature review [7] has used a portable attendance system that is small and convenient. This literature review has benefited HADER in a way to develop a solution to be used in smartphones rather than PCs since everyone owns a smartphone, smartphones are always available, and are known to be easy to work with, this means to develop a mobile application rather than a desktop application/website. In addition, HADER Application will not use any hardware to eliminate and avoid operational costs and maintenance costs.

7) A Students Attendance System Using QR Code

This research was conducted by Fadi Masalha and Nael Hirzallah, faculties at Applied Science University, under the title of "A Students Attendance System Using QR Code". The system works by requiring students to log in using a mobile app through its Server Module to generate an encrypted QR code. It can be done at any point before class time. At the beginning of each class, the students are asked to scan the displayed QR code through the system Mobile Module and match it with the instructor's encrypted QR code. Each student will also have a personal image to confirm their identity. A list of attendance can be accessed for each instructor that shows the status of the

student along with all the student's details such as personal image and weight. This can be presented to the instructor at any time.[8]

The literature review [8] has covered a major identification and an authentication gap by using a face image of students for authentication to ensure that the meant student is attending their exam, and no student is taking the exam on behalf of other students. This method has been applied in HADER Application to cover many security gaps. And to increase the benefits and the value of the HADER Application.

8) Smart Attendance Management System

The architecture of an autonomous system for tracking and managing student attendance data is proposed in this research. The design consists of three key elements that work together to provide a smooth user experience. The use of basic beacons is a critical component since it makes the system both cost-effective and simple to operate. When you combine them with a mobile platform and a web service, you may track attendance in a variety of ways. The suggested method is unique in terms of tackling typical attendance tracking difficulties since it combines periodic detection of nearby beacons with authorized server-side logic. In addition, the architecture allows for the future inclusion of advanced data analysis modules [9].

The literature review [9] developed an attendance system that has friendly user interfaces and a good UX, this is the same concept the team is using while developing HADER Application. in addition, it's beneficial to the HADER system to have a good UI and an easy to navigate application since it's a critical system and the learning time of application should be low since the application is being developed to shorten the time of taking attendance and to be used in exam time where all users have limited time for attendance.

9) Near-Field Communication (NFC)

Near-Field Communication (NFC) technology may be used to implement a student attendance system in the education sector. NFC is a term that refers to a device that can detect data and/or commands from a tag by bringing them close together or even touching them. The NFC-based attendance system is intended to improve upon the manual attendance method [10].
The literature review [10] discusses using an NFC as a commutation method this is an excellent method to be chosen instead of the QR Code. the drawback of this technology is that it's expensive and very complicated to develop. This literature review views a different technology and supports the team point of view in choosing QR Code in HADER, it's the fittest technology until now and it's easy to deploy, develop and maintain by team members.

10) Student Examination Attendance

Students' attendance at academic institutions may be tracked via smartphones. Their suggested student examination attendance method is more effective at obtaining information about student attendance during the exam. The primary distinction between their method and current systems is that previous systems make use of an additional hardware device called a barcode reader. While this approach makes use of smartphones, it can be also utilized to track students' attendance at academic institutions. Their suggested student examination attendance method is more effective at obtaining information about student attendance during the exam. The primary difference between their system and previous systems is that present systems rely on an additional hardware

device called a barcode reader, but this system relies on a smartphone, which significantly reduces costs [11].
The literature review [11] discusses the development of an existing system due to the operational and maintenance cost that caused issues and lead to the developing of a mobile application. The literature review supports the HADER point of view in using a mobile application solution due to the prevalence of smartphones, their portability, and the fact that it's always within reach, this literature review has helped mentioned the complexity of having additional hardware that increases the operational and maintenance costs.

- **Similarities:**

1. All systems are related to educational attendance systems.
2. All systems are in the English language.

- **Differences:**

1. Some researchers used portable hardware attendance systems using smart cards, RFID Tags, and pods and the HADER system will use only software components.
2. In Portable Examination Attendance System [6] research is used PDA (portable digital assistant), HADER system is using mobile application software with a Qa R scanner
3. All systems and research listed do not support uploading exam absence excuses and the HADER system allows students to upload their exam absence excuses on the app to be reviewed by the exam committee in CCSIT.

As shown below in Table 1, some characteristics of each literature review are listed to compare all systems with HADER Application. Each column shows a characteristic, and each row aligns the characteristics with all systems mentioned in section 4.

Table 1: Comparison between all systems

Literature Reviews compared with HADER Application	Characteristics										
	User friendly	Use hardware components to identify student's identity.	Using biometric technologies to mark students' attendance	Marks Exam attendance	Uploading exams excuses	Uses QR Code	Attendance modification	Absence calculation	Send notifications to users.	Supports English language	Systems developed to support educational industries.
HADER Application	√			√	√	√	√	√	√	√	√
1. Smart Student Attendance						√	√	√	√	√	√
2.Planning the Web-Based Attendance Information System	√			√		√				√	√
3.biometrics-based attendance registration system	√	√	√	√	√	√				√	√
4.Face recognition combined with RFID		√	√	√						√	√
5.computerized attendance management systems			√	√						√	√
6.Portable Examination Attendance System	√		√	√						√	√
7. A Students Attendance System Using QR Code	√			√							√
8. Smart Attendance Management System	√						√			√	√
9. Near-Field Communication (NFC)		√	√						√	√	√
10. Student Examination Attendance	√	√		√						√	√

6. Justification

HADER team members have selected this project to make the monitoring process of exam rooms to take students' attendance at examinations easier and to provide late reasons to Student Affairs, saving time and effort. The student may use the detecting device to scan a code and enter their

attendance information straight into the system. If a student is absent, the student may utilize the system to submit an absence excuse, which is then forwarded to Student Affairs. Each exam will be managed by the Deanship of student affairs, and the instructor can see and share the attendance report with any party in CCSIT. The team has inspired to choose the QR code mechanism by referring to the first literature review since it states that their system generates a barcode on the server-side by instructors and students can scan the generated barcode to mark the attendance [2]. This approach encouraged the team since it requires only software resources to design the app and eliminates the need of purchasing any hardware. This means the application the team is building is more convenient and all the hardware resources required by users are their smartphones in their hands.

7. Aims & Objectives

Hader's application solves a variety of exam attendance issues at CCSIT and mainly aims to streamline the process of taking attendance in exam halls.
The aims and objectives are as follows:

- Facilitate and expedite the process of attendance.
- Reduce the disturbance in the exam room.
- Utilizing modern technology at the IAU, CCSIT division.
- Assisting students by consolidating all exam-related information into a single platform.
- Facilitate the process of submitting absence and tardiness excuses and responding to them.
- Assisting the Academic Affairs Unit in CCSIT by automating the registration of student attendance data during exams.

8. Scope/ limitation of the study

The Scope of the system is as follows:

- It includes all activities and actions directly connected to the system's goal.
- It comprises QR barcodes for the attendance system and an accompanying application for scanning them.
- It has three users: a student, an instructor, and the academic exam unit in CCSIT at IAU, and the system will be built for both (iOS) and (Android) systems.
- The system validates if the student is in the wrong/suitable exam hall.

The Limitations of the system are as follows:

- The application will be designed and implemented following CCSIT requirements and the security measures and techniques along with authentication of users, so in the future, CCSIT can adopt this solution and integrate it with the IAU database.
- The system will be application-based, requiring internet connectivity.
- The system will only be available in English.
- The system will only be available to approved users by CCSIT.

9. Professional, ethical, social, and legal implications

9.1 Professional

Hader's System is committed to providing the mobile application with a high level of quality, high-tech, and professionalism by utilizing all the skills, techniques, and knowledge acquired throughout prior academic courses. Additionally, the system's installation must adhere to the

highest professional standards and handle all types of maintenance procedures with the same professionalism, ensuring the system's quality and support for a large number of users.

9.2 Ethical

To ensure the operation of Hader's application, the code must be error-free and defect-free to regulate the ethical parts. Additionally, users must provide their ID and password to access the application, and this information will be secured and encrypted.

9.3 Social

HADER application improves Students' exam attendance by eliminating the manual use of paper to take students' attendance, specifically with the current COVID-19 pandemic. HADER application helps to return to college and ensure that examinations are conducted safely without any need to have tangible attendance papers to be touched by invigilators and students to sign student attendance. Furthermore, it eliminates the disruption in exam rooms caused by the traditional process of taking attendance by calling students' names in exam rooms while taking the exam. HADER application shortens the whole traditional exam attendance process and makes the exam experience more comfortable for both invigilators and students by scanning a QR code only. HADER application helps to decrease the amount of time and effort spent on taking exam attendance and enables students to have a more pleasant exam experience.

9.4 Legal

HADER - Exam Attendance System is designed in conformity with legal requirements to ensure that the system's design, name, source code, and concept do not breach or clash with the intellectual property rights of another system or person. Likewise, data collected from users that contain sensitive information must be safeguarded and not shared with a third party without the user's permission. Moreover, the system was created in conformity with the laws governing operating system markets to avoid any violation of terms and conditions that would result in the system being pulled from the market, and IAU regulations were followed as the system's planned destination.

10. Business Impact

Mobiles and smartphones provide a better and more engaging way of imparting education in the education sector. Mobile apps are opening up new avenues of teaching and learning, which is greatly improving education like never before. Technology has become a significant disruptive force in the education sector all over the world. Mobile apps have had a significant impact on the education industry. HADER exam attendance system is a product designed for CCSIT and fits in the entrepreneurial relevance in that is it flexible enough to be used successfully in IAU and in any education institution. It satisfies the needs and desires in that it speeds up the attendance process and eliminates the flaws caused by the traditional attendance process of taking attendance with a pen and paper, which is always going to be slow, inconvenient, and inaccurate. Hader is the new technology for taking attendance in exams, does not require any hardware, has rich features such as generating reports, attendance analysis, user friendly, and is secure. Hader aims to develop a disaster recovery plan in the IT infrastructure, testing and validation, and data security, with the goal of reaching the top by developing a business continuity plan.

11. Project Organization

The project report is divided into the following major chapters:
The report will be divided into eight chapters:

1. Introduction, this chapter includes a brief description of the project.

2. The project management plan, this chapter consists of the steps needed to be taken to complete this project.
3. Requirement's specification contains the Overall Description, Characteristics, and user interfaces.
4. design specifications, this chapter demonstrates the overall design of the application.
5. Conclusion, this chapter provides a summary of the work.
6. Project implementation contains summeray of the proposal, Changes From the Proposal Phase and Justifications, implementation progress and issues faced.
7. Software test plan, this chapter includes approach, pass/fail crateria, testing process, enviromental requirements, publication, risk and assumptions and dessimination plan.
8. Software test report, includes overview of the test results and detailed test results.

CHAPTER 2. PROJECT MANAGEMENT PLAN

1. Introduction

This chapter shows Software Project Management Plan (SPMP) for the Exam Attendance system named HADER. The project will allow instructors to easily share, print, and review their documents, allow CCSIT academic affairs unit to manage student attendance in exams, and allow invigilators to mark student attendance in exams using QR codes. It also allows students to upload their exam excuses without wasting paper, saving time and effort, and making the entire attendance process simple. This section includes the purpose, scope, and objective of the project and the assumptions, constraints, project deliverables also the master schedule and budget summary.

1.1 Definition, acronyms, and abbreviations

Table 3: Project's Abbreviations

IAU	Imam Abdulrahman University
CCSIT	College of computer since and information technology
CIS	Computer Information System
IT	Information technology
QR code	Quick Response code
SPMP	Software Project Management Plan
SRS	Software Requirements Specification
SDS	System design Specification
DB	Database
UI	User interface

1.2 Project Summary

The purpose of this project is to develop a phone application that enables tracking students' attendance in exam halls by scanning their QR code at Imam Abdulrahman bin Faisal University, specifically at the College of Computer and Information Technology. By streamlining the process of taking attendance, storing it in a database, and reviewing it, the application benefits students, invigilators, instructors, and the Academic Affairs Unit. The project team will deliver several documents, including an SPMP, an SRS, and an SDS, that aid in the team's implementation of the project the following semester.

1.3 Purpose, Scope, and Objectives

The purpose of the HADER application is to facilitate the process of obtaining students' attendance in the examination hall and making the process more efficient, accurate, and timely using modern technology, which is all about efficiency and speed. The product assists the student, the invigilator, instructors, and the Exam Unit at Imam Abdulrahman bin Faisal University, specifically at the College of Computer and Information Technology. The procedure is as follows: When the student enters the examination hall, the student shows his/her QR code to the invigilator, the invigilator scans the student’s QR code, and the attendance is automatically logged in the Academic Affairs Unit's databases. The system users are:

Students:

- Log in to view the dates and times of their exams.
- A check-in process for taking an exam that includes a status report (on-time, late, absent).
- Upload and monitor the status of absence excuses.

Invigilators:

- Can identify which examination room they are invigilating.
- Can scan students' QR codes and review students who are present.
- Search per date about invigilating an exam.

Instructors:

- Switch to invigilating mode if they will invigilate.
- Be informed by the administration of the time/day of the class they will be invigilating.
- View their sections and identify which students are absent.
- Receive notification from admissions regarding (declined/accepted) excuses for retaking an exam.

Administration:

- Establish a schedule for the distribution of students to exam rooms daily.
- Manage exams and assign an Exam Invigilator.
- Manage students' excuses by approving or denying them.

1.4 Assumptions and Constraints

The team of HADER Application assumes that:

- The student, the invigilator, instructors, and the Academic Affairs Unit will all use the system.
- Each student and a facility member own a unique university identification number.
- The user has access to the internet.
- The user owns a smartphone.
- The project team members are capable of dealing with any circumstance that may arise.
- The resources of a sufficient quality to achieve the desired outcome.

The system development is planned with the following constraints:

- The requirement is for each student to have a unique and consistent QR code.
- The application must meet the deadline.
- The application must meet the requirements.
- The application should be simple to use, secure, and dependable.
- The application should be compatible with both iOS and Android systems.
- The application supports the English language only.

1.5 Project Deliverables

The HADER Application's primary deliverables are listed below, along with their associated dates. The date is determined by the amount of time required to complete each deliverable.

Table 4: Project Deliverables

Deliverables	Date
Proposal discussion	9/9/2021
Status Report #1	9/9/2021
Problem statement	16/9/2021
Status Report #2	23/9/2021
Project Plan (SPMP)	23/9/2021

Status Report #3	30/9/2021
Midterm Report	7/10/2021
Initial system requirements (SRS)	14/10/2021
System requirements (SRS)	21/10/2021
Status Report #4	21/10/2021
Initial design (SDS)	4/11/2021
Status Report #5	28/10/2021
Final design	18/11/2021
Status Report #6	11/11/2021
Final report	9/12/2021
Final Oral Presentation	9/12/2021, 16/12/2021
Database design	25/01/2022
Application programming	01/04/2022
Initial testing	07/04/2022
User manual	11/04/2022
Final testing report	April 2022

1.6 Master Schedule and budget Summary

The following figures depict the Gantt chart for the master schedule. Microsoft Excel was used to predict working time between the first and second semesters, by setting expected project completion dates that are likely to alter over the semester, depending on the timetable for the second semester. There is no need for cost estimation during the system development process, as it is completely free.

Table 5: Master Schedule and budget Summary.

Task Name	Duration	Start Date	Finish Date
Proposal discussion	1 day	05/05/2021	06/05/2021
Collection requirements	9 days	01/09/2021	10/09/2021
Introduction document (problem statement)	1day	17/09/2021	18/09/2021
Project management plan (SPMP)	10 days	19/09/2021	29/09/2021
Midterm report	4 days	02/10/2021	06/10/2021
Software system requirement (SRS)	10 days	09/10/2021	19/10/2021
System design specification (SDS)	14 days	21/10/2021	04/11/2021
UI designing	13 days	05/11/2021	18/11/2021
Final report	14 days	17/11/2021	01/12/2021
Database design	5 days	20/01/2022	25/01/2022

Application programming	67 days	24/01/2022	01/04/2022
Initial testing	5 days	02/04/2022	07/04/2022
Final testing	22 days	08/04/2022	30/04/2022

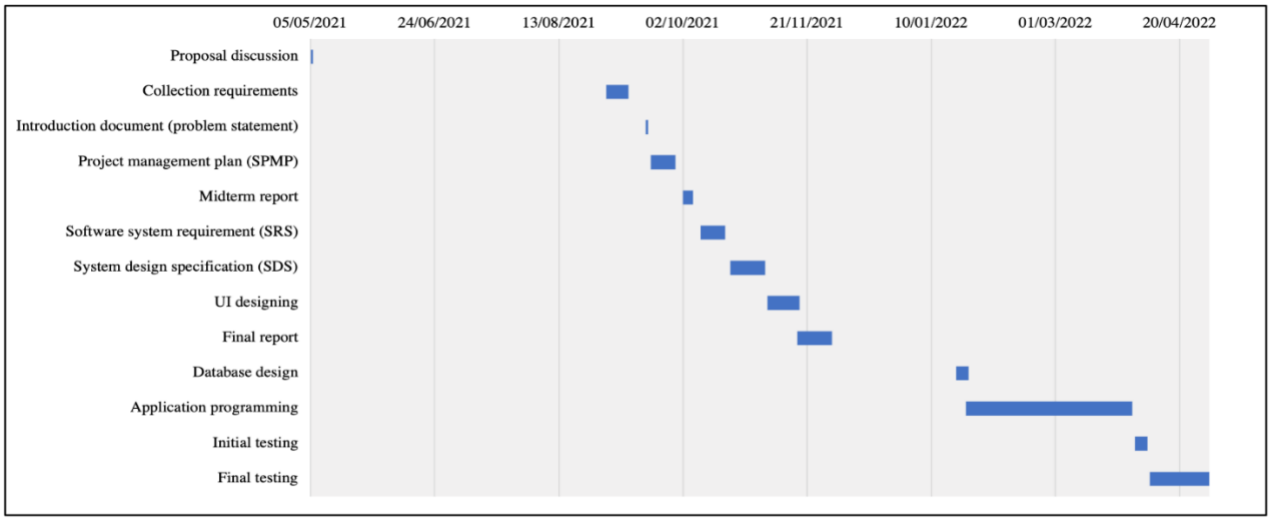


Figure 1 HADER Gantt Chart

2. PROJECT ORGANIZATION

This section explains the external interfaces, internal structures, project roles, and responsibilities.

2.1 External Structure

The system is going to be developed for CCSIT, and the requirements are extracted mainly from the exam unit. Furthermore, a member of the Academic Affairs Committee at CCSIT will be the focal person at CCSIT.

2.2 Internal Structure

The development team consists of the academic supervisor Ms. Asiya Abdus Salam, project manager Shahad Al Tamimi, and team members each hold a role and more than one responsibility.

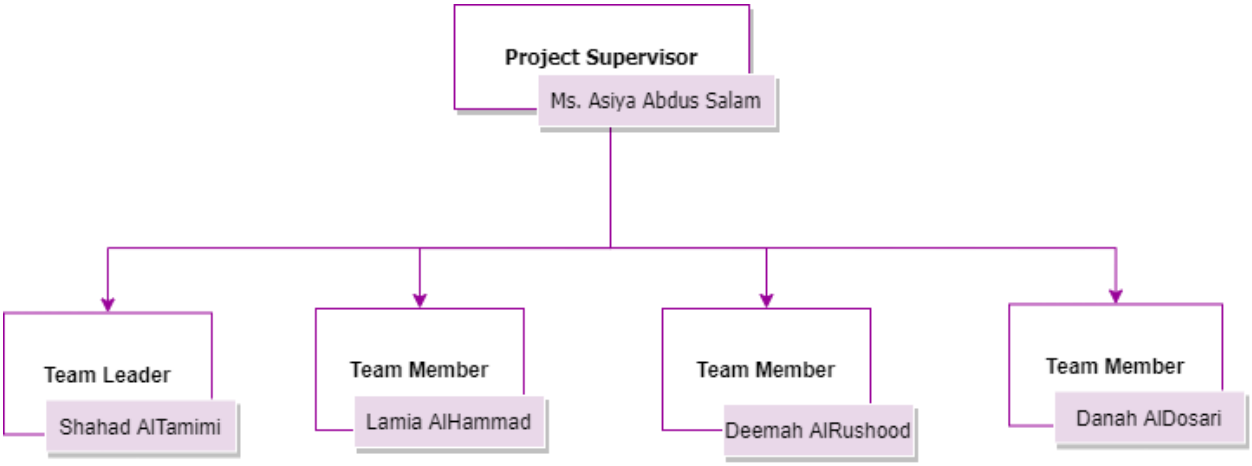


Figure 2: Project's Internal Structure

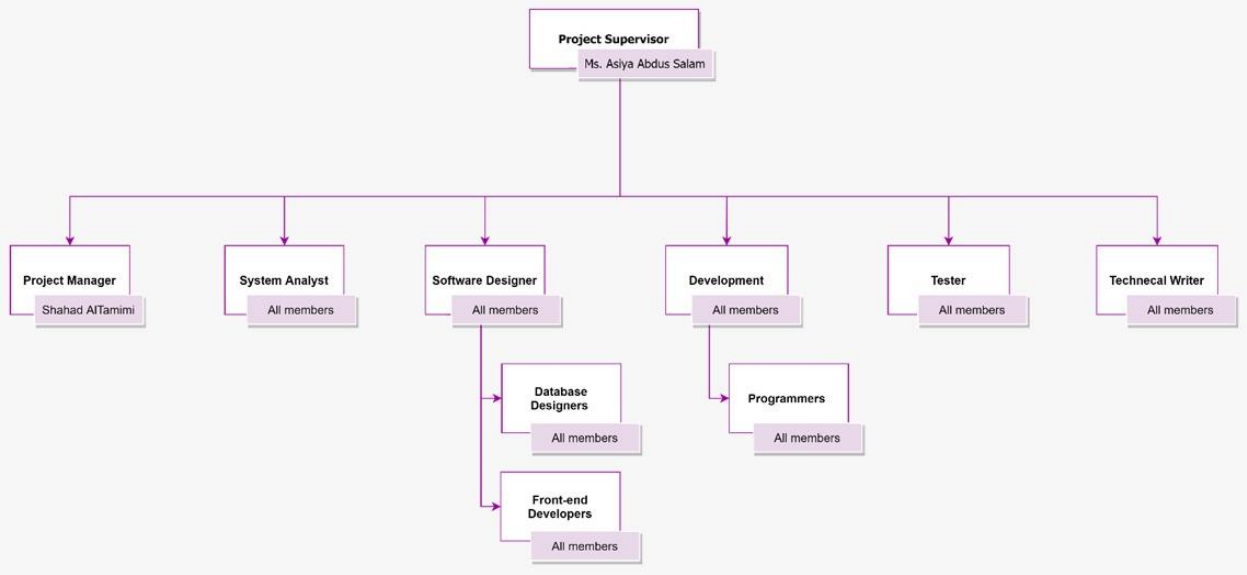


Figure 3: Project's Organizational Structure

2.3 The Project Manager

The project manager Shahad is responsible for conducting follow-up meetings, planning, dividing tasks, organizing the project development process, and delivering and submitting tasks to the supervisor.

2.4 Scope of Authority

The scope of authority demonstrates the privileges Shahad can practice as she holds the project manager position in the project. The rights include assigning tasks, fairly dividing work among team members, accepting or rejecting work, scheduling follow-up meetings with team members or canceling them if they are not required to conduct urgent meetings, and submitting the tasks to the supervisor on time.

2.5 Scope of Responsibility

The scope of responsibility demonstrates the privileges Shahad can practice as she holds the project manager position in the project. The responsibilities include overseeing all project resources, ensuring that work is completed and submitted on time, monitoring the workflow among team members, directing them when problems arise, comparing actual performance to planned performance, building commitment, and ensuring that all team members are committed and motivated to work.

2.6 Internal Responsibilities

To complete the project in an accurate manner the internal responsibilities were assigned among team members. The project supervisor holds the responsibility of leading the whole team members and evaluating the work. The project manager is controlling the project by being responsible for the schedule, work, and risk management. Team members will hold the responsibility of building the project and delivering an efficient and good quality application.

2.7 Project Roles and Responsibilities

Team Roles and responsibilities are illustrated in the table below to show each member’s duties and role to complete the project and meet its goals and objectives. Based on the waterfall process model phases, positions were assigned. Thus, each leader supervises the process, has the authority to discredit the work, and has the last say on tasks, but all team members are participants in each role.

Table 2: Project Roles and Responsibilities

Name	Position	Roles and Responsibilities
Ms. Asiya Abdus Salam	Project Supervisor	- Approve or decline the submitted work. - Provide feedback about the submitted documentations and review the project work. - Setting project deadlines date and time. - Lead and supervise team members. - Guide, direct and answer questions team members may have. - Increase the productivity of the team by motivating them and developing motivational tactics.
Shahad AlTamimi	Project Manager	- Managing and leading the team. - Developing and maintaining a detailed project schedule and plan. - Define the project goals and establish risk control measurements. - Developing Corrective actions when necessary. - Review and approve reports and work performed. - Submitting deliverables to the project supervisor.
All Team Members	Systems Analyst Leader	- Leads the researching and discovering of the project requirements. - Ensure that the design satisfy and requirements. - Prepare and document all the resource documents. - Analysis of the system to figure out any issues may encounter.

		- Focusing on the project goals meet the actual work and functionalities.
All Team Members	Software Designers Leader: - Database designer - Front-end developer	- Leads the design of the system software and its architecture. - Propose an effective design solution to meet the project objectives and goals. - Preforming system monitoring strategies. - Building and designing a database.
All Team Members	Development Team Leader: - Programmer	- Lead the coding development in accordance with the designs. - Propose solutions about authentication and security in the project. - Working on preforming the code and authentication of the applications. - Allocating and using the right resources. - Configuring and installing the database.
All Team Members	Testing Team Leader: - System Testers	- All test planning duties. - Develop testing strategy, plan and activities. - Prevent any software issues. - Document and evaluate test results. - Determine if the team has all of the resources required to carry out the testing tasks. - Use the appropriate test measurement and metrics. - Prepare a report on the status of testing efforts.
All Team Members	Technical writers	- Develop software manual. - Prepare and review technical documents.

3. MANAGEMENT PROCESS

This section contains and discusses the management processes of HADER application they are the start-up plan, work planning, project control, and risk management, to start working on the project in an effective and accurate manners and applicable project management controls to deliver a working application on time.

3.1 Start-up

This section represents the staffing plan, resource acquisition plan, and staff training plan.

Table 3: Start-Up Table

Project Phase	Human Resource	Duration
Project proposal	Team members	One week
SPMP	Project Manager	Two weeks
SRS	System analyst	Two weeks
Status report	Team members	Biweekly
SDS	Software Architect and Software developer	Four weeks
STP	Software Architect, Software developer, GUI designer	Two weeks
User manual	Team members	One week

3.2 Staffing

The HADER team consists of five valued members who agreed on delivering the project before the end of the academic year (2021/2022). The table below outlines the project's different phases as well as the time needed to finish each.

3.3 Resource Acquisition

The hardware and software resources that will be employed in the project are listed in the table below:

Table 4: Hardware and software resources

Hardware resources	**Smartphones**
	Laptops
Software resources	Firebase
	Flutter
	Dart language
	Microsoft Office
	Trello
	Figma

3.4 Staff Training

Previously, the team has developed applications and databases from the "Database Concepts and Design" and " Human Computer Interaction" and learning dart language along with flutter that will be used during the project implementation phase to build a cross-platform application that allows the application to work on iOS and Android smartphones. The members of the project team will rely heavily on their past work experience and education in order to successfully complete the project.

4. WORK PLANNING

This section details the activities, timetable, and resource allocation of the workday.

4.1 Work Activities

Over the course of two semesters, the team will work in a group to create and develop an application for Android and iOS operating systems. During the first semester, the planning and design phase will cover all aspects of the project, prepare a list of directives for each phase, gather information, and explore several ideas that will assist in the final plan's development. In the second semester, the attention will be on the implementation and testing phases and , the actual execution of the project phases described previously.

4.2 Schedule Allocation

Table 5: Schedule Allocation

Week	Dates	Topics	Deliverables
1 **2**	Aug 29th - Sept 2nd Sept 5th – Sept 9th	Proposal discussion	Status Report #1
3	Sept 12th – Sept 16th	Collecting data and defining the domain of the problem – (Problem statement)	
4	Sept 19th – Sept 23rd	Project Plan (SPMP)	Status Report #2
5 **6**	Sept 26th – Sept 30th Oct 3rd – Oct 7th	Midterm report*	Status Report #3 Midterm Report
7	Oct 10th – Oct 14th	Analyzing phase (System requirements) (SRS)	
8	Oct 17th – Oct 21st	Analyzing phase (System requirements) (SRS)	Status Report #4
9 **10**	Oct 24th – Oct 28th Oct 31st – Nov 4th	Initial design (SDS)	Status Report #5
11 **12**	Nov 7th – Nov 11th Nov 14th – Nov 18th	Final design	Status Report #6
13 **14**	Nov 21st – Nov 25th Nov 28th – Dec 2nd	Final report	Final report

4.3 Resource Allocation

HADER system is strongly dependent on programming, so for development and implementation, the team must use Flutter along with Dart language, which is used to build mobile applications for iOS and Android operating systems. The most popular Microsoft Office products are available in packages at Imam Abdulrahman bin Faisal University, which enables students to work on project documentation more quickly and easily. Likewise, the CCSIT IT department supplies IT support to both students and instructors.

The project team met with Ms. Teflah Alqurnas, Ms. Sarah Almulhim, Ms. Malak Alfosail, and Dr. Mamoon Ibrahim, all of whom are members of the Exam unit at CCSIT, to gather requirements.

5. Project Controls

The project controls section includes requirements control, schedule control, quality control, project reporting and communication, and metrics collection that will be used to plan, develop, and manage the project.

5.1 Requirements Control

The application serves exam invigilators, lecturers, academic affairs, and students in CCSIT in final and midterm exams only, by managing exams, marking attendance, and uploading excuses efficiently by building an exam attendance system. The project team conducted meetings with the exam unit and academic affairs unit in CCSIT to clarify the procedures taken in exams, and collect and analyze functions, features, and services that need to be automated.

5.2 Schedule Control

To create the timeline based on the project stages, the team chose to work on Trello and submit their work to a shared OneDrive. It assists in keeping track of chores and keeping track of the schedule under control. The calendar of work activities, their connections, and the length of each day's labor, including weekends, are all calculated. The team leader oversees keeping the calendar up to date and alerting members of deadlines. And the team leader is responsible for either increasing the time or reducing the number of jobs each day in the event of delays and disputes. Further meetings may be required to ensure that the work is finished on time.

5.3 Quality Control

Team members should test the application regularly along with test plans while working on it to verify that the features suit the requirements of the users and are of good quality, as well as free of errors. Furthermore, the tools utilized to create the Application and project documentation should be of the highest quality.

5.4 Project Reporting and Communication

The project team members communicate and interact with the supervisor through Zoom meetings, E-mail, and WhatsApp to exchange essential project materials and coordinate weekly updates.

5.5 Metrics Collection

A metric is defined as a method that is used to verify and enhance efficiency, and evaluate performance in a measurable manner, to ensure the successful completion of the project.

Table 6: Metrics Collection

Metric	Unit	Frequency of collection
Effort	Member, Weeks, or days	During initial plan and SPMP
Working time	Member, hours per day	During initial plan and SPMP
Elapsed time to completion	Days	After review meetings
Resource allocation	% Assigned task / total task	SPMP
Work complete	Percentage “%”	SPMP

5.6 Risk Management

Table 7: Project Risk Management

No.	Risk	Description	Probability	Effect	Prevention Action	Correction Action	Impact
1	Data Loss	Loss or corruption of critical data	High	High	Taking a backup scheduled periodically	Restore latest backup	-Time -Quality -Cost
2	Authenticati on of check-in process	Authenticate that the real student is entering exam	Medium	High	Prevent unauthorized students to enter the exam	Creating a smart QR code with student image while checking-in	-Time -Quality -Cost
3	Requirement changes	Not applying the right planning and analysis phases to the project	High	High	Divide the work equally and interviewing the right people to gather requirements.	Spend more time on project planning and analysis phases to define the right requirements.	-Time -Quality
4	Unavailabilit y of team members	Absence of members due to other activities or excuses.	High	Low	Inform the team earlier about their assigned tasks and remind them days before the deadline	Solve any issue between project manager and team members. If there is no change then the supervisor will be included to take the correct actions	-Time -Quality
5	Not meeting deadlines	Not accomplishing and submitting tasks on time	Medium	High	Divide tasks effectively and apply time management skills at all times	Discuss with supervisor about warning members who neglects deadlines	-Time -Quality -Cost

5.7 Project Closeout

This section explains the project's completion, including the application and final report, which will be completed by the end of the second semester in 2022. To assure the project's effective conclusion, all necessary plans will be incorporated accurately and will be evaluated by the project supervisor.

6. TECHNICAL PROCESS

This section identifies the development process model, technical models, tools, and techniques that will be applied to develop and deliver the project in the best way, and it identifies the project infrastructure and product acceptance plan.

6.1 Process Model

After understanding the different software development methodologies, the team has agreed that the Waterfall Model is the best fit as shown in figure 4, since the waterfall model segregates project activities into linear sequential phases unidirectionally, each phase is dependent on the deliverables of the previous one and validating the deliverables to go to the next phase. Waterfall methodology is best when teams are small and the objectives of projects are clearly outlined from the beginning, Waterfall methodology is used when the project requires a high level of reliability and when project goals and objectives rarely change. Following waterfall methodology phases in a systematic manner, it will result in a successful software application. HADER Exam Attendance System follows a waterfall process model because all phases move step by step and each outcome from every phase serves as an entry to another phase to ensure that the project deliverables are accurate and as expected. In addition, the HADER group is small, and goals and objectives are known from the beginning and are not changing during the project lifecycle.

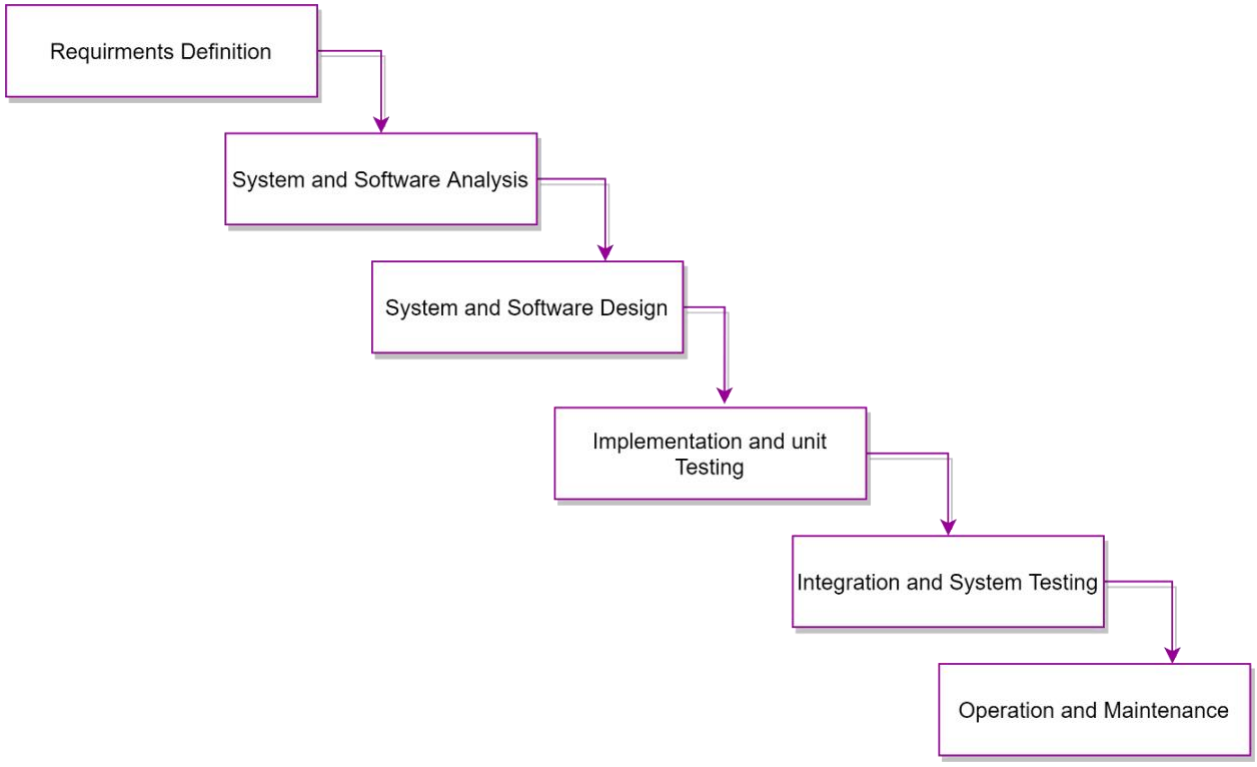


Figure 4: HADER Waterfall Process Model

6.2 Methods, Tools, and Techniques

All team members used Trello to manage the project's progress. Additionally, the Firebase database structure will be utilized to build the database and its relationships and OneDrive shared folder to work on shared Microsoft word documents. Finally, the HADER team will utilize various tools to create the necessary sketches and graphs for the project. The table below shows all categories and their tools:

Table 8: HADER Methods, Tools, and Techniques

Categories	Tools
Communication and planning	WhatsApp Zoom meetings Email
Documentation	Microsoft Word OneDrive
Diagram Designing	draw.io Microsoft Word
Interfaces Designing	Figma
Implementation and Testing	Firebase Dart language Flutter platform
Technical Standards	IEEE Standards

6.3 Project Infrastructure

In this project, the software front-end development will be developed on a personal computer using the Flutter platform along with Dart language. For back-end development Firebase to build the database will be used. Each member of the project team will install the necessary software throughout development and implementation. Additionally, the analytical stage will be conducted using the shared OneDrive folder that contains all Microsoft Word documents. Utilize Draw.io to create diagrams and Figma to create mock-ups and user interfaces.

6.4 Product Acceptance

Acceptance of the final product is primarily determined by the project's supervisor and team members. However, the team must first plan and agree on what will be accomplished in the project, particularly in each phase. Acceptance of projects is centered on ensuring user satisfaction and system criteria. Team members must be committed to specified deadlines while drafting documents. IEEE Standards must be followed. After being approved and accepted from the senior project unit, the product will be implemented next semester.

7. Configuration Management

Configuration management establishes a baseline to safeguard documents produced during the project's development period and the current version of the software from unauthorized modification. Configuration elements must be chosen, and requirements must be assigned. Before making any changes, all phases' configurations should be monitored and assessed. This will aid in the proper implementation of the project. The supervisor must approve any changes, and who is responsible for monitoring and reporting on modifications made to various versions of the work. Provide safe and secure storage to prevent damage and unauthorized access. There is a prescribed procedure for seeking modifications implemented:

- Evaluate change requests and maintain control over system changes.
- Any team member can suggest changes to the project.
- The team should evaluate and research the order modification.

- When modifications are made, all parties will be informed.
- If the supervisor approves the change, the team will carry it out.
- The supervisor makes guidance to modify.

8. Independent Verification and Validation

Verification and validation testing are focused on confirming that the requirements of the HADER's system conform to the documented requirements and to the satisfaction of the client. Verification is typically performed before validation to ensure that the program adheres to its specifications. The verification test will be conducted by the team members by going over the requirements and design specifications, the software code, the testing plan, and the test cases. They next validate the system by ensuring that all requirements are met. Criteria for successful completion: The application should be verified and approved without any defects or errors. Additionally, it should meet the specifications.

9. Documentation

HADER's application will be tracked and monitored via milestone reports. The table details the document type, format standard, page count, and peer review type.

Table 9: HADER DOCUMENTATION

Document Type	Format Standard	Estimated Page Count	Peer Review Type
Proposal discussion	IEEE 1058-1998 Standard.	1	All team members & Supervisor
Introduction document (problem statement)	IEEE 1058-1998 Standard.	1	All team members
Project management plan (SPMP)	IEEE 1058-1998 Standard.	20	All team members & Supervisor
Midterm report	IEEE 1058-1998 Standard.	25	All team members & Supervisor
Software system requirement (SRS)	IEEE 1058-1998 Standard.	25	All team members & Supervisor
System design specification (SDS)	IEEE 1058-1998 Standard.	50	All team members & Supervisor
Final report	IEEE 1058-1998 Standard.	140	All team members & Supervisor
Database design	IEEE 1058-1998 Standard.	8	All team members & Supervisor
Application programming	-	-	All team members
User manual	IEEE 1058-1998 Standard.	10	All team members & Supervisor

Document Type	Format Standard	Estimated Page Count	Peer Review Type
Final testing report	IEEE 1058-1998 Standard.	180	All team members & Supervisor

10. Quality Assurance

Quality assurance actions must be implemented throughout the project's lifecycle. As a result, each member must adhere to the established procedures and standards in order to provide high-quality services. The following quality assurance activities will be implemented:

• The application's performance will be evaluated to identify and eliminate problems and to ensure that the application operates as intended.
• Any hint of a serious issue will be detected, reported, and corrected.
• Conducting reviews and audits.
• If any changes are necessary, the implemented components will be enhanced prior to going on to further components.

11. Reviews and Audits

The SPMP outlines the strategies and procedures for addressing and resolving issues that arise during the course of the project. Team members should communicate concerns to their supervisors personally or during online weekly meetings. Additionally, the document must be changed in the event of errors. The following are the primary stages that team members must do in order to address problems:

• Identify the issue.
• Generate solutions to the situation.
• Decide on a course of action.
• Utilize the solution.
• Evaluate the outcomes.

CHAPTER 3. SOFTWARE REQUIREMENTS SPECIFICATION

1. Introduction

The SRS document goes in depth on the system as it is being developed and its functions. This document will provide a general description of the system, including product viewpoint and product function, and Specific Requirements, which contain functional and non-functional requirements, as well as a use case diagram. These sections will be addressed in the SRS Chapter.

1.1 Purpose

The software requirements specification (SRS) document's purpose is to identify and describe Hader's application's requirements precisely, and it summarizes Hader's many needs.

The SRS will define the scope, perspective, and functions of the product. Additionally, to the requirements for external interfaces, system features, and nonfunctional requirements.

The SRS will serve as a contract between the Exam unit and the project team outlining how the system would operate. As a result, requirements conflicts, and development efforts will be reduced in the subsequent phases.

1.2 Document Conventions

This document is formatted in accordance with the IEEE standard for Software Requirements Specifications.

1.3 Intended Audience and Reading Suggestions

The intended audience are CCSIT students, Exam unit, Invigilators, and Instructors. HADER team members are highly encouraging them to read this paper. In addition, this paper differs according to the reader:

- Project supervisors and evaluators: to get a thorough understanding of the product to make necessary comments or ideas to assist team members in improving and updating the document.
- Application users: to ensure product satisfaction and future development.
- Team members, for advice since all subsequent stages, are dependent on them.
- It is strongly advised that you read the whole paper to understand the Hader application thoroughly.

1.4 Product Scope

Hader is one of the products that would facilitate students and exam units in CCSIT in the Exam attendance process and make them feel comfortable since it will solve the problem of calling each student's name and disturb their focus during exams, saving time. In addition, an authentication method is to provide student’s personal image along with their unique QR Code. Providing a personal image would solve the issue concerning if a student entered an exam behalf of another student.

2. Overall Description

This section is about the product perspective and product functions.

2.1 Product Perspective

The Hader is an application for CCSIT. The main purpose of this application is to enable CCSIT students to join their exam room smoothly and without any distraction and enables admins manage students and Invigilators. There are different types of Hader users: Admins, CCSIT students and invigilators. Each type of users will use the website based on their specified authority. The user must be connected to the internet in order to access Hader application, users can export sheets of

various functions to save them in the mobile and share them via email to any desired party in CCSIT.

2.2 Product Functions

The Hader application provides a set of different functions for each individual user as well as common functions for all users. The main users of the application are the Invigilator, students, and admins. The following table reviews all the major application functions.

➢ Admin Features:

Table 10: Admin Features

FEATURE	DESCRIPTION
MANAGE THE STUDENT'S EXCUSES	In this feature the admin can view and change the excuse status (Approve, decline and pending)
SHOW THE STUDENT'S ABSENT PERCENTAGES	In this feature the admin can view absent percentages for each student from the highest percentage to the lowest percentage.
MANAGE AN EXAM	In this feature the admin can manage, delete, and update exam information.
INVIGILATOR MANAGEMENT	In this feature the admin can scan the QR code for the invigilators, check in for the invigilator list and export the list.
ADMIN PROFILE	Admin can view their name, personal image, position, their QR code, change/reset the password, and log out

➢ Invigilator Features:

Table 11: Invigilator Features

FEATURE	DESCRIPTION
QR SCANNER TO SCAN STUDENTS QR CODE TO CHECK IN	Hader system enables the Invigilator to scan the students QR codes in order to check in for exam.
SHOW INVIGILATOR INFORMATION	Invigilator can view their name, personal image, position, their QR code, change/reset the password, and log out
EDIT, ADD, OR DELETE STUDENT'S ATTENDANCE STATUS	Hader system allows the Invigilator to Edit, add, or delete the student's attendance status manually (in case of an error).
INVIGILATOR DEVICES RECEIVE NOTIFICATION OF STUDENT'S ATTENDANCE (NOTIFICATION)	In this feature the system will send notifications to the Invigilator device after a student attendance scanned and added to the attendance list e.g. (the student Shahad is on time and was added to the list successfully).
EXPORT ATTENDANCE REPORT	In this feature the system allows the user (invigilators) to Export attended students' sheet as PDF.
SEND ATTENDANCE REPORTS VIA E-MAIL	In this feature the system allows the user (invigilators) to send attendance reports via E-mail.
STUDENTS' NAMES LIST	In this feature, the system displays to the user (invigilators) students' names list and status

- **Student Features:**

Table 12: Student Features

FEATURE	DESCRIPTION
DISPLAY QR CODE: TO BE SCANNED BY THE EXAM INVIGILATOR	Hader system displays few students' information like student's name, ID, and image for authentication and the unique QR Code to be scanned by the exam invigilator for student's check in and conducing exams by tapping the QR Code icon on the bottom menu of HADER application.
SHOW STUDENT INFORMATION	Hader system displays few students' information like student's name, ID, and image, major and their unique QR Code. This interface has functions such as reset password edit information and sign out.
UPLOAD EXAM EXCUSE	Hader system enables students to upload their exam excuse via the excuse interface and then choose (Upload excuses) from the menu, a student will be asked to submit the type of exam the student missed, date of excuse and will be asked to upload the excuse from their mobile.
VIEW EXCUSE STATUS	Hader system enables students to view their exam excuse status via the excuse interface and then choose (View Excuses Status) from the menu.

- **Instructor Features:**

Table 13 Instructor Features

FEATURE	DESCRIPTION
SHOW INSTRUCTOR INFORMATION	Hader system allows user (Instructor) to view their information in a profile page: (NAME, POSITION, QR code).
INSTRUCTORS RECEIVE NOTIFICATION	In this feature the system will send notifications to the instructor device about different activates assigning to invigilate and attendance lists.

- **Common Features:**

FEATURE	DESCRIPTION
LOGIN	In this feature the system allows Each user to be able to sign in into Hader system by providing their email and password.
FORGOT PASSWORD	Hader system allows the users to retrieve their forgotten passwords.
LOGOUT	In this feature the user can Sign out of the system.

2.3 User Classes and Characteristics

The table below demonstrates each user and the required skills, educational level, and technical experience needed to use the application:

Table 14: User Classes and Characteristics

Actor	Skills	Educational level	Technical experience
Admin	Understanding English language.	Master's degree and above.	- Medium knowledge in mobile application. - Manipulating (add/delete/edit).
Student	Understanding English language.	High educational level.	Basic knowledge in mobile application.
Invigilator	Understanding English language.	High educational degree and above.	Basic knowledge in mobile application.
instructor	Understanding English language.	High educational degree and above.	Basic knowledge in mobile application.

2.4 Operating Environment

Table 14 illustrates the needed software for the Hader application.

Table 15: Operating Environment

Software	Version
draw.io	13.9.9
Microsoft Office	Office 365
Figma	105.2
Flutter	2.5
Dart	2.7
Firebase	20.6.0

2.5 Design and Implementation Constraints

The following are the constraints of the Hader application:

- The application is for smartphones with iOS and Android Operating systems only.
- The user must be connected to the internet to access the application.
- The users must be a member of CCSIT to log in with their email and password.
- Only Admins can manage exams by assigning invigilators, date, time, and course.
- Only Admins can edit the Invigilator attendance list.
- Only the Invigilator can edit the student's attendance list.
- Only Admins can approve or decline students' excuses.

2.5 User Documentation

To ensure ease of usability, a user manual will be delivered along with the application at the end of the second semester of 2022.

2.6 Assumptions and Dependencies

The Hader application is predicated on the following assumptions and constraints:

- Hardware and software: The user has a smart mobile phone, downloaded the application, and is connected to the internet.
- Operating system: iOS and Android operating systems only.

- User characteristics: The application is targeted at four types of users: admins, instructors, invigilators, and students in CCSIT at IAU.

3. External Interface Requirements

This section describes the user interface, hardware interface, software interface, and communication interfaces.

3.1 User Interfaces

When any user opens the application, the login interface should be displayed at first. The user can click on the (Forgot Password) link on the log-in interface to reset the password. The user should Log In to interact with the application. Depending on the user type (Student, Admin, Invigilator, instructor) a specific interface by default should appear. For example, the student's QR code interface should appear first by default after logging in, so the students can check in into exams.

➢ Common Interfaces:

1. Startup Page

This page will appear every time when any user open HADER Application.

Figure 6: Startup Page

2. Sign in

Forgot Password

Users can reset their passwords by, clicking on forgot password and enter their phone number and then they'll receive a message with a 4-digit code with a link to reset the password.

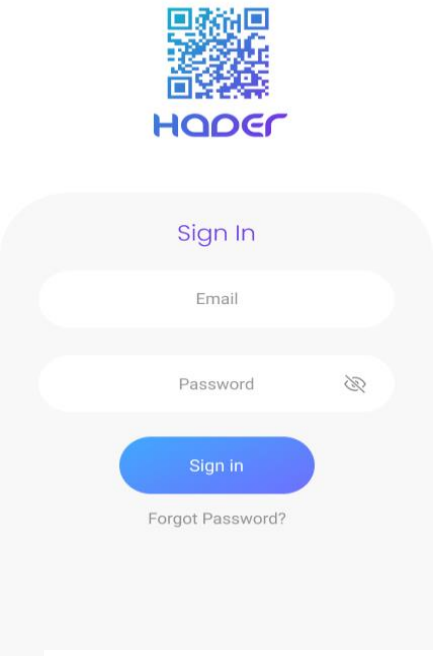
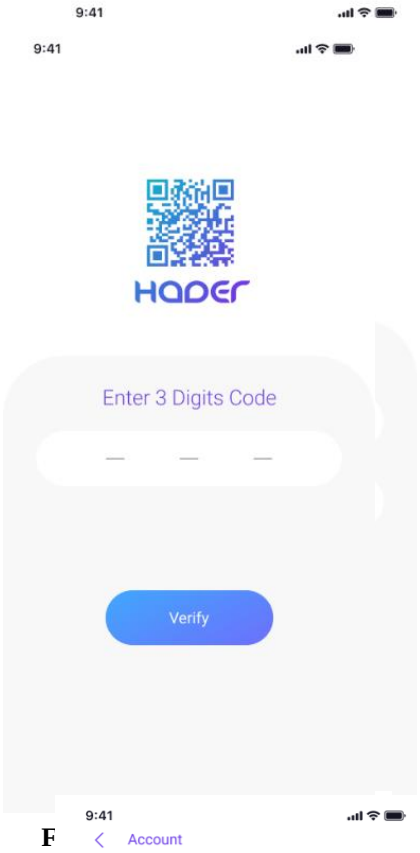


Figure 7 Forgot password interface

Figure 5: Sign in Interface

3. **Verifying Code**
4. **Change Password**

This interface appears after code verification and allows the user to change their password, also if the user already logged in and wants to change their password, the link will redirect to this page by visiting their profile and click on change password.



F

➢ Admin Interfaces:

1. **Admin Profile**

Profile interface shows the information of the admin, such as position and QR Code. In addition, it gives him the ability to change their password or log out.

Figure 10: Admin Profile

2. Student Management Menu Interface:

Manage interface contain two options manage invigilator's or manage students.

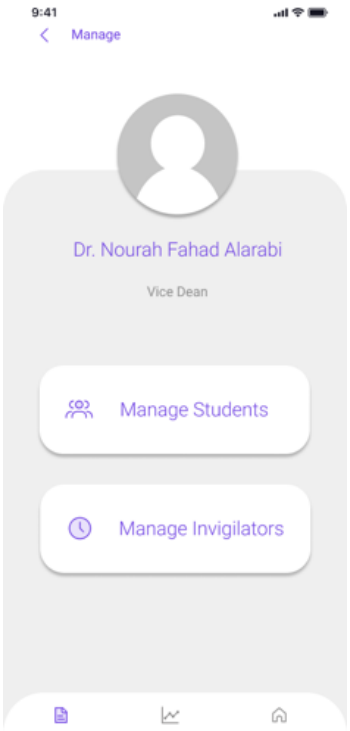


Figure 11: Student Management Menu

3. View Absent students Interface

The absent student's list with their excuses interface will allow the admin to manage the submitted exam excuses by(approve/decline) and share excuses if required.

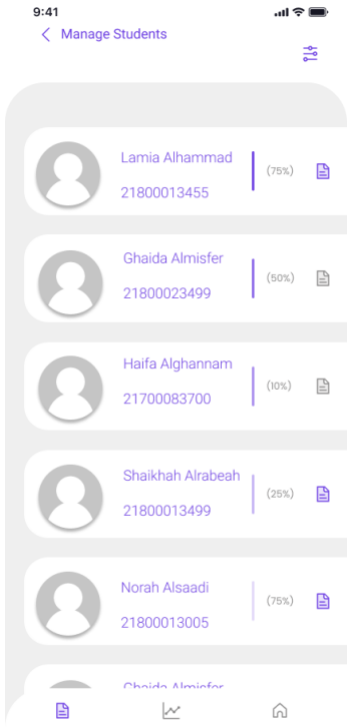


Figure 12: Absent student

4. View students' excuses

After clicking on a student this interface will land and shows the student's information such as student's name, ID, and absent percentage per each semester and the submitted excuse. In addition, this interface allows the admin to choose an excuse status(approve/decline/pend) and allows to share the excuse below the screen at the left corner.

Figure 13: View student's excuses

5. Share student's excuses

This is the popup action message if the admin choses to share or export student's excuse.

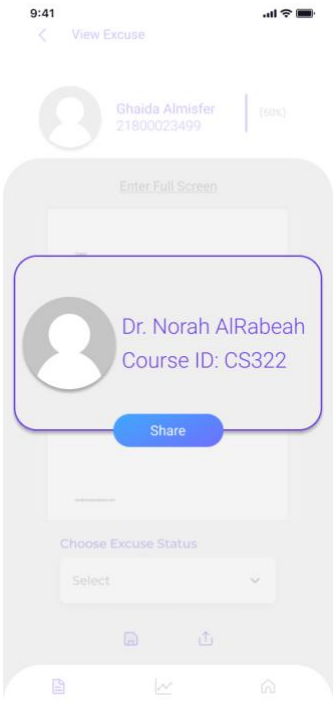


Figure 14: Share student's excuses

6. Manage invigilators

From the manage menu the admin can visit the manage instructor's page, the admin can manage the exam by viewing, adding, deleting, and updating exam information such as assigning or changing exam invigilators, exam time, exam date, or exam hall or course.

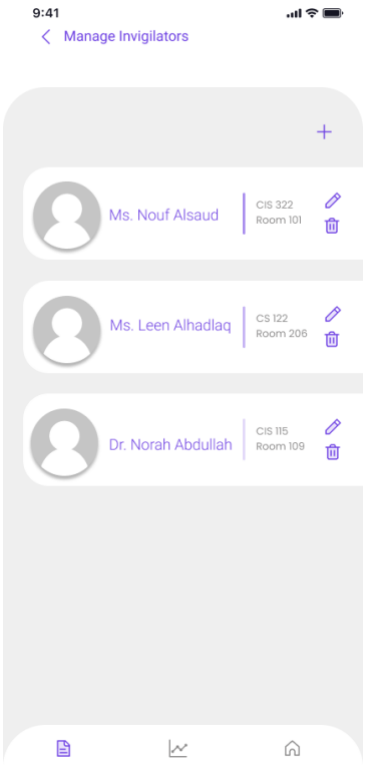


Figure 15: manage invigilators

7. Manage an exam

After clicking on the plus sign at the top right of the previous interface, this interface will land and allow the admin to add information and manage exam.

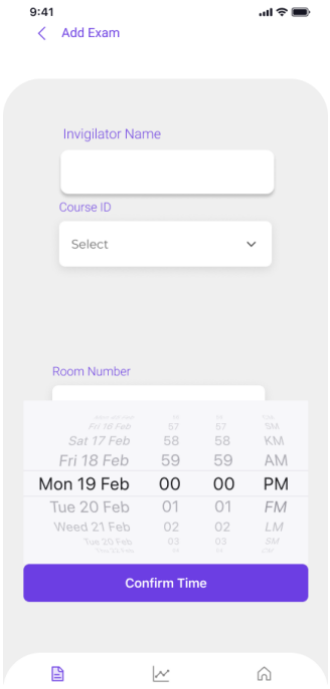


Figure 16: Manage an exam

8. Dashboard interface

After clicking on the dashboard icon in the below navigation bar, the dashboard interface will appear. The dashboard interface provides general statistics on the number of absent and present students and absenteeism rates of all exams per day and per month of all students in CCSIT.

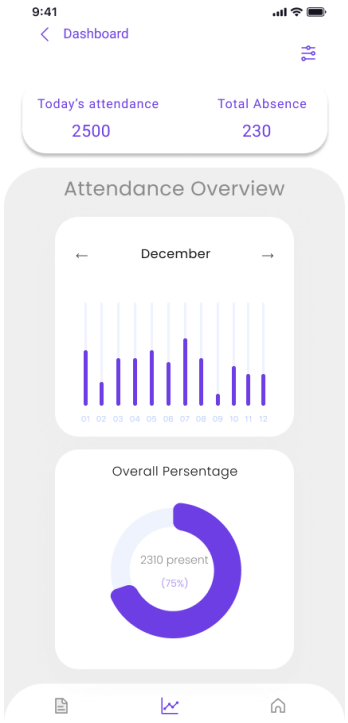


Figure 17: Dashboard interface

➢ Invigilator's Interfaces:

1. Monitoring Exam Schedule

The Invigilator chooses the date of the day on which the invigilating will take place.

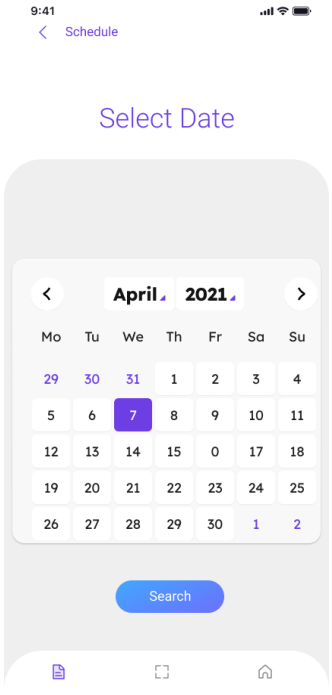


Figure 18: Monitoring exam Schedule

2. Invigilating Days Interface

After clicking on the date, the invigilating days interface will appear that shows information about exams.

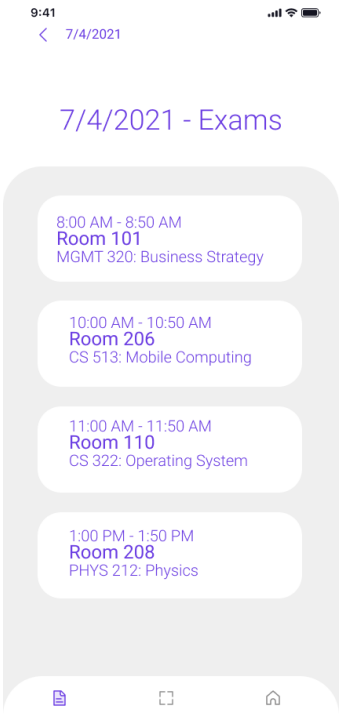


Figure 19: Invigilating days

3. QR Code Scanner

The QR code scanner interface allows invigilators to scan student's QR code.

4. Notification of Identified Student

After Scanning student's QR Code this notification should appear. It contains the student personal image, name, and ID.

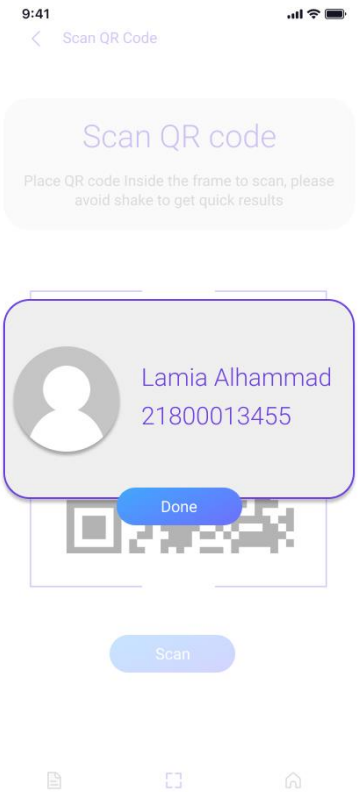
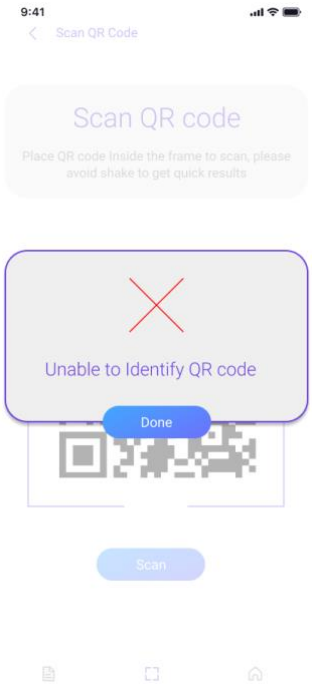


Figure 21: Identified Student

5. Notification of Unidentified Student

After Scanning student's QR Code and it fails to identify the QR code of the student, this error message should appear.



6. Student's Exam Attendance List

After Scanning student's QR Code and its successes to identify them, this list should appear containing the students' names, ID, personal photo, and a checked checkbox next to the student that indicates that the student attended the exam. In this interface, the invigilator can edit student's attendance in case of any error. In addition, the invigilator can share and export the list if required.

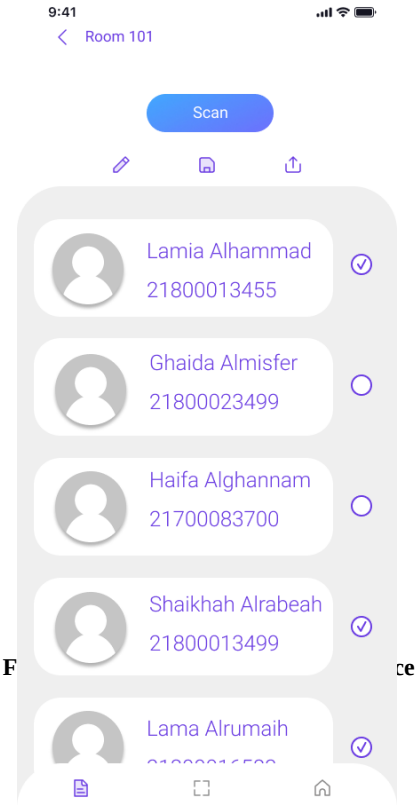


Figure 23: Student's attendance

7. Notification of Saved Exam Attendance List

The invigilator can save the attendance and this notification will appear.

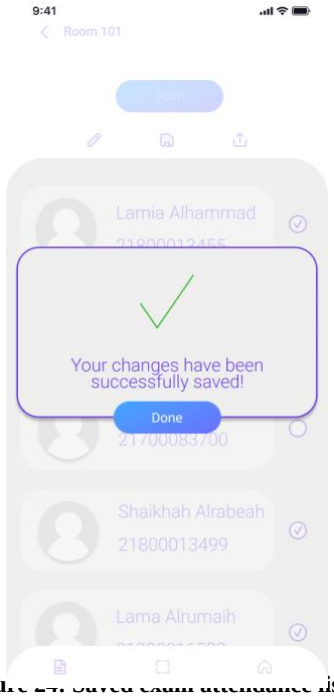


Figure 24: Saved exam attendance list

8. Notification of Shared Exam Attendance List

The invigilator after saving can share the attendance list with the course instructor or share and export the list as a PDF if required.

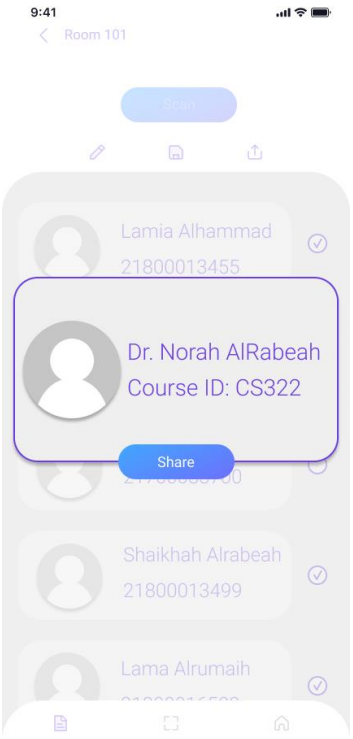


Figure 25: Shared Exam attendance list

9. Invigilator Profile

The profile contains the invigilator's personal photo, their name, Position, QR Code. The invigilator can logout and change their password through this page.

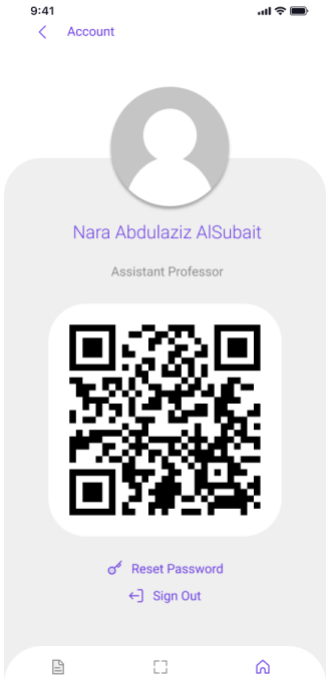


Figure 26: Invigilator Profile

➢ Instructor Interfaces:

1. Notifications Interface

The instructor is notified when assigned as an invigilator and can switch to the invigilator interface when the three points are pressed. Also, upon receiving a list of students' names, the instructor is notified, and the list is uploaded via the three dots.

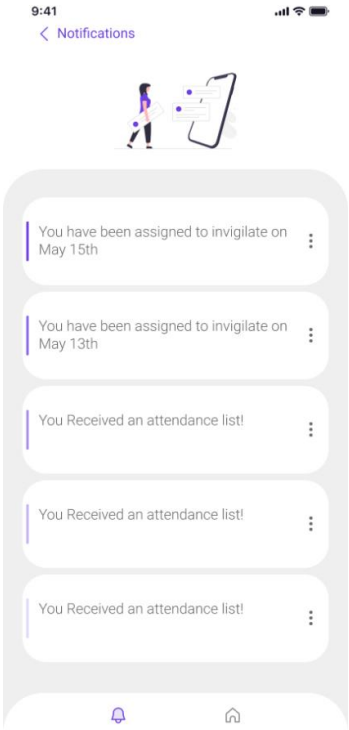


Figure 27: Notifications Interface

2. Profile Interface

The profile contains the instructor's personal photo, their name, Position, QR Code. The instructor can logout and change their password through this page.

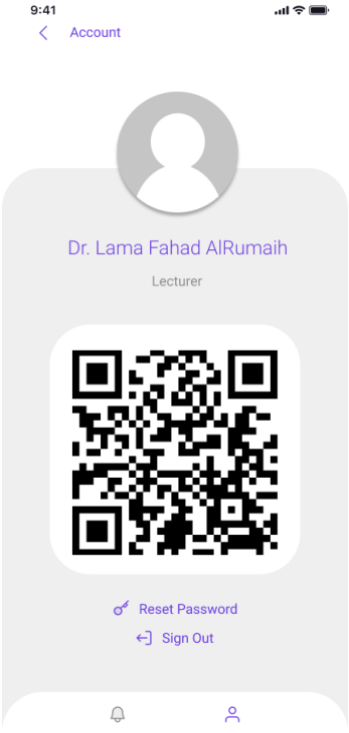


Figure 28: Instructor's Profile

➢ Student Interfaces:

1. Student's Profile

The profile contains the student's personal photo, their name, College, QR Code.

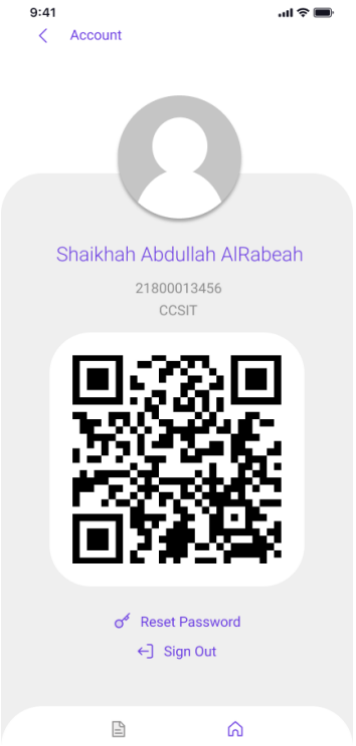
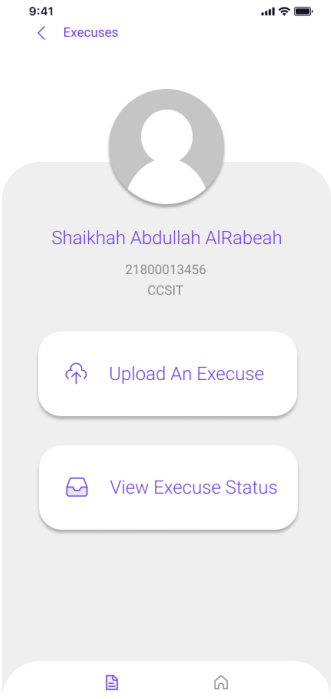


Figure 29: Student's Profile

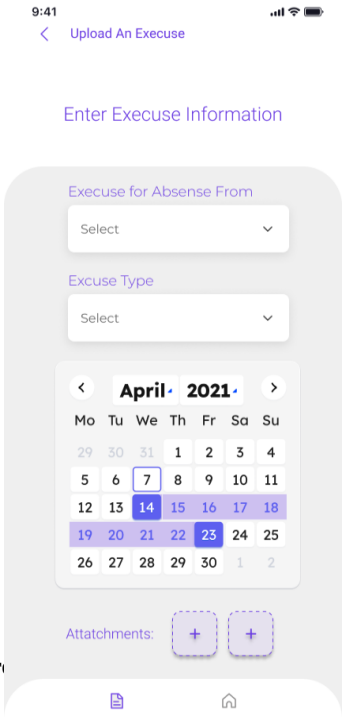
2. Manage Excuses

The student must select two options from the menu whether to upload an excuse or see the status of excuses that were previously submitted.



3. Upload an Excuse

The student must select two options from the menu whether to upload an excuse or see the status of excuses that were previously submitted.



Figur

Figure 31: Upload an Excuse

4. View Excuses Status

The status of the submitted excuse is shown here.

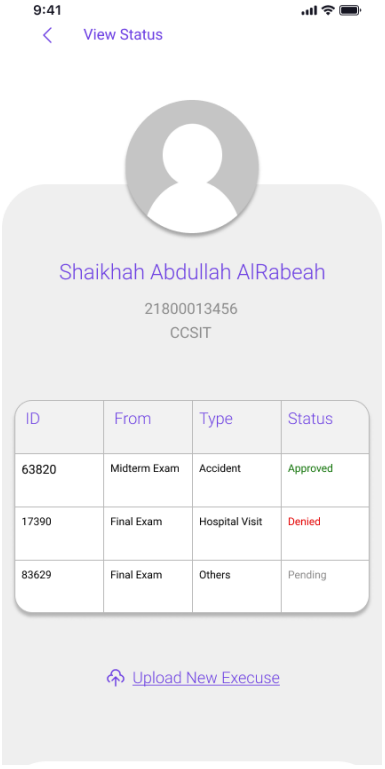


Figure 32: View Excuse Status

3.2 Hardware Interfaces

HADER works on all versions of iOS and Android smartphones only, by that it requires Dart and Flutter to develop the application. In addition, since the mobile application requires a stable wireless connection to function, internet access is necessary such as (a wireless LAN, WAN, Wi-Fi modem, or a 3G, 4G, 5G connection via telecommunication providers … etc.).

3.3 Software Interfaces

To work with the HADER application, any smartphone with iOS and Android operating system is needed with a stable internet connection is required to connect to a Firebase database for the timetabled classes to load, store, and update all types of user information. In addition, students can upload exam excuses from files or photo libraries as a PDF or PNG. The admin can export and share sheets of (exam invigilators/ Invigilators check-in/student excuses) list via E-mail as a PDF, to share it with any desired party in CCSIT. Invigilators can export and share sheets of student's attendance lists via E-mail as a PDF to share with any party in CCSIT that requires attendance sheets.

3.4 Communications Interfaces

The communication between with the application and different system users is vital since application components work as one unit. The client-server model must be followed in the communication architecture. The system must be connected to the internet and use a database server as a communication medium among different users to perform communication functions.

3.4.1 Student Features

1) Display QR Code: To be scanned by the exam invigilator

Description and priority

Hader system displays a few students' information like student's name, ID, and image for authentication and the unique QR Code to be scanned by the exam invigilator for student's check-in and conducting exams by tapping the QR Code icon on the bottom menu of HADER application.

Priority:	High

Stimulus/Response Sequences

Stimulus:	Students show their QR code to invigilators to be scanned.
Response:	System sends a notification to the invigilator about attendance status (late/ on time) and shows exact time of checking in.

Functional Requirements

Feature Requirements: Display QR Code: To be scanned by the exam invigilator.

Functional Requirements	Requirement Feature
REQ-1	User (Student) must be logged in.
REQ-2	User (Invigilator) must be logged in and connected to the internet.

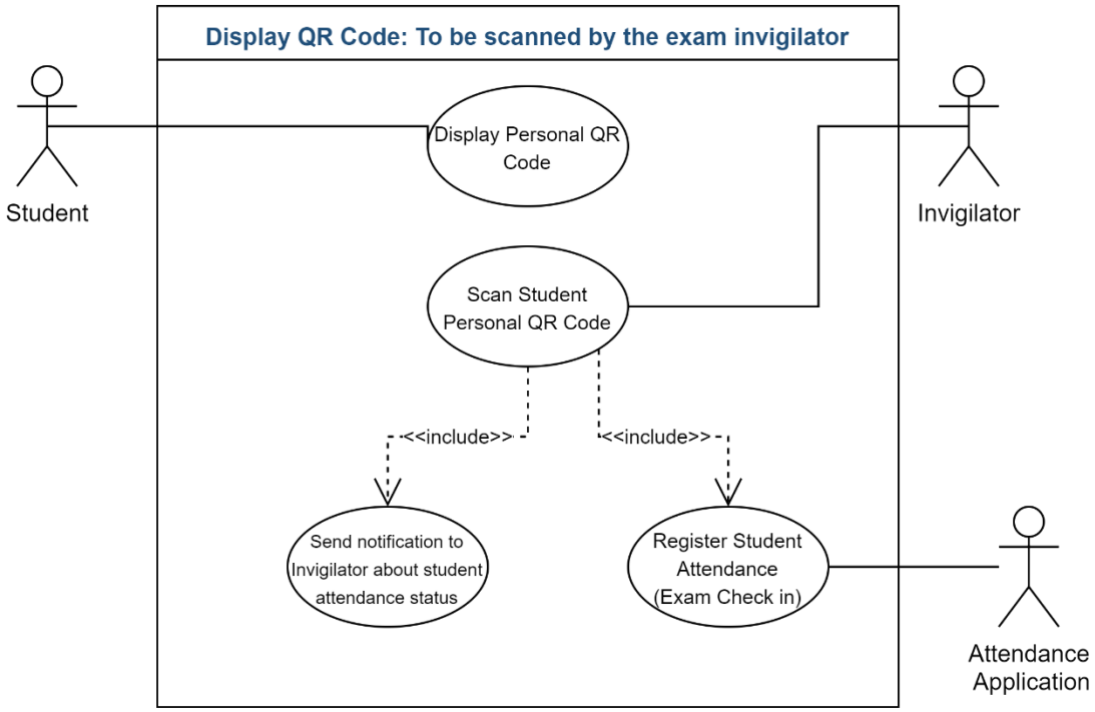


Figure 33: Register Attendance

2) Show Student information

Description and priority:

Hader system displays a few students' information like student's name, ID, image, major, and their unique QR Code. This interface has functions such as resetting password edit information and signing out.

Priority:	Medium

Stimulus/Response Sequences

Stimulus:	Students click on the profile icon on the right side at the bottom of the page
Response:	System uploads all information.

Functional Requirements

Feature Requirements: Show Student information

Functional Requirements	Requirement Feature
REQ-1	User (Student) must be logged in.
REQ-2	User (Invigilator) must be logged in and connected to the internet.

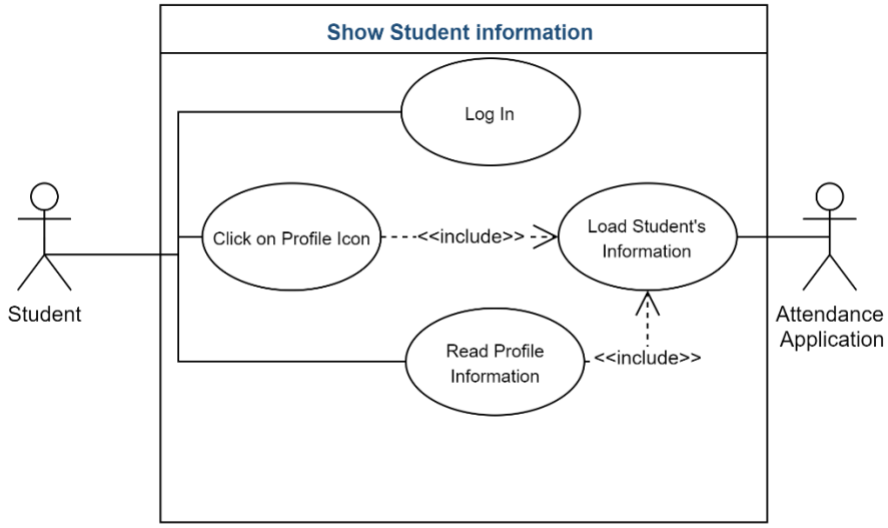


Figure 34: Show Student's Information

3) Upload Exam Excuse

Description and Priority

Hader system enables students to upload their exam excuses via the excuse interface and then choose (Upload excuses) from the menu, a student will be asked to submit the type of exam the student missed, the date of excuse and will be asked to upload the excuse from their mobile and then generates a Reference ID of the uploaded excuse.

Priority:	High

Stimulus/Response Sequences

Stimulus:	Students fill all the requires fields to upload an excuse.
Response:	System will send a notification to state if the excuse has been uploaded successfully or not and generates a reference ID to students about their excuse so student can check it later

Functional Requirements

Feature Requirements: Upload exam excuse

Functional Requirements	Requirement Feature
REQ-1	Student must be logged in.
REQ-2	Student must fill all the fields.
REQ-3	Student should upload the excuse as a PDF or PNG only.

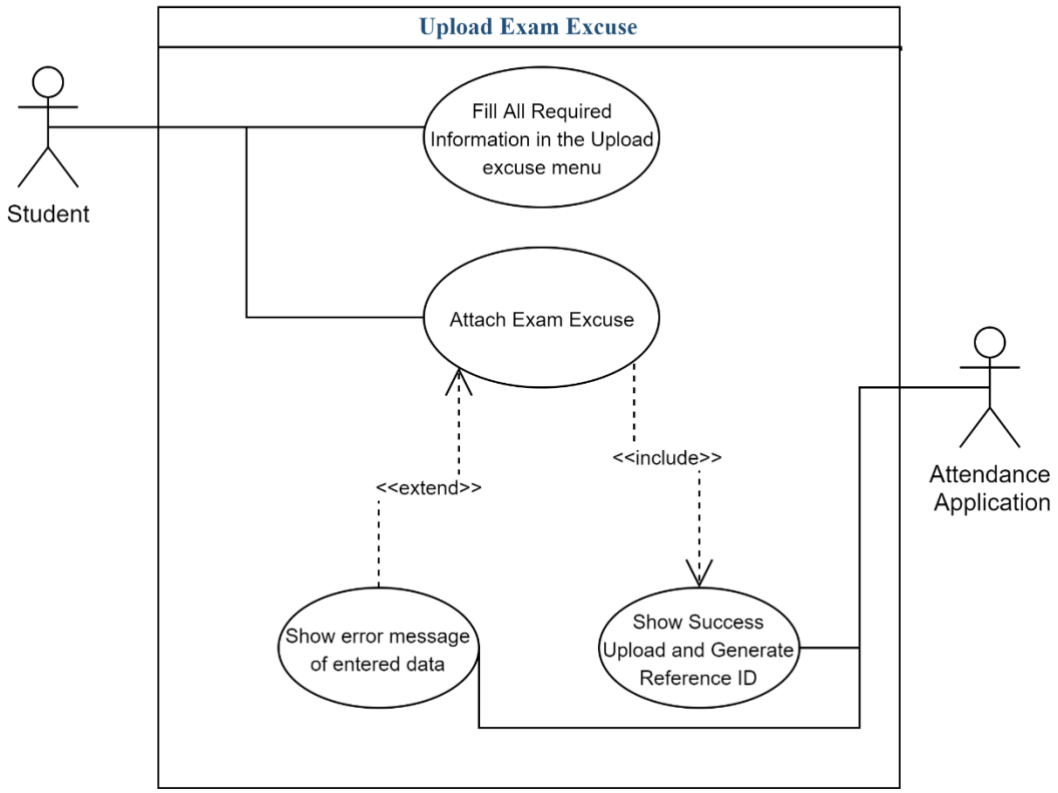


Figure 35: Upload Exam Excuse

4) View Excuse status

Description and Priority

Hader system enables students to view their exam excuse status via the excuse interface and then choose (View Excuses Status) from the menu, a student must choose from the dropdown list their excuse Reference ID.

Priority:	High

Stimulus/Response Sequences

Stimulus:	Students must choose from the view excuses from the excuse's menu.
Response:	System will upload a table with each excuse status (Pending, Approved, Declined).

Functional Requirements

Feature Requirements: View Excuse Status

Functional Requirements	Requirement Feature

REQ-1	Student must be logged in.

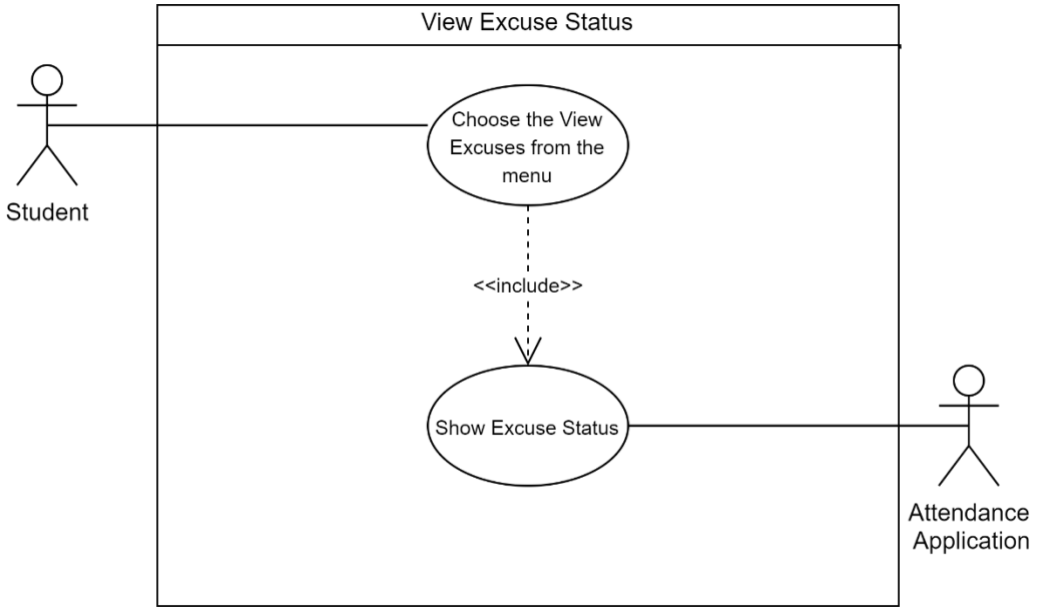


Figure 36: View Excuse Status

❖ Admin Features

1. Manage the student's excuses.

Description and Priority:

In this feature, the admin can view and change the excuse status (Approve, decline, and pending)

Priority:	High

Stimulus/Response Sequences

Stimulus:	Admin view the excuses.
Response:	System shows the excuse status.
Stimulus:	Admin change excuse status.
Response:	System shows the result and allows the admin to export the excuses list as PDF

Functional Requirements:

Feature Requirements: Excuses Management.

Functional Requirements	Requirement Feature
REQ-1	Admin must be logged in.

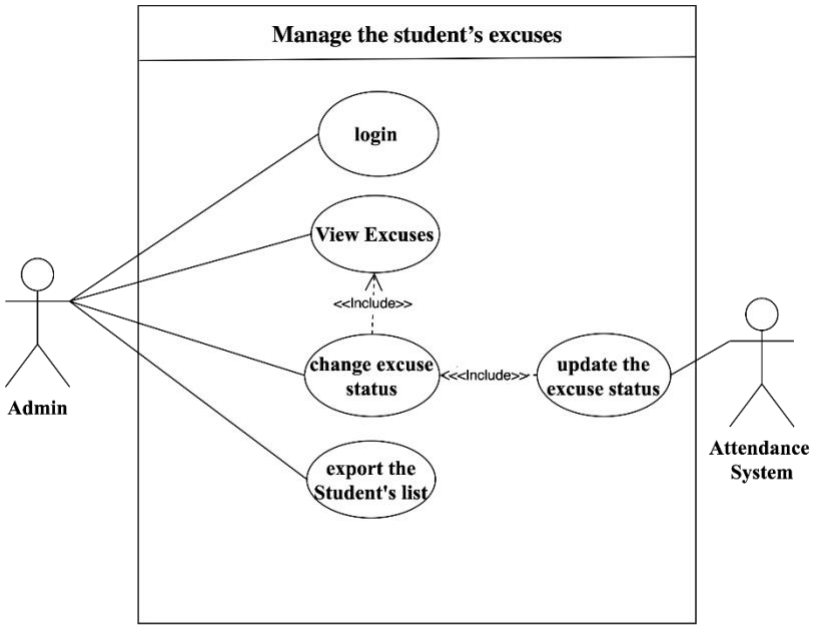


Figure 37: Manage Student's Excuses

2. Show the student's absent percentages and absent numbers (Dashboard).

Description and Priority

In this feature, the admin can view the absent rate and number of absent students.

Priority:	medium

Stimulus/Response Sequences

Stimulus:	Admin view the student's absent information (for the midterm and final exam).
Response:	System shows the statistics of exams for each day

Functional Requirements

Feature Requirements: Student Management.

Functional Requirements	Requirement Feature
REQ-1	Admin must be logged in.

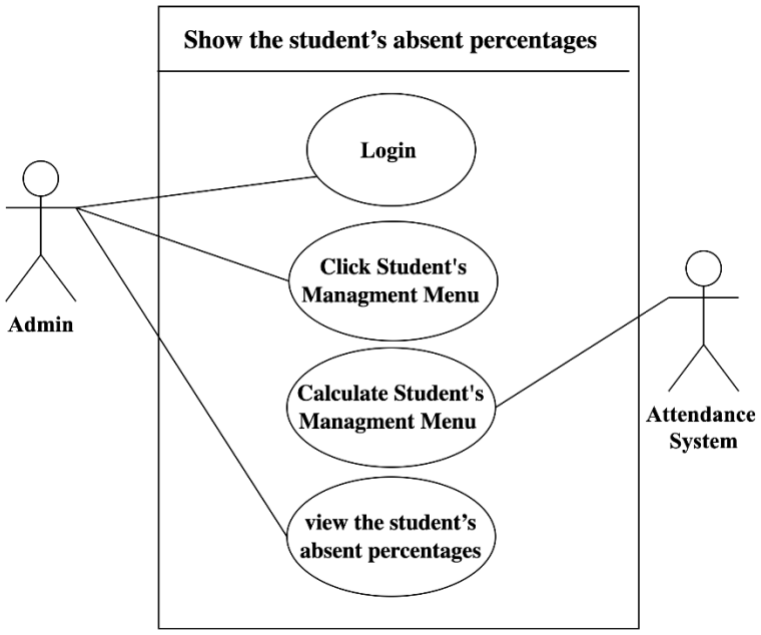


Figure 38: Show Student's Absent Percentage

3. Manage an exam.

Description and Priority

In this feature, the admin can create, delete, and update exam information.

Priority:	High

Stimulus/Response Sequences

Stimulus:	Admin manages new exam by choosing the room, invigilator, course, date, and time.
Response:	Instructors/ invigilators will receive notification about invigilating an exam
Stimulus:	Admin can delete and update exam information.
Response:	The system will ask the admin if he is sure to delete or update and then save it.

Functional Requirements

Feature Requirements: Exam Management.

Functional Requirements	Requirement Feature

REQ-1	Admin must be logged in.
REQ-2	Admin must fill all the fields.

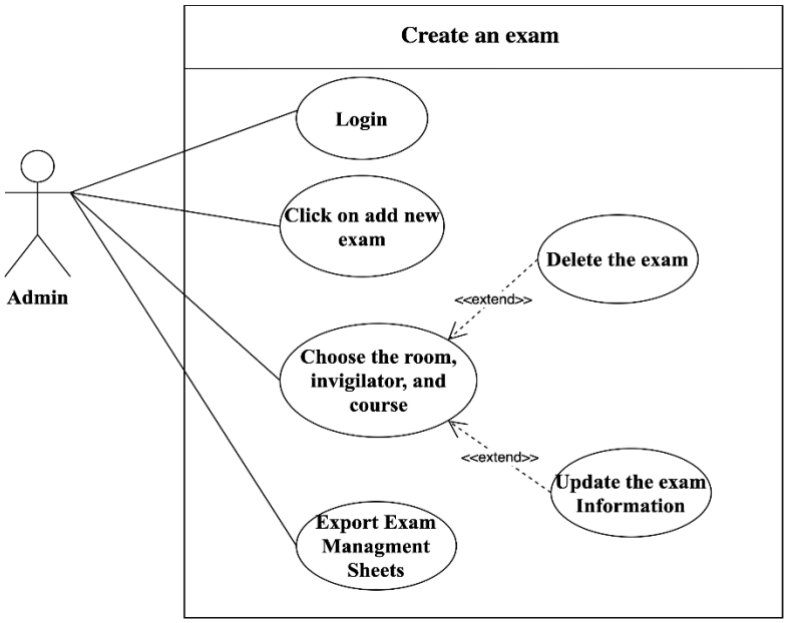


Figure 39: Create an exam

4. Invigilator management.

Description and Priority

In this feature, the admin can scan the QR code for the invigilators and export the invigilator's attendance list.

Priority:	High

Stimulus/Response Sequences

Stimulus:	Admin will scan the QR code for the invigilators.
Response:	System will generate invigilators attendance list.
Stimulus:	Admin can view and edit the invigilators attendance list.
Response:	The system will allow the admin to export the invigilators list as PDF.

Functional Requirements

Feature Requirements: Invigilator Management.

Functional Requirements	Requirement Feature
REQ-1	Admin must be logged in.

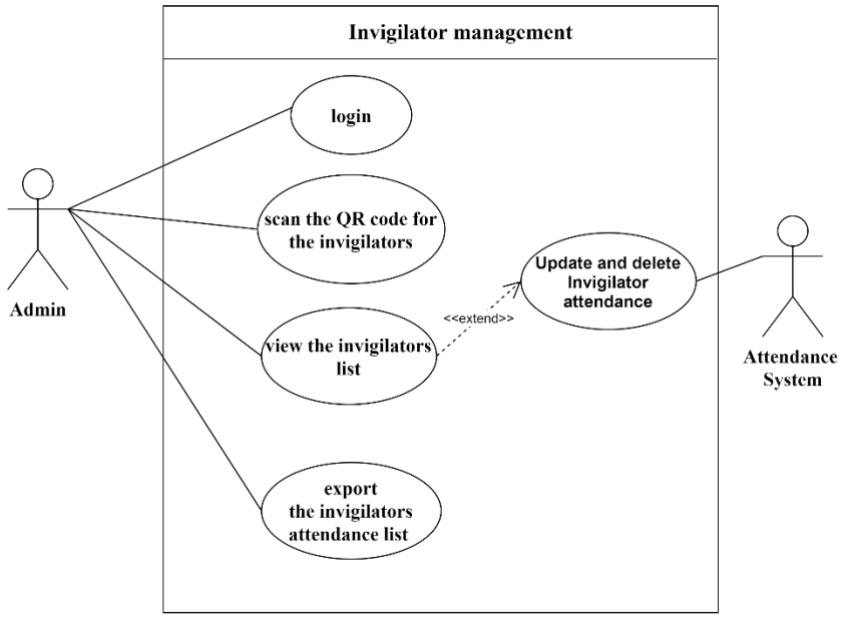


Figure 40: Invigilator's management

5. View Admin Profile.

Description and Priority

Admin can view the position, change/reset the password, and log out.

Priority:	medium

Stimulus/Response Sequences

Stimulus:	Admin can view the position
Stimulus:	Admin can change/reset the password
Respond:	Admin can log out

Functional Requirements

Feature Requirements: Admin Profile

Functional Requirements	Requirement Feature
REQ-1	Admin must be logged in.

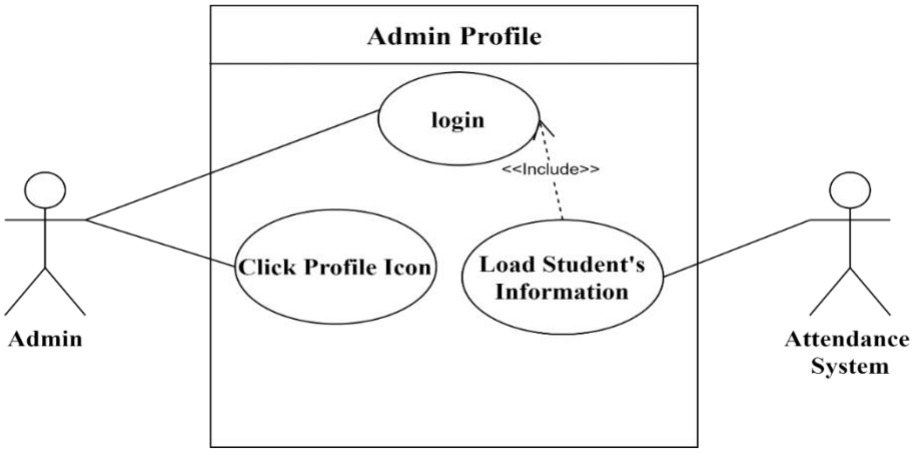


Figure 41: Admin Profile

3.4.2 Invigilator Features

1. QR reader to Scan student's QR codes to check in

Description and Priority

Hader system enables the Invigilator to scan the student's QR codes to check in the exam.

Priority:	High

Stimulus/Response Sequences

Stimulus:	Invigilators scan the students QR code
Response:	System registers the student attendance in the list.

Functional Requirements

Feature Requirements: QR reader to Scan student's QR code to check in.

Functional Requirements	Requirement Feature
REQ-1	User (invigilators) must be logged in.

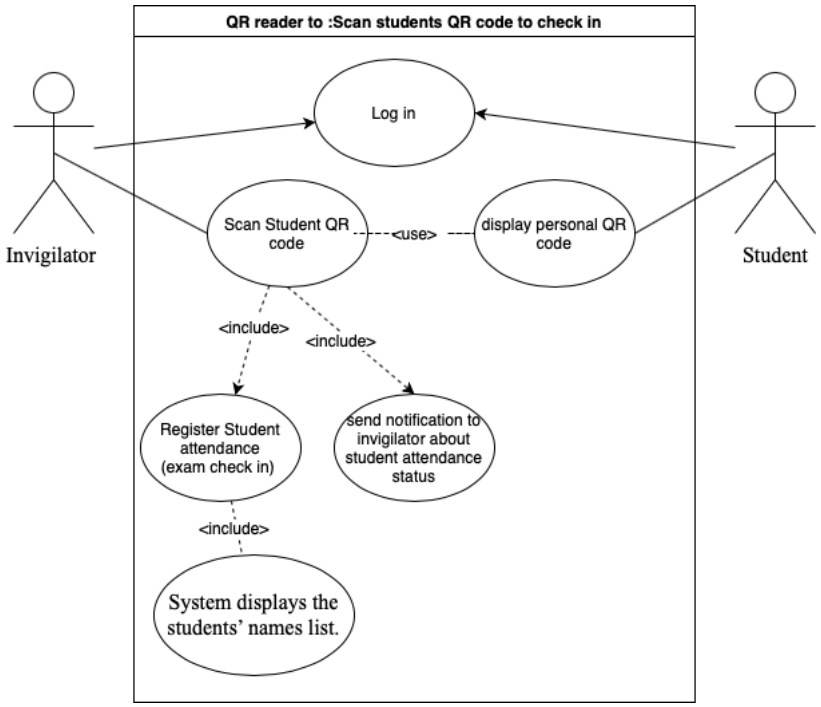


Figure 42: Scan QR

2. Show Invigilator Information

Description and Priority

Hader system allows the user (Invigilator) to view their information on a profile page: (NAME, ID, EMAIL, POSITION, QR code).

Priority:	medium

Stimulus/Response Sequences

Stimulus:	User (Invigilators) display profile page.
Response:	System shows the user information.

Functional Requirements

Feature Requirements: Show Invigilator Information.

Functional Requirements	Requirement Feature
REQ-1	user (invigilators) must be logged in.
REQ-2	user (invigilators) Request to display profile page.

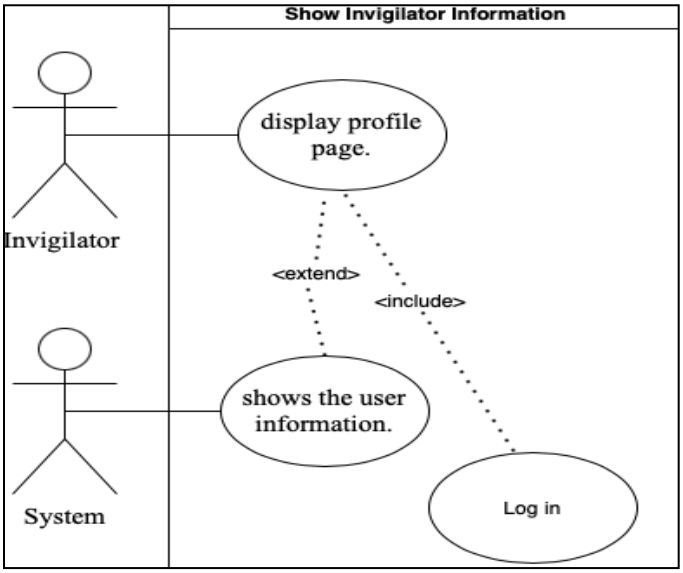
Figure 43: Show Invigilator's Information

3. Edit, add, or delete a student's attendance status.

Description and Priority

Hader system allows the Invigilator to Edit, add, or delete the student's attendance status manually (in case of an error).

Priority:	High



Stimulus/Response Sequences

Stimulus:	User (Invigilators) display the attendance list.
Response:	System shows the attendance list.
Stimulus:	User (Invigilators) search for student name or id.
Response:	System shows the result and allows the User (Invigilators) to change student status (Edit, add, or delete).
Stimulus	User (Invigilators) saves the attendance list.
Response:	System saves the changes.

Functional Requirements

Feature Requirements: Show Invigilator Information.

Functional Requirements	Requirement Feature
REQ-1	user (invigilators) must be logged in.
REQ-2	Request to display attendance list.
REQ-3	Search for Student using name or id.

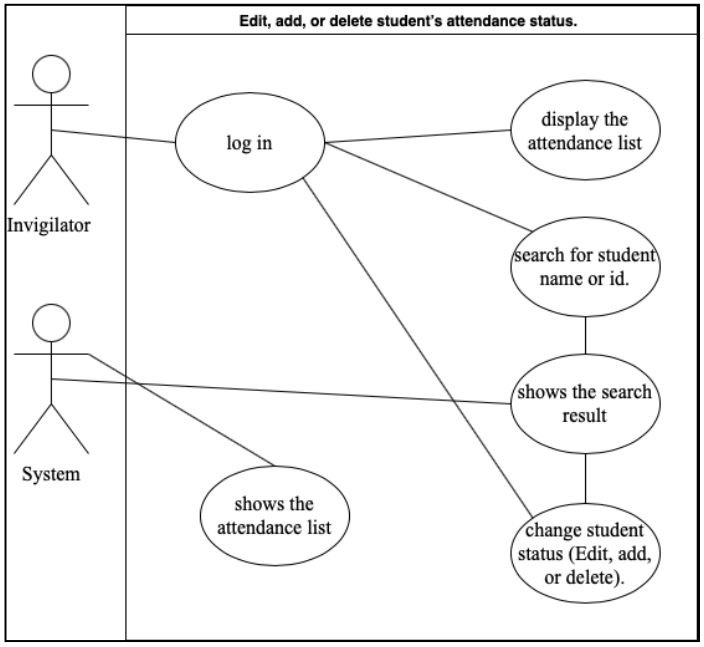


Figure 44: Maintain Student's Status

4. Invigilator devices receive notification of student's attendance

Description and Priority

In this feature, the system will send notifications to the Invigilator device after a student's attendance is scanned and added to the attendance list e.g. (the student Shahad is on time and was added to the list successfully).

Priority:	medium

Stimulus/Response Sequences

Stimulus:	User (Invigilators) scan a student QR code.
Response:	System registers the student attendance in the list.
Stimulus:	The system registered the student.
Response:	System send notification to Invigilator device.

Functional Requirements

Feature Requirements: Show Invigilator Information.

Functional Requirements	Requirement Feature
REQ-1	user (invigilators) must be logged in.
REQ-2	A student QR code must be scanned and registered.

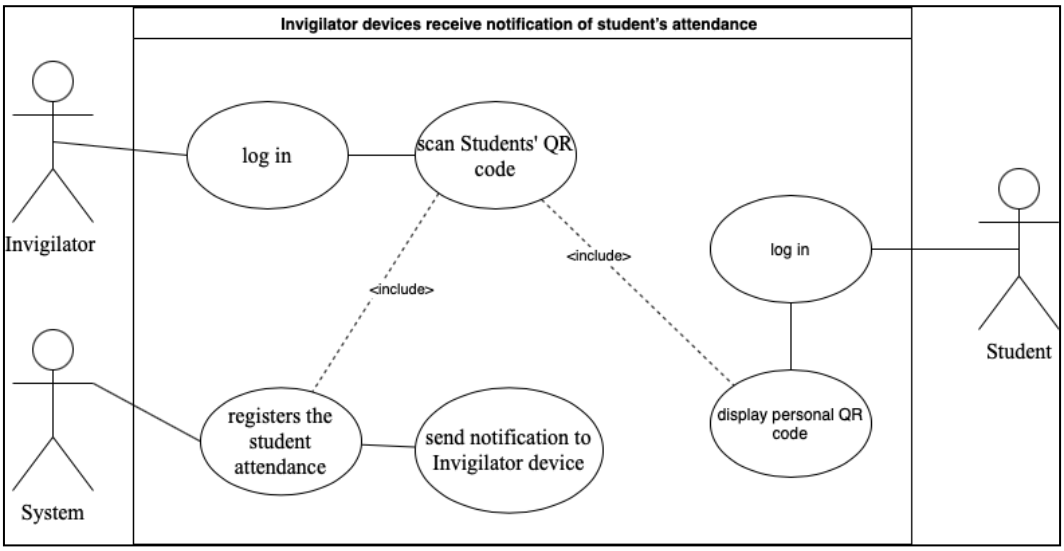


Figure 45: Receive Notification of Student's Attendance

5. Export attendance report as PDF.

Description and Priority

In this feature the system allows the user (invigilators) to Export attended students' sheets as PDF, XML, etc.

Priority:	medium

Stimulus/Response Sequences

Stimulus:	User (Invigilators) request to export attendance report
Response:	System gives options of report file types like; PDF, XML or etc.
Stimulus:	User (Invigilators) choose the appropriate type.
Response:	System generates the report.

Functional Requirements

Feature Requirements: Show Invigilator Information.

Functional Requirements	Requirement Feature
REQ-1	user (invigilators) must be logged in.
REQ-2	User must choose a report file type.

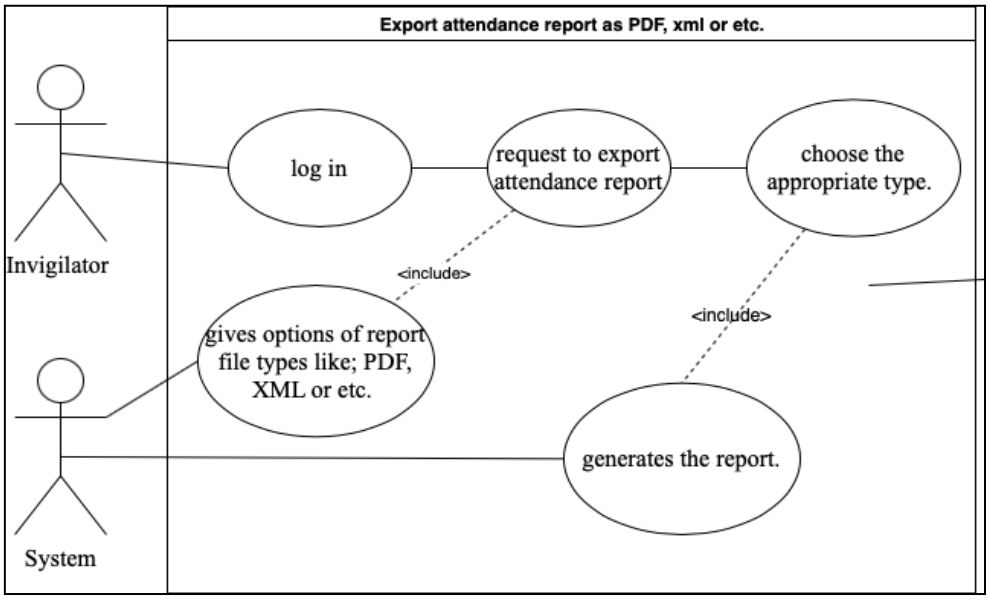


Figure 46: Export Attendance Report

6. Save and share attendance reports.

Description and Priority

In this feature the system allows the user (invigilators) to send attendance reports via E-mail.

Priority:	low

Stimulus/Response Sequences

Stimulus:	User (Invigilators) request to send attendance reports via E-mail.
Response:	System generates the report.
Stimulus:	After the system generates the report.
Response:	System opens the default email app then send the email.

Functional Requirements

Feature Requirements: Show Invigilator Information.

Functional Requirements	Requirement Feature
REQ-1	user (invigilators) must be logged in.

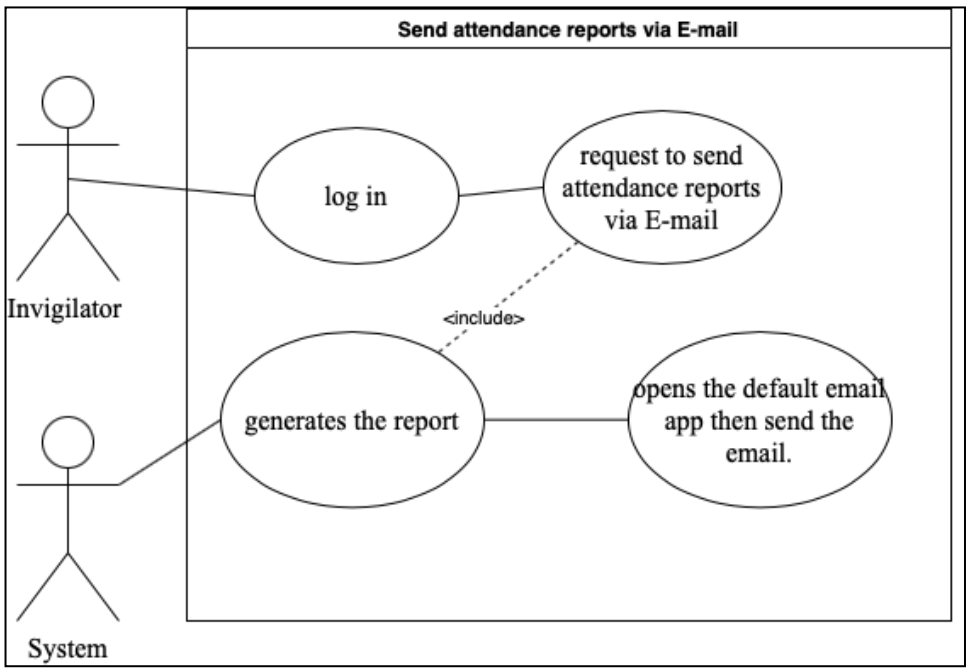


Figure 47: Send Attendance Report

7. Scan and View students' attendance names list.

Description and Priority

In this feature, the system displays to the user (invigilators) students' names list and status.

Priority:	High

Stimulus/Response Sequences

Stimulus:	User (Invigilators) scans the first student QR code.
Response:	System registers the student.
Stimulus:	After the system registers the student.
Response:	System displays the students' names list.

Functional Requirements

Feature Requirements: Show Invigilator Information.

Functional Requirements	Requirement Feature
REQ-1	User (invigilators) must be logged in.
REQ-2	User students' names list is disabled by default until first student QR code scanned.

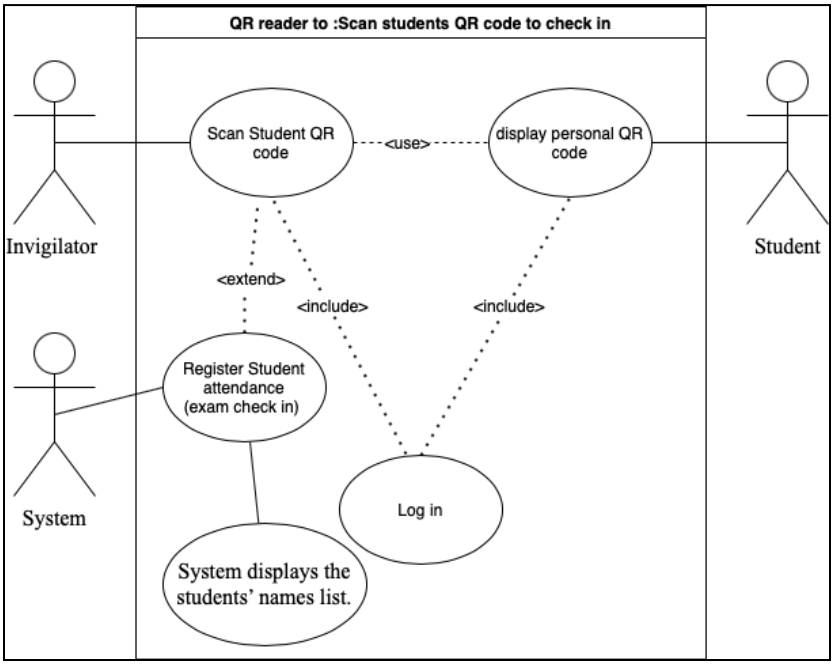


Figure 48: Scan and View students' attendance names list.

3.4.3 Instructor Features

1. Show Instructor Information

Description and Priority

Hader system allows the user (Instructor) to view their information on a profile page: (NAME, POSITION, QR code).

Priority:	medium

Stimulus/Response Sequences

Stimulus:	User (Instructor) display profile page.
Response:	System shows the user information.

Functional Requirements

Feature Requirements: **Show Instructor Information**

Functional Requirements	Requirement Feature
REQ-1	User (instructor) must be logged in.
REQ-2	User (instructor) Request to display profile page.

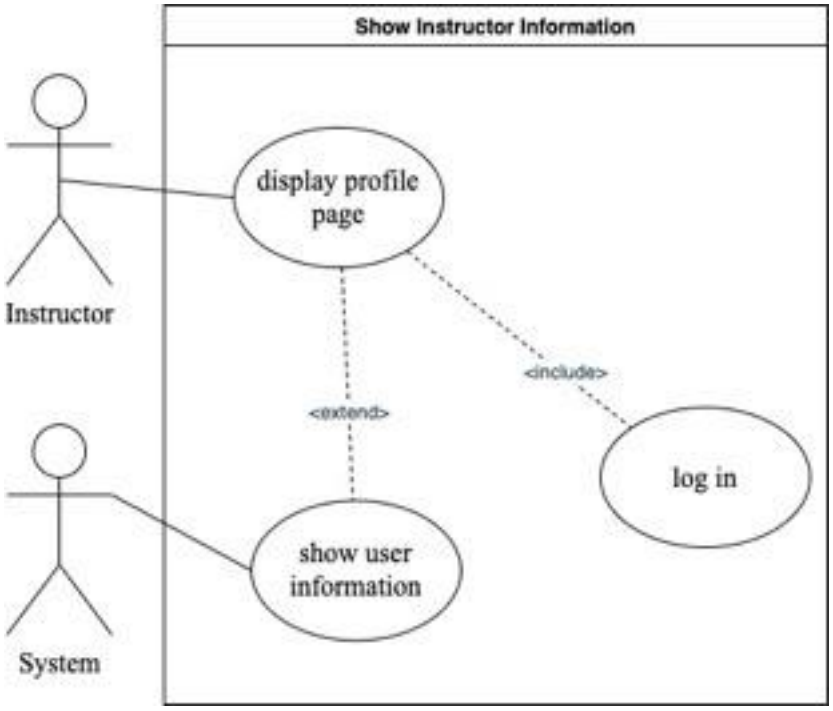


Figure 49 Show Instructor Information

2. Instructors receive notification

Description and Priority

In this feature, the system will send notifications to the instructor device about different activities assigned to invigilate and attendance lists.

Priority:	medium

Stimulus/Response Sequences

Stimulus:	User (instructor) is involved in an activity
Response:	the system will notify the user.

Functional Requirements

Feature Requirements: **Instructor receive notification**

Functional Requirements	Requirement Feature
REQ-1	User (instructor) must be logged in.

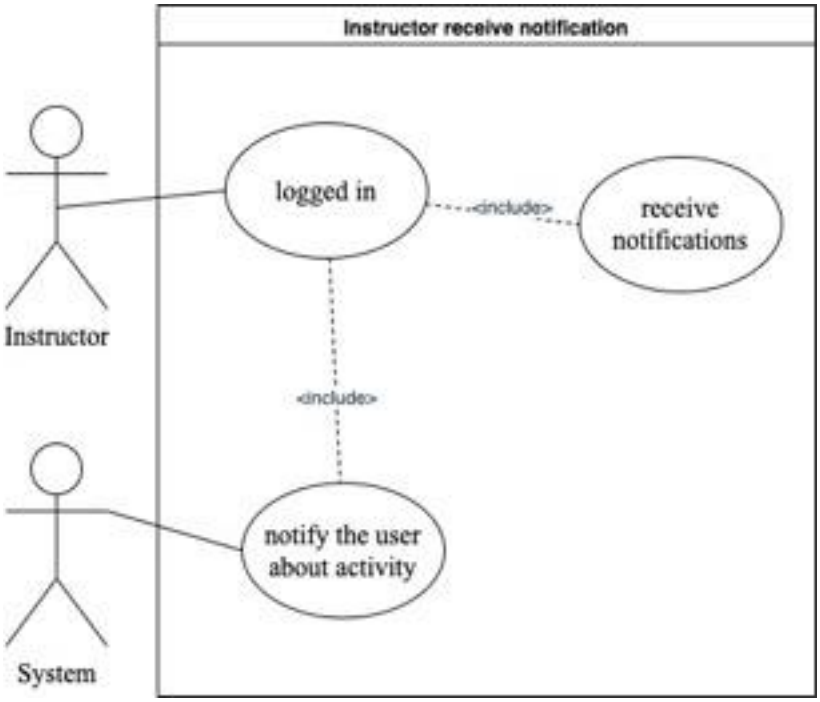


Figure 50 Instructor receive notification

3.4.4 Common Features

1. Sign in

Description and Priority

In this feature the system allows Each user to be able to sign in to the Hader system by providing their email and password.

Priority:	High

Stimulus/Response Sequences

Stimulus:	The user enters their e-mail and password in sign in page.
Response:	Entering to the application home page.

Functional Requirements

Feature Requirements: Sign in.

Functional Requirements	Requirement Feature
REQ-1	User must enter valid data.

Forgot password.

Description and Priority

Hader system allows the users to retrieve their forgotten passwords.

Priority:	medium

Stimulus/Response Sequences

Stimulus:	User request the forgot password function.
Response:	System asks for the user's phone number.
Stimulus:	User enters the data.
Response:	The system shows the reset password page.

Stimulus	User enters new password.
Response:	System saves the changes.

Functional Requirements

Feature Requirements: Forgot password.

Functional Requirements	Requirement Feature
REQ-1	The user must provide correct data.

2. Sign out

Description and Priority

In this feature, the user can Sign out of the system.

Priority:	High

Stimulus/Response Sequences

Stimulus:	User request to sign out.
Response:	System signs the user out and go back to sign in page.

Functional Requirements

Feature Requirements: Sign out.

Functional Requirements	Requirement Feature
REQ-1	The user must be logged in.

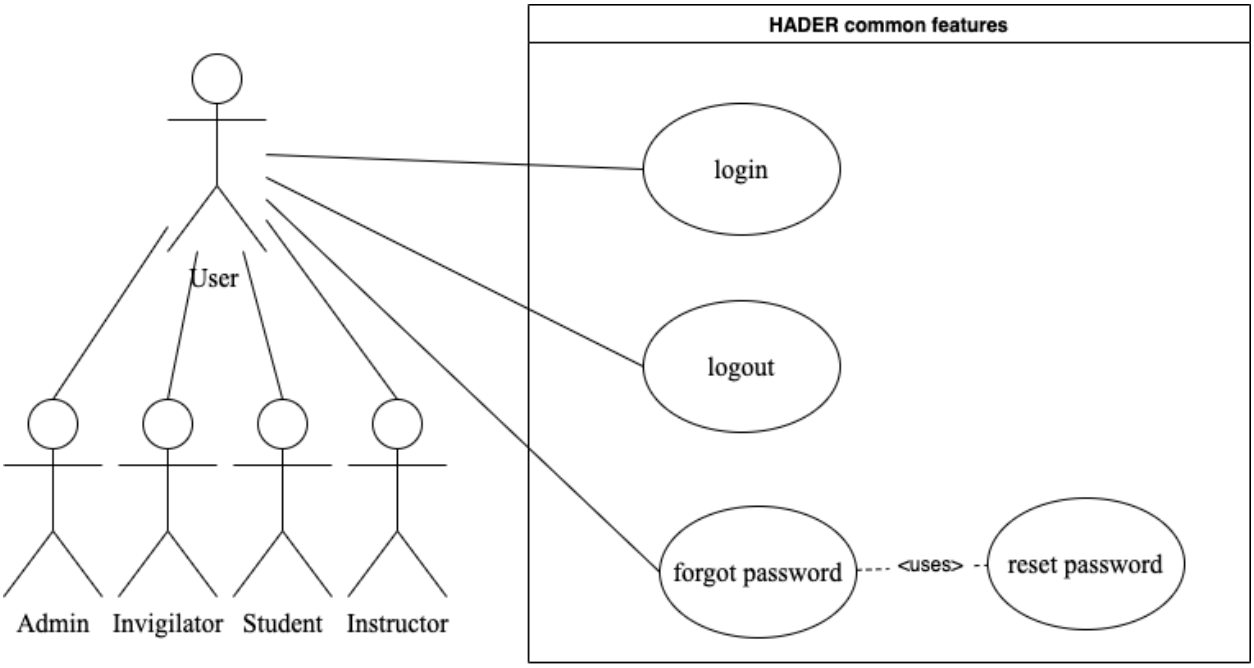


Figure 51: Common Features

4. Other Nonfunctional Requirements

4.1 Performance Requirements

To make the application simple and ready to use, the response time must be quick, which is a critical requirement in this system, as speed is critical due to the time constraints imposed by the exams. Another critical factor is the workload; the system must operate synchronously and without errors, as well as reconcile all requesting users, including the student, the invigilator, and the administrator. Finally, the QR code must be scanned in a period not exceeding thirty seconds.

4.2 Safety Requirements

The system must be updated continuously to address any malfunctions or damage and to avoid data loss, and users must also be able to report any problems encountered so that appropriate measures can be implemented.

4.3 Security Requirements

Because the primary concern of users is the possibility of personal fraud, this application has been secured by providing each student and invigilator with each own unique code accompanied by a photograph of each user's face to ensure integrity. All photos and information submitted by users will be safeguarded and encrypted in the database.

4.4 Software Quality Attributes

- **Availability:** the application should be freely downloadable by anyone.
- **Usability:** the application should be simple to use, efficient, intuitive, and simple to comprehend.
- **Reliability:** the application should accurately store and retrieves the data provided by the user.
- **Portability**: the application will be developed in such a way that it can be used on both Android and iOS devices.
- **Maintainability:** the application should be adaptable to modifications such as new business rules or the correction of previous errors.

4.5 Business Rules

As previously stated, the ADER application will be used by a variety of users. It is critical to note that administrators, invigorators, and students each have their own set of features and interfaces. Some of them are common, such as sign-in and log out. And when a user forgets their password and attempts to recover it, the application will send a recovery link via email. The admin is the superuser and thus has access to functions that other system users do not have. This includes the ability to add and edit exams with their associated details. Additionally, only the administrator has the authority to accept or decline student absence excuses.

CHAPTER 4. DESIGN SPECIFICATIONS

1. Introduction

The System Design Specification (SDS) document covers in-depth the system design and architecture of the system.

1.1 Purpose

The purpose of the software design specification (SDS) document is to describe the HADER examination attendance system design and architecture in precise detail. This document demonstrates how the system's design can be communicated and understood by giving a diversity of viewpoints.

1.2 Scope

Software design specifications (SDS) documents describe how the system will meet the functional and non-functional requirements specified in the software specification (SRS). Architecture, functionality, interfaces, and database design will be described using sequence diagrams and class diagrams, and user-interface diagrams in this work.

1.3 Definitions, Acronyms, and Abbreviations

Table 16: Definitions, Acronyms, and Abbreviations

Abbreviations	Term
SRS	Software requirement specification
SDS	Software design specification
EER	Extended Entity Relationship Diagram
DB	Database
CCSIT	College of computer science and information technology
UI	User interface

2. System overview

HADER will automate the process of registering students' attendance in the examination hall by using a QR code based mobile application. Students, invigilators, instructors, and the Academic Affairs Department at CCIST will all benefit from the system's features by managing exams and/or excuses.

3. Design Considerations

The section describes the constraints on the design system that havehas an influence on HADER.

3.1 Assumptions and Dependencies

3.1.1 Related software or hardware

- HADER team members aim to make the mobile application reliable and don’t encounter problems during operation or maintenance.
- HADER team members also assume the application will work properly if connected to the internet.

3.1.2 Operating systems

HADER exam attendance system was developed using Flutter along with dart programming language, and can run across different platforms or operating environments, so HADER team members took into consideration making the mobile Application work on Android and iOS operating systems.

3.1.3 End-user characteristics

Four types of users will be using the HADER application Admin, instructors, Invigilators, and Students the application has different characteristics based on the role of the user such as:

- Admin: is responsible for managing exams, managing student's excuses, view students' reports.
- Instructors: are responsible for managing exams, viewing their section's exam attendance reports, viewing admin's exam approvals/declination, switchswitching to invigilator mode to take studentsstudents' attendance by scanning their QR Code, share attendance reports to any party in CCSIT via email,
- Invigilators: take students' attendance by scanning their QR Code, sand hare attendance reports to any party in CCSIT via email.
- Students: upload/view exam excuses status. Check-in to exams via QR code.

3.2 General Constraints

HADER team members are designing a mobile application that is user-friendly with easy functions and the user can interact with the mobile application from the first attempt the constraints are:

- The user owns a smartphone.
- The application is for smartphones with iOS and Android Operating systems only.
- The users must be a member of CCSIT in order to sign in with their email and password.
- All user data entered must be stored and retrieved at any time.
- Unique user ID for all different users.
- The system supports the English language only.
- A stable internet connection is mandatory to interact with the application.

4. User Interface Design.

This section describes the Hader application's interfaces in detail and provides an overview of each interface. Additionally, it includes the interface design rules.

4.1 Overview of User Interfaces

➢ Common Interfaces

- Login: The user logs in by entering the email and password.
- Forgotten password: If the password is forgotten, the user must enter a phone number to send the Digit code.
- Enter Digit Code: The user enters the digit numbers that were sent via messages, then confirm.
- Change the password: The user enters the new password and types it again to confirm, then update.
- Profile: Shows the user's information, name, profile picture, and QR code.

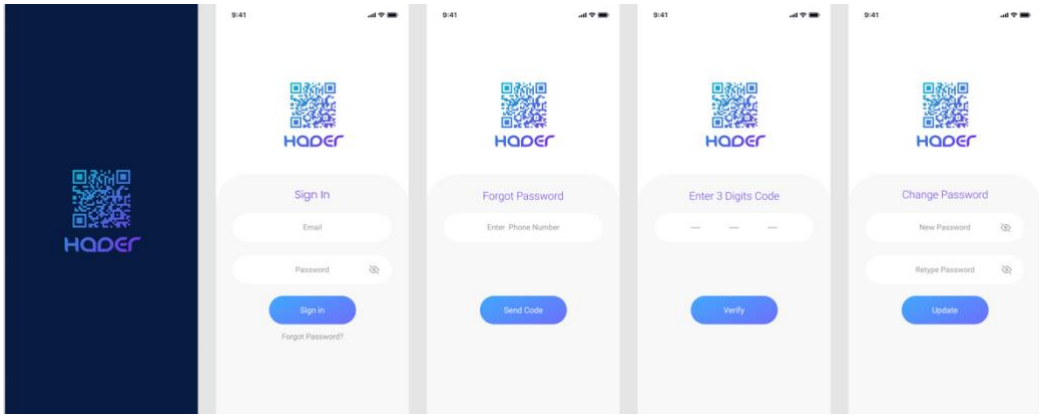


Figure 52 Login, forgotten password, Enter Digit Code, Change the password interfaces

Figure 53 Profile interfaces

➢ Admin Interface:

- Administration: The admin chooses to manage the invigilators or manage the students.
- Manage invigilators: a plus sign appears at the top of the list to add a new invigilator to the exam, as well as a list of the names of the invigilators who have been added previously.
- Adding a new Exam: When the plus sign is pressed, this page appears, and the exam information is entered, such as the name of the invigilator, room number, course ID, date, and time.
- Manage students: A list of absent students, their percentage of absence, and an icon to view excuses for absence appear.
- Absence Excuse Management: The admin looks at the student's excuse and determines the status of the excuse, whether it is accepted, rejected, or suspended, and then the excuse is sent to the designated user.
- Dashboard: There are general statistics on the number of absent and present students and absenteeism rates, the admin can filter the data by gender.

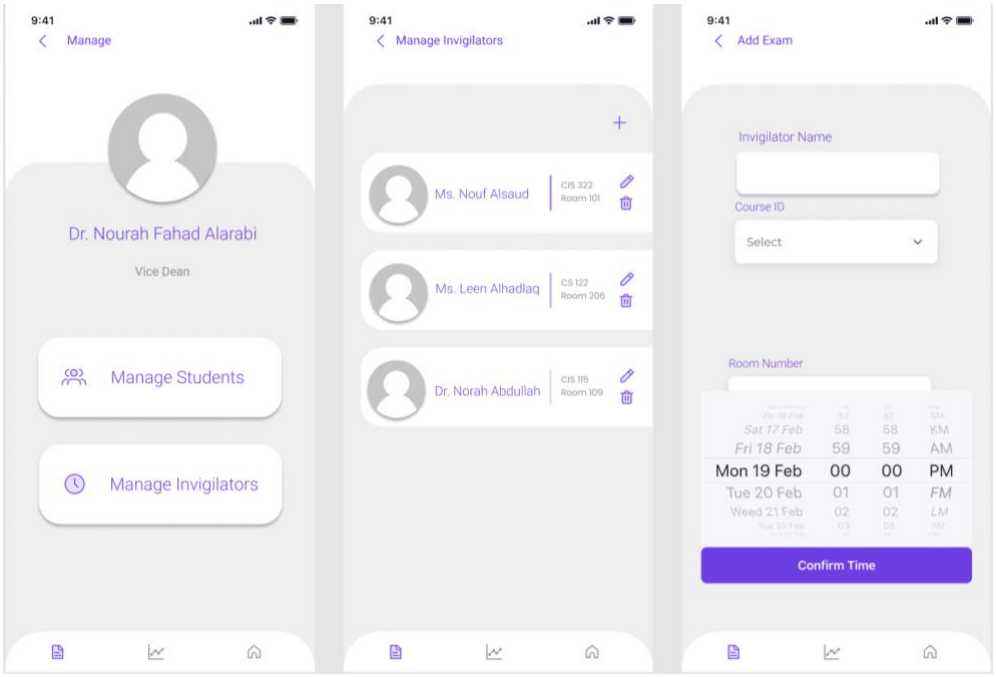


Figure 54 manages invigilators, adding a new exam interfaces

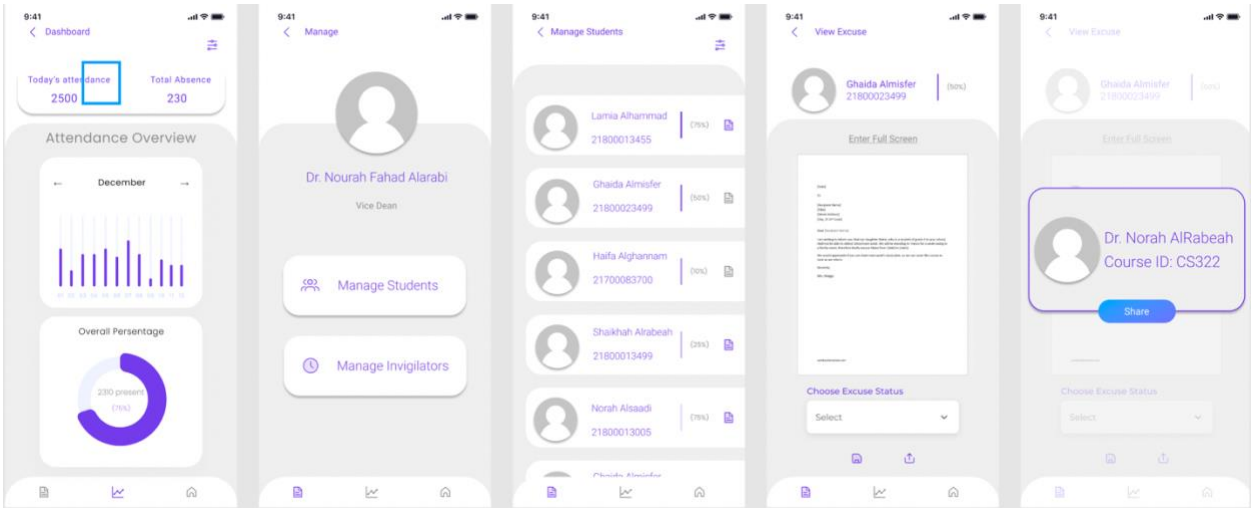


Figure 55 manage student, Absence Excuse Management, Dashboard interfaces

➢ Student interface:

- Excuses Menu: The student has the option to raise an excuse or see the case of an excuse that was previously submitted.
- Submitting an excuse: The student enters the excuse information, such as the reason, type of absence, the date, and a photo or file of the excuse.
- View Excuse Status: The status of the request for an excuse for absence is shown here.

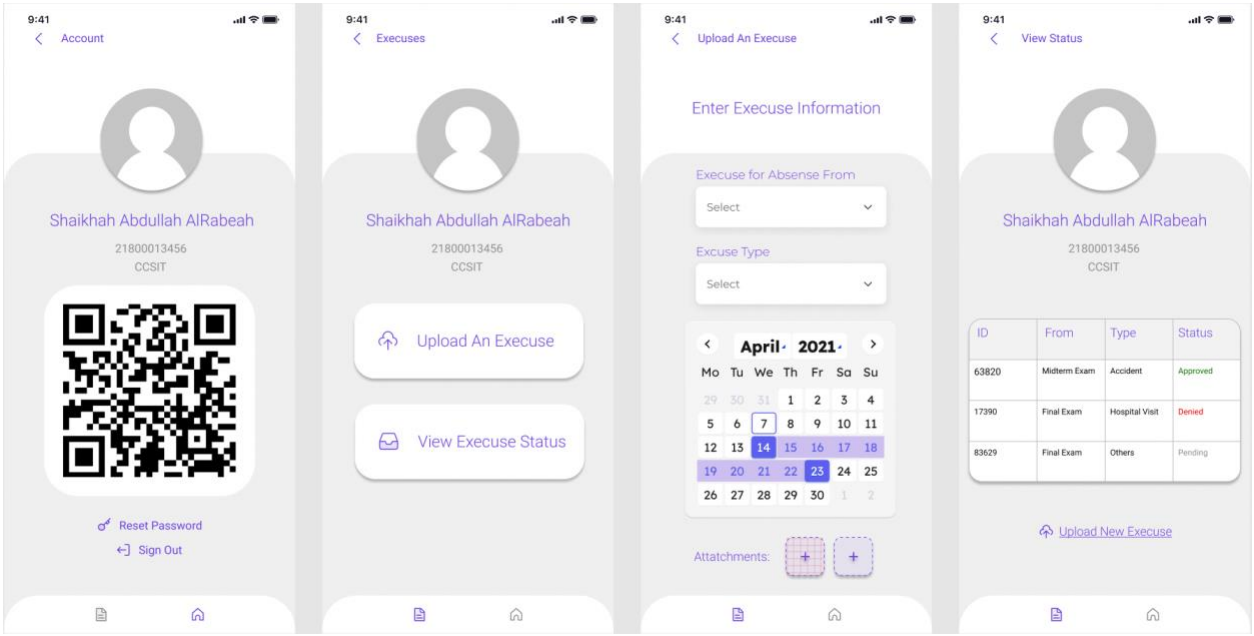


Figure 56 Excuses Menu, Submitting an excuse, View Excuse Status interfaces

➢ Invigilator interface:

- Manage students: this interface allows the admin to view exam excuses and (approve/decline) it will also allow the admin to share any student's excuses via E-mail to any party in CCSIT.
- Notifications: The Instructor is notified after being assigned as an invigilator and can switch to the invigilator interface when the three points are pressed. Also, upon receiving a list of students' names, the instructor is notified, and the list is uploaded via the three dots.
- Monitoring schedule: The Invigilator chooses the date of the day on which the monitoring will take place.
- List of exams: When choosing the date, a list of exams appears with the room number.
- Today's exams: A list of students whose attendance will be taken in the exam appears and a button to scan the code.
- Scan a QR code: When the "Scan Code" button is pressed, the QR code scanning page appears, and the student's attendance is taken by passing the mobile phone over the QR code of the student.
- Error message: An error message appears when the QR code cannot be identified.
- Success message: When you save the list after the change, a success message appears.
- Student identification: Student information appears as his/her name, university number, and photo for confirmation.

- Share list: The attendance list for the subject instructor is shared.

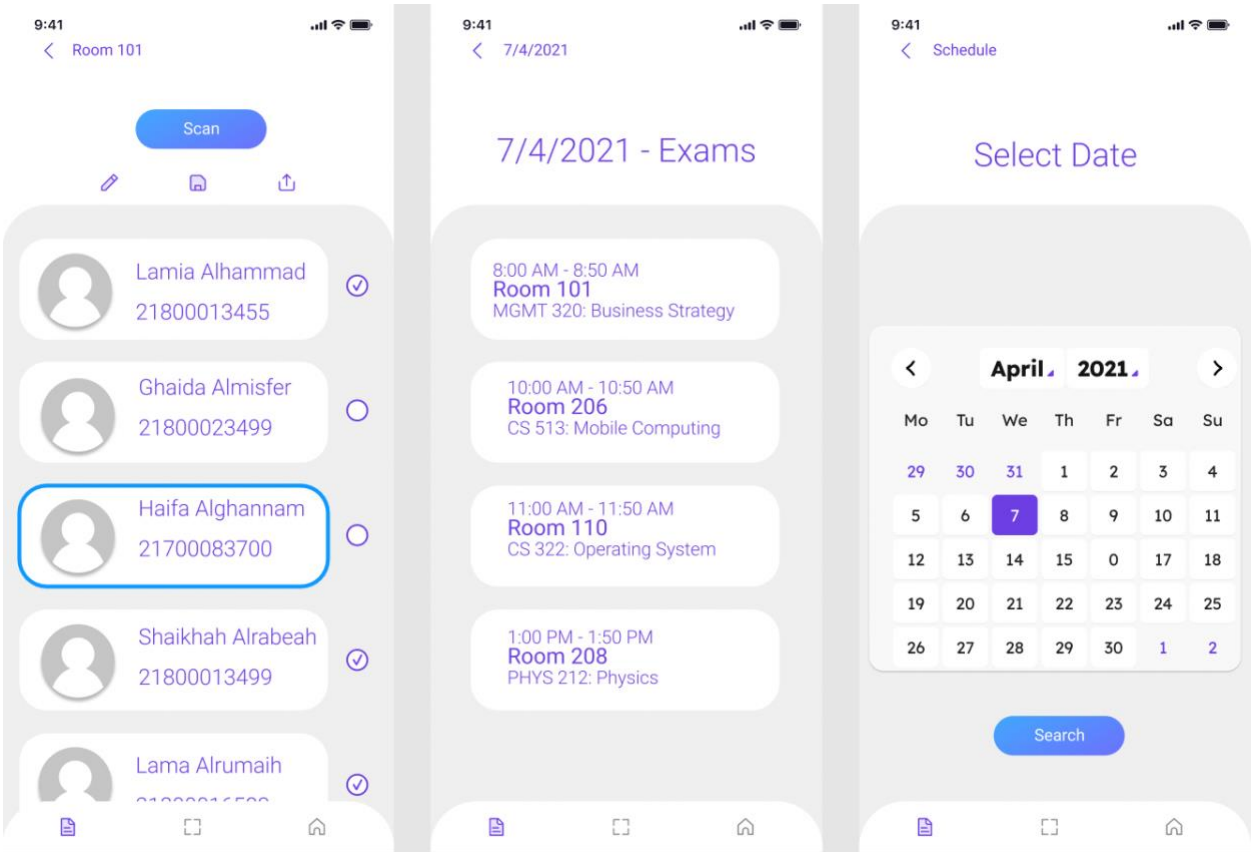


Figure 57 Monitoring schedule, List of exams, Today's exam interfaces

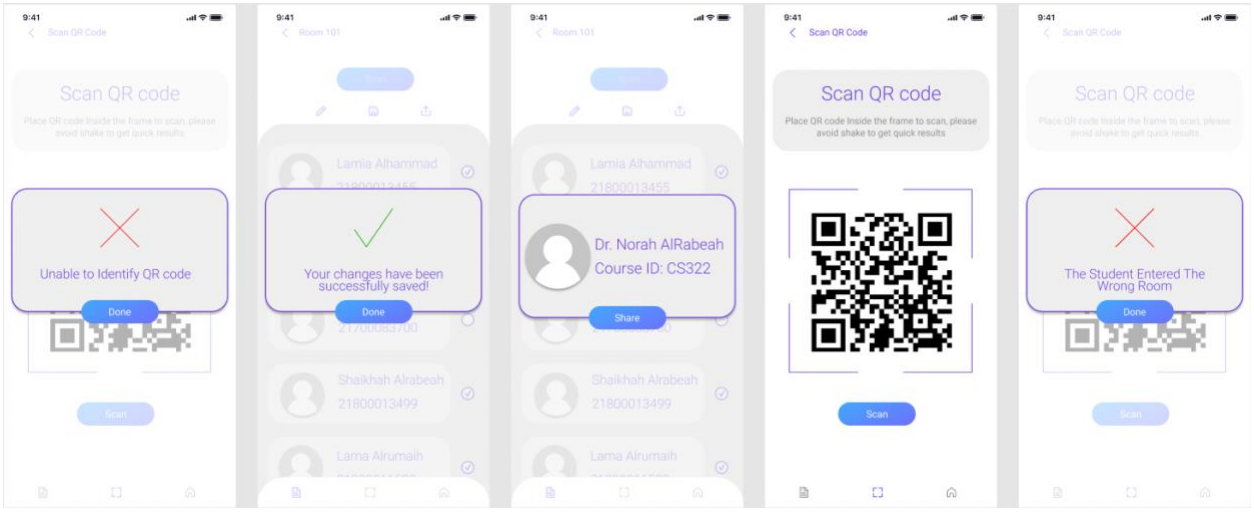


Figure 58 Scan a QR code, Error message, Success message, Student identification, Share list interfaces

➢ Instructor interface:

- Notifications: The Instructor is notified when appointed as an invigilator and can switch to the invigilator interface when the menu icons are pressed. Also, upon receiving a list of students' names, the instructor is notified, and the list is uploaded via the three dots.

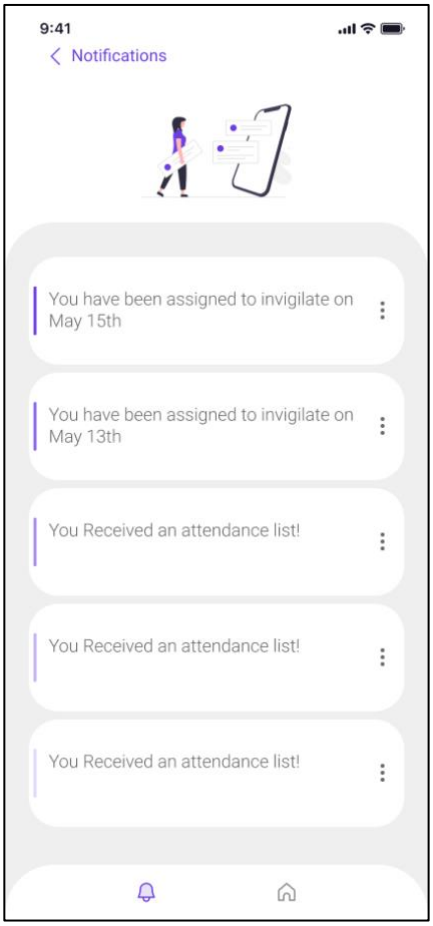


Figure 59 Notification interface

4.2 Interface design rules

- Make Navigation Simple: Simple navigation offers an obvious method of navigating between screens and locating desired items.
- Make a big touch area: Ensure that control sizes are easy to tap with the thumb. Smaller controls will irritate the user while using the application.
- Eliminate Clutter: It is not important to display all available data on the user interface. Organize the content in such a way that the user understands the available features. Minimize clutter and keep the screen's content to a minimum.
- Consistency: The colors, language icons/pictures, font format, and layout of all interfaces on the application must be consistent.

4.3 Screen objects and actions

This section describes the interface's objects and the actions that will occur when the user interacts with them.

4.3.1 Admin

Table 17 Admin - Screen objects and actions

Object	Type	Action
Profile Page		
Reset password	Icon/Link	Takes the user to reset password page.
Log out	Icon/Link	Takes the user to the sign in page.
Dashboard Page		
Attendance overview	Left/Right Arrows	Navigates between months.
Filter	Icon	Filters the data by gender.
Management Page		
Manage students	Button	Takes the user to absent students list page.
Manage invigilator	Button	Takes the user to manage invigilator page.
Absent Students List Page		
Filter	Icon	Filters the list by absence percentage or date.
View Excuse	Icon	Takes the user to view excuse page.
View Excuse Page		
Choose excuse status	Dropdown list	A dropdown list appears, and the user selects the proper excuse status.
Save	Icon	Success message appears.
Share	Icon	Shares excuse status to designated instructor.
Manage Invigilator Page		
Plus	Icon	Takes the user to add exam page.
Edit	Icon	Edits entered information.
Delete	Icon	Deletes invigilator.
Add Exam page		
Invigilator name	Textbox	The user writes the invigilator’s name for exam.
Course ID	Dropdown list	A dropdown list with courses id appears.
Room Number	Dropdown list	A dropdown list with courses room numbers appears.
Choose date and time	Date/Time picker	The user chooses date and time for the new exam.

4.3.2 Student

Table 18 Student - Screen objects and actions

Object	Type	Action
Profile Page		
Reset password	Icon/Link	Takes the user to reset password page.
Log out	Icon/Link	Takes the user to the sign in page.
Manage Excuses Page		
Upload an excuse	Button	Takes the user to upload excuse page.
View excuse status	Button	Takes the user to view excuse status page.
Upload Excuse Page		
Excuse for absence from	Dropdown list	A dropdown list with absent from midterm/final exam appears.
Excuse type	Dropdown list	A dropdown list with absence reasonings appears.
Enter date from/To	Date picker	The user selects the absence from/to dates.
Add attachments	Icon	The user adds the excuse attachment.
View Excuse Status Page		
Upload new excuse	Icon/Link	Takes the user to upload excuse page.

4.3.3 Invigilator

Table 19 Invigilator - Screen objects and actions

Object	Type	Action
Profile Page		
Reset password	Icon/Link	Takes the user to reset password page.
Log out	Icon/Link	Takes the user to the sign in page.
Monitoring Schedule Page		
Search by date	Date picker	The user chooses a day to view its exam.
Search	Button	Takes the user to the chosen date.
Today's List Page		

Exam details	Button	Takes the user to students list page.
Students List Page		
Scan	Button	Takes the user to scan page.
Edit	Icon	Enables the user to edit student's attendance status.
Save	Icon	Success message appears.
Share	Icon	Shares students list to the designated instructor.
Radio button	Radio button	Changes student's attendance status, present or absent.
Scan Page		
Scan	Buttons	Scans the student's QR code then shows scanned student information.

4.3.4 Instructor

Table 20 Instructor - Screen objects and actions

Object	Type	Action
Profile Page		
Reset password	Icon/Link	Takes the user to reset password page.
Log out	Icon/Link	Takes the user to the sign in page.
Notifications Page		
Menu	Icon	A menu with delete, download list, switch to invigilator mode appears.

5. System Architecture

This section indicates HADER architecture. It covers the architectural design approach and Architectural Design for the overall system

5. Architectural Design Approach

The suggested HADER application employs a multilayered architecture (N-tier). Three fundamental levels comprise the architecture: the presentation layer, the web system layer, and the data layer.

- **Presentation layer:** This is the top layer, which is responsible for managing the user's interaction with the application by presenting information and accepting relevant input from the user.
- **Web system layer:** This is the application's intermediate layer, which manages and executes all application functions. This layer will alter and process the user's inputs.

- **Data layer:** This layer gives access to the database within the limits of the application. By conducting queries, data may be saved and retrieved. The multilayered design aids in the application's maintenance, adaptability, and security. Security concerns would be concentrated in a single layer without a system layer, making them extremely difficult to manage and remediate.

5.1 Architectural Design

this section shows the architectural design of the application, as shown in the figure below a mobile phone can interact with all elements.

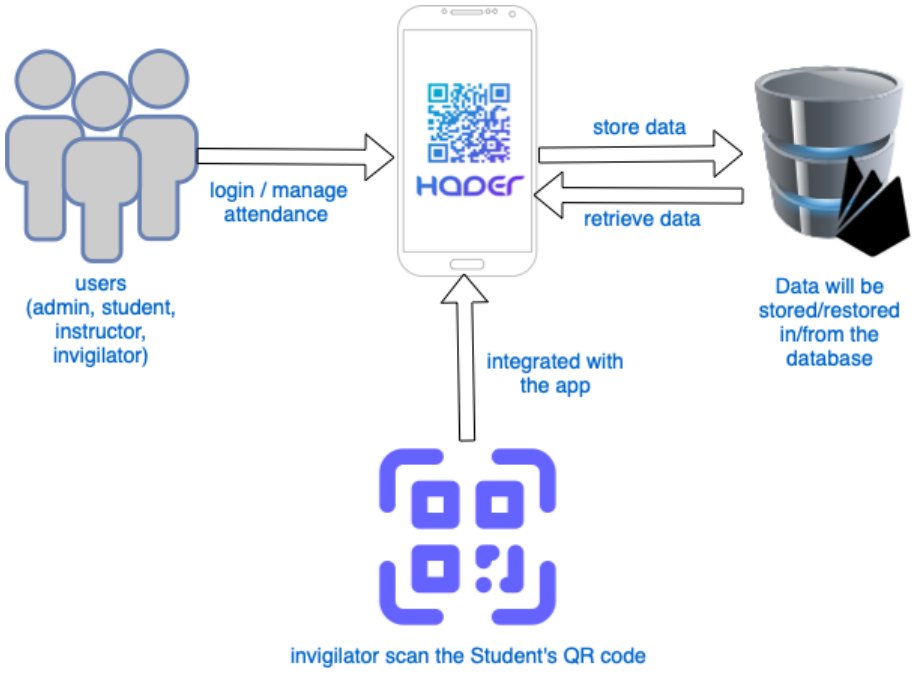


Figure 60 Architectural Design

5.2 Subsystem architectural design.

The context diagram of the HADER application shows the main process and data flow between the system and external entities as shown in the figure below.

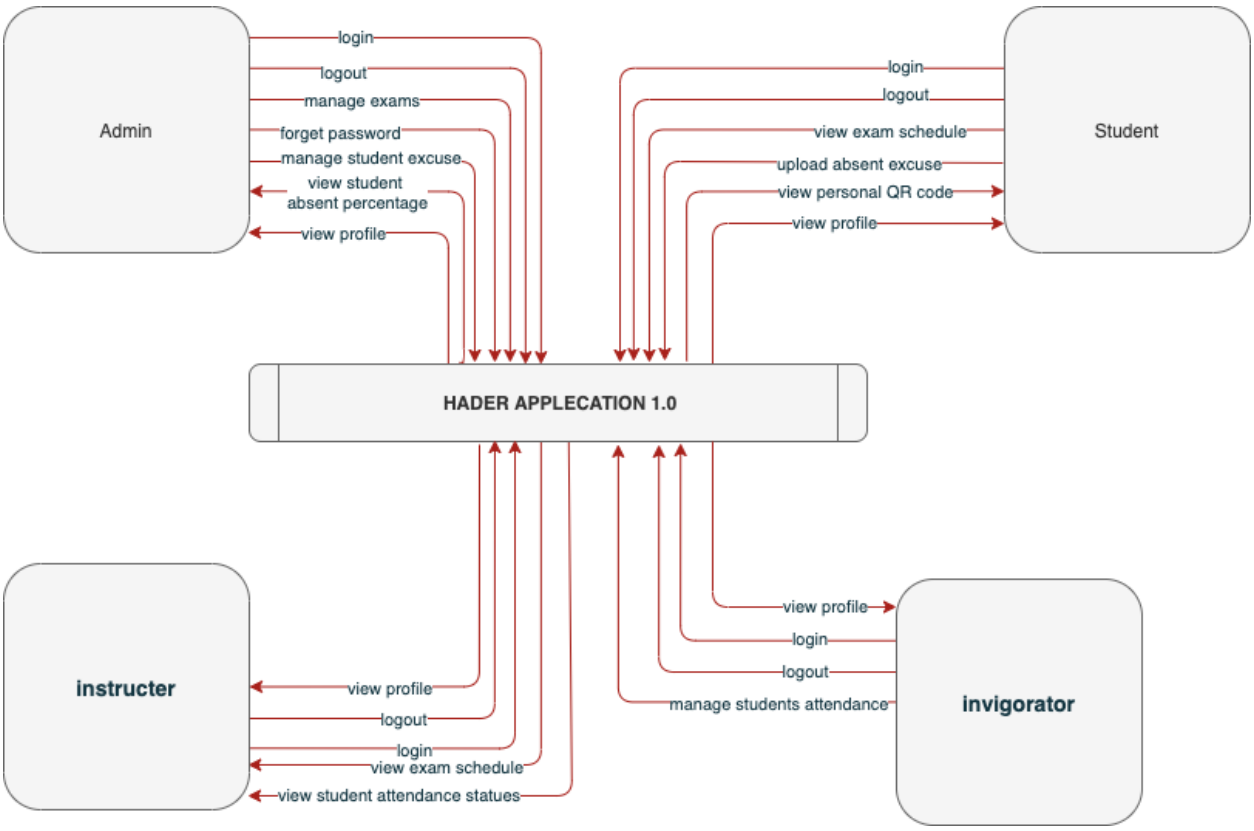


Figure 61 DFD context diagram of HADER application

6. Data Design

This section lists all the database designs including (data description, data dictionary and database description, Architectural Design, EER, ERD, and DFD).

6.1 Database Description

This section includes the description of the Entity-Relationship Diagram ER for the HADER application in addition to the Database Schema.

Table 21 Database Description

	Relationship Between Entities		Relationship Type	Description
1	**users**	**admin**	**one-to-one**	**Each admin has one user.**
2	**users**	**invigilator**	**one-to-one**	**Each invigilator has one user.**
3	**users**	**student**	**one-to-one**	**Each student has one user.**
4	**users**	**instructor**	**one-to-one**	**Each instructor has one user.**
5	**student**	**courses**	**many-to-many**	**many students have many courses**
6	**instructor**	**courses**	**many-to-many**	**many instructors have many courses**
7	**courses**	**Exam**	**many-to-many**	**many courses have multiple exams**
8	**admin**	**Exam**	**many-to-many**	**many admins manage several exams**
9	**attendance_sheet**	**Exam**	**one-to-many**	**each exam has one attendance sheet**
10	**Exam**	**invigilator**	**one-to-many**	**each exam has many invigilators**
11	**admin**	**S_Absent_Excuses**	**many-to-many**	**many admins manage Absent Excuses**
12	**student**	**S_Absent_Excuses**	**many-to-many**	**many students have multiple Absent Excuses**
13	**student**	**attendance_sheet**	**many-to-many**	**many students appear on multiple attendance sheet**

6.1.1 Extended Entity Relationship Diagram (EER)

The EER below for HADER contains 9 related entities as shown in the figures of the diagrams.

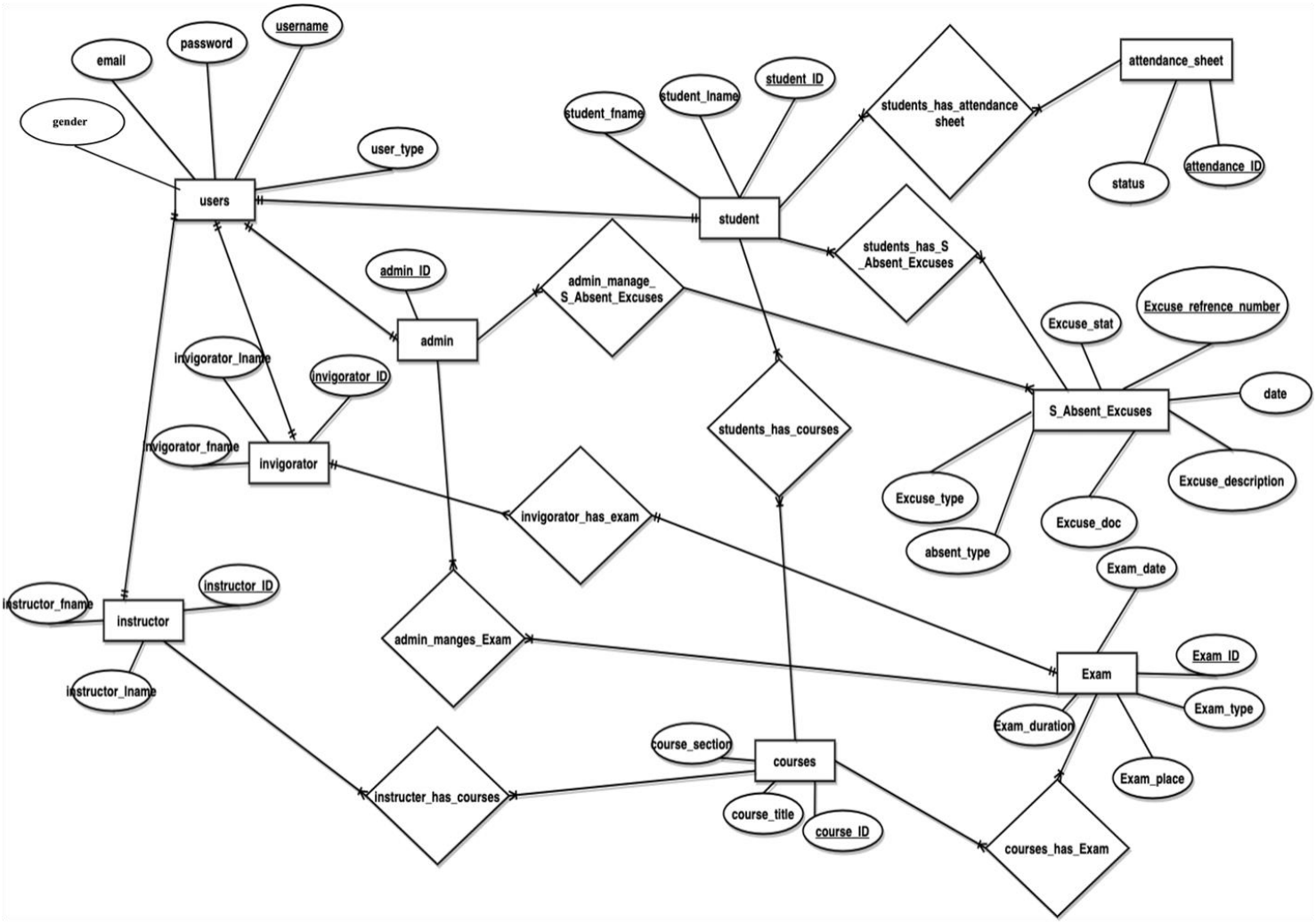


Figure 62 Extended Entity Relationship Diagram (EER)

6.1.2 Entity Relationship Diagram (ERD)

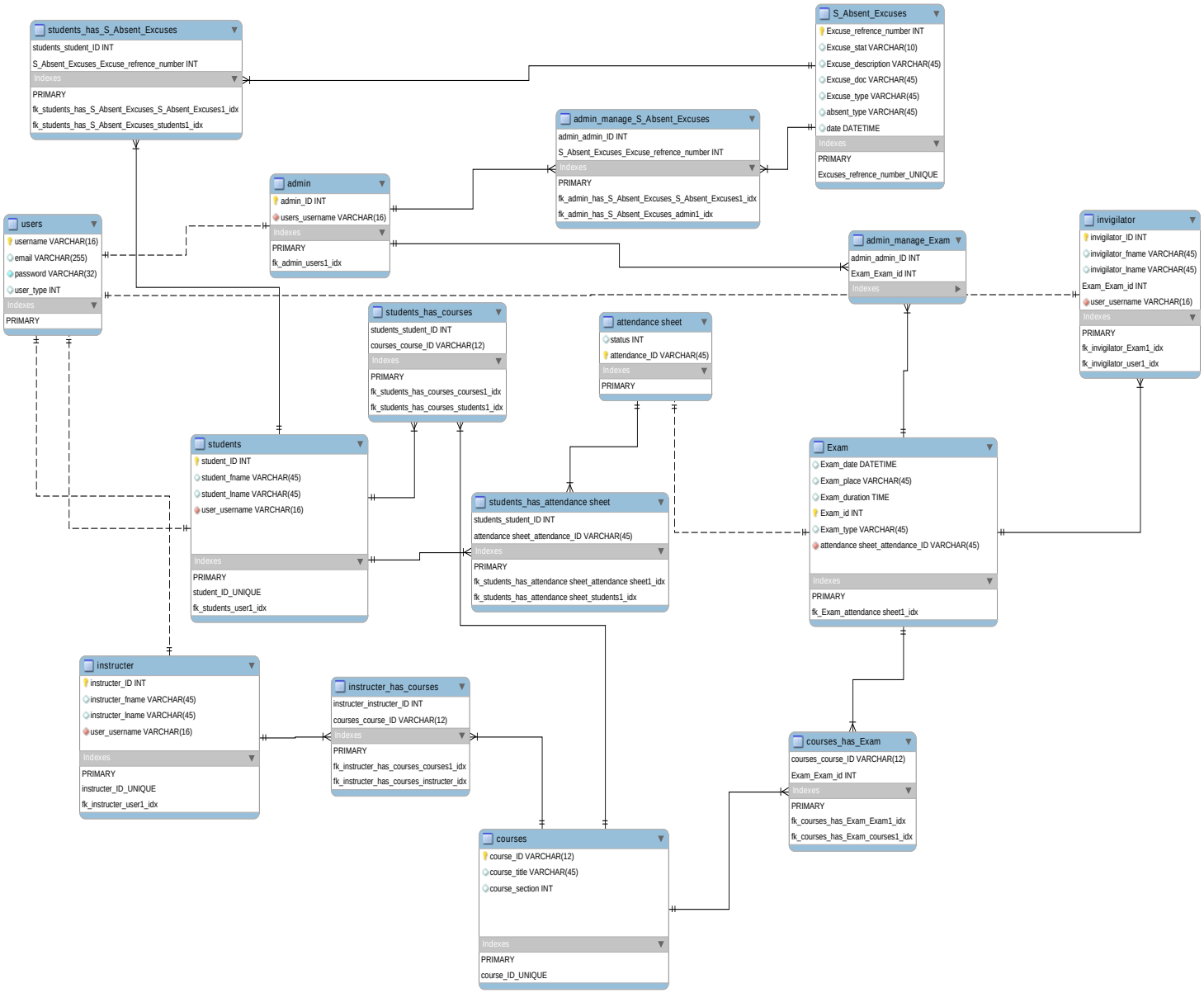


Figure 63 Entity Relationship Diagram (ERD)

6.1.3 Database Schema

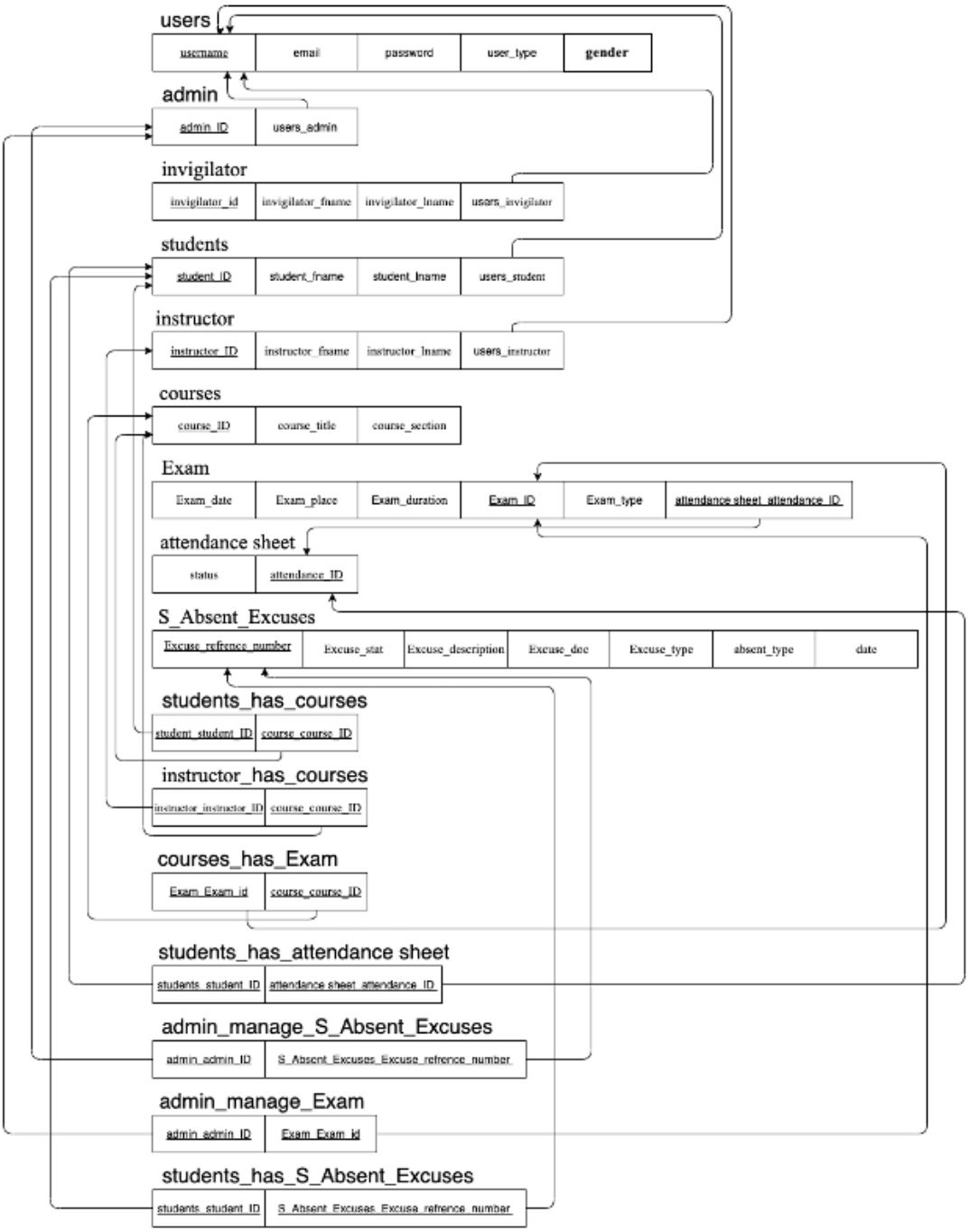


Figure 64 HADER Database Schema

6.2 Data Dictionary

Table 22 Data Dictionary

	Entity	Field	Data type	length	Primary key	Foreign key	Allow null	default
1	users	username	VARCHAR	16	*		-	-
		email	VARCHAR	255			-	-
		password	VARCHAR	32			-	-
		user_type	INT	-			-	-
		gender	Boolean	2			-	-
2	admin	admin_ID	INT	-	*		-	-
		users_admin	VARCHAR	16		*	-	-
3	invigilator	Invigilator_ID	INT	-	*		-	-
		invigilator_fname	VARCHAR	45			-	-
		invigilator_lname	VARCHAR	45			-	-
		users_invigilator	VARCHAR	16		*	-	-
		Exam_Exam_ID	INT			*	-	-
4	students	student_ID	INT	-	*		-	-
		student_fname	VARCHAR	45			-	-
		student_lname	VARCHAR	45			-	-
		users_student	VARCHAR	16		*	--	-
5	instructor	Instructor_ID	INT	*			-	-
		Instructor_fname	VARCHAR	45			-	-
		Instructor_lname	VARCHAR	45			-	-
		users_instructor	VARCHAR	16		*	-	-
6	courses	course_ID	VARCHAR	12			-	-
		course_title	VARCHAR	45			-	-
		course_section	INT	-			-	-
7	Exam	Exam_date	DATETIME				-	-
		Exam_place	VARCHAR	45			-	-
		Exam_duration	TIME	-			-	-
		Exam_ID	INT	-	*		-	-
		Exam_type	VARCHAR	45			-	-
		Attendance sheet_attendance_ID	VARCHAR	45		*	-	-
8	attendance sheet	status	INT	-			-	-
		attendance_ID	VARCHAR	45	*		-	-
9	S_Absent_Excuses	Excuse_refrence_number	INT	-	*		-	-
		Excuse_stat	VARCHAR	10			-	-
		Excuse_description	VARCHAR	45			-	-
		Excuse_doc	VARCHAR	45			-	-
		Excuse_type	VARCHAR	45			-	-
		absent_type	VARCHAR	45			-	-
		date	DATETIME	-			-	-

7.Detailed System Design:

7.1 Sequence Diagrams:

7.1.1 Common Features:

7.1.1.1 Log In / log out:

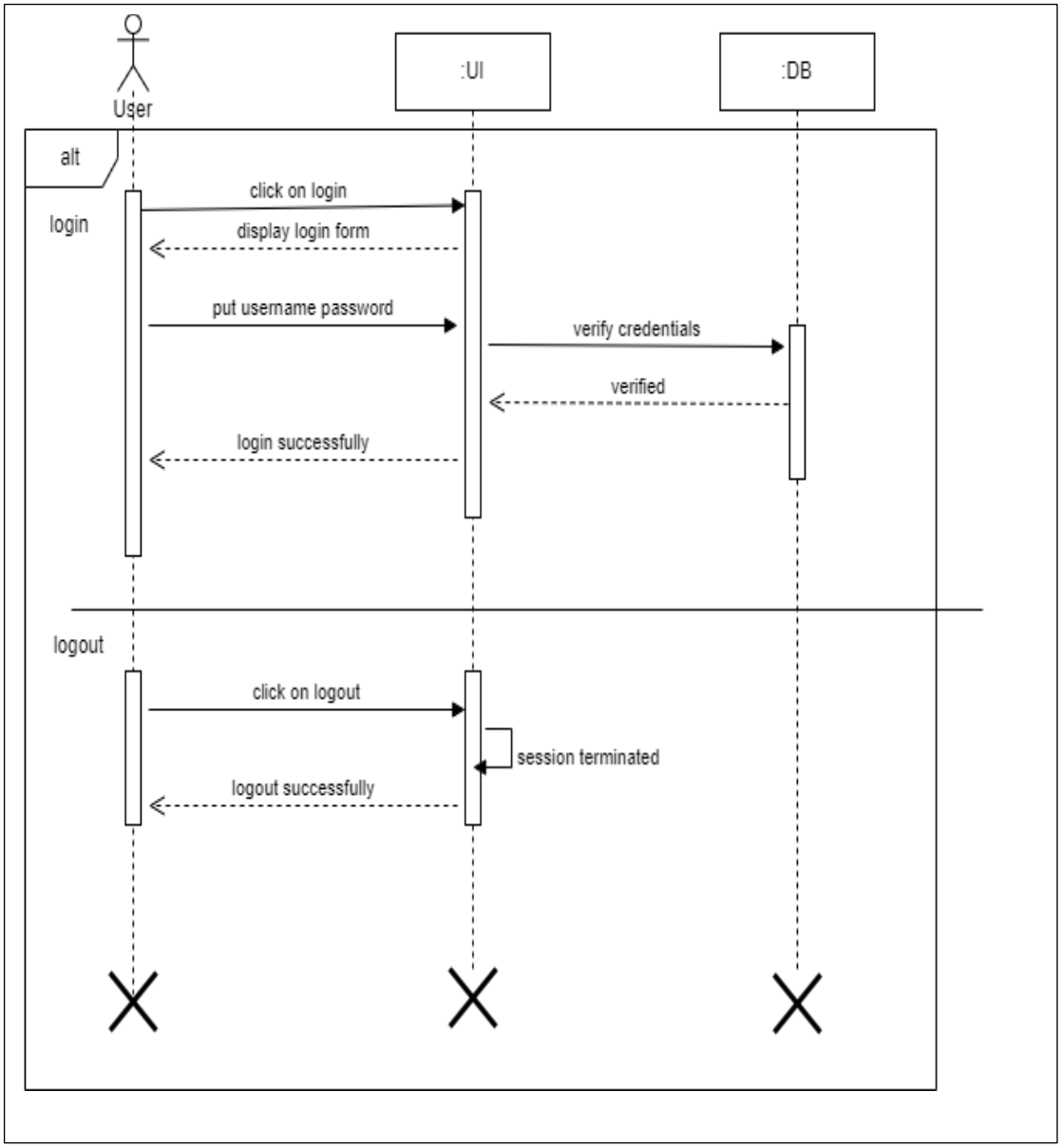


Figure 65 Log In and log out sequence diagram

The sequence diagram above shows how the system allows Each user to log in to the Hader system by providing their username and password and logout.

7.1.1.2 Forgot / reset password:

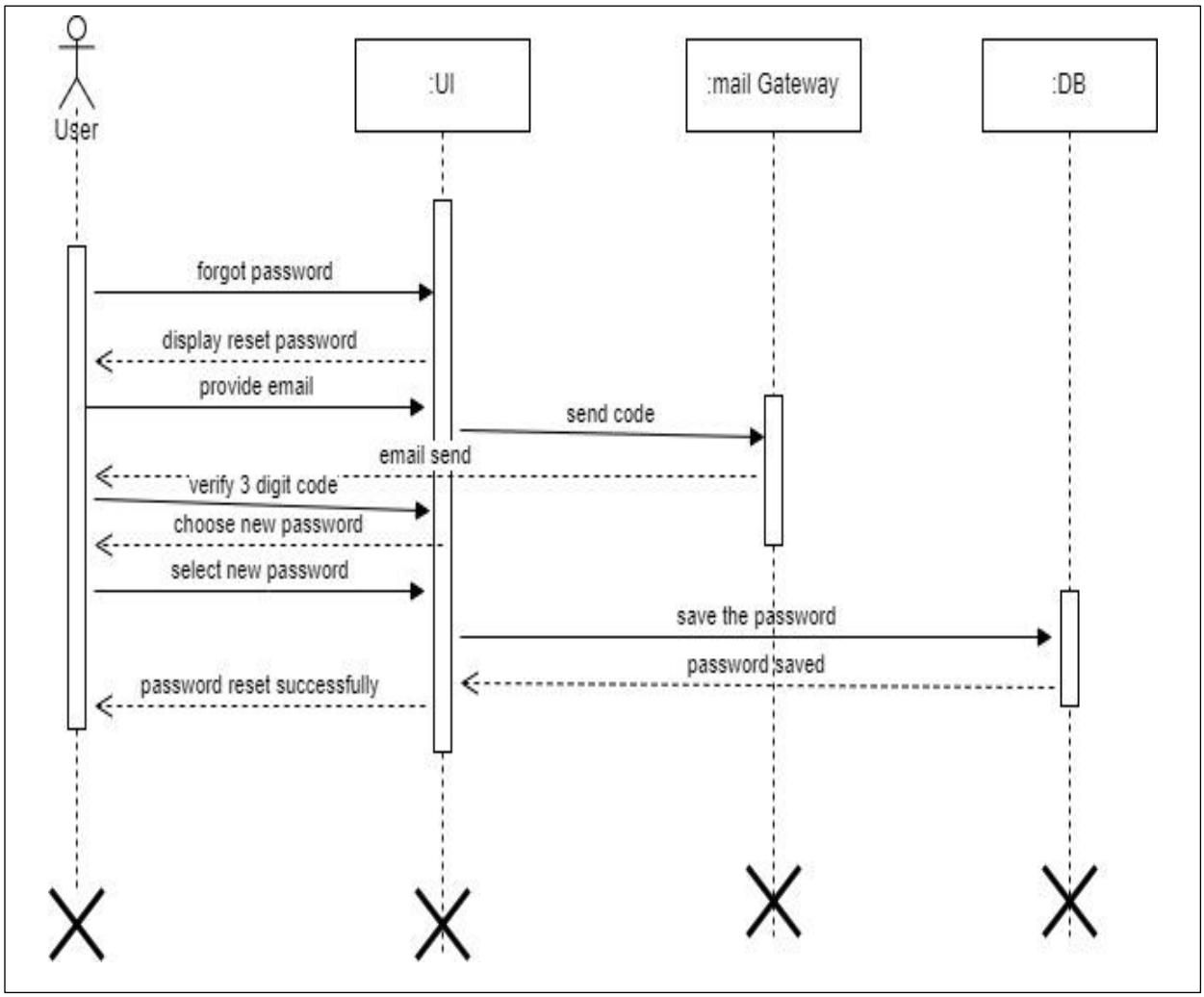


Figure 66 Forgot and reset password sequence diagram

The sequence diagram above shows how the system allows the users to retrieve their forgotten passwords and reset their passwords.

7.1.1.3 Scan QR Code for Students:

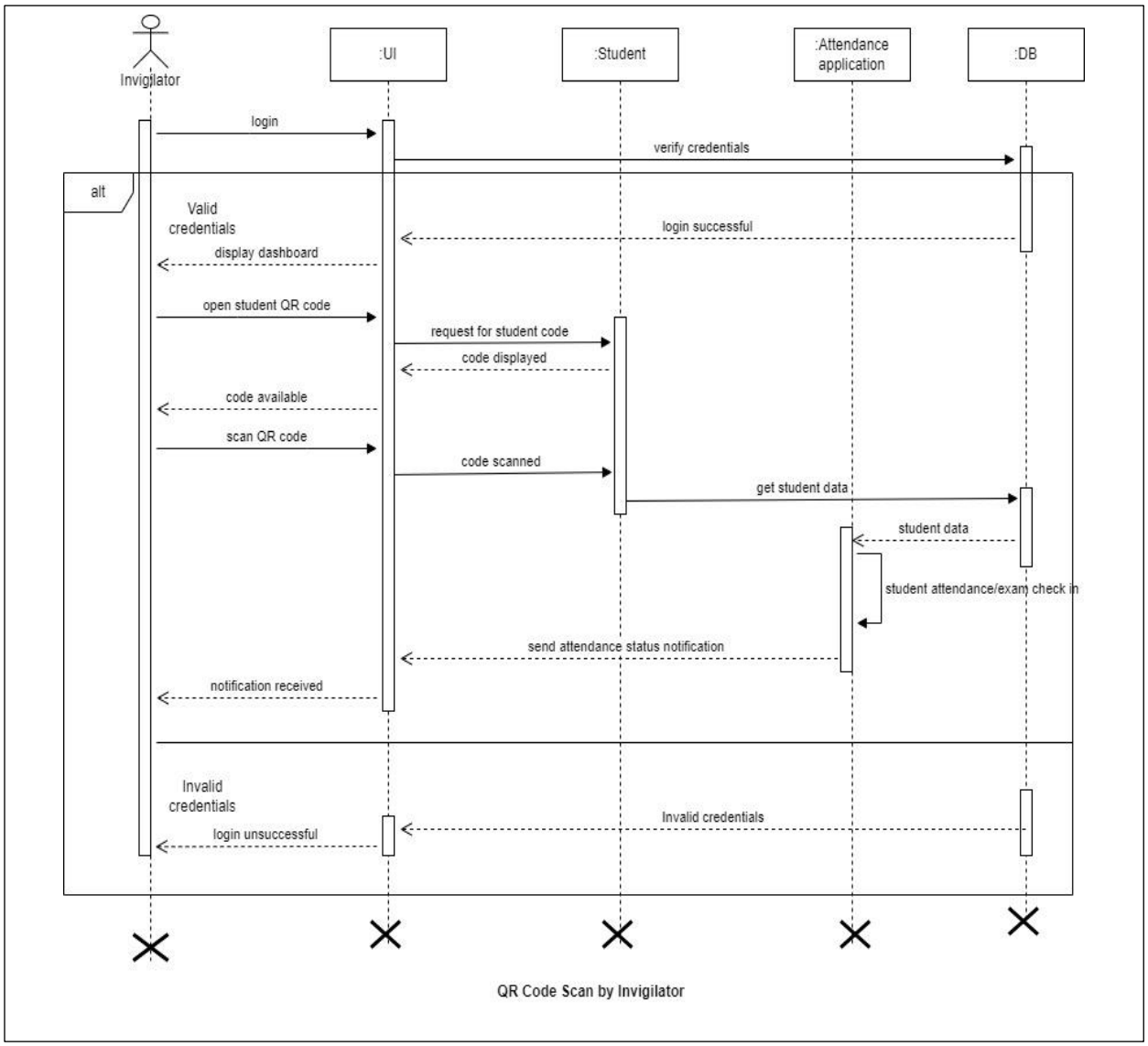


Figure 67 Scan QR code for student's sequence diagram

The sequence diagram shows how the Hader system scans the unique QR Code by the exam invigilator for students.

7.1.1.4 Show Student Information:

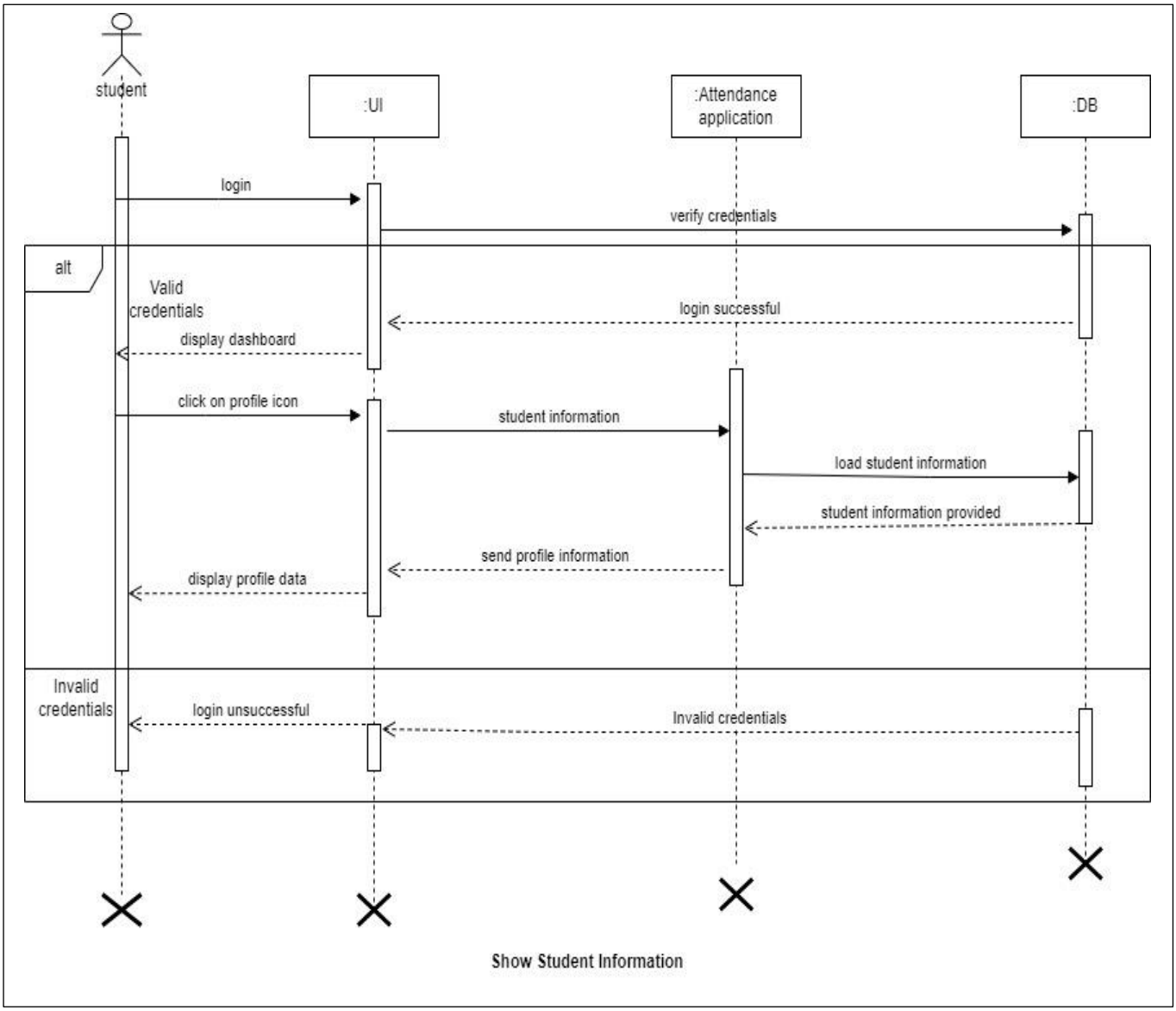


Figure 68 Show Student information sequence diagram

The sequence diagram shows how the Hader system displays some of the students' information.

7.1.1.5 Upload Exam Excuse:

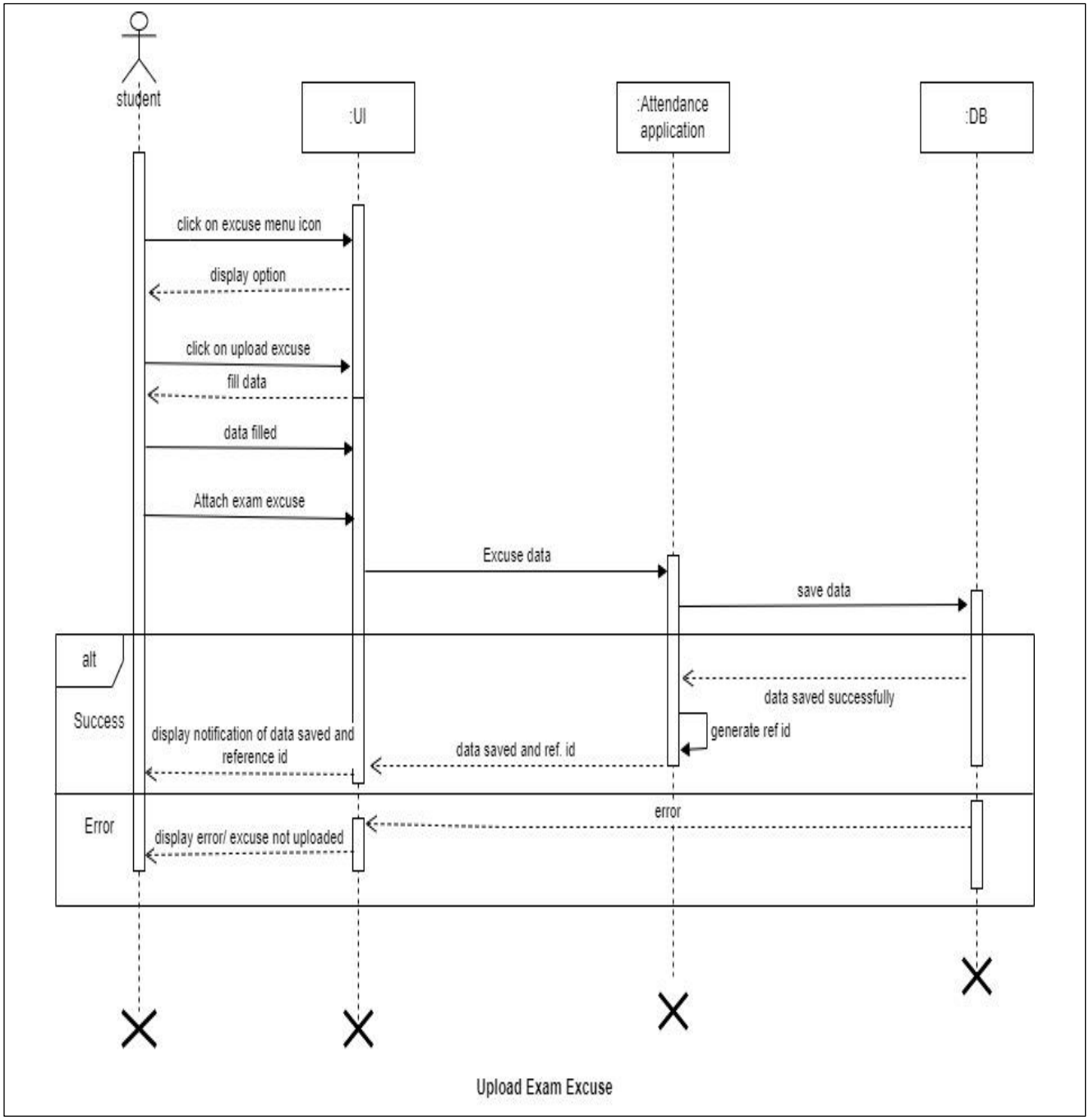


Figure 69 Upload exam excuse sequence diagram

The sequence diagram above shows how the Hader system enables students to upload their exam excuses.

7.1.1.6 View Excuse Status:

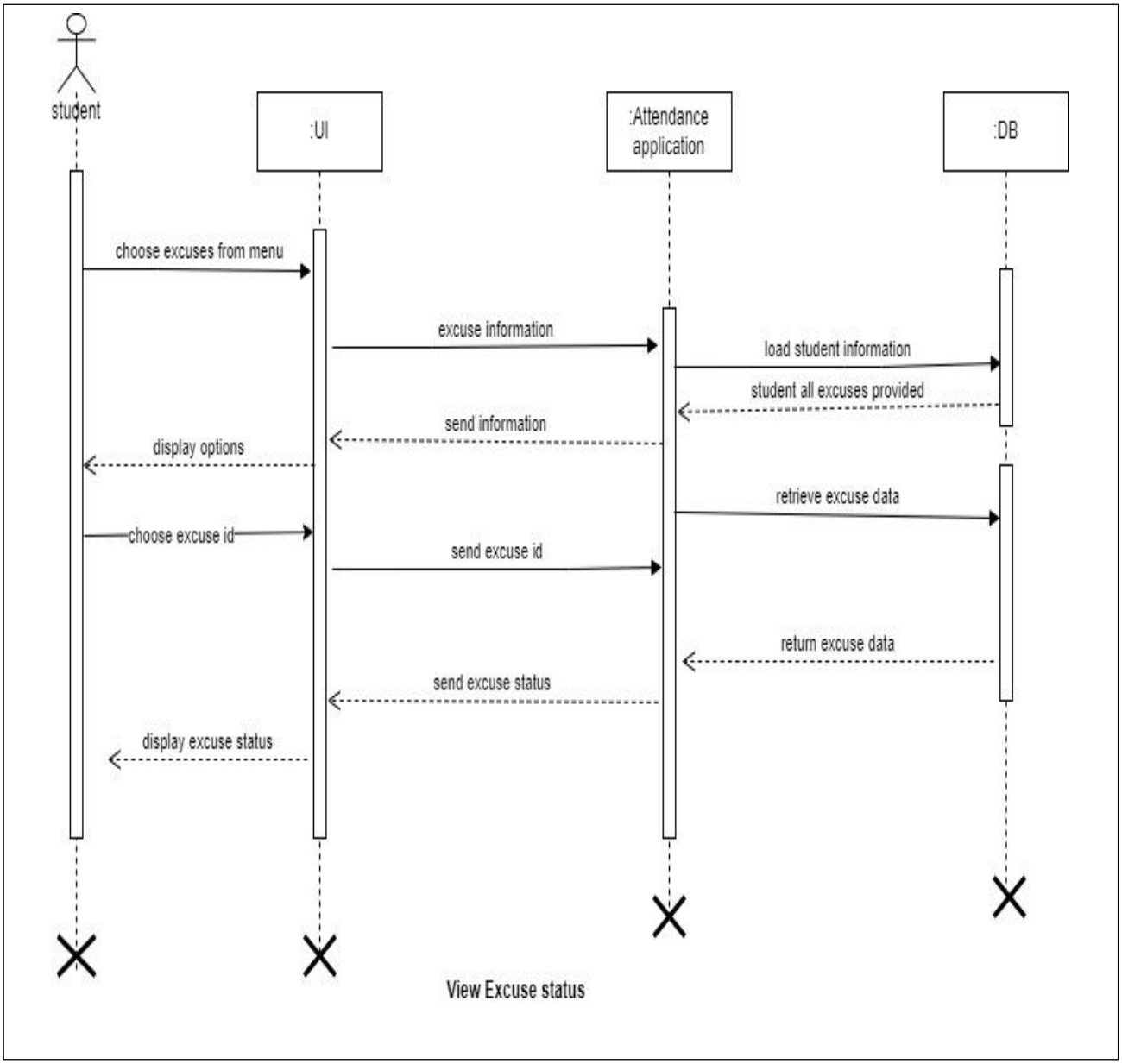


Figure 70 View excuse status sequence diagram

The sequence diagram above shows how the Hader system enables students to view their exam excuse status.

7.1.1.7 Manage the Student's Excuses:

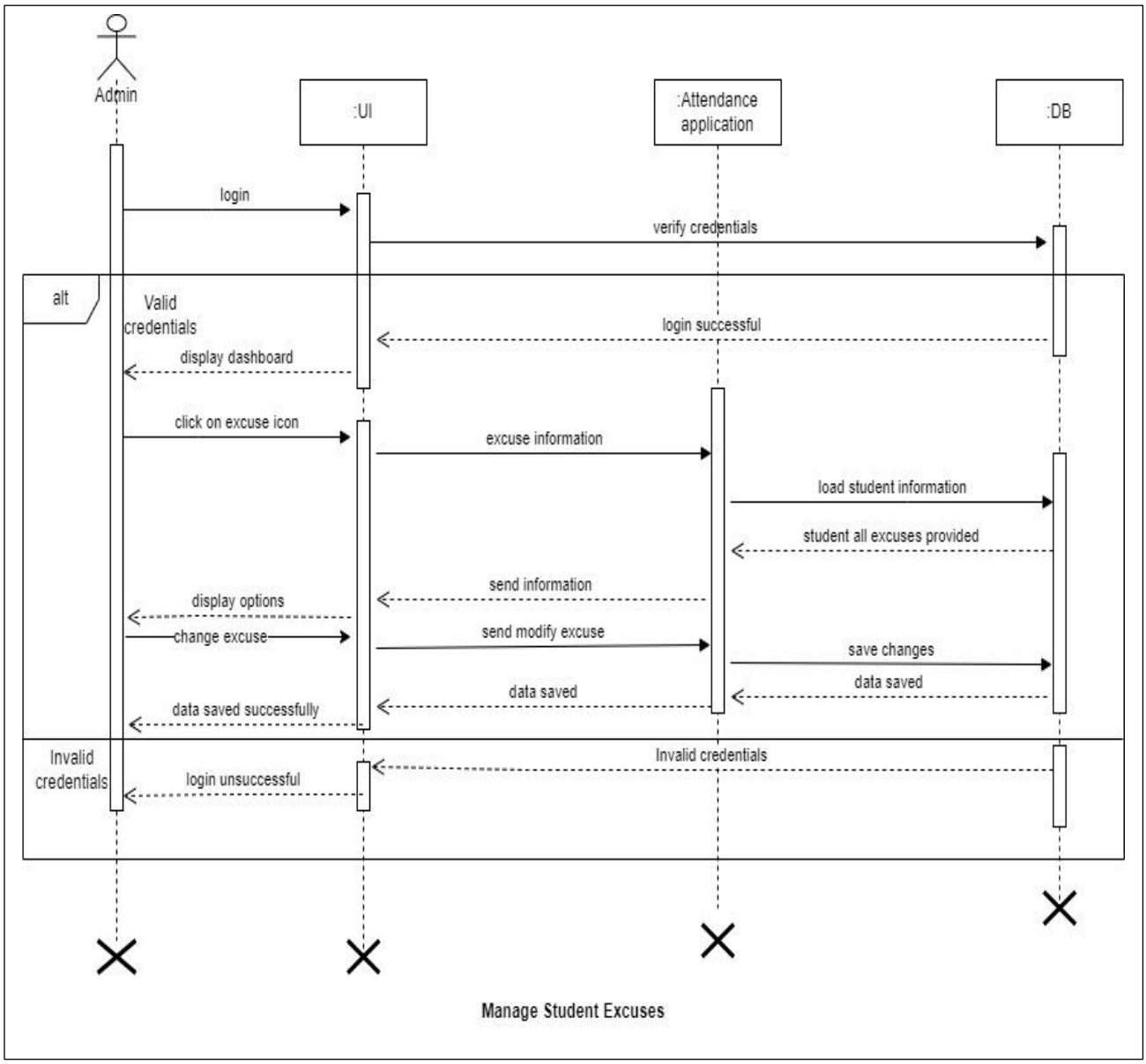


Figure 71 Manage the student's excuses sequence diagram

The sequence diagram above shows how the admin can view, change, and modify the excuse status.

4.5.1.8 Show the student's absence percentages for all courses:

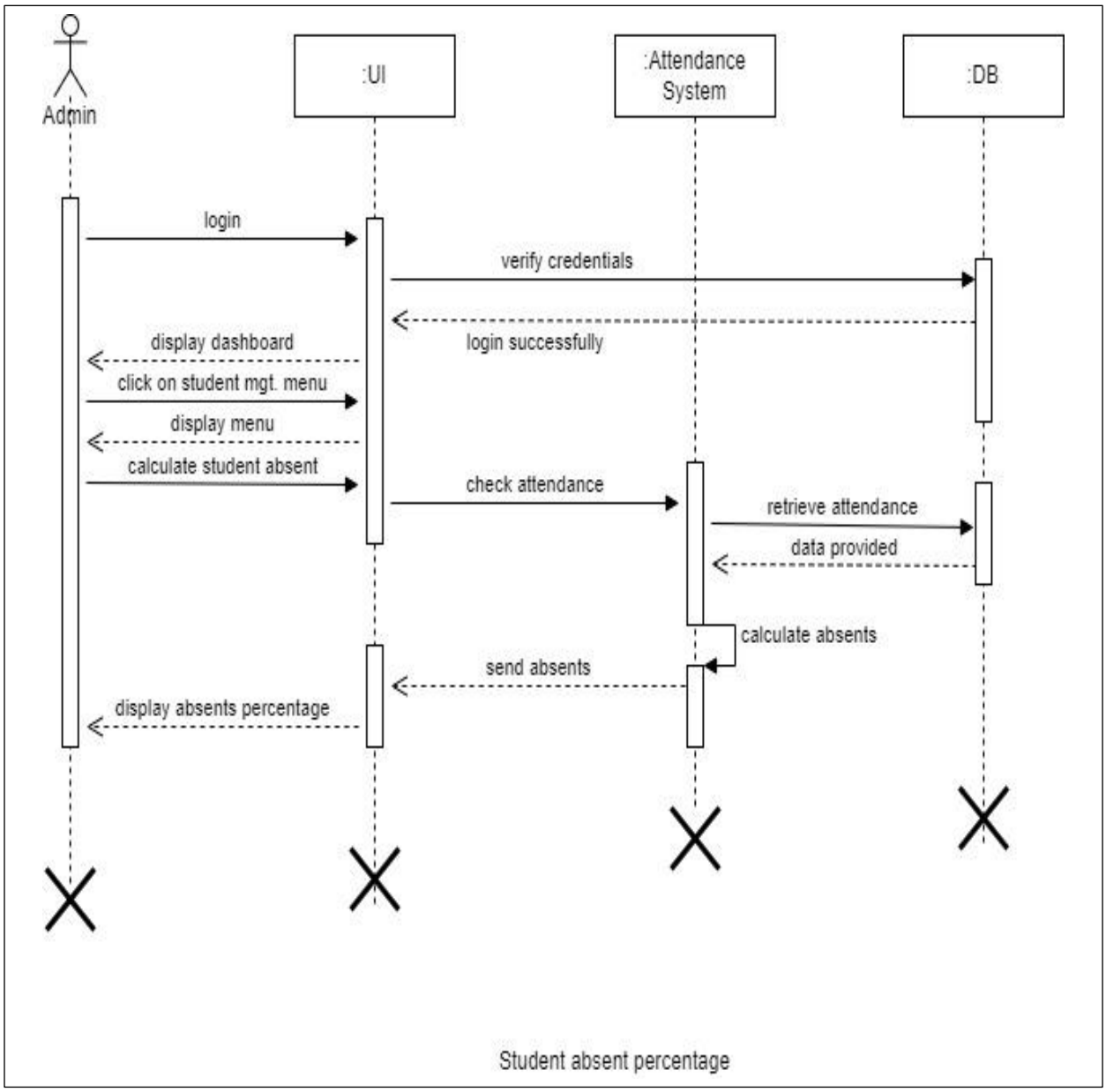


Figure 72 Show the student's absence percentages for all courses sequence diagram

This sequence diagram above shows how the admin can view absent percentages for each student.

7.1.1.9 Create an Exam:

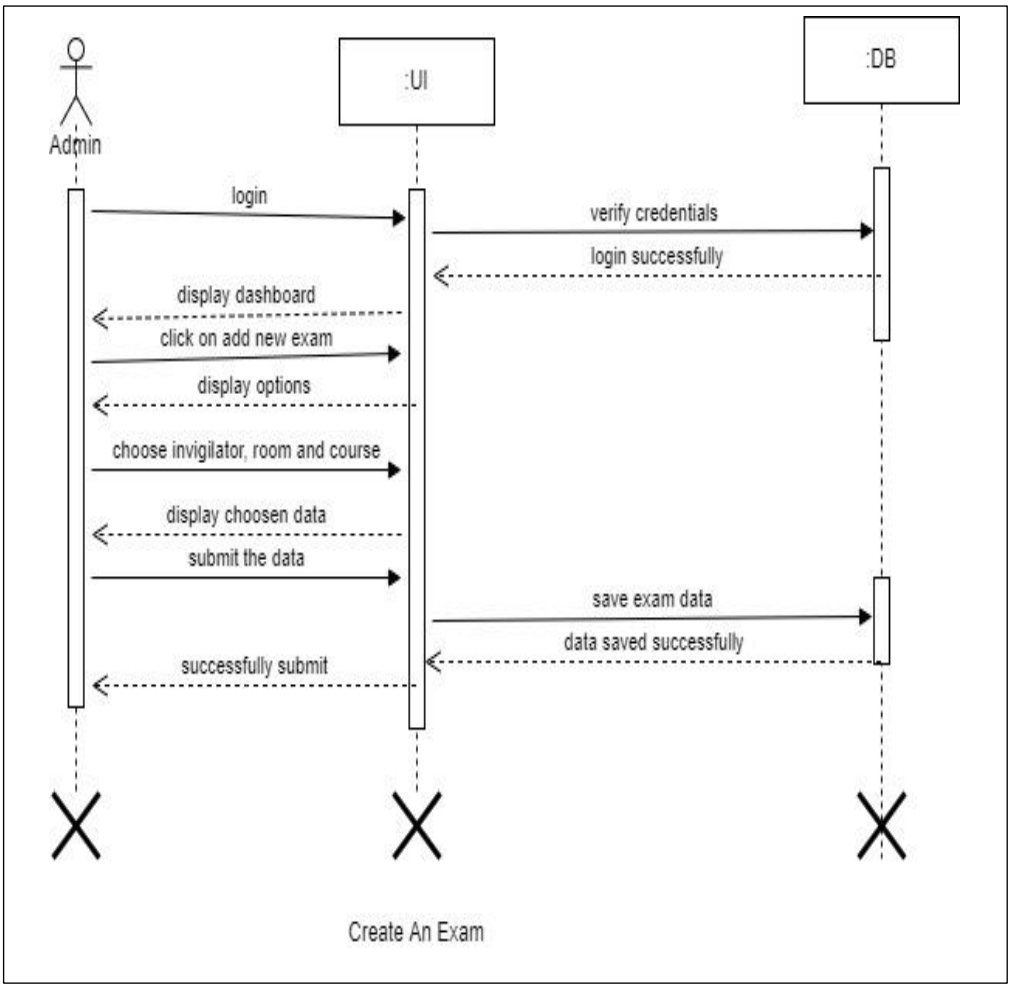


Figure 73 Create an exam sequence diagram

The sequence diagram above shows how the admin can create an exam.

7.1.1.10 Invigilator Management:

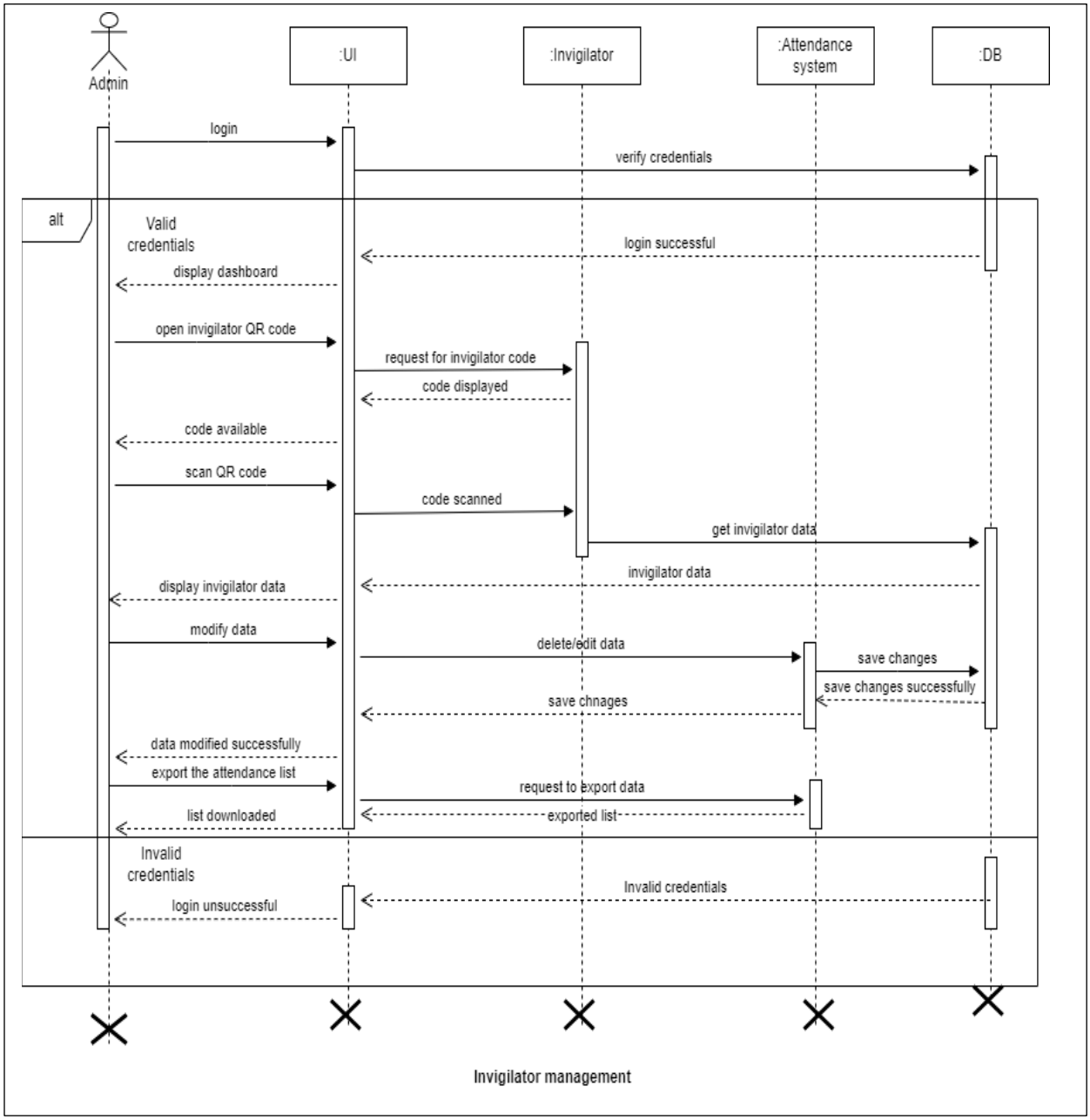


Figure 74 Manage invigilator sequence diagram

The sequence diagram above shows how the admin can scan the QR code, modify the data of the invigilators, and export the attendance list.

4.5.1.9 View admin profile:

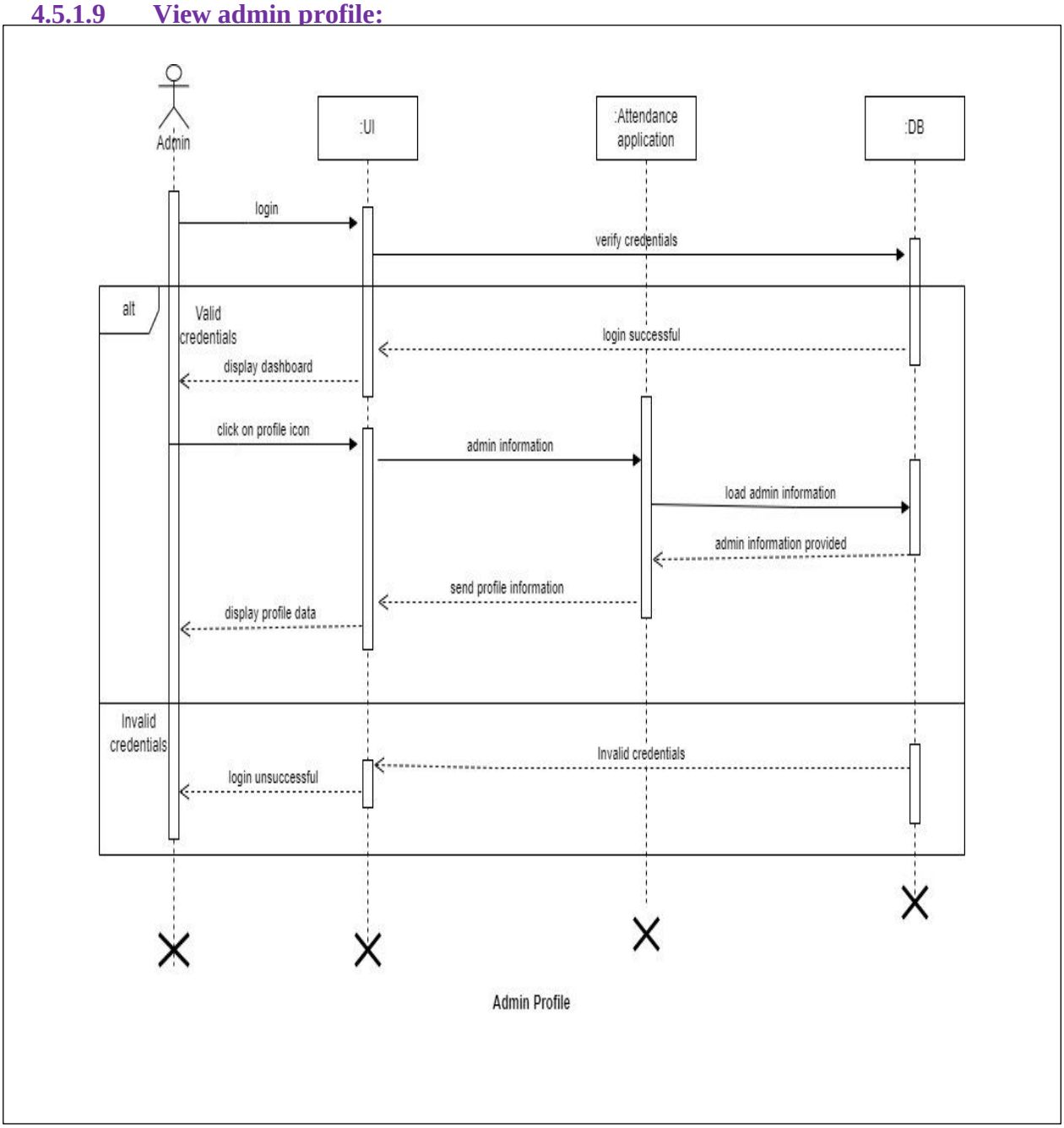


Figure 75 View admin profile sequence diagram

The sequence diagram above shows how the Hader system enables the admin to view their profile.

7.1.1.12 Scan student's QR code to check-in:

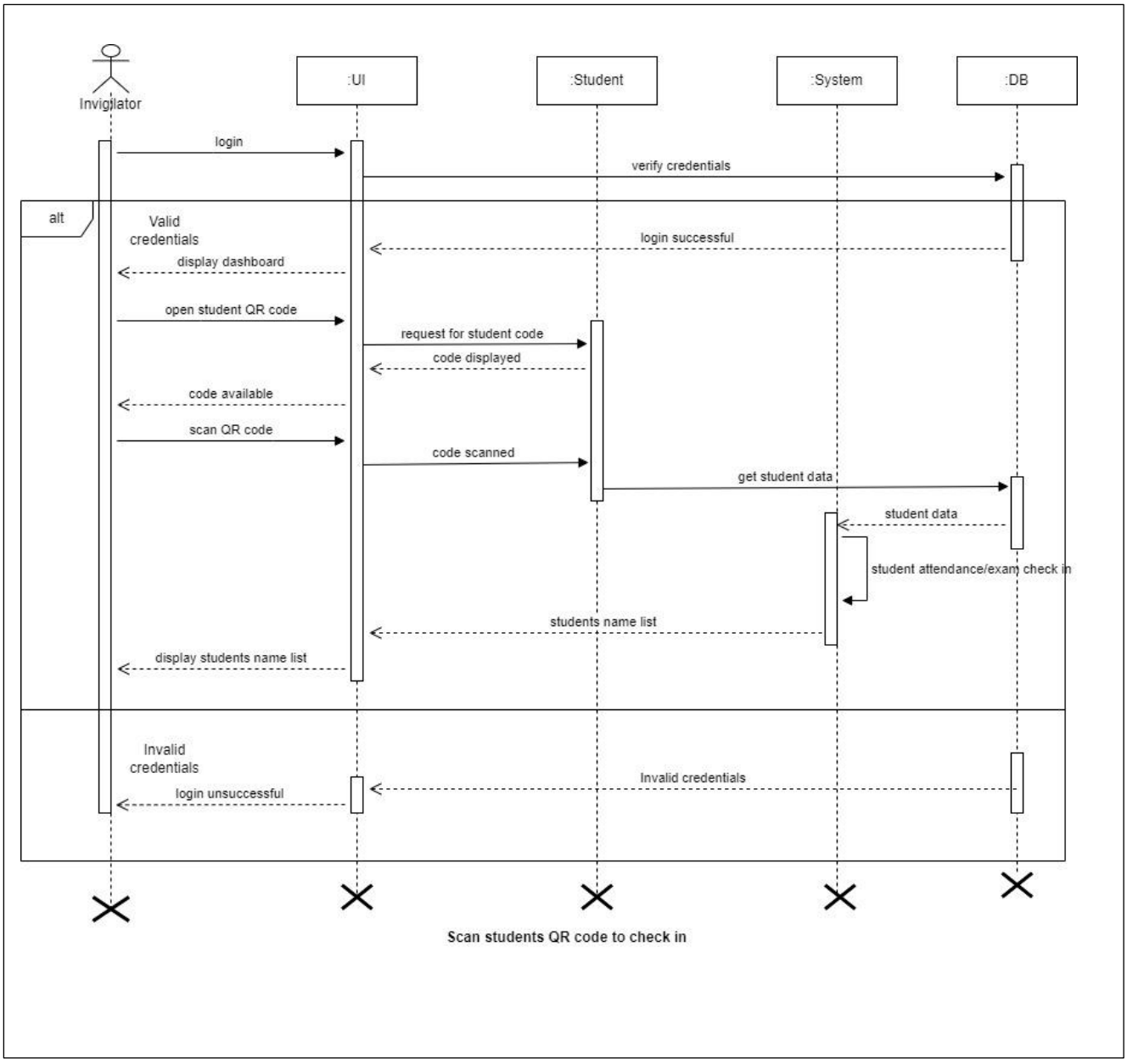


Figure 76 Scan students' QR code to check in the sequence diagram

The sequence diagram above shows how the Hader system enables the Invigilator to scan the student's QR codes to check in the exam.

7.1.1.13 Show Invigilator Information:

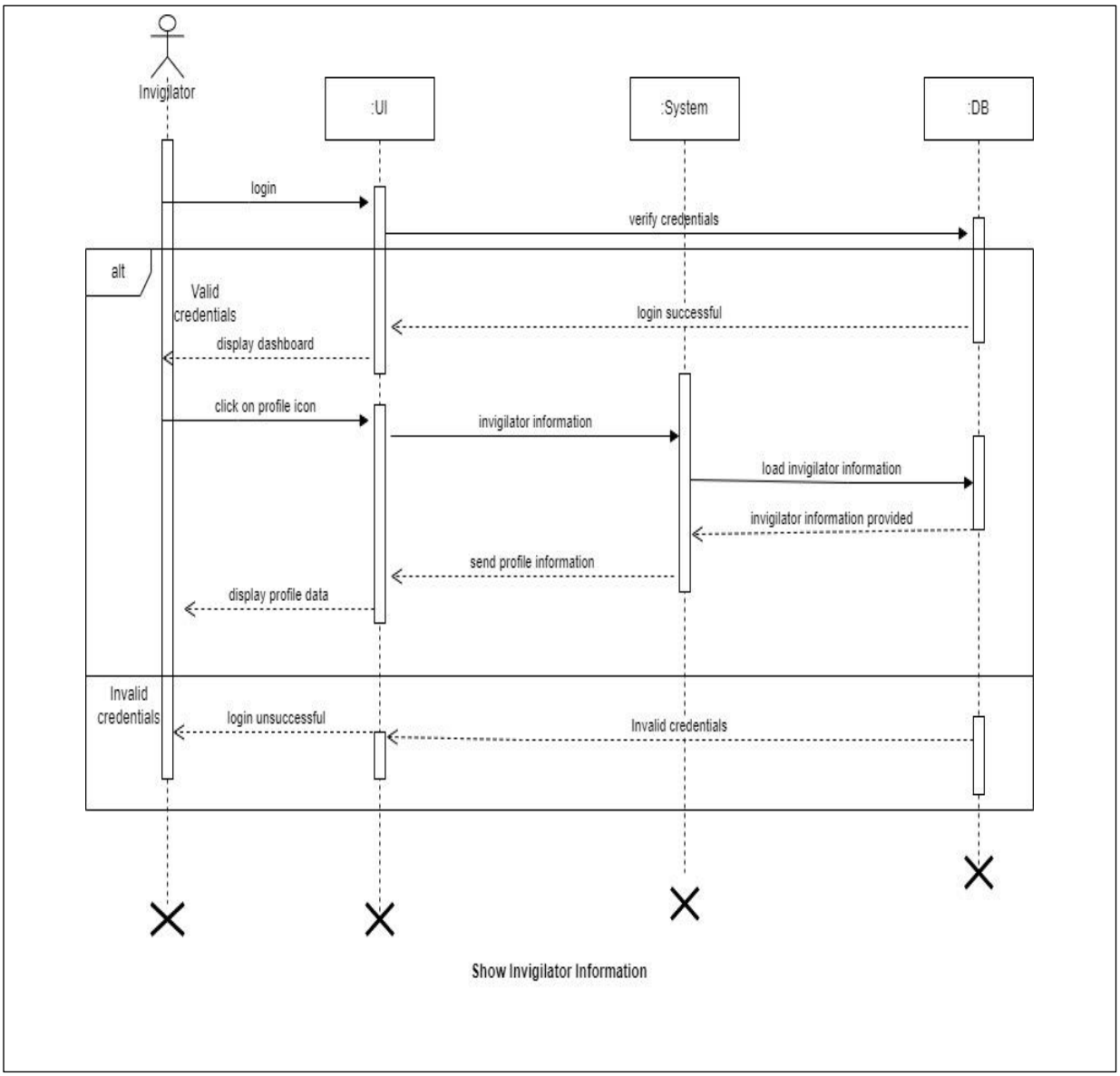


Figure 77 Show invigilator information sequence diagram

The sequence diagram above shows how the Hader system allows the user (Invigilator) to view their information on the profile page.

7.1.1.14 Modify Student's Attendance Status:

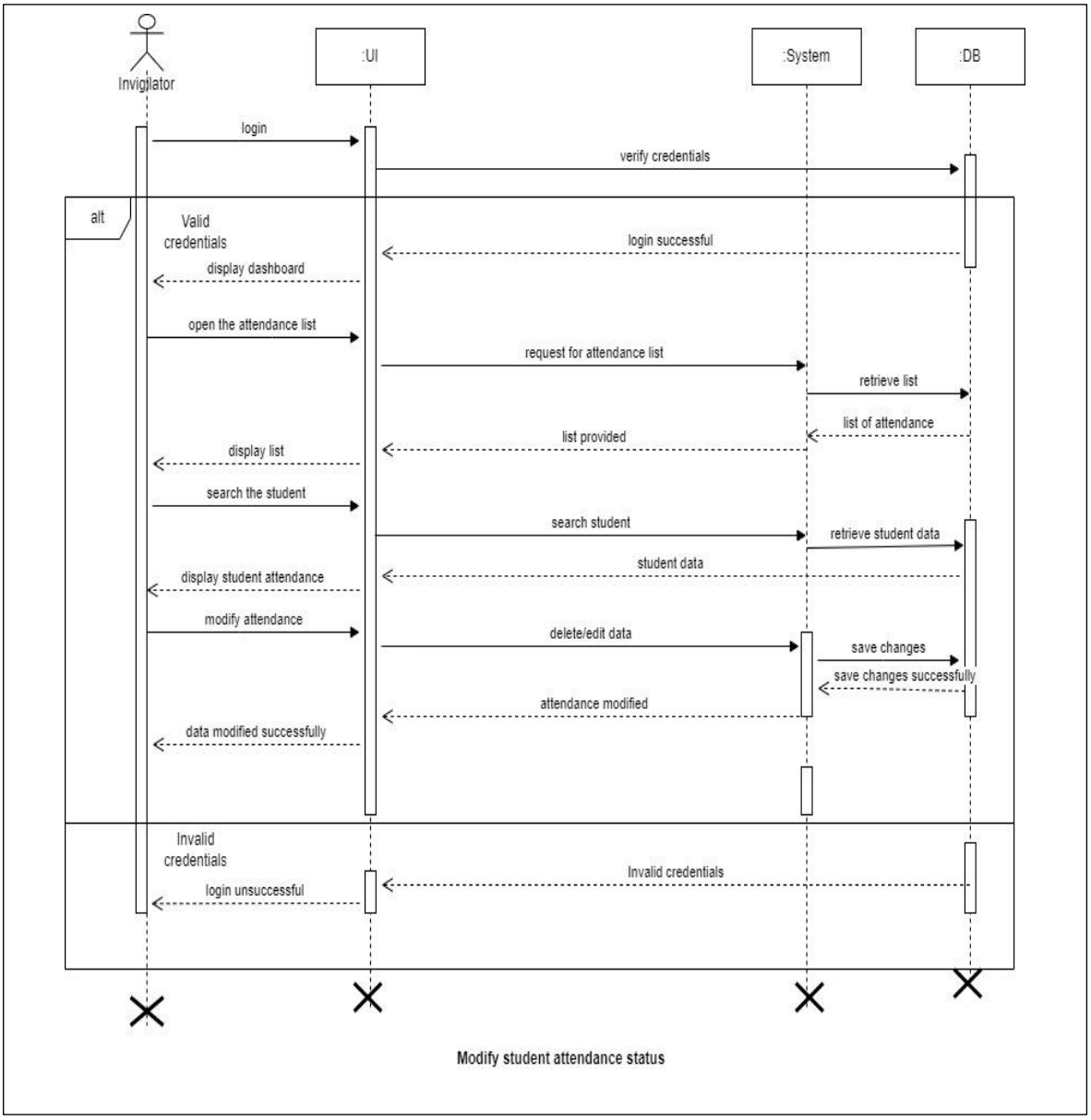


Figure 78 modify Student's Attendance Status sequence diagram

The sequence diagram above shows how the Hader system allows the Invigilator to modify the student's attendance status manually (in case of an error).

7.1.1.15 Invigilator Devices Receive Notification of Student's Attendance:

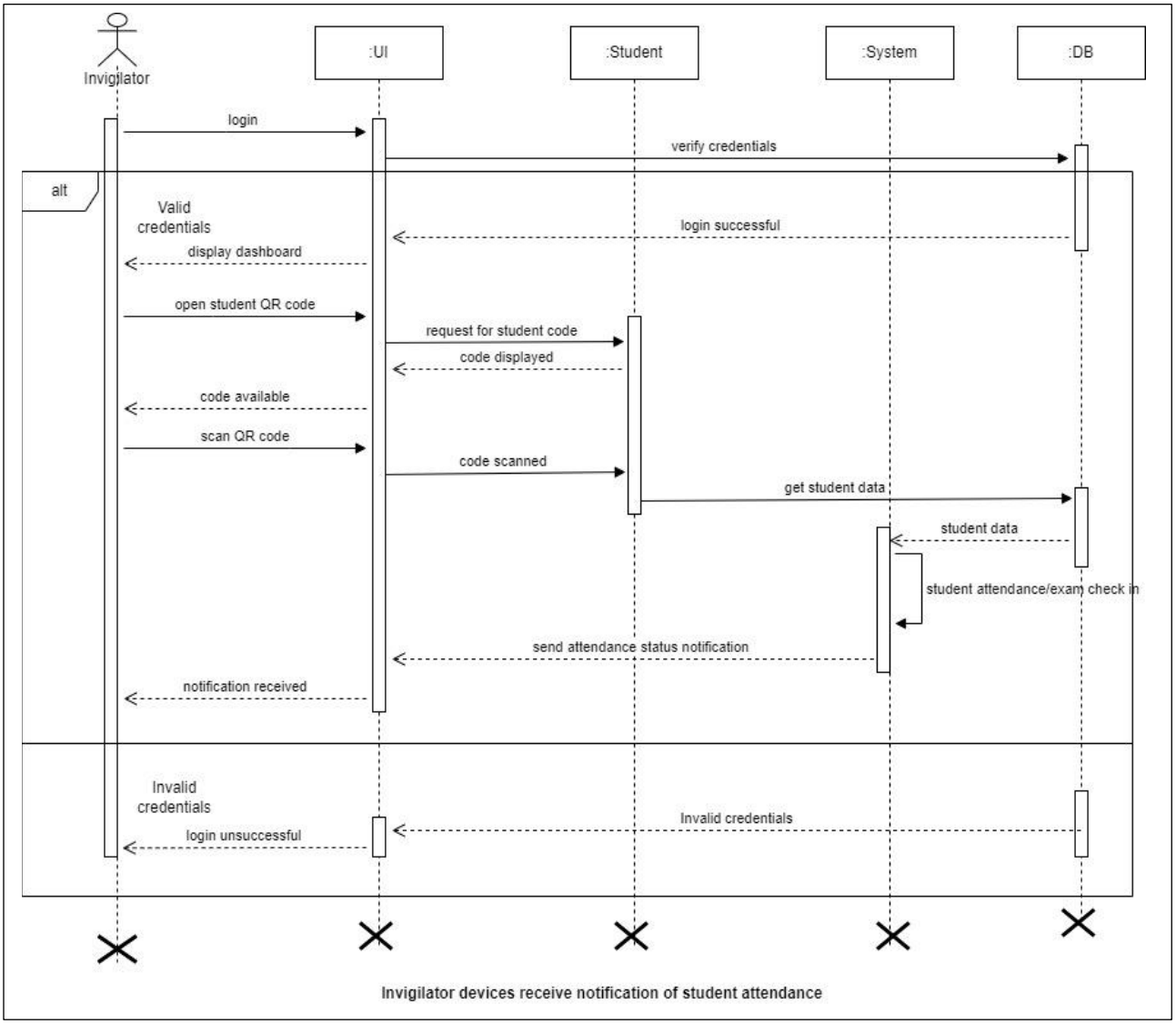


Figure 79 Invigilator receives attended student notification sequence diagram

The sequence diagram above shows how the system will send notifications to the Invigilator device after a student's attendance is scanned and added to the attendance list.

7.1.1.16 Export Attendance Report As PDF, XML etc.:

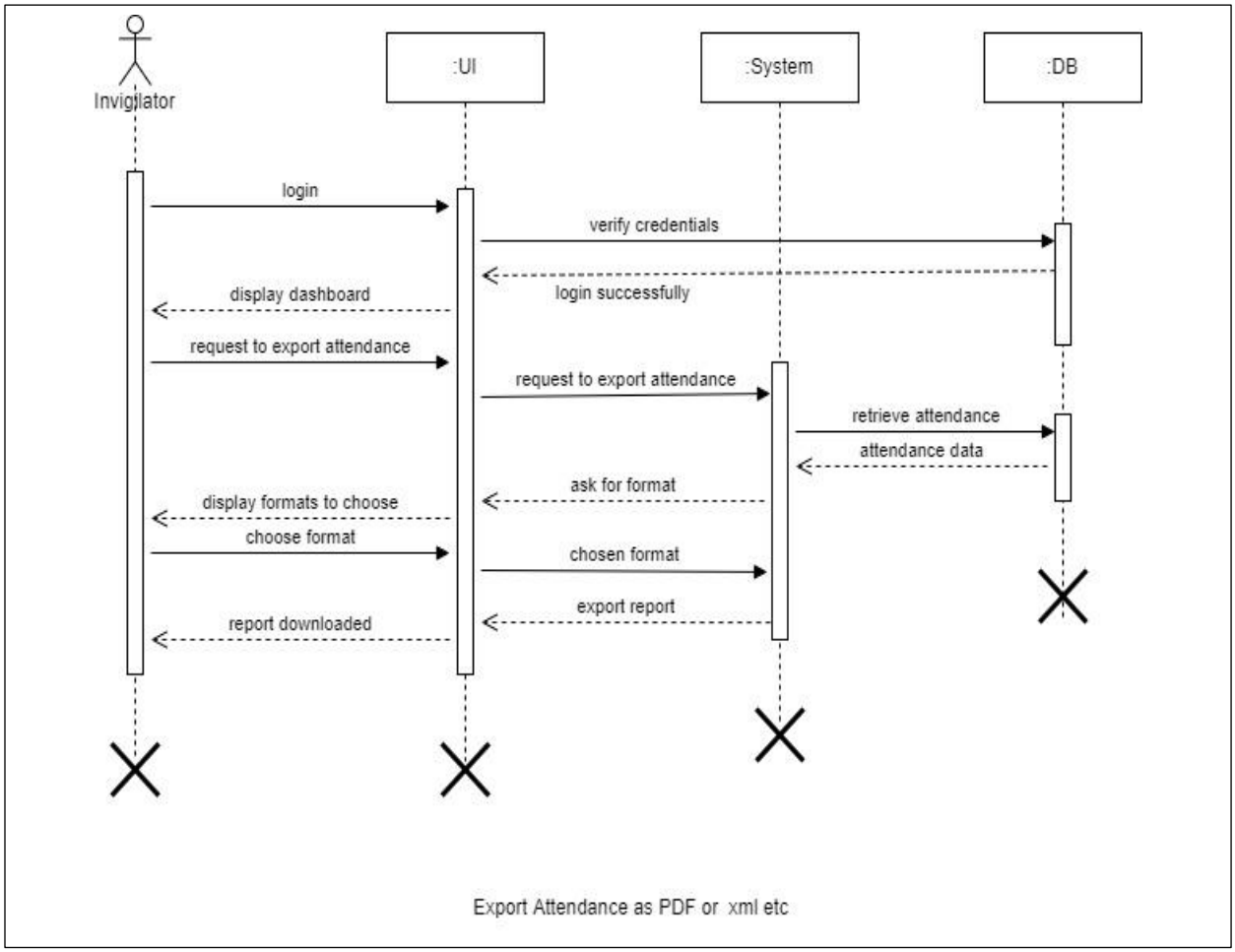


Figure 80 Export attendance report as a PDF sequence diagram

The sequence diagram above shows how the system allows the user (invigilators) to Export attended students' sheets as PDF, XML, etc.

7.1.1.16 Send Attendance Reports Via E-Mail:

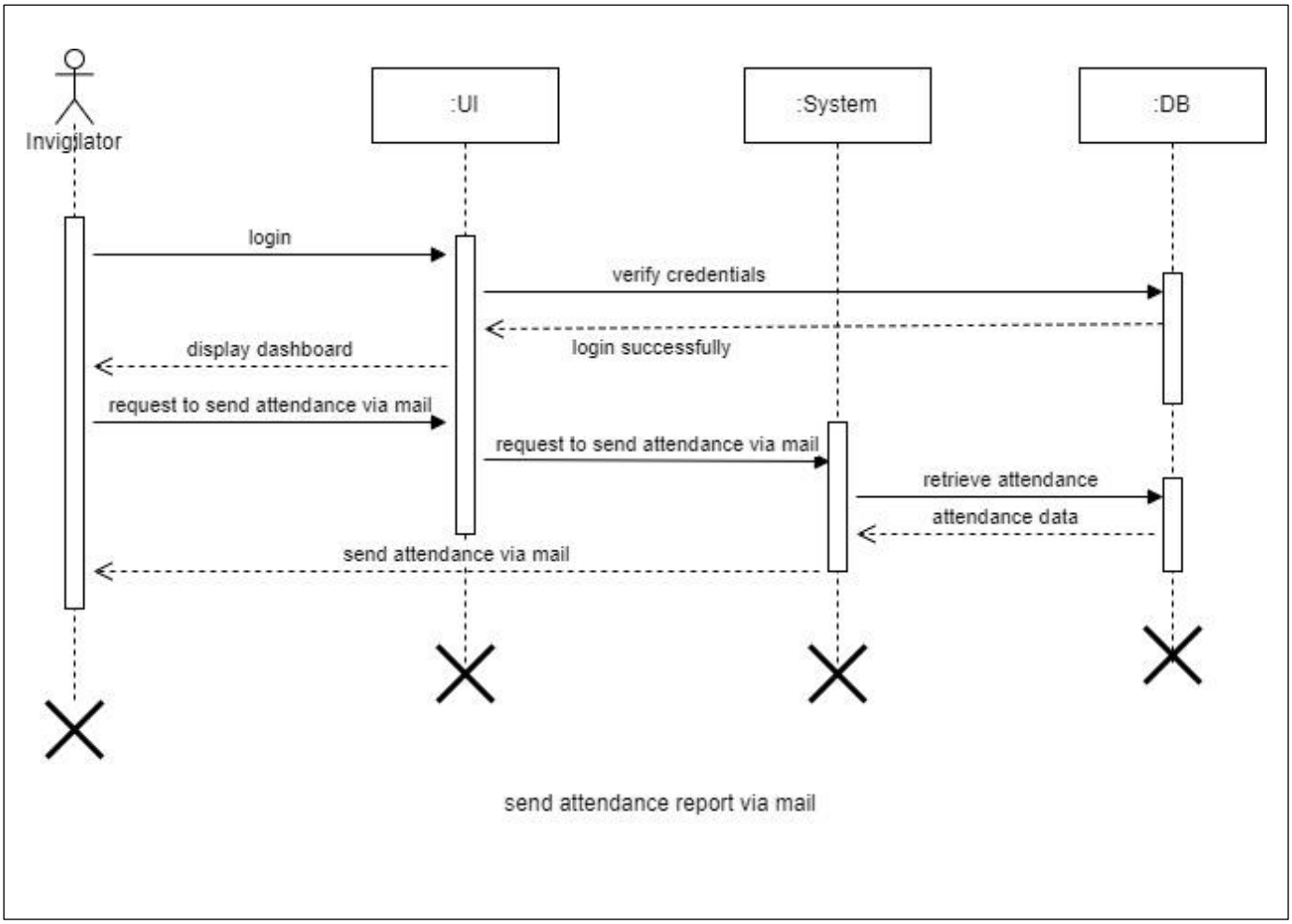


Figure 81 Send attendance reports via E-mail sequence diagram

The sequence diagram above shows how the system allows the user (invigilators) to send email attendance reports.

7.2Activity Diagrams:

7.2.1 Sign In/Sign Out:

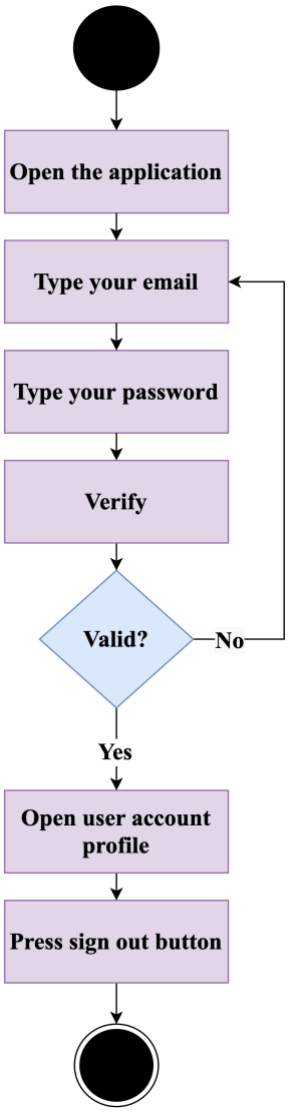


Figure 82 Sign in/Sign out activity diagram

The activity diagram above shows how the admin, student, Instructor, and Invigilator Sign into the system and sign out the form.

7.2.2 Forgot Password:

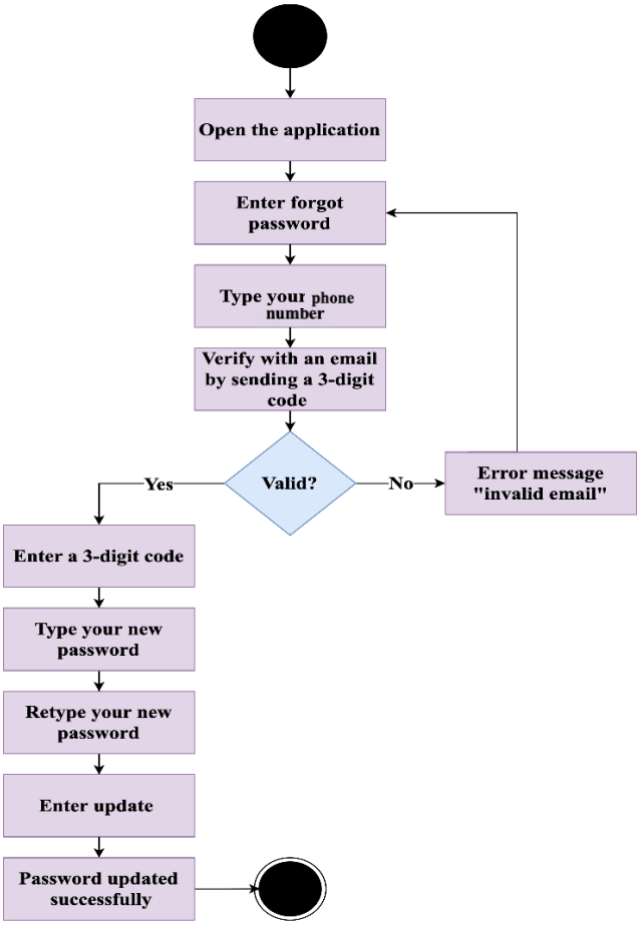


Figure 83 Forgot password activity diagram

The activity diagram above shows how the admin, student, Instructor, and Invigilator update their password when forgotten.

7.2.3 Dashboard:

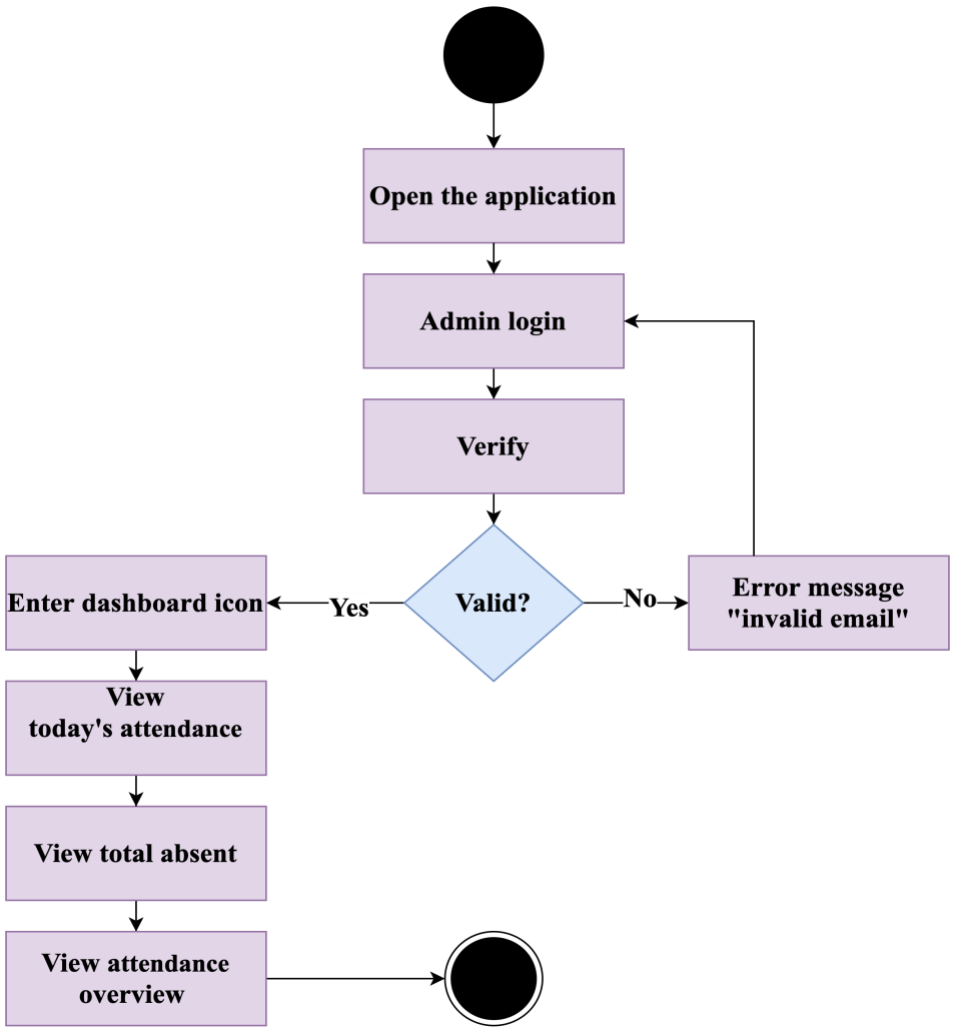


Figure 84 Dashboard activity diagram

The activity diagram above shows how the admin, can view today's attendance, totally absent, and attendance overview.

7.2.4 Manage Students:

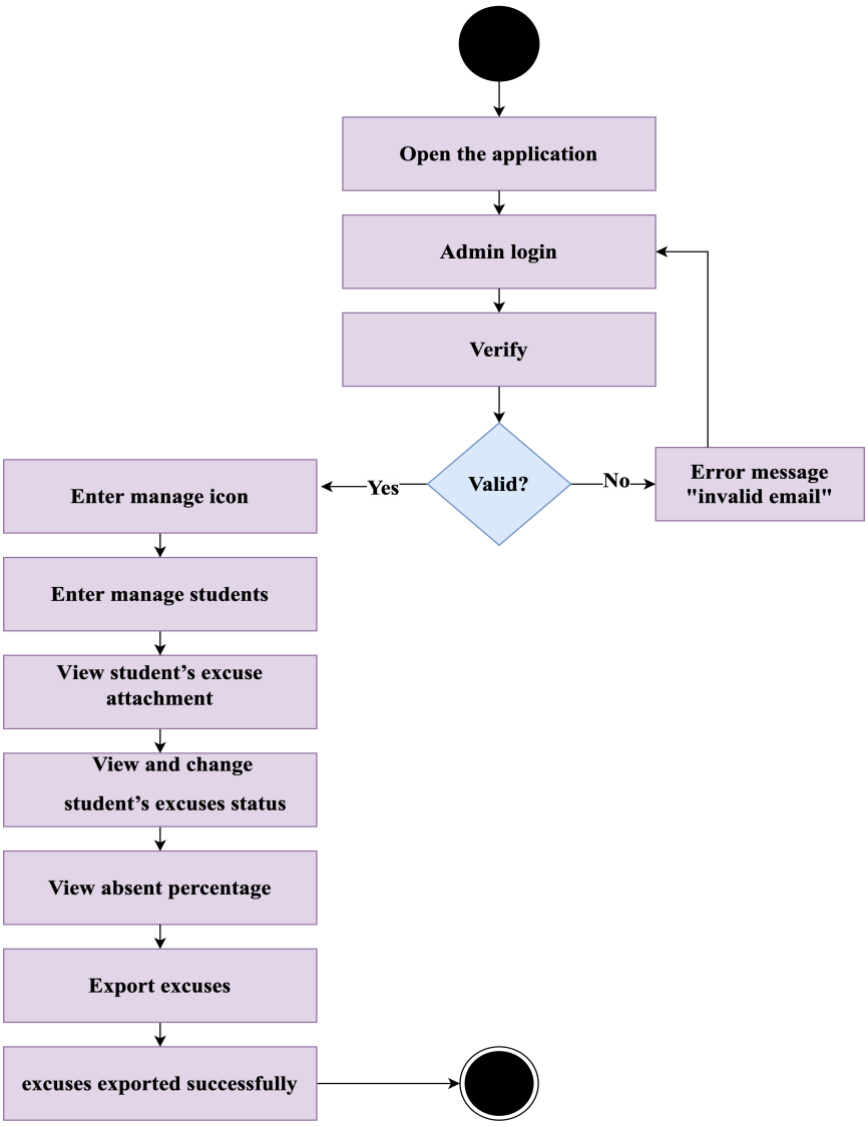


Figure 85 Manage student activity diagram

The activity diagram above shows how the admin can view a student's excuse attachment, and aabsent percentage, change the student's excuses status and export the excuses.

7.2.5 Manage Invigilator:

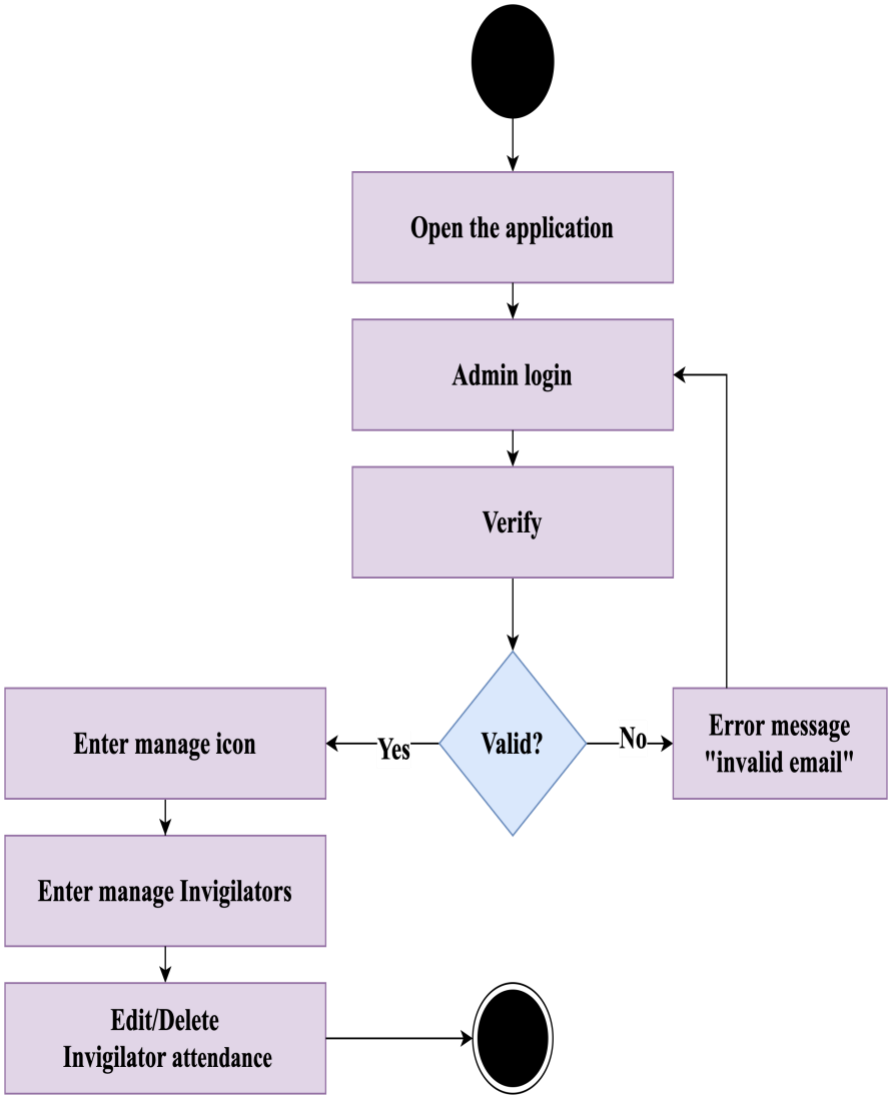


Figure 86 Manage invigilator activity diagram

The activity diagram above shows how the admin manages the Invigilator.

7.2.6 Admin: Add Exam:

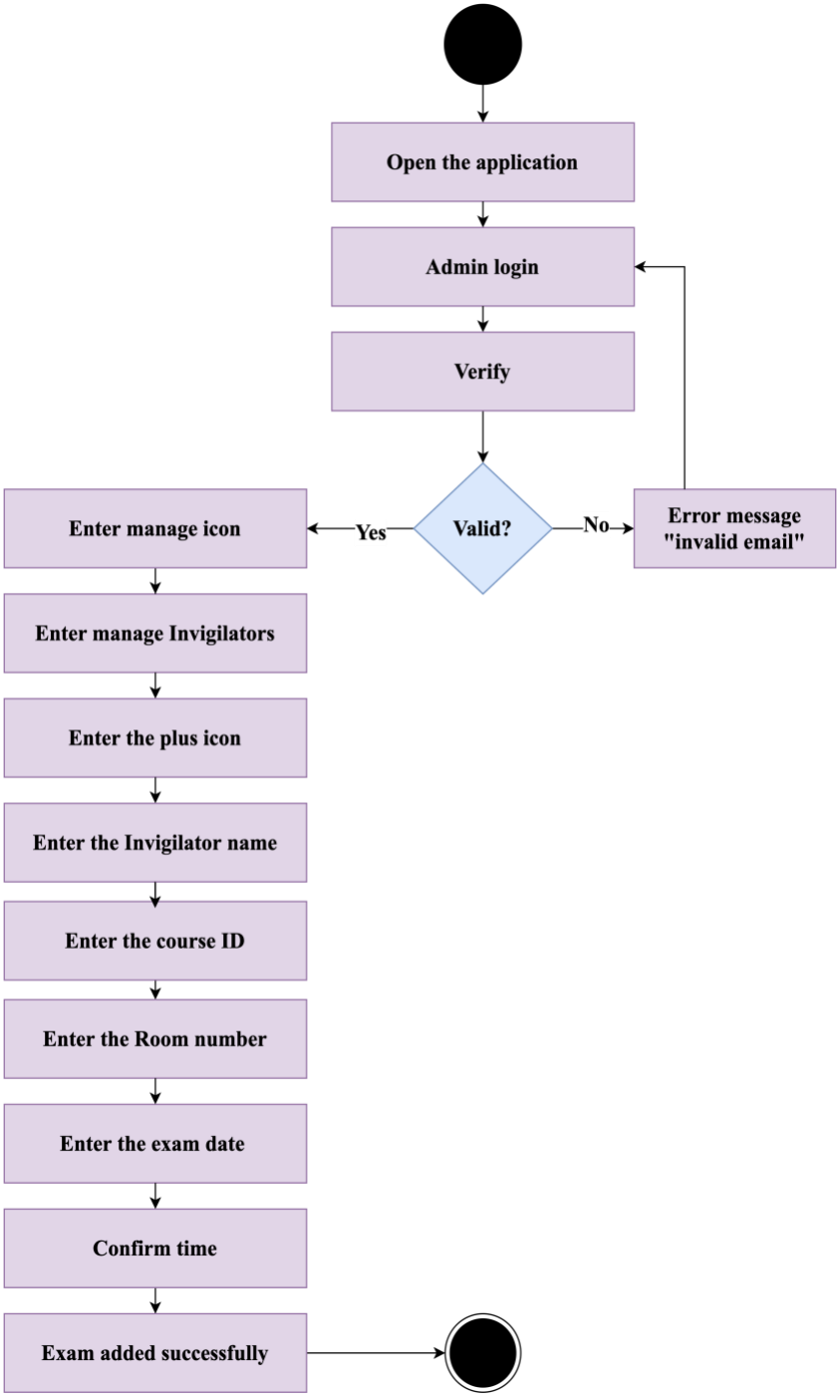


Figure 87 Manage exam activity diagram

The activity diagram above shows how the admin can add an exam.

7.2.7 Upload excuse for the student:

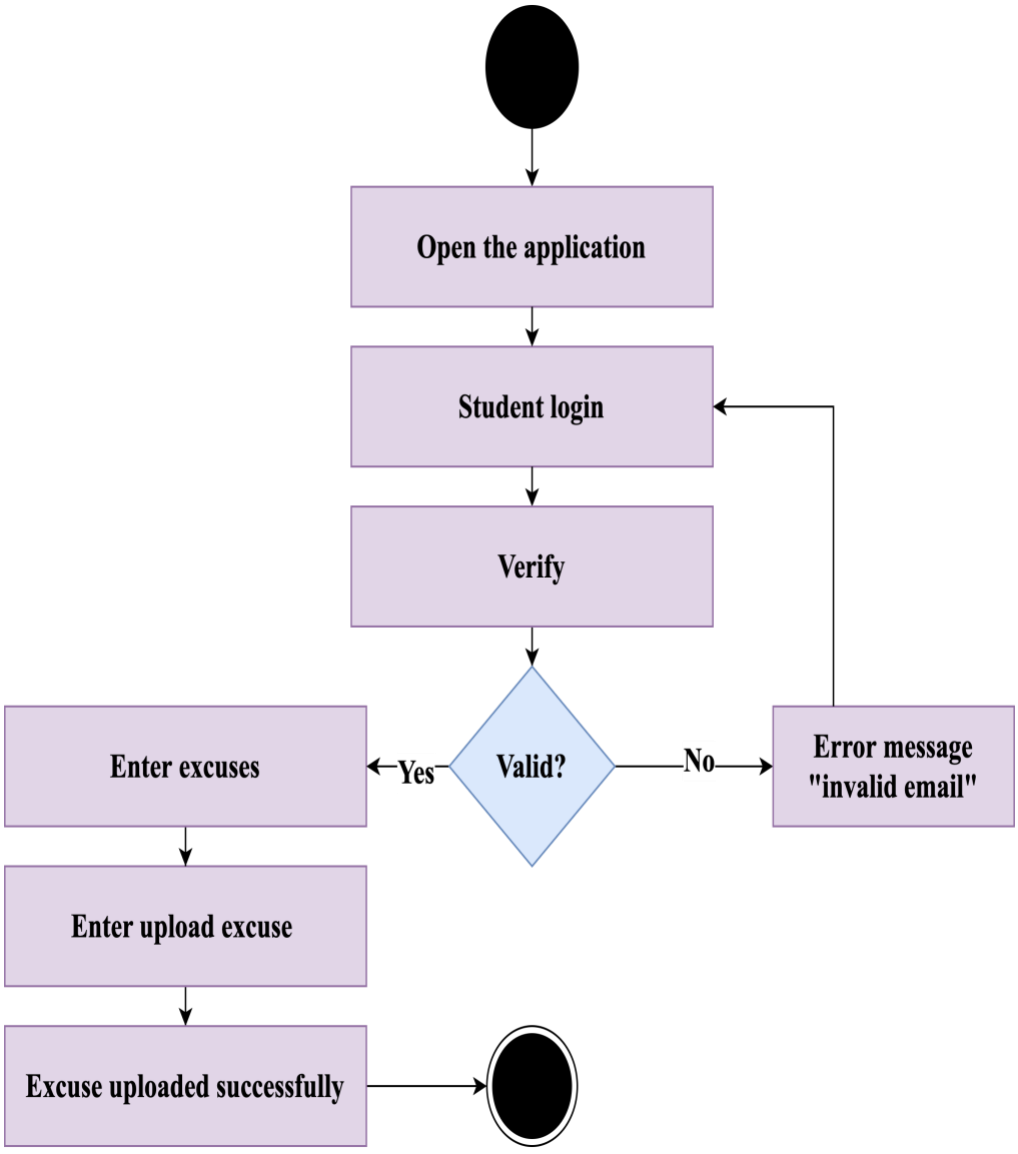


Figure 88 Upload excuse for the student activity diagram

The activity diagram above shows how the student can upload an excuse.

7.2.8 View excuse status for the student:

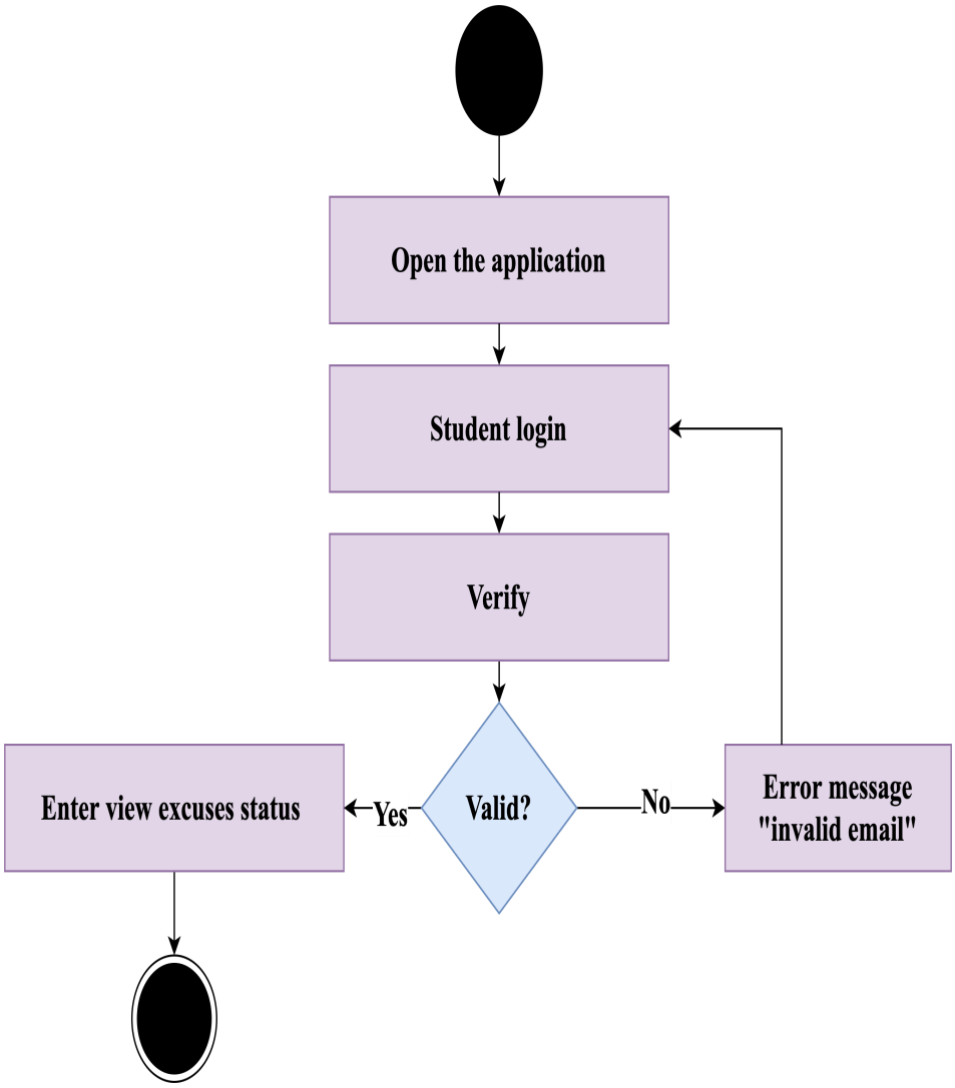


Figure 89 View excuse status for the student activity diagram

The activity diagram above shows how the student can view excuse status.

7.2.9 Invigilator: View/Edit/Export Student's Attendance Sheet:

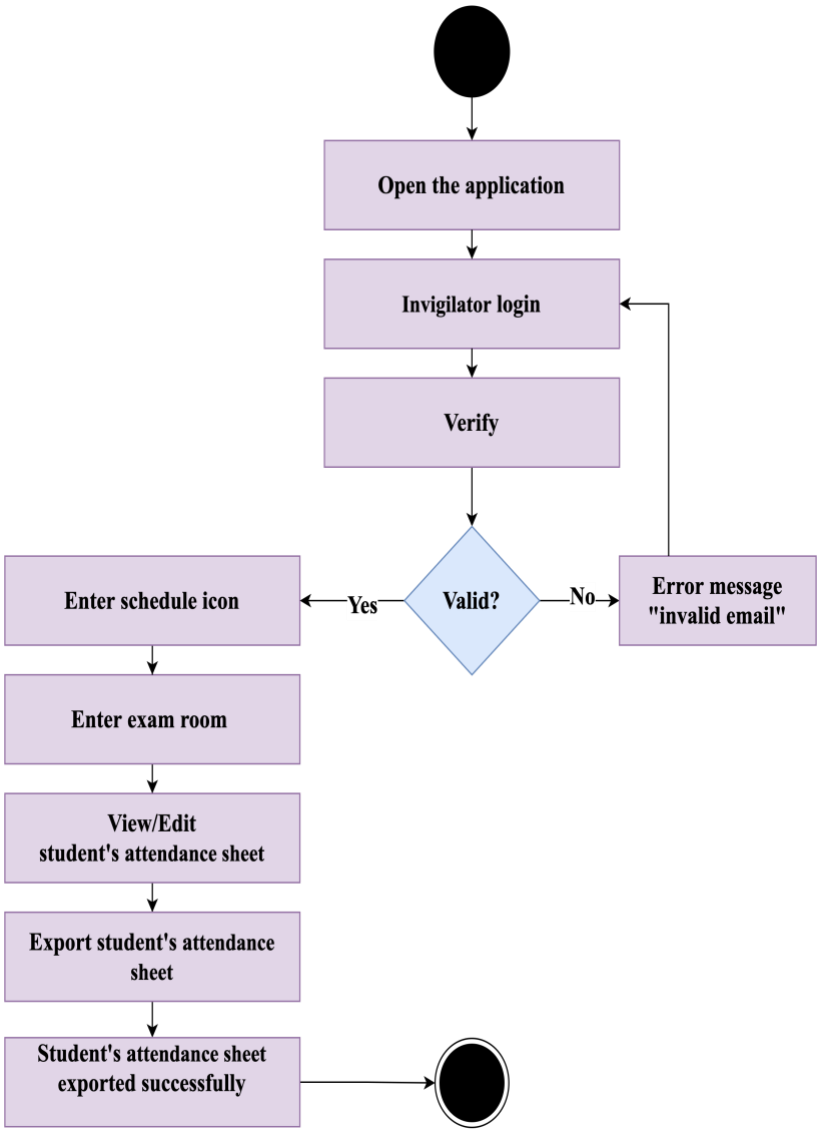


Figure 90 Invigilator: View/Edit/Export student's attendance sheet activity diagram

The activity diagram above shows how the Invigilator can view, edit, and export students' attendance sheets.

7.2.10 View Exam Schedule for The Invigilator:

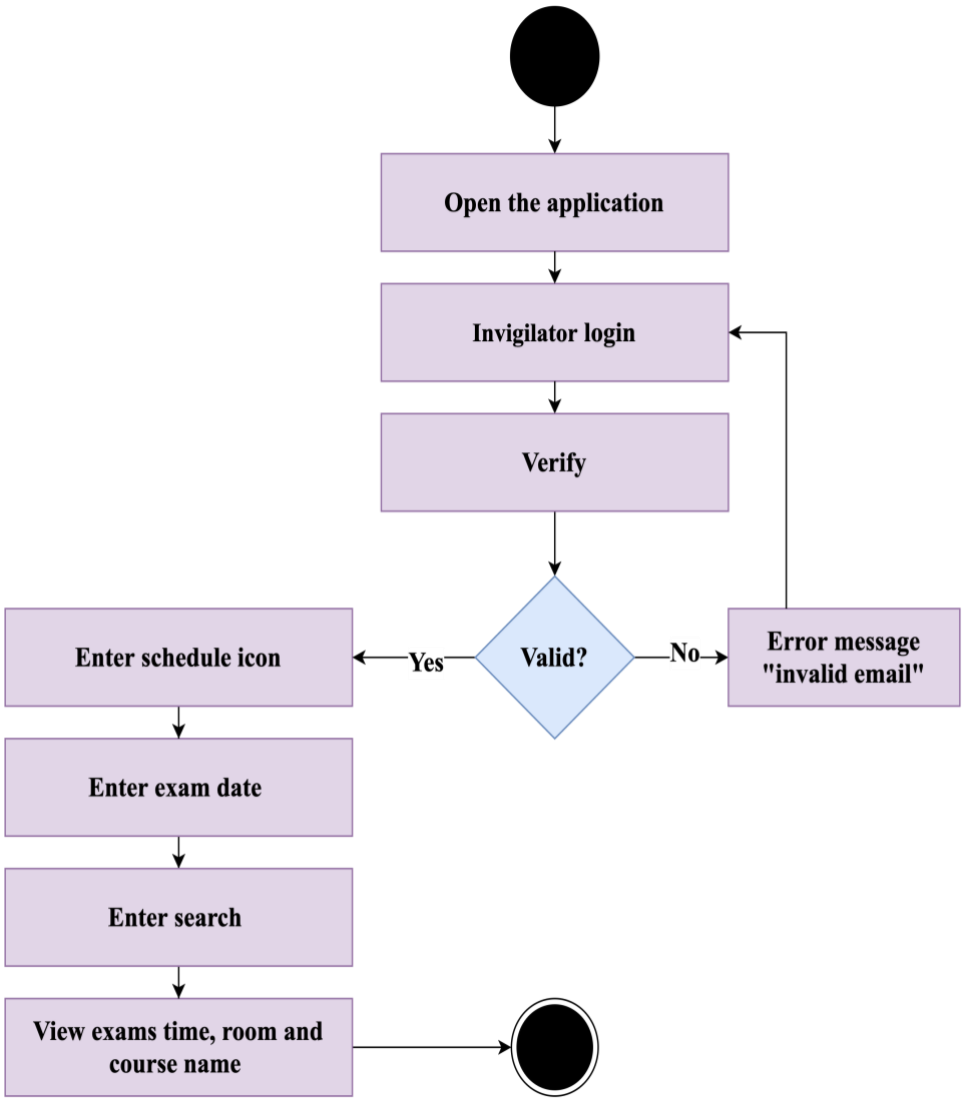


Figure 91 View the exam schedule for the invigilator activity diagram

The activity diagram above shows how the Invigilator can view the exam schedule.

7.2.11 Scan Student QR Code:

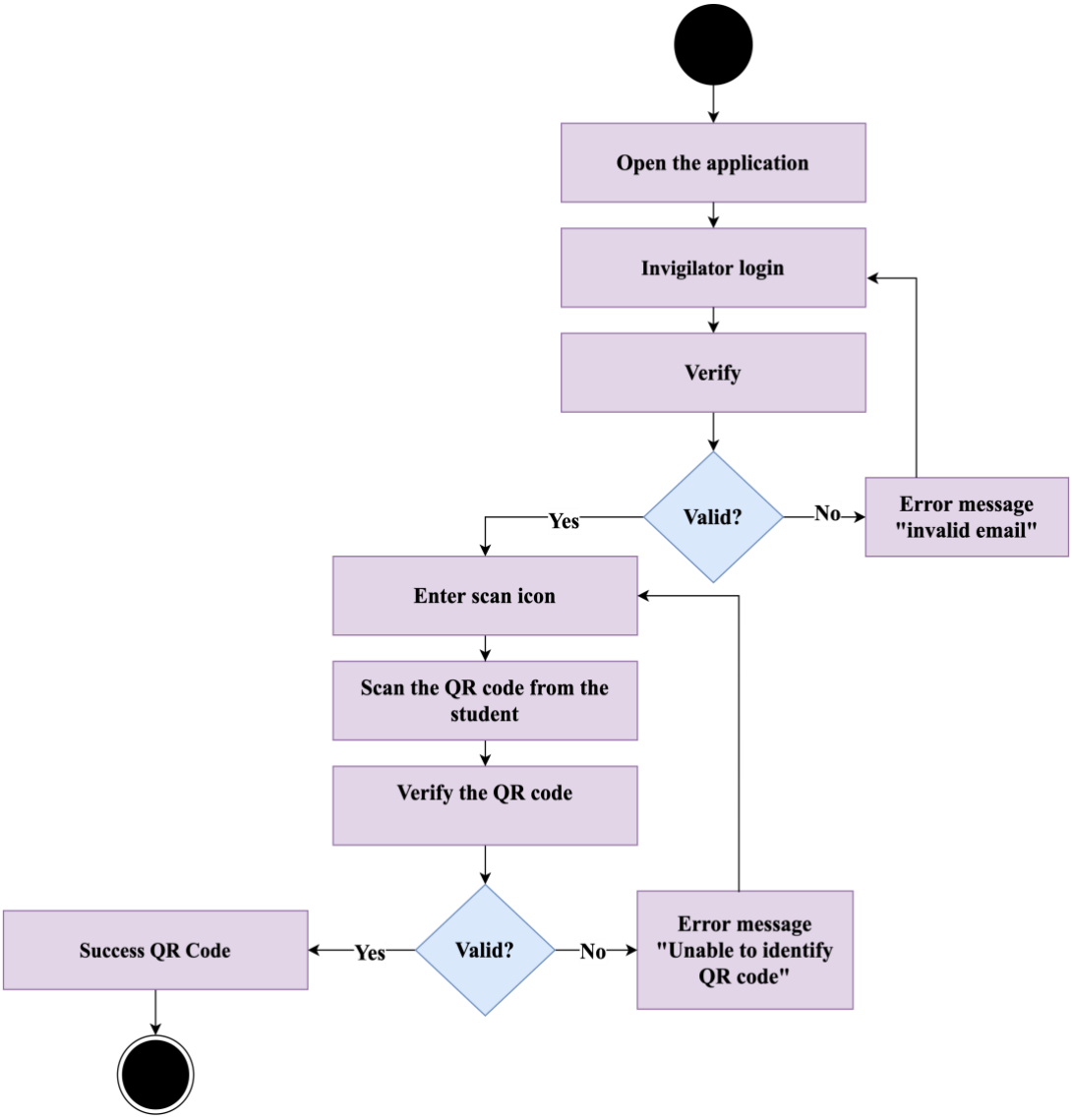


Figure 92 Scan student QR code activity diagram

The activity diagram above shows how the Invigilator can Scan the QR code from the student.

7.2.12 Instructor Profile Information and Notifications:

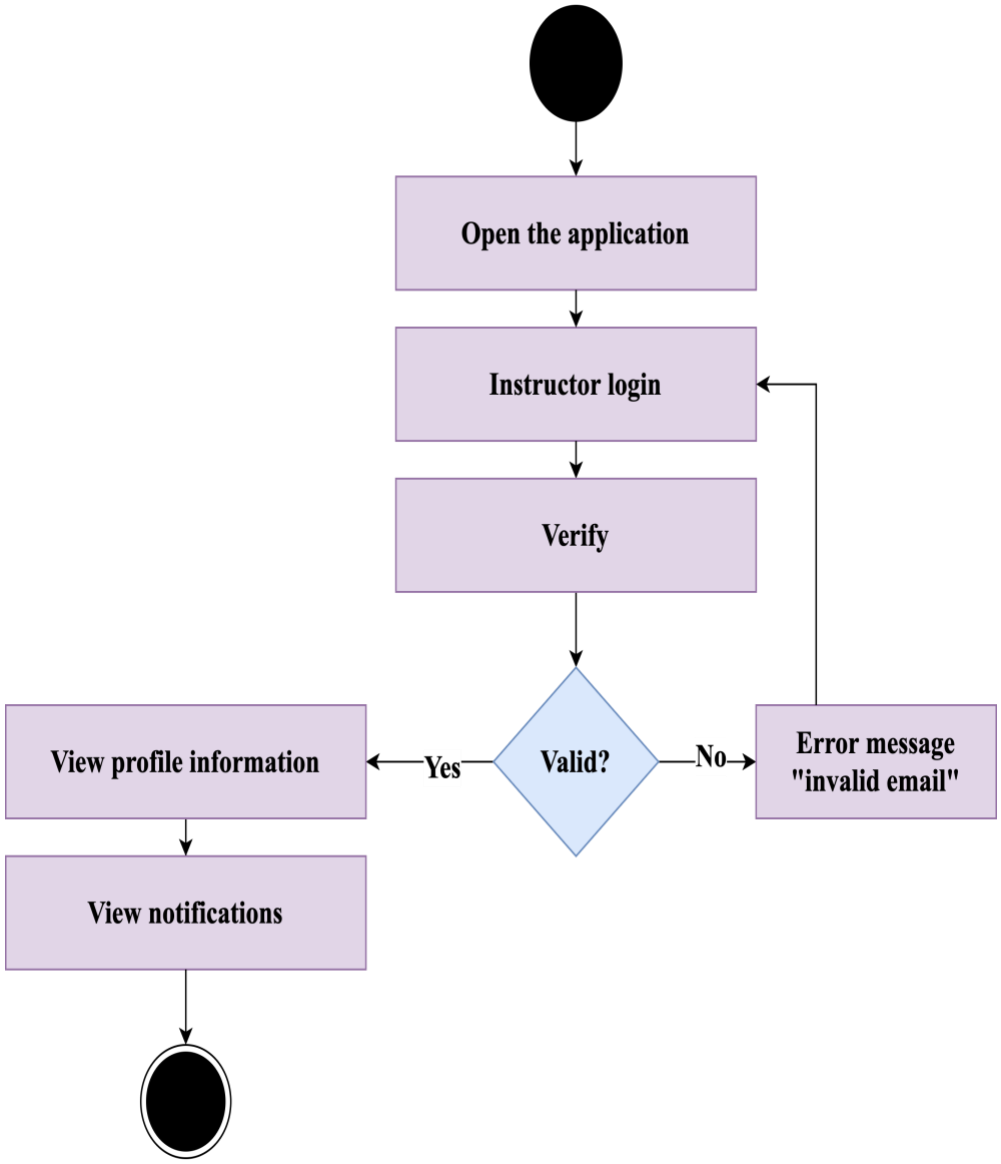


Figure 93 Instructor profile information and notifications activity diagram

The activity diagram above shows how the instructor can view profile information and notifications.

CHAPTER 5. PROJECT IMPLEMENTATION

1. Introduction

Nowadays, Attendance records and reporting are fundamental for any institution, whether educational or commercial to operate effectively. Many academic institutions in economically developing nations are using pen and paper to record attendance, which takes more time, is inconvenient, and sometimes factually incorrect.

In CCSIT at IAU, an invigilator would pass an attendance sheet for students to sign or call out their names. The project's goal is to address and resolve concerns related to exam attendance recording, because individuals nowadays carry their phones everywhere, according to statistics, the average age of smartphone users is 26 years old, with 42% being under the age of 26[1]. With the widespread use of smartphones among university students a software solution based on a mobile application is the most natural choice along with QR code technology. The proposed solution provides each student with a QR code that can be scanned by an invigilator using a QR Code scanner and the HADER Application to take attendance information.

The initiative involves students, invigilators, teachers, test units, and academic affairs at IAU. Hader test attendance system is an iOS and Android native mobile app. Students, lecturers, and the Academic Affairs Department of CCSIT at IAU. This application is developed using Flutter and dart languages to be used on both Android and iOS operating systems and the firebase database. The student shows the invigilator/instructors their QR code, the invigilator/instructors scan the student's QR code, and the student's attendance is automatically recorded in the database. The student can also upload excuses, which are reviewed by the Academic Affairs Unit.

All The evaluators' comments on Hader to expand the scope and increase the challenges have been addressed and the project has been updated accordingly. In addition, the testing plan has been created and assessed to test the HADER application before delivering the project to make sure that it is working effectively and efficiently. Also, HADER team members are developing a dissemination plan to publish a research paper, it is included in the dissemination plan section of this report. The working software for HADER has been developed successfully along with the database for the common interfaces, and the HADER team currently moving towards coding other interfaces.

Finally, the design of the test plan document is included which will show approaches, resources, and the schedule of all testing activities.

2. Summary of the Proposal

Hader's application solves a variety of exam attendance issues at CCSIT and mainly aims to streamline the process of taking attendance in exam halls.

The HADER program will be used by four sorts of users: Admin, instructors, Invigilators, and Students. Depending on the user's role, the program has varied characteristics:

- Admin: is responsible for managing exams, managing students' excuses, view students' reports.
- Instructors: are responsible for managing exams, viewing their section's exam attendance reports, viewing admin's exam approvals/declination, switching to invigilator mode to take students' attendance by scanning their QR Code, share attendance reports to any party in CCSIT via email,

- Invigilators: take students' attendance by scanning their QR Code and share attendance repowiths to any party in CCSIT via email.
- Students: upload/view exam excuses status. Check-in to exams via QR code.

For the work of all the implementation phases, the project team will continue with the waterfall model which is the most suitable approach to be used for this project, because the waterfall methodology best when teams are small and the objectives of projects are clearly outlined from the beginning, waterfall methodology is used when the project requires a high level of reliability and when project goals and objectives rarely change. The project will be implemented using different software tools such as Flutter which is used for the coding part -including the design-. For the data part, firebase will be used. The project will be tested as planned in the proposal and mentioned in the test plan in detail.

3. Changes From the Proposal Phase and Justifications

This section includes all suggestions and comments from the reviewers and responses to each comment:

Evaluator: Ms. Elham Shaflout

Table 23: EvaluatorsEvaluator's Feedback (1)

Evaluator's Comments:	Team's Justification and Response:
1. **Abstract:** needs improvement in the style of writing. there is no links between sentences. also, some sentences need rephrasing into using scientific words.	Noted and updated.
2. **4.BACKGROUND AND LITERATURE REVIEW:** have you considered the existing apps for attendance tracking? they are very similar to your idea, and they are free to install by anyone and use. I found many apps that apply similar concept using fingerprint, cloud, barcode and some of them gives you options to choose how to take attendance. What your system would deliver compared to those apps?	We have added more literature reviews.
3. **1.7 SCOPE/ LIMITATION OF THE STUDY:** have you considered if the system would be integrated with the university system or not? and limitations was not mentioned here.	Noted and updated.
4. **1.9 PROJECT ORGANIZATION:** I'm not sure if this is what should be written in this section.	Table has been changed to bullet point and the remaining was kept the same.

5. 2.6.1 Process Model: The model presented in the graph is not a waterfall model, this is a hybrid model. you need to build enough understanding of the difference between waterfall/ Agile/ Hybrid models. Be specific.	The mistake was in the arrows as it was going down to the next phase and another arrow that is looping to every phase. The arrow that was looping has been deleted to follow the waterfall process model.
6. 2.6.3 Project Infrastructure: which version of JavaScript? and what about other component of the software (e.g., backend)? do you have an initial design of the software architecture? if yes Then provide it here.	We have updated and changed few of the tools and infrastructure we are using and according to that JavaScript is no longer needed.
7. Make sure to study all possible scenarios regarding exam invigilation, when more than one invigilator is invigilating the same room, then either one of them do the scanning or both but make sure that no one student get scanned more than once.	Considered.
8. Using an uploaded image to do the authentication, I think this could have some security issues, so make sure to study the best practices if you intend to implement this, otherwise, I will not recommend that you force the users to upload their images in the system to ensure privacy and protection of their data.	Considered.
9. When more than one instructor is teaching the same course and two sections for two instructors are combined in the same class.	Each instructor has their own list.
10. You may consider implementing the same authentication process for invigilators attendance as well.	Considered.
11. There are some features allows to share some files such as student's excuses, make sure that this would not break any privacy issue as this file can be considered a confidential between student and the college. You might add some restrictions on the sharing function (e.g. not to share them through social media,..etc).	Considered.

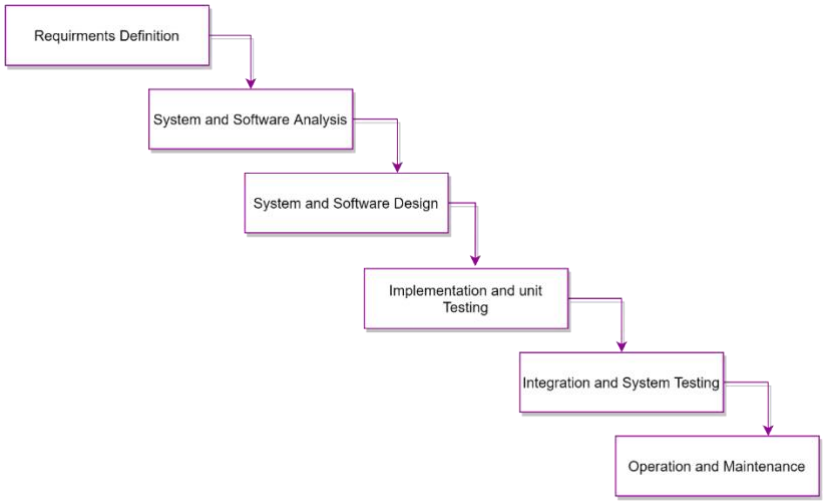
12. Usually when designing the software quality metrics, you have to define a specific measure in which you can test your system against once it is implemented, this way you can say confidently that you met your requirements, and you can show numbers to your audiences. Same practice.	Considered
13. can be followed also when you define your objectives, they have to be SMART, means it has to be measurable and tested for the level of satisfaction.	Considered
14. In the architectural design, it is very generic one, if you show this to an investor for example or even a specialist in the field, they will not understand anything, or in other word they cannot see what the idea of your system/app is and what makes it unique. you must fix this design and show how the different components of your system are interacting, this is like design a building/house.	Considered and updated.

Evaluator: Dr. Rami Mohammad

Table 24: Evaluator's Feedback (2)

Evaluator's Comments:	Team Response:
1. **Introduction:** needs more information just add up little background purpose scope etc. (At least make it one page) secondly intext citation must be provided.	Noted and updated.
2. **4.BACKGROUND AND LITERATURE REVIEW:** Literature study needs to be further expanded as only 6 previous studies are not enough. Also, the LR should be rewritten so that it reflects the pros and cons of each reviewed technique.	We have rewritten few literature reviews and we have added more literature reviews.
3. **1.5 JUSTIFICATION**: Justification must be shown by quoting the references of the literature or some proved stats or data etc.	Noted and updated.
4. **2.6.1 Process Model:** Revise the formatting under this title: "Process Model"	Noted and updated.
5. **Over all comments:** The project idea seems interesting. Usability, user security and data flow management are important for the project	Considered.

my suggestion is to focus on these points as well. Lastly, I would like to encourage to go for the research paper as we have one specific rubric for the research this time.	
6. Pay attention to the formatting. This should be consistent all over. Especially, font style, capital and small letters, ….	Noted and updated.
7. In the introduction section please revise this "Academic Affairs Department at Imam Abdulrahman bin Faisal University"	Noted and updated.
8. Avoid using "he, she, we, they, our, …."	Noted and updated.
9. The LR section is like a summarization rather than a LR that justifies the work and shows the gap.	Noted and updated.
10. Check the formatting under title 5.2.	Noted and updated.
11. More explanation should be added to Table 1.	Noted and updated.
12. In the scope section, the last point should be considered in implementing the project and not as a limitation.	Noted and updated.
13. Justification of using waterfall is generic and should	Noted and updated.

be discussed from the project point of view.	
14. In section 13.4, there should be a clear method for assessing these.	Considered.

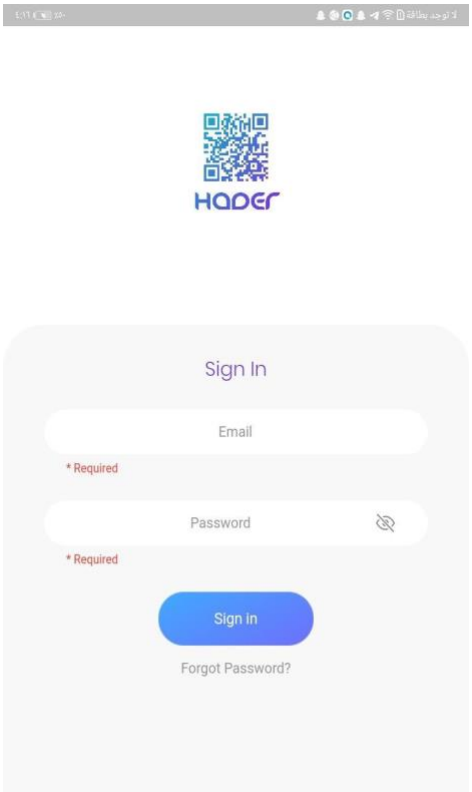
4. Implementation progress

The team has taken into, considered, and worked on the evaluators' comments and updated the project accordingly. The team has also completed the testing plan to start the test. Afterthat, the team has progressed and has a worked on the software of common interfaces using Flutter and Dart languages, as well as a database created with Firebase. Then worked on the coding of the whole application including user interfaces and functionality and completed the application on 1/5/2022 and atarted testing and implementing test cases and maintenance until 15/5/2022. In addition, the team has submitted the dissemination plan for publication. The common HADER application interfaces are done and illustrated below:

1. Startup Page:

This page will appear every time when any user open HADER Application.

Figure 94: Start-up Page



2. Sign in Interface:

The user can log in using their email and password.

Figure 95: Sign in interface

3. Forgot Password Interface:

Users can reset their passwords by, clicking on forgot password and entering their phone number.

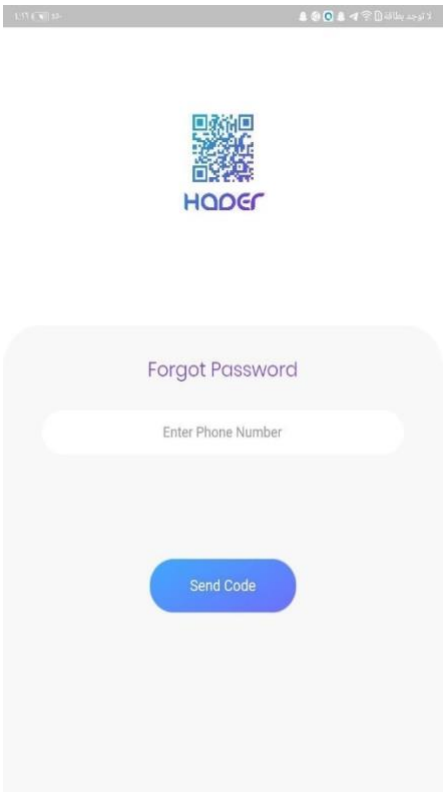


Figure 96: Forgot password interface

4. Verifying Code Interface:

In this interface the interface is ready but not its operation.

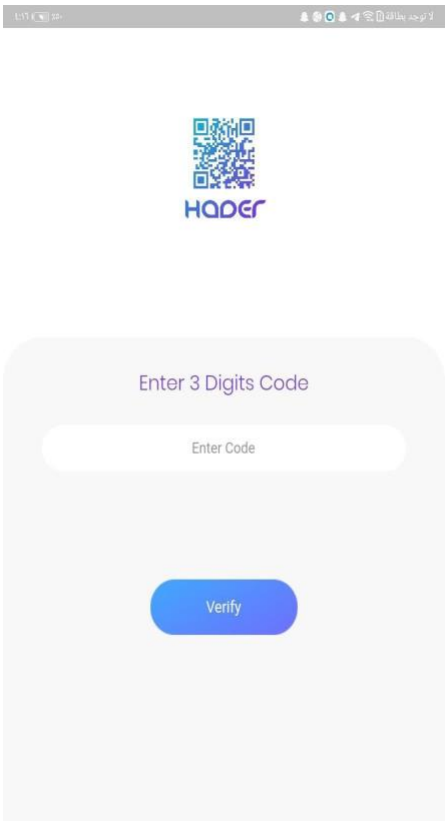


Figure 97: Verifying Code interface

5. Change Password Interface:

This interface appears after code verification and allows the user to change their password, also if the user already logged in and wants to change their password, the link will redirect to this page by visiting their profile and click on change password. Currently the interface is ready but not its operation.

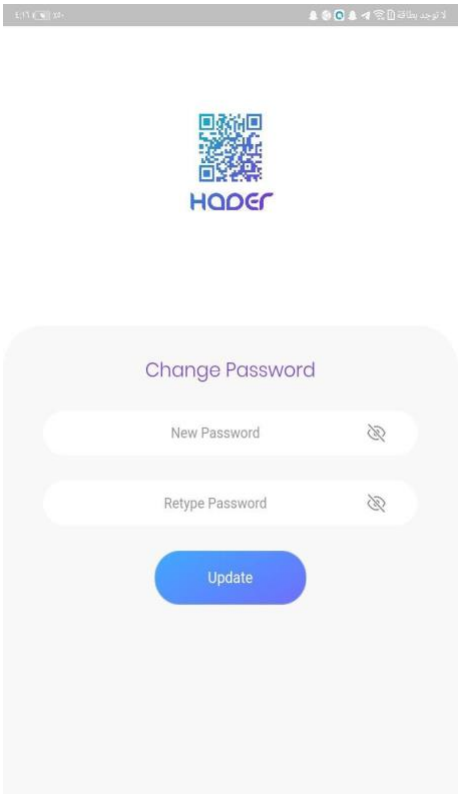


Figure 98: Change Password interface

6. Invigilator's profile:

The profile contains the invigilator's photo, their name, Position, QR Code. The invigilator can logout and change their password through this interface.

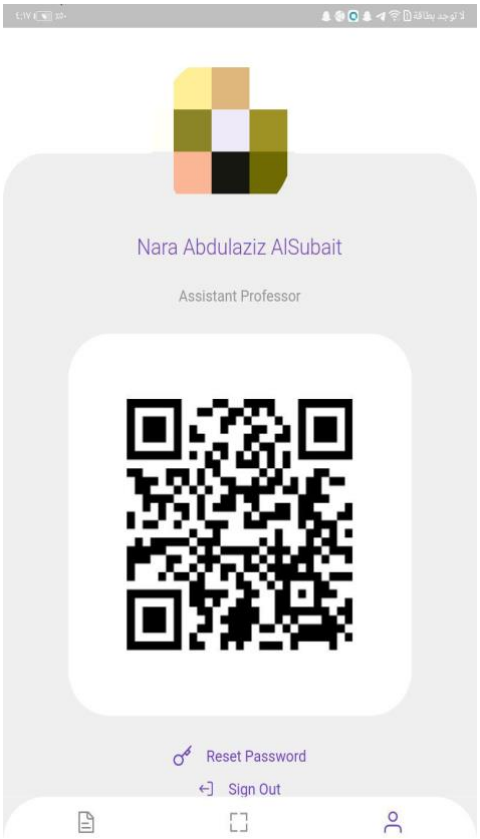


Figure 99: Invigilator's profile

5. Issues faced

This section includes the issues that the team experienced while developing the software.
Technical issues:

1- Downloading the software and its packages.
2- Collecting the code from every member.
3- Errors and finding solutions.

Managerial issues:

1- Absence of a team member and change management.
2- Having a lot of work in comparison to the time available.

CHAPTER 6. SOFTWARE TEST PLAN

1. Introduction

The purpose of the Software Test Plan (STP) is to define the scope, approach, resources, and timetable for all testing operations. The plan must specify the items to be tested, their related features, the types of testing to be performed, the persons responsible for testing, the resources and timetable needed to complete testing, and the risks associated with the plan.

1.1 Objectives

The main objectives of the STP document are:

- Establish the testing strategy that will be used throughout the testing process.
- Describe the test's approach, resources, and timetable.
- Clearly define the primary criteria and the testing procedure.
- Specify the environmental conditions under which the testing will occur.
- Specify the test cases' expected outcomes.
- Identifying errors in the HADER Application.
- Define correct techniques for resolving errors.

1.2 Testing Strategy

Testing is the procedure for analyzing and evaluating a system's performance. Additionally, it assists in identifying inconsistencies between existing and desired results and in analyzing the features of system components. The HADER application will evaluate the components using two distinct methodologies. The first is Gray Box Testing, which is a testing technique that is used when only a limited amount of information about the system's internal functionality is known [12]. The second methodology is White Box Testing, which is a type of software testing in which the internal structure, architecture, and coding of software are examined to validate the input-output flow and to improve the design, usability, and security of the software [13]. The components will next be integrated to test the system as a whole.

1.3 Scope

The purpose of this STP is to outline how the HADER application will be tested following the completion of the implementation phase. It will test the application thoroughly using component, integration, validation and verification, interface, security, performance, and regression testing.

1.4 Reference Material

The list of documents referenced in the Software Test Plan (STP) are:

- Software Project Management Plan (SPMP) for HADER Application.
- Software Requirements Specification (SRS) for HADER Application.
- Software Design Specification (SDS) for HADER Application.
- Designing the User Interface: Strategies for Effective Human-Computer Interaction.

1.5 Definitions and Acronyms

Table 25 Definitions and Acronyms

Acronyms	Definition
SDS	Software Design Specification: Description written by software designers to assist the development team by describing the data, architecture, interface, and procedural design.
SPMP	Software Project Management Plan
SRS	Software Requirement Specification: A detailed description of how the software should be produced, including both functional and non-functional requirements.
STP	Software Test Plan

1.6 Test Items

This section provides a brief description of program modules, job control procedures, user procedures, and operator procedures.

1.6.1 Program Modules

Rather than testing the entire program, small portions or modules will be tested to detect any faults or bugs in the system and to simplify the process of resolving those bugs. These modules are divided between the user interface, which will be tested against a collection of test cases, and the system's database, which will be tested against a set of queries to ensure that all of those queries are appropriately reflected in the database.

1.6.2 User Procedures

The members of the team will ensure that the system's output conforms to the expectations specified in the user manuals, such as the problem statement, requirement, and design requirements. These outcomes must live up to the expectation in terms of accuracy, completeness, and scope of the work performed.

1.6.3 Operator Procedures

The HADER application will be tested on both iOS and Android mobile phones to verify proper operation. The system must be tested in a real-world context with several concurrent users. Additionally, the system will be tested by the system's developers.

1.6.4 Features to Be Tested

The table below summarizes the HADER application's features that will be tested.

Table 26: Features to be tested

Actor	Features To Be Tested
Admin	• Manage the student's excuses. • Show the student's absent percentages. • Manage an exam. • Invigilator management.
Invigilator	• QR scanner to scan students QR codes to check-in. • Edit student's attendance status.
Instructor	• Instructors receive notification.
Student	• Upload exam excuse. • View excuse status. • Display QR code to be scanned by the exam invigilator.
common (all users)	• Login. • Logout. • Forgot password.

1.6.5 Features Not to Be Tested

The table below summarizes the features that will be excluded from testing.

Table 27: features not to be tested

Actor	Features Not to Be Tested
Admin	• Show admin profile.
Invigilator	• Show invigilator profile. • Export attendance report. • Send attendance reports via e-mail. • Show students' name list.
Instructor	• Show instructor profile.
Student	• Show student profile.

2. Approach

HADER application will feature a grey box method. This strategy is examined by testing individual components of the system in order to confirm theirthe completeness and accuracy of each. Completion Criteria: Once defects are discovered, they should be corrected immediately. Component testing is done when components perform functions as intended and with a proper outcome.

➢ Common features

Component: Log in

Table 28: Component Log in -Positive test case

Test Scenario ID	Log in -1	**Test Case ID**	Log in -1A
Pre-condition	The user must be registered in HADER.		
Login-Positive test case	Test component in Log in to HADER by ensuring that the user access the application with valid information.		
Test procedure	1. Enter Log in page. 2. Enter valid username. 3. Enter valid password. 4. Click login button.		
Expected result	Access the application when enter the correct username and password and click login button, The user will be navigated to verification interface.		
Actual result	Access the application when enter the correct username and password and click login button, The user will be navigated to verification interface.		

Table 29: Component Log in - Negative test case 1

Test Scenario ID	Log in -1	**Test Case ID**	Log in -1B
Pre-condition	The user must be registered in HADER.		
Login-Negative test case 1	Test component in Log in to HADER by ensuring that the user access the application with valid information.		
Test procedure	1. Enter Log in page. 2. Enter invalid username. 3. Enter invalid password. 4. Click login button.		
Expected result	Access the application when enter the correct username and password and click login button, The user will be navigated to verification interface.		
Actual result	An error message appears to notify the user that the username or password is incorrect.		

Table 30: Component Log in - Negative test case 2

Test Scenario ID	Login -1	Test Case ID	Login -1C
Pre-condition	The user must be registered in HADER.		
Login-Negative test case 2	Test component in Log in to HADER by ensuring that the user access the application with valid information.		
Test procedure	1. Enter Log in page. 2. Enter valid username. 3. Enter invalid valid password. 4. Click login button.		
Expected result	Access the application when enter the correct username and password and click login button, The user will be navigated to verification interface.		
Actual result	An error message appears to notify the user that the username or password is incorrect.		

Component: Logout

Table 31: Component Logout

Test Scenario ID	Logout -1	Test Case ID	Logout -1A
Pre-condition	The user must be logged in HADER.		
Logout	Test component in logout of the app.		
Test procedure	1. Click on logout button.		
Expected result	The user will sign out from the application.		
Actual result	The user will sign out from the application.		

Component: Forgot password

Table 32: Component Forgot password

Test Scenario ID	Forgot password 1	Test Case ID	Forgot password -1A
Pre-condition	The user must be registered in HADER.		
Forgot password	Test the Forgot password component in HADER.		
Test procedure	1. Enter Log in page. 2. Click on “Forgot password?” option. 3. Enter valid phone number. 4. Click “Send code” button. 5. Enter the sent 3-digit code.		

	6. Click Verify button. 7. Enter the new password. 8. Reenter the new password. 9. Click on update.
Expected result	The user password is updated.
Actual result	The user password is updated.

➢ Admin features:

Component: Manage the student's excuses

Table 33 Component: Manage the student's excuses

Test Scenario ID	MSE -1	**Test Case ID**	MSE -1A
Pre-condition	The user must be logged in HADER.		
Manage the student's excuses	The admin looks at the student's excuse and determines the status of the excuse, whether it is accepted, rejected, or suspended.		
Test procedure	1. Click on the Manage Students button from the Administration interface. 2. Choose from the list of absent students. 3. View the uploaded excuse file. 4. Change the excuse statues.		
Expected result	Student excuse statues will be updated, and confirmation message appears.		
Actual result	Student excuse statues will be updated, and confirmation message appears.		

Component: Show the student's absent percentages

Table 34 Shows the student's absent percentages

Test Scenario ID	SSAB -1	**Test Case ID**	SSAB -1A
Pre-condition	The user must be logged in HADER.		
Show the students absent percentage	Admin can view absent rate and number of absent students.		
Test procedure	View the student's absent information on the dashboard interface.		
Expected result	General statistics on the number of absent and present students and absenteeism rates will render.		
Actual result	General statistics on the number of absent and present students and absenteeism rates will render.		

Component: Invigilator management.

Table 35 Component: Invigilator management – add new.

Test Scenario ID	MI -1	**Test Case ID**	MI -1A
Pre-condition	The user must be logged in HADER.		
Invigilator management – add new	Admin can check in for the invigilator list and add new invigilators.		
Test procedure	1. Click on the Manage Invigilator button from the Administration interface. 2. View the names of the invigilators who have been added previously. 3. Press on the plus sign to add new invigilator. 4. Choose the room, invigilator, course ID, date, and time. 5. Click on Confirm Time button.		
Expected result	A new invigilator will be assigned, and new exam will be created.		
Actual result	A new invigilator will be assigned, and new exam will be created.		

Table 36 Component: Invigilator management – Edit/Delete.

Test Scenario ID	MI -1	**Test Case ID**	MI -1B
Pre-condition	The user must be logged in HADER.		
Invigilator management – Edit/Delete	Admin edits/deletes the pre-assigned invigilator.		
Test procedure	1. Click on the Manage Invigilator button from the Administration interface. 2. View the names of the invigilators who have been added previously. 3.Press on the edit/delete sign to edit/delete invigilators. 5. Click on Confirm button.		
Expected result	The invigilator will be edited or deleted.		
Actual result	The invigilator will be edited or deleted.		

Component: Manage an exam

Table 37 Component: Manage an exam- Add a new exam

Test Scenario ID	ME -1	**Test Case ID**	ME -1A
Pre-condition	The user must be logged in HADER.		
Manage an exam- add new exam	Admin manages new exam by choosing the room, invigilator, course, date, and time.		
Test procedure	1. Click on the Manage Invigilator button from the Administration interface. 2. Press on the plus sign.		

	3. Choose the room, invigilator, course ID, date, and time. 4. Click on Confirm Time button.
Expected result	A new invigilator will be assigned, and new exam will be created.
Actual result	A new invigilator will be assigned, and new exam will be created.

Table 38 Manage an exam- Edit exam

Test Scenario ID	ME -1	**Test Case ID**	ME -1B
Pre-condition	The user must be logged in HADER.		
Manage an exam- add new exam	Admin manage edits the pre-created exams, changing room, invigilator, course, date, and/or time.		
Test procedure	1. Click on the Manage Invigilator icon from the Administration interface. 2. Press on the edit sign of the desired exam. 3. Apply changes on the room, invigilator, course ID, date, and/or time information. 4. Click on Confirm Time button.		
Expected result	The exam will be edited.		
Actual result	The exam will be edited.		

➢ Student features:

Component: Upload exam excuse.

Table 39 Component: Upload exam excuse

Test Scenario ID	UEE -1	**Test Case ID**	Test -1A
Pre-condition	The user must be logged in HADER.		
Upload exam excuse	The student enters the excuse information, such as the reason, type of absence, the date and a photo or file of the excuse.		
Test procedure	1. Click on the upload an excuse icon. 2. Select excuse for absent from. 3. Select excuse type. 4. Choose the date. 5. Upload attachments.		
Expected result	The excuse will be uploaded		
Actual result	The excuse will be uploaded		

Component: View excuse status.

Table 40 Component: View excuse status

Test Scenario ID	VES -1	**Test Case ID**	Test -1A

Pre-condition	The user must be logged in HADER.
View excuse status	The status of the request for an excuse for absence is shown here.
Test procedure	1. Click on the View Excuse Status icon. 2. View the excuse status interfaces.
Expected result	View the excuse status.
Actual result	View the excuse status.

Component: Display QR code to be scanned by the exam invigilator.

Table 41 Component: Display QR code to be scanned by the exam invigilator

Test Scenario ID	DQRC -1	**Test Case ID**	Test -1A
Pre-condition	The user must be logged in HADER.		
Display QR code	Students Display QR code to be scanned by the exam invigilator		
Test procedure	1. Go to the profile interface. 2. QR code will render. 3. QR code can be scanned by the invigilator.		
Expected result	Students' QR code will render.		
Actual result	Students' QR code will render.		

➢ Invigilator features:

Component: Scan student's QR code.

Table 42 Component: Scan student's QR code

Test Scenario ID	SQRC -1	**Test Case ID**	Test -1A
Pre-condition	The user must be logged in HADER.		
Scan QR code	Scan students QR codes to check-in.		
Test procedure	1. Click on the list icon. 2. Select date. 3. Select exam. 4. Click on the scan button.		
Expected result	Students' information appears.		
Actual result	Students' information appears.		

Table 43 Component: Scan student's QR codes to check in – Negative test case

Test Scenario ID	SQRC -1	**Test Case ID**	Test -1A
Pre-condition	The user must be logged in HADER.		
Scan QR code	Scan students QR codes to check-in.		
Test procedure	1. Click on the list icon.		

	2. Select date. 3. Select exam. 4. Click on the scan button.
Expected result	Students' information appears on the screen.
Actual result	An error message appears on the screen.

Component: Edit student's attendance status.

Table 44 Component: Edit student's attendance status – Positive case test.

Test Scenario ID	ESAS -1	**Test Case ID**	Test -1A
Pre-condition	The user must be logged in HADER.		
Edit student's attendance status	Edit student's attendance status.		
Test procedure	1. Click on the list icon. 2. Select date. 3. Select exam. 4. Edit attendance status. 5. Click on save icon.		
Expected result	A success message appears.		
Actual result	A success message appears.		

Table 45 Component: Edit student's attendance status – Negative case test.

Test Scenario ID	ESAS -2	**Test Case ID**	Test -1A
Pre-condition	The user must be logged in HADER.		
Edit student's attendance status – Negative	Edit student's attendance status.		
Test procedure	1. Click on the list icon. 2. Select date. 3. Select exam. 4. Edit attendance status. 5. Click on save icon.		
Expected result	A success message appears.		
Actual result	An error message appears.		

➢ Instructor features:

Component: Receive notification.

Table 46 Component: Receive notification

Test Scenario ID	RN -1	**Test Case ID**	Test -1A
Pre-condition	The user must be logged in HADER.		
Receive notification	The instructor receives a notification about attendance lists or invigilating tasks.		
Test procedure	1. Click on the notification icon. 2. View notifications received.		
Expected result	View notifications received.		
Actual result	View notifications received.		

2.2 Integration Testing

The integration test will be performed following the grouping of unit-tested units to ensure that the design objectives, system performance, and functional requirements are met. The bottom-up approach will be used for integration testing, with units being integrated beginning with the lowest unit of the system and progressing to the upper units. This test will be repeated until the integration is complete, at which point the system will be tested as a whole.

Test approach: Grey box testing.

Module tested: Login

Table 47: Grey box testing: Login

Test No.	Task	Input	Precondition	Excepted result
1	User Login	Username (2180000001), Password (Shahad$11)	The user must be registered in the system.	The user will be navigated to the verification interface.
2	User Login	Unregistered Username (2918402001), Password (xxxx)	The user must be registered in the system.	An error message appears to notify the user that the username or password is incorrect.

3	User Login	Username (2170001466), incorrect Password (sshahad)	The user must be registered in the system	An error message appears to notify the user that the username or password is incorrect.

Module tested: Logout

Table 48: Grey box testing: Logout

Test No.	Task	Input	Precondition	Excepted result
1	User logout	-	The user must be logged in.	The user will sign out from the application

2.3 Interface Testing

Interface testing is performed to verify that the connection of various component systems is done efficiently and effectively. The main objective of this testing is to ensure that all users have no issues when using the HADER application and that the functionality works as expected, therefore the code will be tested after it is completed.

Interface testing consists of two sections:

1. InterfaceThe interface between a web server and an application server.
2. The application server and the database server connection.

The interface testing is performed to:

1. Verify that servers are performed appropriately.
2. For each inquiry done by an application, faults are managed properly, or an error message is returned.

2.4 Security Testing

The primary objective of security testing is to identify and eliminate any potential points of system failures. gray-box testing will be used to locate all potential defects in the software. The measures of confidentiality, integrity, authentication, authorization, and availability will be implemented to

ensure security. For example, in the login test case, the tester will log in with an unregistered account, an incorrect password, and no password at all to see if the security measures are effective.

Module tested: Login (authorization)

Table 49: Security testing (Authorization)

Test No.	Task	Input	Precondition	Excepted result
1	Login	Username (2180000001), Password (Shahad$11)	The user must be registered in the system.	The user will be navigated to the verification interface.

Module tested: Login (confidentiality)

Table 50: Security testing (Confidentiality)

Test No.	Task	Input	Precondition	Excepted result
1	Login	Username (2180000001), Password (**********)	The user must be registered in the system.	The password will be encrypted in asterisk symbol.

Module tested: Login (Authentication)

Table 51: Security testing (Authentication)

Test No.	Task	Input	Precondition	Excepted result
1	Login	Username (2180000001), incorrect Password (sssss)	The user must be registered in the system.	The system will be verifying the password.

2.5 Performance Testing

The purpose of the performance test is to validate that the system performs in accordance with user expectations, such as (response time, speed, availability) the application 's performance should remain constant. Users expect a response time of less than 15 seconds, and testing should make sure that it is available round-the-clock and in a variety of settings. For example, the tester will evaluate if the system is capable of handling the workload and whether or not the system produces the expected results.

2.6 Acceptance Testing

In acceptance testing, the alpha testing will be used which is a type of testing that is performed on an application near the conclusion of the development phase, when the product is nearly ready for use. is used to test the system before it is delivered to application users. It is important in that it reduces the amount of time spent on error correction and increases the level of user satisfaction before delivering the final version of the system. The system will be used by the user to ensure that the system meets the acceptance criteria for the system under consideration.

3. Pass / Fail Criteria

This section specifies the criteria which is used to determine whether to suspend, resume and approve each test.

3.1 Suspension Criteria

Suspension criteria is the criteria that will be used to decide if testing should be stopped or finished before the plan is done.

1. When an error occurs that cannot be corrected
2. When a project's deadline is missed.
3. When the assigned test resources for the test team are not available when they are required.
4. When the software is not available during scheduled testing activities.

3.2 Resumption Criteria

It is the criteria that determine whether or not the testing can be rebooted, restarted, or resumed after a suspension has been put in place. For the item to restart the test, all test cases must pass successfully, and any detected problems must be corrected.

1. When the bugs are successfully fixed, the testing team is alerted to resume testing.
2. when the contract is adjusted with the client to extend the delivery date
3. hardware and software resources are becoming available.

3.3 Approval Criteria

Approval criteria are the conditions that must be met in order to approve the outcome. The test case result must match the result description before it is considered approved. The approval process requires the user requirements to determine whether the testing result meets the acceptance criteria.

1. The result meets the quality and operational standards.
2. End-user acceptance and satisfaction with the system's operation and quality
3. Error-free application
4. Quick response time
5. Test the system to ensure that the results and quality are as expected.

4. Testing Process

4.1 Test Deliverables

The test deliverables will comprise a test report that contains test cases, a record of every test execution, and an overview of the test's current status.

4.2 Testing Tasks

There are several steps that must be completed prior to execute testing activities, including test plan, test cases, execute tests, report defects, error handling, complete test reports and manage the change.

4.3 Responsibilities

The project manager will be in charge of approving or rejecting the test plan, test cases, and the final acceptance of the whole work, while the development team will be in charge of testing all the

tasks and the testing team will be in charge of all the functional testing, use case validation testing and the performance testing.

4.4 Schedule

The schedule below explains who will perform the testing activities and how long it will take to accomplish them.

Table 52 Testing schedule

Task	Owner	Time needed
Developing the test plan	Project Manager	One week
Developing the test cases	Project Team	One week
Execute the tests	Testing Team	Two weeks
Reporting the defects	Testing Team	One week
Completing the test report	Project Team	One week

5. Environmental Requirements

This section discusses the environmental needs for completing the test process, which include hardware, software, security, tools, publication, and risks and assumptions.

5.1 Hardware

The hardware and network requirements that are needed to complete testing procedures are:

- Internet connection**.**
- Computers**.**
- Smart phones.

5.2 Software

The software requirements that are needed to complete testing procedures are:

- Android studio with flutter extensions.
- XCode.
- Firebase.
- MS word for documentation.
- Visual Studio Code.
- Mac/ Windows operating systems.

5.3 Security

HADER application will be protected through:

- Login authentication: All types of users need to be authorized.
- Firewalls.

5.4 Tools

The tools needed to apply the testing activities are:

- Microsoft word for documentation.
- Visual Studio Code.
- Android studio.
- flutter & dart.
- firebase.
- XCode.

6. Publications

The documents and publications that are required to support the testing procedures are:

- Software Project Management Plan (SPMP).
- Software Requirements Specification (SRS).
- System Design Specification (SDS).

7. Risks and Assumptions

Table 53 Risks and Mitigation actions

Risks	Mitigation action
Schedule constraints	Prioritize the application's functions so that the most important ones are tested first.
Lack of experience	Searching and learning.
Test Resources Availability	The system will be tested with the available test resources or search for other available test resources.
Loss of data.	Recover from backup copies.

7.1 Change Management Procedures

To manage any changes that might happen:

- First the supervisor, team leader and the rest of the members need to be notified about the change needed to evaluate it.
- The supervisor, team leader and the rest of the team members must approve.
- After approval change will be implemented based on the approved modifications.
- Any disagreement or rejection from the supervisor, team leader or any team member will prevent the change from implementation.
- If any change or modification was applied, it should be recorded and documented.

7.2 Plan Approvals

Project Team:

Table 54: Project team plan approval

Team Member's Name	Signature	Date
Shahad Saud AlTamimi	Shahad	2/8/2022
Lamia Abdullah AlHammad	Lamia	2/8/2022
Danah Nasheer AlDosari	Danah	2/8/2022
Deemah Adel AlRushood	Deemah	2/8/2022

Project Supervisor:

Table 55 Project supervisor plan approval

Name	Signature	Date
Ms. Asiya Abdus Salam		2/8/2022

8. What Have We Learned?

HADER team members chose flutter and dart languages and know how powerful they are in the way that it is a cross-platform framework, a piece of code can work on both android and iOS operating systems, members have learned more via online courses and videos to complete the project. In addition, Hader team members have successfully developed time management skill and being more flexible and more corporate together. More importantly, Hader team members learned how to motivate each other to make the project successful. In addition, team members have learned more about usability for users and how to make the application simple to use by taking an overview of the literature reviews and studied how to optimize the application performance successfully .

9. Dissemination Plan

This section shows the dissemination plan including submission type, status, date of submission and targeted event as shown in Table 58.

Table 56 Dissemination plan

Submission type	Research paper
Status	Submitted (Pending status)
Date of submission	May 5th, 2022
Targeted event	1. Journal (International Journal of Emerging Technologies in Learning iJET).

10. Conclusion

In conclusion, the project is a mobile application that will make the exam attendance process easy and short by using a QR Code technology to take attendance faster in CCSIT at IAU. The application serves to improve exam attendance by providing invigilators and instructors the ability to scan students' QR codes, allowing the Academic Affairs Unit to automatically record the student's attendance. In addition, students can upload their excuses, which the Academic Affairs Unit will either approve or decline. Finally, all of this will culminate in a high-security, high-authentication application built for verification and validation the application guarantees to deliver benefits to CCSIT.

CHAPTER 7. SOFTWARE TEST REPORT

1. Introduction

1.1 Document overview

This document is the software test report of the testing phase of Hader Application project. It contains the results of tests, which were executed during the testing phase of the project.

1.2 Abbreviations and Glossary

1.2.1 Abbreviations

Table 57: Abbrevations List

Abbreviations	Defenitions
NOK	Not OK
NR	Not RUN
POK	Partial OK
NC	Not completed

1.2.2 Glossary

Table 58: Glossary List

Glossary	Defenitions
Test case	Test case is a set of procedures, conditions and variables that a laboratory uses to specify whether the system meets the requirements and is working properly.
Test log	it is a category of the testing artifacts done during the testing process. It gives a detailed summary of the comprehensive test run and shows the tests that were passed and failed, details and information about testing processes.
Partial OK	Means that the output is partially compatible with the expected result.

1.3 References

1.3.1 Project References

Table 59: Project References

#	Document Identifier	Document Title
[R1]	1	STP
[R2]	2	SRS
[R3]	3	SPMP
[R4]	4	SDS

1.3.2 Standard and Reglatory References

Table 60: Standard and Reglatory References

#	Document Identifier	Document Title
[STD1]	1	IEEE SPMP
[STD2]	2	IEEE SRS
[STD3]	3	IEEE SDS

1.4 Conventions

Everyone on the team is responsible for completing the work efficiently deliver it as scheduled. M oreover, to achieve the legal aspect, the data collected from all users that contains sensitive information will be secured and the information wont be shared to any third parties without the user's information and conformation.

2. Overview of Tese Results

2.1 Tests log

Hader Application (Version 1) was tested on a mobile device that works on android operating system, from 14/04/2022 to 13/05/2022 tests of the test phase were executed.

Testers where:

- Danah ALDosari
- Deemah AlRushood
- Lamia Alhammad
- Shahad AlTamimi

2.2 Rationale for decision

After executing a test, the decision is defined according to the following rules:

- **OK:** The test sheet is set to "OK" state when all steps are in "OK" state. The real result is compliant to the expected result.
- **NOK:** The test sheet is set to "NOK" state when all steps of the test are set to "NOK" state or when the result of a step differs from the expected result.
- **Partial OK:** The test sheet is set to "Partial OK" state when at least one step of the test is set to "NOK" state or when the result of a step is partially compliant to the expected result.
- **NOT RUN:** Default state of a test sheet not yet executed.
- **NOT COMPLETED:** The test sheet is set to "Not Completed" state when at least one step of the test is set "Not Run" state.

3. Detailed Tests Results

This section will cover the common, admin, invigilator, instructor and student features.

3.1 Test case result

- Common features

Table 61: Test Log in

Test ID	Log in -1a	Comment	Decision
Test description	This function allows the user to log in to their account in HADER be entering their email and password.	The user with valid account information can log in to HADER.	OK
Initial conditions	The user must be registered in HADER.		OK
Test inputs	Email and password.		OK
Test outputs	The user accesses the application successfully.		OK
Assumptions and constraints	• Valid user information. • No missing fields. • Internet connection.		OK
Expected results and criteria			OK
Test procedure			
Step number	Operator actions	Expected result and evaluation	Result
1	Open the application	The user is directed to log in page	OK
2	Enter log in page	Log in page is rendered	OK
3	Enter valid email	Validity of the email be checked	OK
4	Enter valid password	Validity of the password be checked	OK
5	Click login button	The user accesses the application and be navigated to the profile interface.	OK
Check condition			
1	Repeat the steps above with wrong email or password	Validity of the email and password be checked	OK

Table 62: Test Log out

Test ID	Logout -1	Comment	Decision
Test description	This function allows the user the user to sign out of HADER.		OK
Initial conditions	The user must be logged in HADER.		OK
Test inputs	n/a		OK
Test outputs	User logout out from the application successfully.		OK
Assumptions and constraints	• user must be logged in. • Internet connection.		OK
Expected results and criteria	The user will sign out from the application.		OK
Test procedure			
Step number	Operator actions	Expected result and evaluation	Result
1	Click on logout button.	user successfully signed out of the application.	OK
Check condition			
1	n/a		

Table 63: Test Forgot password

Test ID	Forgot password -1	Comment	Decision
Test description	Users can reset their passwords by, clicking on forgot password and entering their		OK
Initial conditions	The user must be registered in HADER.		OK
Test inputs	• Email address • Verification code • New password	Verification code will be sent to the user's email.	OK
Test outputs	The user password is updated.		OK
Assumptions and constraints	• user must enter a valid email address. • User must enter a valid verification code. • No missing fields. • Internet connection.		OK

Expected results and criteria	The user password is updated.		OK
Test procedure			
Step number	Operator actions	Expected result and evaluation	Result
1	Enter log in page	log in page renders	OK
2	Click on “forgot password?” Option	Forgot password renders	OK
3	Enter valid email address	Validity of the entered email is checked	OK
4	Click “send code” button	A verification link is sent to the user email	OK
7	Enter the new password.	Validity of the entered password is checked	OK
8	Reenter the new password.	Check if the reentered password matches the new password	OK
9	Click on update	The user password is updated.	OK
Check condition			
1	Repeat the steps above with wrong email or password	Validity of the email and password be checked	OK

- Admin features

Table 64: Test Manage the student’s excuses

Test ID	MSE -1	Comment	Decision
Test description	This function allows the admin to view the student’s excuses and determines the status of the excuse, whether it is accepted, rejected, or suspended.		OK
Initial conditions	The user must be logged in HADER.		OK
Test inputs	n/a		OK
Test outputs	Successful pop-up message will render		OK
Assumptions and constraints	• not all types of files are accepted. • Internet connection.		OK
Expected results and criteria	Student excuse statues will be updated, and		OK

	confirmation message appears.		
Test procedure			
Step number	Operator actions	Expected result and evaluation	Result
1	Click on the manage students' button from the administration page	the user is directed to view excuses page	OK
2	Choose from the list of absent students	the chosen student excuse is rendered	OK
3	View the uploaded excuse file	excuse file is shown	OK
4	change the excuse statues	statues is changed.	OK
5	save changes	Changed statues will be saved	OK
Check condition			
1	Check the students excuse status after step 4	statues is changed.	OK

Table 65: Test Show the student's absent percentages

Test ID	SSAB -1	Comment	Decision
Test description	In this feature the admin can view absent percentages for each student from the highest percentage to the lowest percentage.		OK
Initial conditions	The user must be logged in HADER.		OK
Test inputs	n/a		OK
Test outputs	General statistics of absents and presents of the students will render.		OK
Assumptions and constraints	• Internet connection.		OK
Expected results and criteria	Admin can view absent rate and number of absent students.		OK
Test procedure			
Step number	Operator actions	Expected result and evaluation	Result
	Click on the dashboard button	Dashboard page renders	OK

1	View the student's absent information on the dashboard page.	Information renders	OK
Check condition			
1	n/a		OK

Table 66: Test Invigilator management - add new invigilator

Test ID	MI -1	Comment	Decision
Test description	Admin can check in for the invigilator list and add new invigilators.		OK
Initial conditions	The user must be logged in HADER.		OK
Test inputs	• Exam room no. • Invigilator name • course ID • Subject name • exam date • exam start time • exam end time	Exam room no, Invigilator name and course ID is in a drop list and the data is retrieved from the database.	OK
Test outputs	Successful pop-up message will render		OK
Assumptions and constraints	• cannot create new exam date without valid input data • Valid user information. • No missing fields. • Internet connection.		OK
Expected results and criteria	A new invigilator will be assigned, and new exam will be created.		OK
Test procedure			
Step number	Operator actions	Expected result and evaluation	Result
1	Click on the Manage Invigilator button from the Administration page	Manage Invigilator page renders	OK
2	View the names of the invigilators who have been added previously	The names of the invigilators who have been added previously renders	OK
3	Press on the plus sign to add new invigilator	Add exam page renders	OK

4	Click on submit button.	A new invigilator will be assigned, and new exam will be created.	OK
Check condition			
1	Check the Manage Invigilator page for the new exam added	New exam is shown on the Manage Invigilator page	OK

Table 67: Test Invigilator management - Edit/Delete.

Test ID	MI -2	Comment	Decision
Test description	Admin edits/deletes the pre-assigned invigilator.		OK
Initial conditions	The user must be logged in HADER.		OK
Test inputs	• Exam room no. • Invigilator name • course ID • Subject name • exam date • exam start time • exam end time	Exam room no, Invigilator name and course ID is in a drop list and the data is retrieved from the database.	OK
Test outputs	Successful pop-up message will render		OK
Assumptions and constraints	• cannot create new exam date without valid input data • Valid user information. • No missing fields. • Internet connection.		OK
Expected results and criteria	A new invigilator will be assigned, and new exam will be created.		OK
Test procedure			
Step number	Operator actions	Expected result and evaluation	Result
1	Click on the Manage Invigilator button from the Administration page	Manage Invigilator page renders	OK
2	View the names of the invigilators who have been added previously	The names of the invigilators who have been added previously renders	OK
3	Press on the edit/delete sign to edit/delete invigilators.	If edit is clicked invigilator information renders.	OK

		If delete is clicked the invigilator will be deleted	
4	Edit the invigilator information	invigilator Information will be changed	
5	Click on submit button.	Save the Changes	OK
Check condition			
1	Check the invigilators list on the Manage Invigilator page	invigilator information is changed/deleted on the Manage Invigilator page	OK

Table 68: Test Manage an exam- add new exam

Test ID	ME -1	Comment	Decision
Test description	Admin manages new exam by choosing the room, invigilator, course, date, and time.		OK
Initial conditions	The user must be logged in HADER.		OK
Test inputs	• Exam room no. • Invigilator name • course ID • Subject name • exam date • exam start time • exam end time	Exam room no, Invigilator name and course ID is in a drop list and the data is retrieved from the database.	OK
Test outputs	Successful pop-up message will render		OK
Assumptions and constraints	• cannot create new exam date without valid input data • Valid user information. • No missing fields. • Internet connection.		OK
Expected results and criteria	A new invigilator will be assigned, and new exam will be created.		OK
Test procedure			
Step number	Operator actions	Expected result and evaluation	Result
1	Click on the Manage Invigilator button from the Administration page	Manage Invigilator page renders	OK

2	View the names of the invigilators who have been added previously	The names of the invigilators who have been added previously renders	OK
3	Press on the plus sign to add new exam	Add exam page renders	OK
4	Click on submit button.	A new invigilator will be assigned, and new exam will be created.	OK
Check condition			
1	Check the Manage Invigilator page for the new Invigilator added	New exam is shown on the Manage Invigilator page	OK

Table 69: Test Manage an exam- edit exam

Test ID	ME -2	Comment	Decision
Test description	Admin edits/deletes the pre-assigned exams.		OK
Initial conditions	The user must be logged in HADER.		OK
Test inputs	• Exam room no. • Invigilator name • course ID • Subject name • exam date • exam start time • exam end time	Exam room no, Invigilator name and course ID is in a drop list and the data is retrieved from the database.	OK
Test outputs	Successful pop-up message will render		OK
Assumptions and constraints	• cannot create new exam date without valid input data • Valid user information. • No missing fields. • Internet connection.		OK
Expected results and criteria	A new invigilator will be assigned, and new exam will be created.		OK
Test procedure			
Step number	Operator actions	Expected result and evaluation	Result

1	Click on the Manage Invigilator button from the Administration page	Manage Invigilator page renders	OK
2	View the names of the invigilators who have been added previously	The names of the invigilators who have been added previously renders	OK
3	Press on the edit/delete sign to edit/delete exam.	If edit is clicked exam information renders. If delete is clicked the invigilator will be deleted	OK
4	Edit the exam information	exam Information will be changed	OK
5	Click on submit button.	Save the Changes	OK
Check condition			
1	Check the invigilators list on the Manage Invigilator page	exam information is changed/deleted on the Manage Invigilator page	OK

- Invigilator features

Table 70: Test Scan students QR code

Test ID	SQRC -1	Comment	Decision
Test description	In this feature the invigilator Scan students QR codes to check-in the exam attendance		OK
Initial conditions	The user must be logged in HADER.		OK
Test inputs	• Student QR code		OK
Test outputs	Students' information appears on a pop-up message.		OK
Assumptions and constraints	• Only students on the attendances list can be checked-in. • Internet connection.		OK
Expected results and criteria	A student is checked-in as present at the exam.		OK
Test procedure			
Step number	Operator actions	Expected result and evaluation	Result
1	Click on the list icon	A select date page renders	OK

2	Select date	A list of all the exam's that day for this invigilator will render	OK
3	Select exam	An attendances list will render	OK
4	Click on the scan button	Scan the student QR code	OK
5	Click on the save button	Changes is saved	OK
Check condition 1			
1	Check the attendance list after scanning a student a ✔ icon will show next to their name.	A student is checked-in as present at the exam.	OK

Table 71: Test Edit student's attendance status

Test ID	ESAS -1	Comment	Decision
Test description	In this feature the invigilator can Edit student's attendance status		OK
Initial conditions	The user must be logged in HADER.		OK
Test inputs	• n/a		OK
Test outputs	Students' attendance status edited.		OK
Assumptions and constraints	• Only students on the attendances list can be edited. • Internet connection.		OK
Expected results and criteria	A student is checked-in as present at the exam.		OK
Test procedure			
Step number	Operator actions	Expected result and evaluation	Result
1	Click on the list icon	A select date page renders	OK
2	Select date	A list of all the exam's that day for this invigilator will render	OK
3	Select exam	An attendances list will render	OK
4	Click on Edit attendance status icon.	Edit option is activated.	OK
5	Click on the save button	Changes is saved	OK
Check condition 1			
1	Check the attendance list after scanning a student a	A student is checked-in as present at the exam.	OK

	✔ icon will show next to their name.		

➢ Instructor features

Table 72: Test Receive notification

Test ID	RN -1	Comment	Decision
Test description	In this feature the instructor receives a notification about attendance lists or invigilating tasks.		OK
Initial conditions	The user must be logged in HADER.		OK
Test inputs	n/a		OK
Test outputs	View notifications received		OK
Assumptions and constraints	• Internet connection		OK
Expected results and criteria	View notifications received		OK
Test procedure			
Step number	Operator actions	Expected result and evaluation	Result
1	Click on the notification icon	Notifications page renders	OK
2	View notifications received	Notifications are shown in the notifications page	OK
Check condition 1			
1	n/a		OK

➢ Student features

Table 73: Test Upload exam excuse

Test ID	UEE -1	Comment	Decision
Test description	In this feature the student enters the excuse information, such as the reason, type of absence, the date and a photo or file of the excuse		OK
Initial conditions	The user must be logged in HADER		OK
Test inputs	• excuse for absent from (Midterm, Final) • Excuse type • Start date • End date • Attachment file		OK
Test outputs	Successful pop-up message will render		OK
Assumptions and constraints	• No missing fields • Internet connection		OK
Expected results and criteria	The excuse will be uploaded		OK
Test procedure			
Step number	Operator actions	Expected result and evaluation	Result
1	Click on the upload an excuse button	Upload an excuse page renders	OK
2	Select excuse type	type will be chosen	OK
3	Choose the date	Date will be chosen	OK
4	Upload attachments	User will be directed to there files and can choose one to upload	OK
5	Click on submit button.	Save the Changes	
Check condition 1			
1	Click on view status button to view the uploaded excuses	The uploaded excuse and the statues is shown	OK

Table 74: Test View excuse status

Test ID	VES -1	Comment	Decision
Test description	In this feature the student can view the status of the request for an excuse for absence is shown here		OK
Initial conditions	The user must be logged in HADER		OK
Test inputs	• n/a		OK
Test outputs	Table with the uploaded excuses and their statues rendered		OK
Assumptions and constraints	• Internet connection		OK
Expected results and criteria	view the excuses status table.		OK
Test procedure			
Step number	Operator actions	Expected result and evaluation	Result
1	Click on the View excuse button	View excuse page renders	OK
Check condition 1			
1	n/a		OK

Table 75: Test Display QR code

Test ID	DQRC -1	Comment	Decision
Test description	In this feature the student Students Display QR code to be scanned by the exam invigilator		OK
Initial conditions	The user must be logged in HADER		OK
Test inputs	• n/a		OK
Test outputs	Students' QR code will render.		OK
Assumptions and constraints	• Internet connection.		OK
Expected results and criteria	Students' QR code will render.		OK
Test procedure			

Step number	Operator actions	Expected result and evaluation	Result
1	Log in HADER app	Profile page render with the QR code	OK
Check condition 1			
1	Log in with a student account to cheek the QR functionality	QR code is shown on the profile page	OK

CHAPTER 9. CONCLUSION

1. Conclusion

Exam attendance is one of the tasks that should be automated in CCSIT. The traditional method of taking exam attendance is inefficient, requires additional effort, creates disruption in exam rooms, and is inaccurate. The role of HADER is to improve the quality of exam attendance at CCSIT by enabling the academic affairs unit to manage attendance and exams, allowing invigilators and instructors to scan the student's QR code, and automatically recording the student's attendance in the Academic Affairs Unit's databases. Additionally, students can upload their excuses, which will be approved or denied by the Academic Affairs Unit. Finally, the application will be built with high security, and high authentication techniques to verify and validate the whole attendance process.

This chapter demonstrates the following things:

- The findings and the contributions of the system.
- The system limitations.
- The lessons learned and recommendations for future work.

2. Findings and Contributions

HADER application allows CCSIT to automate exam processes with a single application. This application has several aims that will be delivered with it. The first and most important aim is to reduce disturbance in the exam rooms. Second, by consolidating all exam-related information into a single platform, the IAU, CCSIT division will be able to facilitate and speed up exam attendance by utilizing modern technology at the IAU, CCSIT division. Thirdly, to make the process of submitting and responding to absence and late excuses easier. Furthermore, HADER Application is a cutting-edge and unique solution in CCSIT and IAU. Finally, the project components were delivered on time. Finally, all work was completed while respecting the cooperative role of the team members, which was driven in this manner to complete the project.

3. Limitations

The following limitations are considered in HADER exam attendance system:

1. The system will not work if there is no internet connection.
2. Due to time and capability constraints, the system will only allow the invigilator and instructors if assigned as invigilators to take students attendance, the admin is not a party in this process.
3. The system will not validate or authenticate if the student has two mobile phones and checked in twice in the same exam hall.
4. The system is not currently integrated with IAU/CCSIT system.
5. Due to time limitation, there is no desktop solution, and the system will be designed for mobile use only.

4. Lessons Learned

This project helped the team members to learn many things, such as:

1. HADER team members have learned from literature reviews collected about what are the different authentication methods along with their different accuracy results.
2. HADER team members took an overview of the literature reviews and studied how we can optimize HADER application performance.

3. HADER team members have learned and improved technical writing and research skills by writing a literature review.
4. HADER team members have taken CO-OP experience into this project by choosing the correct platform to build the application by using a cross-platform application development framework to allow us save time by coding a single codebase to build hader application to be used on two platforms that are Android and iOS.
5. The team now have a better understanding of the risks, constraints, time, and resources required to complete the project. Also, gained more skills in how to best utilize the available resources.
6. Team members learned how to plan, identify their priorities, and determine their future work direction.
7. The team learned how to deal with change management during the implementation phase.

5. Recommendations for future works

1. Design and publish the desktop version for the system to be used by all users.
2. Use IAU database to integrate the solution to be used by all IAU collages and facilities.
3. Adding location access and GPS feature to notify students and invigilators by sending a notification that your exam is in this hall if the student pass by the exam hall.

References

1) Unified National Platform. (2021, 11 24). *Digital Transformation*. Retrieved from Unified National Platform: https://www.my.gov.sa/wps/portal/snp/aboutksa/digitaltransformation

2) K. J. Somaiya, "SSRN," Computer Engineering Department, 8 4 2019. [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3370769. [Accessed 3 9 2021].

3) Mukhamad, T.N., dan Diana, L.F Sistem Absensi Asisten Dosen Menggunakan QR Code Scanner Berbasis Android Pad Program Studi Sistem Informasi Universitas Muria Kudus. ISBN: 978-602-1-1180-21-1 - Prosiding SNATIF Ke- 2 Tahun 2015. Fakultas Teknik-Universitas Muria Kudus.

4) S. Chennattu, A. Kelkar, A. Anthony, and S. Nagdeote, "Portable Biometric Attendance System Using IOT," 2019 4th International Conference on Information Systems and Computer Networks (ISCON), 2019, pp. 245-249, doi: 10.1109/ISCON47742.2019.9036275.

5) M. S. Akbar, P. Sarker, A. T. Mansoor, A. M. Al Ashray and J. Uddin, "Face Recognition and RFID Verified Attendance System," 2018 International Conference on Computing, Electronics & Communications Engineering (iCCECE), 2018, pp. 168-172, doi: 10.1109/iCCECOME.2018.8658705.

6) Patel and D. S. P. R, “Development of a Student Attendance Management System Using RFID and Face Recognition: A Review,” International Journal of Advance Research in Computer Science and Management Studies, vol. 2, no. 8, Aug. 2014.

7) H. F. Hanafi, C. M. Meyer, M. H. Abd Wahab, and H. A. Kader, “Peas portable examination attendance - core,” core, 2009. [Online]. Available: https://core.ac.uk/download/pdf/12007449.pdf. [Accessed: 29-Sep-2021].

8) Fadi Masalha and Nael Hirzallah (2014). A Students Attendance System Using QR Code, *International Journal of Advanced Computer Science and Applications*, *5*(3).

9) A review of student attendance system using near-field ... (n.d.). Retrieved November 22, 2021, from https://www.researchgate.net/publication/300791064_A_Review_of_Student_Attendance_System_Using_Near-Field_Communication_NFC_Technology.

10) Mobile barcode based examination attendance system. (n.d.). Retrieved November 22, 2021, from https://www.researchgate.net/profile/Rehman-Khan-3/publication/333044740_Mobile_Barcode_Based_Examination_Attendance_System/links

/5cd932e192851c4eab9a3922/Mobile-Barcode-Based-Examination-Attendance-System.pdf.

11) examination attendance system. (n.d.). Retrieved November 22, 2021, from https://www.researchgate.net/profile/Rehman-Khan-3/publication/333044740_Mobile_Barcode_Based_Examination_Attendance_System/links/5cd932e192851c4eab9a3922/Mobile-Barcode-Based-Examination-Attendance-System.pdf.

12) Grey Box Testing?, W. is. (2022). *What is Grey Box Testing?* Tutorials Point. Retrieved April 14, 2022, from https://www.tutorialspoint.com/software_testing_dictionary/grey_box_testing.htm

13) Hamilton, T. (2022, February 19). *What is white box testing? techniques, example & types*. Guru99. Retrieved April 14, 2022, from https://www.guru99.com/white-box-testing.html

APPENDICES

Appendix A

Interview Questions

These questions were asked to CCSIT Exam committee during the interview

1- How does the administration contact with instructors to inform them about exam room planning?
2- How does the admission inform instructors about accepting and declining an exam excuse?
3- If you could develop the current use the process what changes will you make?
4- How many days/ weeks does the exam room planning take? And who is involved in certain tasks?
5- If we involved late students and late instructors in the system while attending the exam, will that be beneficial? If yes, what time a student is considered late? and the instructor is considered late?
6-Describe the outgoing process of taking exam attendance?
7- What are the common problems faced in this process?
8- If a software is available for attendance how would've benefit from it?
9- How long does taking exam attendance to take?
10- what is the process of dealing with late or absent students?
11- If there is a system or an app for easier exam attendance, will you use it?
12- What feature would be helpful for you?
13- would you like a system, website, or app?
14- How does the administration contact instructors to inform them about exam room planning?
15- How are the exam Invigilators distributed on exam days and classes?
16- How are the exam monitors distributed on days and classes?
17- who developed My IAU app?

Appendix B

Evaluation

The tables below contain all suggestions and comments from the reviewers and responses to each comment:

Evaluator: Ms. Elham Shaflout

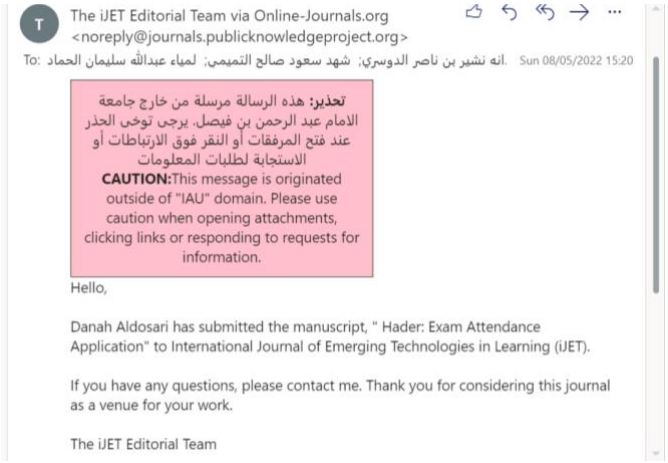
Evaluator's Comments:	Team's Response:
1. **Abstract:** needs improvement in the style of writing. there is no links between sentences. also, some sentences need rephrasing into using scientific words.	Noted and updated.
2. **4BACKGROUND AND LITERATURE REVIEW:** have you considered the existing apps for attendance tracking? they are very similar to your idea, and they are free to install by anyone and use. I found many apps that apply similar concept using fingerprint, cloud, barcode and some of them gives you options to choose how to take attendance. What your system would deliver compared to those apps?	We have added more literature reviews.
3. **1.7 SCOPE/ LIMITATION OF THE STUDY:** have you considered if the system would be integrated with the university system or not? and limitations was not mentioned here.	Noted and updated.
15. 1.9 PROJECT ORGANIZATION: I'm not sure if this is what should be written in this section.	Table has been changed to bullet point and the remaining was kept the same.
16. 2.6.1 Process Model: The model presented in the graph is not a waterfall model, this is a hybrid model. you need to build enough understanding of the difference between waterfall/ Agile/ Hybrid models. Be specific.	The mistake was in the arrows as it was going down to the next phase and another arrow that is looping to every phase. The arrow that was looping has been deleted to follow the waterfall process model.
17. 2.6.3 Project Infrastructure: which version of JavaScript? and what about other component of the software (e.g., backend)? do you have an initial design of the software architecture? if yes Then provide it here.	We have updated and changed few of the tools and infrastructure we are using and according to that JavaScript is no longer needed.

Evaluator: Dr. Rami Mohammad

Evaluator's Comments:	Team Response:
15. **Introduction:** needs more information just add up little background purpose scope etc. (At least make it one page) secondly intext citation must be provided.	Noted and updated.
16. **4.BACKGROUND AND LITERATURE REVIEW:** Literature study needs to be further expanded as only 6 previous studies are not enough. Also, the LR should be rewritten so that it reflects the pros and cons of each reviewed technique.	We have rewritten few literature reviews and we have added more literature reviews.
17. **1.5 JUSTIFICATION**: Justification must be shown by quoting the references of the literature or some proved stats or data etc.	Noted and updated.
18. **2.6.1 Process Model:** Revise the formatting under this title: "Process Model"	Noted and updated.
19. **Over all comments:** The project idea seems interesting. Usability, user security, and data flow management are important for the project my suggestion is to focus on these points as well. Lastly, I would like to encourage to go for the research paper as we have one specific rubric for the research this time.	Considered.

Appendix C

Publication of research paper

Submission type	Research paper (Journal)
Status	Submitted
Date of submission	May 5th, 2022
Targeted event	1. Journal (International Journal of Emerging Technologies in Learning iJET).
Screenshot	

Appendix D

Hader Team has received an acceptance to participate in Munshaat Program:

Ezaz Mohammed Saad Al-dahasi
Fri 29/04/2022 23:13

عزيزي المتقدم، عزيزتي المتقدمة

نبارك قبول مشروعك في برنامج المعسكرات الريادية الجامعية
نتمنى لفريقكم كل التوفيق والنجاح

دعنا نوضح لك الآتي:

- سيتم تقديم البرنامج في تاريخ 16 مايو 2022 بدلاً عن 8 مايو 2022 .
- سيتم تقديم البرنامج على مدار أسبوعين، يبدأ الأسبوع الأول 16 مايو 2022 حضورياً في مركز دعم منشآت في مدينة الخبر والاسبوع الثاني من البرنامج سيتم استكماله افتراضياً.
- سيتم توضيح بقية تفاصيل البرنامج لاحقاً بمشيئة الله.

يرجى التأكيد بالموافقة من أحد أفراد المجموعة على استمرارية الرغبة بمشاركة المشروع في المعسكر بالرد على هذه الرسالة
بـ (نؤكد استمرارية مشاركة (اسم المشروع) ومناسبة آلية تقديم البرنامج لفريق المشروع)

رئيسة قسم الشراكات والمبادرات بوحدة ريادة الأعمال
وكالة البحث العلمي والابتكار
م. عزاز محمد الدهاسي

Appendix E

Hader code *NOTE: due to the huge size of the app the document could't contain all of the source code.

Auth code

File - /Users/lulu awesome/Desktop/hader-master/lib/auth/authentication.dart

```dart
import 'dart:convert';

import 'package:email_auth/email_auth.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:http/http.dart' as http;

class AuthenticationHelper {
  final FirebaseAuth _auth = FirebaseAuth.instance;
  final _firebaseAuth = FirebaseAuth.instance.currentUser;
  EmailAuth emailAuth = new EmailAuth(sessionName: "Sample session");
  get user => _auth.currentUser;

  //SIGN UP METHOD
  // Future signUp({required String email, required String password}) async {
  //   try {
  //     await _auth.createUserWithEmailAndPassword(
  //       email: email,
  //       password: password,
  //     );
  //     return null;
  //   } on FirebaseAuthException catch (e) {
  //     return e.message;
  //   }
  // }

  //SIGN IN METHOD
  Future signIn({required String email, required String password}) async {
    try {
      await _auth.signInWithEmailAndPassword(email: email, password: password);
      return null;
    } on FirebaseAuthException catch (e) {
      return e.message;
    }
  }

```

Page 1 of 3

File - /Users/luluawesome/Desktop/hader-master/lib/auth/authentication.dart

```dart
  //SIGN OUT METHOD
  Future signOut() async {
    await _auth.signOut();

    print('signout');
  }

  Future sendOtp({required String email}) async {
    await emailAuth.sendOtp(recipientMail: email);
  }

  Future verifyOTP(String email, String otp) async
   {
    await emailAuth.validateOtp(recipientMail:
email, userOtp: otp);
  }

  Future changePassword(String newPassword, String
token) async {
    // SharedPreferences sharedPreferences = await
SharedPreferences.getInstance();
    // _token = sharedPreferences.getString("token
");
    final url =
        'https://identitytoolkit.googleapis.com/v1/
accounts:update?key=""';
    try {
      await http.post(
        Uri(host: url),
        body: json.encode(
          {
            'idToken': token,
            'password': newPassword,
            'returnSecureToken': true,
          },
        ),
      );
    } catch (error) {
      throw error;
    }
  }
```

File - /Users/lulu awesome/Desktop/hader-master/lib/auth/authentication.dart

```
71
72   Future updatePassword(String password) async {
73     await _firebaseAuth?.updatePassword(password);
74   }
75
76   Future resetPassword(String email) async {
77     await _auth.sendPasswordResetEmail(email:
   email);
78   }
79 }
80
```

Constants

File - /Users/luluawesome/Desktop/hader-master/lib/constants/colors.dart

```dart
import 'package:flutter/material.dart';

const color_1 = Color(0xffFFFFFF);
const color_2 = Color(0xffF8F8F8);
const color_3 = Color(0xff6C3EE3);
const color_4 = Color(0xff969696);
const color_5 = Color(0xffEFEFEF);
const color_6 = Color(0xffCDC0F0);
```

Page 1 of 1

File - /Users/tuluawesome/Desktop/hader-master/lib/constants/validators.dart

```dart

import 'package:flutter/material.dart';
class Validator {
  Validator();

  String? email(String? value) {
    String pattern = r'^[a-zA-Z0-9.]+@[a-zA-Z0-9]+\.[a-zA-Z]+';
    RegExp regex = RegExp(pattern);
    if (!regex.hasMatch(value!.trim()))
      return 'validator.email';
    else
      return null;
  }

  String? password(String? value) {
    String pattern = r'^.{6,}$';
    RegExp regex = RegExp(pattern);
    if (!regex.hasMatch(value!.trim()))
      return 'validator.password';
    else
      return null;
  }

  String? name(String? value) {
    String pattern = r"^[a-zA-Z]+(([',. -][a-zA-Z])?[a-zA-Z]*)*$";
    RegExp regex = RegExp(pattern);
    if (!regex.hasMatch(value!.trim()))
      return 'validator.name';
    else
      return null;
  }

  String? number(String? value) {
    String pattern = r'^\D?(\d{3})\D?\D?(\d{3})\D?(\d{4})$';
    RegExp regex = RegExp(pattern);
    if (!regex.hasMatch(value!.trim()))
      return 'validator.number';
    else
```


File - /Users/tuluawesome/Desktop/hader-master/lib/constants/validators.dart

```dart
      return null;
  }

  String? amount(String? value) {
    String pattern = r'^\d+$';
    RegExp regex = RegExp(pattern);
    if (!regex.hasMatch(value!.trim()))
      return 'validator.amount';
    else
      return null;
  }

  String? notEmpty(String? value) {
    String pattern = r'^\S+$';
    RegExp regex = RegExp(pattern);
    if (!regex.hasMatch(value!.trim()))
      return 'validator.notEmpty';
    else
      return null;
  }
}

```

File - /Users/lulusawesome/Desktop/hader-master/lib/constants/my_textfield.dart

```dart
import 'dart:ui';

import 'package:flutter/material.dart';
import 'package:flutter_screenutil/
flutter_screenutil.dart';
import 'package:flutter_svg/flutter_svg.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:hader/constants/colors.dart';
import 'package:hader/constants/validators.dart';

class MyTextField extends StatefulWidget {
  MyTextField(
      {Key? key,
      required this.controller,
      required this.hinttext,
      required this.textInputType,
      this.validator,
      required this.textAlign,
      this.obscure = false})
      : super(key: key);
  final String hinttext;
  final TextEditingController controller;
  final bool obscure;
  final TextAlign textAlign;
  final TextInputType textInputType;
  String? Function(String?)? validator = Validator
().notEmpty;

  @override
  _MyTextFieldState createState() =>
_MyTextFieldState();
}

class _MyTextFieldState extends State<MyTextField
> {
  bool showPass = false;
  OutlineInputBorder outlineInputBorder =
OutlineInputBorder(
    borderSide: BorderSide(color: Colors.white,
width: 0.0),
    borderRadius: BorderRadius.circular(100.r),
```


File - /Users/lulusawesome/Desktop/hader-master/lib/constants/my_textfield.dart

```dart
  );

  @override
  void initState() {
    showPass = widget.obscure;
  }

  @override
  Widget build(BuildContext context) {
    return Container(
        width: 352.w,
        // height: 63.h,
        // color: Colors.red,
        child: Theme(
          data: Theme.of(context).copyWith(
splashColor: Colors.transparent),
          child: TextFormField(
            cursorColor: color_3,
            validator: widget.validator,
            keyboardType: widget.textInputType,
            textAlign: widget.textAlign,
            obscureText: showPass,
            controller: widget.controller,
            style: GoogleFonts.roboto(
                fontSize: 17.sp, color: color_4,
fontWeight: FontWeight.w400),
            decoration: InputDecoration(
              suffixIcon: widget.obscure == true
                  ? InkWell(
                      onTap: () {
                        setState(() {
                          showPass = !showPass;
                        });
                      },
                      child: showPass
                          ? Container(
                              padding: EdgeInsets.
only(right: 30.w),
                              child: SvgPicture.
asset(
                                "assets/images/eye.
```

```
svg",
                                ),
                            )
                          : Container(
                              padding: EdgeInsets.only(right: 30.w),
                              child: SvgPicture.asset(
                                "assets/images/eye.svg",
                                color: color_3,
                              ),
                            ),
                      )
                    : Container(
                        height: 18.h,
                        width: 20.w,
                      ),
                suffixIconConstraints:
                    BoxConstraints(minHeight: 18.h, minWidth: 20.w),
                filled: true,
                fillColor: Colors.white,
                hintText: widget.hinttext,
                hintStyle: GoogleFonts.roboto(
                    fontSize: 17.sp, color: color_4, fontWeight: FontWeight.w400),
                contentPadding:
                    EdgeInsets.only(left: 20.0.w, top: 10.h, bottom: 10.h),
                border: outlineInputBorder,
                focusedBorder: outlineInputBorder,
                enabledBorder: outlineInputBorder,
                errorBorder: outlineInputBorder,
                focusedErrorBorder: outlineInputBorder,
                errorStyle: GoogleFonts.roboto(
                    fontSize: 14.sp,
                    color: Colors.red,
                    fontWeight: FontWeight.w400),
            ),
```

File - /Users/luluawesome/Desktop/hader-master/lib/constants/submit_button.dart

```dart
import 'package:flutter/material.dart';
import 'package:flutter_screenutil/
flutter_screenutil.dart';
import 'package:google_fonts/google_fonts.dart';

import 'colors.dart';

class SubmitButton extends StatefulWidget {
  const SubmitButton({Key? key, required this.
submitText}) : super(key: key);
  final String submitText;

  @override
  State<SubmitButton> createState() =>
_SubmitButtonState();
}

class _SubmitButtonState extends State<SubmitButton
> {
  @override
  Widget build(BuildContext context) {
    return  Container(
      width: 193.w,
      height: 63.h,
      decoration: BoxDecoration(
        // color: Color(0xff6082FD),
        borderRadius: BorderRadius.circular(100.r),
      ),
      child: Stack(
        children: [
          Center(
            child: Image.asset("assets/images/gradi
.png",
            ),
          ),
          Center(
            child: Text(
              widget.submitText,
              style: GoogleFonts.roboto(
                color: color_1,
                fontSize: 18.sp,
```

File - /Users/tuluawesome/Desktop/hader-master/lib/constants/small_submit_button.dart

```dart
import 'package:flutter/material.dart';
import 'package:flutter_screenutil/
flutter_screenutil.dart';
import 'package:google_fonts/google_fonts.dart';

import 'colors.dart';

class SmallSubmitButton extends StatefulWidget {
  const SmallSubmitButton({Key? key, required this.
submitText}) : super(key: key);
  final String submitText;

  @override
  _SmallSubmitButtonState createState() =>
_SmallSubmitButtonState();
}

class _SmallSubmitButtonState extends State<
SmallSubmitButton> {
  @override
  Widget build(BuildContext context) {
    return Container(
      width: 143.w,
      height: 43.h,
      decoration: BoxDecoration(
        // color: Color(0xff6082FD),
        borderRadius: BorderRadius.circular(100.r),
      ),
      child: Stack(
        children: [
          Center(
            child: Image.asset("assets/images/gradi
.png",
            ),
          ),
          Center(
            child: Text(
              widget.submitText,
              style: GoogleFonts.roboto(
                color: color_1,
                fontSize: 18.sp,
```

Database

File - /Users/lulu awesome/Desktop/hader-master/lib/db/database.dart

```dart
import 'package:firebase_helpers/firebase_helpers.dart';
import 'package:hader/models/class.dart';
import 'package:hader/models/exam.dart';
import 'package:hader/models/instructor.dart';
import 'package:hader/models/student_details.dart';

import '../models/admin.dart';
import '../models/attendace.dart';
import '../models/courses.dart';
import '../models/exam_rooms.dart';
import '../models/excuse.dart';
import '../models/excuse_status.dart';
import '../models/invigilators.dart';

class HaderDatabase {
  static const String STUDENT_DB = "STUDENT";
  static const String INVIGILATOR_DB = "INVIGILATOR";
  static const String ADMIN_DB = "ADMIN";
  static const String ATTENDENCE_DB = "ATTENDENCE";
  static const String COURSES_DB = "COURSES";
  static const String EXAM_DB = "EXAM";
  static const String EXAMROOMS_DB = "EXAMROOMS";
  static const String EXCUSE_DB = "EXCUSE";
  static const String EXCUSE_STATUS_DB = "EXCUSE_STATUS_DB";
  static const String CLASS_DB = "CLASS";
  static const String INSTRUCTOR_DB = "INSTRUCTOR";

  DatabaseService<StudentDetails> studentDb = DatabaseService<StudentDetails>(
      STUDENT_DB,
      fromDS: (id, data) => StudentDetails.fromDS(id, data),
      toMap: (data) => data.toJson());

  DatabaseService<Invigilators> invigilatorDb = DatabaseService<Invigilators>(
      INVIGILATOR_DB,
      fromDS: (id, data) => Invigilators.fromDS(id
```

Page 1 of 9

File - /Users/luluawesome/Desktop/hader-master/lib/db/database.dart

```
35 , data),
36       toMap: (data) => data.toJson());
37
38   DatabaseService<Admin> adminDb = DatabaseService<
   Admin>(ADMIN_DB,
39       fromDS: (id, data) => Admin.fromDS(id, data),
40       toMap: (data) => data.toJson());
41
42   DatabaseService<Instructors> instructorsDb =
   DatabaseService<Instructors>(
43       INSTRUCTOR_DB,
44       fromDS: (id, data) => Instructors.fromDS(id,
   data),
45       toMap: (data) => data.toJson());
46
47   DatabaseService<Attendence> attendenceDb =
   DatabaseService<Attendence>(
48       ATTENDENCE_DB,
49       fromDS: (id, data) => Attendence.fromDS(id,
   data),
50       toMap: (data) => data.toJson());
51
52   DatabaseService<Courses> coursesDb =
   DatabaseService<Courses>(COURSES_DB,
53       fromDS: (id, data) => Courses.fromDS(id, data
   ),
54       toMap: (data) => data.toJson());
55
56   DatabaseService<Exam> examDb = DatabaseService<
   Exam>(EXAM_DB,
57       fromDS: (id, data) => Exam.fromDS(id, data),
58       toMap: (data) => data.toJson());
59
60   DatabaseService<ExamRooms> examRoomsDb =
   DatabaseService<ExamRooms>(
61       EXAMROOMS_DB,
62       fromDS: (id, data) => ExamRooms.fromDS(id,
   data),
63       toMap: (data) => data.toJson());
64
65   DatabaseService<Excuse> excuseDb =
```

```
65 DatabaseService<Excuse>(EXCUSE_DB,
66       fromDS: (id, data) => Excuse.fromDS(id, data
   ),
67       toMap: (data) => data.toJson());
68
69   DatabaseService<ExcuseStatus> excuseStatusDb =
   DatabaseService<ExcuseStatus>(
70       EXCUSE_STATUS_DB,
71       fromDS: (id, data) => ExcuseStatus.fromDS(id
   , data),
72       toMap: (data) => data.toJson());
73
74   DatabaseService<Class> classDb = DatabaseService
   <Class>(CLASS_DB,
75       fromDS: (id, data) => Class.fromDS(id, data
   ),
76       toMap: (data) => data.toJson());
77
78   Future addStudent(StudentDetails studentDetails
   ) async {
79     await studentDb.create(studentDetails.toJson
   ());
80   }
81
82   Future addInvigilators(Invigilators invigilators
   ) async {
83     await invigilatorDb.create(invigilators.toJson
   ());
84   }
85
86   Future addAdmin(Admin admin) async {
87     await adminDb.create(admin.toJson());
88   }
89
90   Future addInstructor(Instructors instructors)
   async {
91     await instructorsDb.create(instructors.toJson
   ());
92   }
93
94   Future addAttendence(Attendence attendence)
```

File - /Users/ruduawesome/Desktop/naderi-master/lib/models/exam.dart

```dart
/// ExamDate : ""
/// ExamTimeStart : ""
/// ExamTimeEnds : ""
/// ExamAssignedTo : ""
/// ExamRoomNo : ""
/// ExamCourse : ""
/// ExamSubject  : ""
/// ClassIDs : ""

class Exam {
  Exam({
    this.examDate,
    this.examTimeStart,
    this.examTimeEnds,
    this.examAssignedTo,
    this.examRoomNo,
    this.examCourse,
    this.examSubject,
    this.classIDs,
    this.avaratarUrl,
  });

  Exam.fromJson(dynamic json) {
    examDate = json['ExamDate'];
    examTimeStart = json['ExamTimeStart'];
    examTimeEnds = json['ExamTimeEnds'];
    examAssignedTo = json['ExamAssignedTo'];
    examRoomNo = json['ExamRoomNo'];
    examCourse = json['ExamCourse'];
    examSubject = json['ExamSubject '];
    classIDs = json['ClassIDs'];
    avaratarUrl = json['AvaratarUrl'];
  }

  Exam.fromDS(String mId, dynamic json) {
    id = mId;
    avaratarUrl = json['AvaratarUrl'];
    examDate = json['ExamDate'];
    examTimeStart = json['ExamTimeStart'];
    examTimeEnds = json['ExamTimeEnds'];
    examAssignedTo = json['ExamAssignedTo'];
```

Page 1 of 2

```
42      examRoomNo = json['ExamRoomNo'];
43      examCourse = json['ExamCourse'];
44      examSubject = json['ExamSubject '];
45      classIDs = json['ClassIDs'];
46    }
47
48    String? id;
49    String? avaratarUrl;
50    String? examDate;
51    String? examTimeStart;
52    String? examTimeEnds;
53    String? examAssignedTo;
54    String? examRoomNo;
55    String? examCourse;
56    String? examSubject;
57    String? classIDs;
58
59    Map<String, dynamic> toJson() {
60      final map = <String, dynamic>{};
61      map['ExamDate'] = examDate;
62      map['AvaratarUrl'] = avaratarUrl;
63      map['ExamTimeStart'] = examTimeStart;
64      map['ExamTimeEnds'] = examTimeEnds;
65      map['ExamAssignedTo'] = examAssignedTo;
66      map['ExamRoomNo'] = examRoomNo;
67      map['ExamCourse'] = examCourse;
68      map['ExamSubject '] = examSubject;
69      map['ClassIDs'] = classIDs;
70      return map;
71    }
72  }
73
```

```dart
/// AvaratarUrl : ""
/// UserName : ""
/// position : ""
/// Email : ""

class Admin {
  Admin({
    this.avaratarUrl,
    this.userName,
    this.position,
    this.email,
  });

  Admin.fromJson(dynamic json) {
    avaratarUrl = json['AvaratarUrl'];
    userName = json['UserName'];
    position = json['Position'];
    email = json['Email'];
  }

  Admin.fromDBJson(dynamic json) {
    try {
      id = json['id'];
    } catch (e, s) {
      print(e);
      print(s);
    }
    avaratarUrl = json['AvaratarUrl'];
    userName = json['UserName'];
    position = json['Position'];
    email = json['Email'];
  }
  Admin.fromDS(String mId, dynamic json) {
    id = mId;
    avaratarUrl = json['AvaratarUrl'];
    userName = json['UserName'];
    position = json['Position'];
    email = json['Email'];
  }
  String? id;
  String? avaratarUrl;
```

File - /Users/luluawesome/Desktop/hader-master/lib/models/admin.dart

```
42   String? userName;
43   String? position;
44   String? email;
45
46   Map<String, dynamic> toJson() {
47     final map = <String, dynamic>{};
48     map['AvaratarUrl'] = avaratarUrl;
49     map['UserName'] = userName;
50     map['Position'] = position;
51     map['Email'] = email;
52     map['id'] = id;
53     return map;
54   }
55 }
56
```

File - /Users/luluawesome/Desktop/hader-master/lib/models/class.dart

```dart
/// ClassName : ""

class Class {
  Class({
    this.className,
  });

  Class.fromJson(dynamic json) {
    className = json['ClassName'];
  }

  Class.fromDS(String mId, dynamic json) {
    id = mId;
    className = json['ClassName'];
  }
  String? id;
  String? className;

  Map<String, dynamic> toJson() {
    final map = <String, dynamic>{};
    map['ClassName'] = className;
    return map;
  }
}

```

File - /Users/luluawesome/Desktop/hader-master/lib/models/excuse.dart

```dart
/// ExamName : ""
/// ExcuseType : ""
/// StartDate : ""
/// EndDate : ""
/// attatchedFile : ""
/// Status : ""

class Excuse {
  Excuse({
    this.examName,
    this.excuseType,
    this.startDate,
    this.endDate,
    this.attatchedFile,
    this.status,
    this.avaratarUrl,
    this.userName,
    this.rollNo,
    this.studyCourse,
    this.email,
    this.classId,
  });

  Excuse.fromJson(dynamic json) {
    examName = json['ExamName'];
    excuseType = json['ExcuseType'];
    startDate = json['StartDate'];
    endDate = json['EndDate'];
    attatchedFile = json['AttatchedFile'];
    status = json['Status'];
    avaratarUrl = json['AvaratarUrl'];
    userName = json['UserName'];
    rollNo = json['RollNo'];
    studyCourse = json['StudyCourse'];
    email = json['Email'];
    classId = json['ClassId'];
  }

  Excuse.fromDS(String mId, dynamic json) {
    id = mId;
    examName = json['ExamName'];
```

File - /Users/luluawesome/Desktop/hader-master/lib/models/excuse.dart

```
42      excuseType = json['ExcuseType'];
43      startDate = json['StartDate'];
44      endDate = json['EndDate'];
45      attatchedFile = json['AttatchedFile'];
46      status = json['Status'];
47      avaratarUrl = json['AvaratarUrl'];
48      userName = json['UserName'];
49      rollNo = json['RollNo'];
50      studyCourse = json['StudyCourse'];
51      email = json['Email'];
52      classId = json['ClassId'];
53    }
54
55    String? id;
56    String? examName;
57    String? excuseType;
58    String? startDate;
59    String? endDate;
60    String? attatchedFile;
61    String? status;
62    String? avaratarUrl;
63    String? userName;
64    String? rollNo;
65    String? studyCourse;
66    String? email;
67    String? classId;
68
69    Map<String, dynamic> toJson() {
70      final map = <String, dynamic>{};
71      map['ExamName'] = examName;
72      map['ExcuseType'] = excuseType;
73      map['StartDate'] = startDate;
74      map['EndDate'] = endDate;
75      map['AttatchedFile'] = attatchedFile;
76      map['Status'] = status;
77      map['AvaratarUrl'] = avaratarUrl;
78      map['UserName'] = userName;
79      map['RollNo'] = rollNo;
80      map['StudyCourse'] = studyCourse;
81      map['Email'] = email;
82      map['ClassId'] = classId;
```

File - /Users/luluawesome/Desktop/hader-master/lib/models/excuse.dart

```
    return map;
  }
}

```

File - /Users/luluawesome/Desktop/hader-master/lib/models/courses.dart

```dart
/// CourseName : ""

class Courses {
  Courses({
    this.courseName,
  });

  Courses.fromJson(dynamic json) {
    courseName = json['CourseName'];
  }
  Courses.fromDS(String mId, dynamic json) {
    id = mId;
    courseName = json['CourseName'];
  }
  String? id;
  String? courseName;

  Map<String, dynamic> toJson() {
    final map = <String, dynamic>{};
    map['CourseName'] = courseName;
    return map;
  }
}

```

File - /Users/luluawesome/Desktop/hader-master/lib/models/attendace.dart

```dart
/// RoomId : ""
/// StudentAvatarUrl : ""
/// StudentUserName : ""
/// StudentRollNo : ""
/// isPresent : ""
/// ExamId : ""
/// RoomName : ""
/// ExamName : ""

class Attendence {
  Attendence({
    this.roomNo,
    this.studentAvatarUrl,
    this.studentUserName,
    this.studentRollNo,
    this.isPresent,
    this.courseId,
    this.roomName,
    this.examSubject,
    this.examId,
    this.studentId,
  });

  Attendence.fromJson(dynamic json) {
    roomNo = json['RoomNo'];
    studentAvatarUrl = json['StudentAvatarUrl'];
    studentUserName = json['StudentUserName'];
    studentRollNo = json['StudentRollNo'];
    isPresent = json['isPresent'];
    courseId = json['CourseId'];
    roomName = json['RoomName'];
    examSubject = json['ExamSubject'];
    examId = json['ExamId'];
    studentId = json['StudentId'];
  }

  Attendence.fromDS(String mId, dynamic json) {
    id = mId;
    roomNo = json['RoomNo'];
    studentAvatarUrl = json['StudentAvatarUrl'];
    studentUserName = json['StudentUserName'];
```

File - /Users/fullawesome/Desktop/nader-master/lib/models/attendace.dart

```dart
    studentRollNo = json['StudentRollNo'];
    isPresent = json['isPresent'];
    courseId = json['CourseId'];
    roomName = json['RoomName'];
    examSubject = json['ExamSubject'];
    examId = json['ExamId'];
    studentId = json['StudentId'];
  }

  String? id;
  String? roomNo;
  String? studentAvatarUrl;
  String? studentUserName;
  String? studentRollNo;
  String? isPresent;
  String? courseId;
  String? roomName;
  String? examSubject;
  String? examId;
  String? studentId;

  Map<String, dynamic> toJson() {
    final map = <String, dynamic>{};
    map['RoomNo'] = roomNo;
    map['StudentAvatarUrl'] = studentAvatarUrl;
    map['StudentUserName'] = studentUserName;
    map['StudentRollNo'] = studentRollNo;
    map['isPresent'] = isPresent;
    map['CourseId'] = courseId;
    map['RoomName'] = roomName;
    map['ExamSubject'] = examSubject;
    map['ExamId'] = examId;
    map['StudentId'] = studentId;
    return map;
  }

  @override
  bool operator ==(other) =>
      other is Attendence && (other.studentId ==
 studentId);

```

File - /Users/luluawesome/Desktop/hader-master/lib/models/exam_rooms.dart

```dart
/// RoomName : ""

class ExamRooms {
  ExamRooms({
    this.roomName,
  });

  ExamRooms.fromJson(dynamic json) {
    roomName = json['RoomName'];
  }
  ExamRooms.fromDS(String mId, dynamic json) {
    id = mId;
    roomName = json['RoomName'];
  }

  String? id;
  String? roomName;

  Map<String, dynamic> toJson() {
    final map = <String, dynamic>{};
    map['RoomName'] = roomName;
    return map;
  }
}

```

```dart
/// AvaratarUrl : ""
/// UserName : ""
/// Proffession : ""
/// Email : ""

class Instructors {
  Instructors({
    this.avaratarUrl,
    this.userName,
    this.proffession,
    this.email,
  });

  Instructors.fromJson(dynamic json) {
    avaratarUrl = json['AvaratarUrl'];
    userName = json['UserName'];
    proffession = json['Proffession'];
    email = json['Email'];
  }

  Instructors.fromDS(String mId, dynamic json) {
    id = mId;
    avaratarUrl = json['AvaratarUrl'];
    userName = json['UserName'];
    proffession = json['Proffession'];
    email = json['Email'];
  }

  String? id;
  String? avaratarUrl;
  String? userName;
  String? proffession;
  String? email;

  Map<String, dynamic> toJson() {
    final map = <String, dynamic>{};
    map['AvaratarUrl'] = avaratarUrl;
    map['UserName'] = userName;
    map['Proffession'] = proffession;
    map['Email'] = email;
    return map;
```

```dart
/// AvaratarUrl : ""
/// UserName : ""
/// Proffession : ""
/// Email : ""

class Invigilators {
  Invigilators({
    this.avaratarUrl,
    this.userName,
    this.proffession,
    this.email,
  });

  Invigilators.fromJson(dynamic json) {
    avaratarUrl = json['AvaratarUrl'];
    userName = json['UserName'];
    proffession = json['Proffession'];
    email = json['Email'];
  }

  Invigilators.fromDBJson(dynamic json) {
    try {
      id = json['id'];
    } catch (e, s) {
      print(e);
      print(s);
    }
    avaratarUrl = json['AvaratarUrl'];
    userName = json['UserName'];
    proffession = json['Proffession'];
    email = json['Email'];
  }

  Invigilators.fromDS(String mId, dynamic json) {
    id = mId;
    avaratarUrl = json['AvaratarUrl'];
    userName = json['UserName'];
    proffession = json['Proffession'];
    email = json['Email'];
  }

```

File - /Users/luluawesome/Desktop/hader-master/lib/models/invigilators.dart

```
42   String? id;
43   String? avaratarUrl;
44   String? userName;
45   String? proffession;
46   String? email;
47
48   Map<String, dynamic> toJson() {
49     final map = <String, dynamic>{};
50     map['AvaratarUrl'] = avaratarUrl;
51     map['UserName'] = userName;
52     map['Proffession'] = proffession;
53     map['Email'] = email;
54     map['id'] = id;
55     return map;
56   }
57 }
58
```

```dart
/// RollNo : ""
/// ExcuseStatus : ""

class ExcuseStatus {
  ExcuseStatus({
    this.rollNo,
    this.excuseStatus,
    this.email,
  });

  ExcuseStatus.fromJson(dynamic json) {
    rollNo = json['RollNo'];
    excuseStatus = json['ExcuseStatus'];
    email = json['Email'];
  }
  ExcuseStatus.fromDBJson(dynamic json) {
    try {
      id = json['id'];
    } catch (e, s) {
      print(e);
      print(s);
    }
    rollNo = json['RollNo'];
    excuseStatus = json['ExcuseStatus'];
    email = json['Email'];
  }

  ExcuseStatus.fromDS(String mId, dynamic json) {
    id = mId;
    rollNo = json['RollNo'];
    excuseStatus = json['ExcuseStatus'];
    email = json['Email'];
  }

  String? id;
  String? rollNo;
  String? excuseStatus;
  String? email;

  Map<String, dynamic> toJson() {
    final map = <String, dynamic>{};
```

File - /Users/luluawesome/Desktop/hader-master/lib/models/excuse_status.dart

```
42      map['RollNo'] = rollNo;
43      map['ExcuseStatus'] = excuseStatus;
44      map['Email'] = email;
45      map['id'] = id;
46      return map;
47    }
48 }
49
```

File - /Users/luluawesome/Desktop/hader-master/lib/models/student_details.dart

```dart
/// Id : ""
/// AvaratarUrl : ""
/// UserName : ""
/// RollNo : ""
/// StudyCourse : ""
/// Email : ""
/// ClassId : ""

class StudentDetails {
  StudentDetails({
    this.avaratarUrl,
    this.userName,
    this.rollNo,
    this.studyCourse,
    this.email,
    this.classId,
  });

  StudentDetails.fromJson(dynamic json) {
    avaratarUrl = json['AvaratarUrl'];
    userName = json['UserName'];
    rollNo = json['RollNo'];
    studyCourse = json['StudyCourse'];
    email = json['Email'];
    classId = json['ClassId'];
  }

  StudentDetails.fromDBJson(dynamic json) {
    try {
      id = json['id'];
    } catch (e, s) {
      print(e);
      print(s);
    }
    avaratarUrl = json['AvaratarUrl'];
    userName = json['UserName'];
    rollNo = json['RollNo'];
    studyCourse = json['StudyCourse'];
    email = json['Email'];
    classId = json['ClassId'];
  }
```

File - /Users/luluawesome/Desktop/hader-master/lib/models/student_details.dart

```dart

  StudentDetails.fromDS(String mId, dynamic json) {
    print(">>>>>>>>>>>>>>>>>> $mId");
    id = mId;
    avaratarUrl = json['AvaratarUrl'];
    userName = json['UserName'];
    rollNo = json['RollNo'];
    studyCourse = json['StudyCourse'];
    email = json['Email'];
    classId = json['ClassId'];
  }

  String? id;
  String? avaratarUrl;
  String? userName;
  String? rollNo;
  String? studyCourse;
  String? email;
  String? classId;

  Map<String, dynamic> toJson() {
    final map = <String, dynamic>{};
    map['AvaratarUrl'] = avaratarUrl;
    map['UserName'] = userName;
    map['RollNo'] = rollNo;
    map['StudyCourse'] = studyCourse;
    map['Email'] = email;
    map['ClassId'] = classId;
    map['id'] = id;
    return map;
  }
}

```

```dart
import 'dart:async';

import 'package:flutter/material.dart';
import 'package:flutter_screenutil/
flutter_screenutil.dart';
import 'package:flutter_svg/flutter_svg.dart';
import 'package:get_storage/get_storage.dart';
import 'package:hader/screens/admin/
admin_profile_screen.dart';
import 'package:hader/screens/invigilator/
profile_screen.dart';
import 'package:hader/screens/student/
students_profile_screen.dart';

import '../db/my_config.dart';
import 'auth_screens/sign_in_screen.dart';
import 'instructor/instructor_profile_screen.dart';

class SplashScreen extends StatefulWidget {
  const SplashScreen({Key? key}) : super(key: key);

  @override
  State<SplashScreen> createState() =>
_SplashScreenState();
}

class _SplashScreenState extends State<SplashScreen
> {
  bool isLoggedIn = false;

  @override
  void initState() {
    super.initState();
    Timer(Duration(seconds: 3), () {
      print("_____________++++++++++++++++++");
      final box = GetStorage();
      isLoggedIn = box.read(MyConfig.IS_LOGGED_IN
) ?? false;
      print(isLoggedIn);

      var userType = box.read(MyConfig.USER_TYPE
```

```
34 ) ?? "";
35       print(userType);
36       if (isLoggedIn) {
37         if (userType == MyConfig.STUDENT) {
38           Navigator.of(context).pushReplacement(
39             MaterialPageRoute(
40               builder: (BuildContext context) =>
   StudentsProfileScreen(),
41             ),
42           );
43         } else if (userType == MyConfig.INVIGILATOR
   ) {
44           Navigator.of(context).pushReplacement(
45             MaterialPageRoute(
46               builder: (BuildContext context) =>
   ProfileScreen(),
47             ),
48           );
49         } else if (userType == MyConfig.ADMIN) {
50           Navigator.of(context).pushReplacement(
51             MaterialPageRoute(
52               builder: (BuildContext context) =>
   AdminProfileScreen(),
53             ),
54           );
55         } else if (userType == MyConfig.INSTRUCTOR
   ) {
56           Navigator.of(context).pushReplacement(
57             MaterialPageRoute(
58               builder: (BuildContext context) =>
   InstructorProfileScreen(),
59             ),
60           );
61         }
62       } else {
63         Navigator.of(context).pushReplacement(
64           MaterialPageRoute(
65             builder: (BuildContext context) =>
   SigninScreen(),
66           ),
67         );
```

```
68        }
69      });
70    }
71
72    @override
73    Widget build(BuildContext context) {
74      return Scaffold(
75        body: Stack(
76          fit: StackFit.expand,
77          children: [
78            SvgPicture.asset(
79              "assets/images/Rectangle.svg",
80              fit: BoxFit.cover,
81            ),
82            Center(
83              child: Image.asset(
84                "assets/images/splashLogo.png",
85                height: 200.h,
86                width: 177.w,
87                fit: BoxFit.fitWidth,
88              ),
89            ),
90          ],
91        ),
92      );
93    }
94  }
95
```

```dart
import 'package:flutter/material.dart';
import 'package:flutter_screenutil/
flutter_screenutil.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:hader/constants/my_textfield.dart';
import 'package:hader/constants/small_submit_button
.dart';
import 'package:hader/constants/submit_button.dart'
;

import '../constants/colors.dart';
import '../db/database.dart';
import '../models/student_details.dart';

class StudentAddForm extends StatefulWidget {
  const StudentAddForm({Key? key}) : super(key: key
);

  @override
  _StudentAddFormState createState() =>
_StudentAddFormState();
}

class _StudentAddFormState extends State<
StudentAddForm> {
  TextEditingController userNameController =
TextEditingController();
  TextEditingController rollNoController =
TextEditingController();
  TextEditingController emailController =
TextEditingController();
  String? course;
  @override
  Widget build(BuildContext context) {
    return SafeArea(
      child: Scaffold(
        resizeToAvoidBottomInset: true,
        body: SingleChildScrollView(
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.
center,
```

File - /Users/furqawesome/Desktop/hader-master/lib/screens/students_add_form.dart

```
32              children: [
33                SizedBox(
34                  height: 30,
35                ),
36                Center(
37                    child: MyTextField(
38                        controller:
   userNameController,
39                        hinttext: 'UserName',
40                        textInputType: TextInputType.
   text,
41                        textAlign: TextAlign.center
   )),
42                SizedBox(
43                  height: 30,
44                ),
45                Center(
46                    child: MyTextField(
47                        controller: rollNoController,
48                        hinttext: 'RollNo',
49                        textInputType: TextInputType.
   text,
50                        textAlign: TextAlign.center
   )),
51                SizedBox(
52                  height: 30,
53                ),
54                Center(
55                    child: MyTextField(
56                        controller: emailController,
57                        hinttext: 'Email',
58                        textInputType: TextInputType.
   text,
59                        textAlign: TextAlign.center
   )),
60                SizedBox(
61                  height: 30,
62                ),
63                Container(
64                  height: 58.h,
65                  width: 352.w,
```

```
66                  padding: EdgeInsets.only(left: 14.
   5.w, right: 15.4.w),
67                  decoration: BoxDecoration(
68                    boxShadow: [
69                      BoxShadow(
70                          blurRadius: 1,
71                          color: Colors.grey.
   withOpacity(0.2),
72                          offset: Offset(0.0, 4),
73                          spreadRadius: 0)
74                    ],
75                    borderRadius: BorderRadius.
   circular(6.75.r),
76                    color: color_1,
77                    border: Border.all(
78                        color: color_1, style:
   BorderStyle.solid, width: 0.80),
79                  ),
80                  child: DropdownButtonHideUnderline
   (
81                    child: DropdownButton<String>(
82                      items: <String>['CIS 322', 'CS
    115', 'MGMT 502']
83                          .map((String value) {
84                        return DropdownMenuItem<
   String>(
85                          value: value,
86                          child: Text(value),
87                        );
88                      }).toList(),
89                      onChanged: (value) {
90                        setState(() {
91                          course = (value ?? "");
92                        });
93                      },
94                      style: GoogleFonts.montserrat(
95                        fontSize: 16.sp,
96                        fontWeight: FontWeight.w500,
97                        color: color_4,
98                      ),
99                      value: course,
```

```
100                       hint: Text(
101                         'Select',
102                         style: GoogleFonts.
    montserrat(
103                           fontSize: 16.sp,
104                           fontWeight: FontWeight.
    w500,
105                           color: color_4,
106                         ),
107                       ),
108                     ),
109                   ),
110                 ),
111                 SizedBox(
112                   height: 30,
113                 ),
114                 SizedBox(
115                   height: 30,
116                 ),
117                 InkWell(
118                   onTap: () async {
119                     await HaderDatabase().addStudent
    (StudentDetails(
120                         userName: userNameController
    .text,
121                         email: emailController.text,
122                         rollNo: rollNoController.
    text,
123                         studyCourse: course,
124                         classId: '
    S6q035KVR0BX6k0KkEln'));
125                   },
126                   child: SmallSubmitButton(
    submitText: "Create"),
127                 ),
128               ],
129             ),
130           ),
131         ),
132       );
133     }
```

File - /Users/rudraawesome/Desktop/radel-master/lib/services/storage_data.dart

```dart
import 'dart:async';
import 'dart:io';

import 'package:firebase_storage/firebase_storage.dart';
import 'package:image_picker/image_picker.dart';
import 'package:path/path.dart' as Path;

class StorageData {
  final ImagePicker _picker = ImagePicker();
  late FirebaseStorage storageReferece;

  //Get the url of the file in the database
  Future<String?> getFileUrl({String? key}) async {
    return '';
  }

  //Get all images and videos
  Future<void> getImageVideoFiles({String? key}) async {}

  //Upload and image or video file
  Future<XFile?> uploadImageFile({String? fileType}) async {
    final XFile? file;
    switch (fileType) {
      case 'imageGallery':
        // Pick an image
        file = await _picker.pickImage(
            source: ImageSource.gallery,
            preferredCameraDevice: CameraDevice.front,
            imageQuality: 25,
            maxHeight: 400,
            maxWidth: 400);
        return file;
      case 'imageCamera':
        // Capture a photo
        file = await _picker.pickImage(
          source: ImageSource.camera,
          imageQuality: 10,
```

```
38          );
39          return file;
40        case 'videoGallery':
41          file = await _picker.pickVideo(source:
   ImageSource.gallery);
42
43          return file;
44        case 'videoCamera':
45          file = await _picker.pickVideo(
46            source: ImageSource.camera,
47            preferredCameraDevice: CameraDevice.front
   ,
48          );
49          return file;
50        default:
51          print('Could not find your request!');
52          return null;
53      }
54    }
55
56    //Upload file to firebase storage
57    Future<Map<String, dynamic>> uploadFile(
58        {XFile? xfile, String? folderName}) async {
59      print("**************************");
60      String fileUrl = '';
61      String thumbnailUrl = '';
62      // final Info.MediaInfo _mediaInfo = Info.
   MediaInfo();
63
64      Map<String, dynamic> result;
65      try {
66        print("======================Success
   ============================");
67        if (xfile != null) {
68          storageReferece = FirebaseStorage.instance;
69          Reference ref = storageReferece
70              .ref()
71              .child('$folderName/${Path.basename(
   xfile.path)}');
72
73          UploadTask uploadTask = ref.putFile(File(
```

```
73 xfile.path));
74         var downloadUrl = await (await uploadTask
   ).ref.getDownloadURL();
75         fileUrl = downloadUrl.toString();
76
77         result = {'imageUrl': fileUrl};
78         print("RETUUUUUUUUUUUUUURN $result");
79         return result;
80       }
81       //In case null value were received
82       result = {
83         'videoUrl': '',
84         'imageUrl': '',
85       };
86       return result;
87     } catch (e, stackTrace) {
88       print("manish");
89       print('Error uploading file: $e');
90       // await Sentry.captureException(e,
   stackTrace: stackTrace);
91       result = {'error': e.toString()};
92       return result;
93     }
94   }
95
96   // //Generate thumbnail image from a video url
97   // Future<File?> generateThumbnailUrl(String
   videoUrl) async {
98   //   final fileName = await VideoThumbnail.
   thumbnailFile(
99   //       video: videoUrl,
100  //       thumbnailPath: (await
   getTemporaryDirectory()).path,
101  //       imageFormat: ImageFormat.JPEG,
102  //       quality: 100);
103  //
104  //   var file = File(fileName!);
105  //   return file;
106  // }
107
108  //Query from bucket
```

File - /Users/luluawesome/Desktop/hader-master/lib/services/storage_data.dart

```
109   Future<void> listAllItems() async {}
110
111   //Upload profile photo
112   Future<String> uploadProfilePhoto() async {
113     return '';
114   }
115 }
116
```

Main

File - /Users/luluawesome/Desktop/hader-master/lib/main.dart

```
34          builder: (context, widget) {
35            //add this line
36            ScreenUtil.setContext(context);
37            return MediaQuery(
38              //Setting font does not change with
   system font size
39              data: MediaQuery.of(context).copyWith(
   textScaleFactor: 1.0),
40              child: widget!,
41            );
42          },
43          home: SplashScreen(),
44          // home: AdminProfileScreen(),
45          // home: StudentAddForm(),
46        ),
47      );
48    }
49 }
50
```

```dart
import 'package:flutter/material.dart';
import 'package:flutter_screenutil/
flutter_screenutil.dart';
import 'package:flutter_svg/svg.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:qr_flutter/qr_flutter.dart';

import '../../auth/authentication.dart';
import '../../constants/colors.dart';
import '../auth_screens/change_pass_screen.dart';
import '../auth_screens/sign_in_screen.dart';
import 'instructor_notification_screen.dart';

class InstructorProfileScreen extends
StatefulWidget {
  const InstructorProfileScreen({Key? key}) : super
(key: key);

  @override
  _InstructorProfileScreenState createState() =>
      _InstructorProfileScreenState();
}

class _InstructorProfileScreenState extends State<
InstructorProfileScreen> {
  @override
  Widget build(BuildContext context) {
    return SafeArea(
      child: Scaffold(
        resizeToAvoidBottomInset: false,
        backgroundColor: color_1,
        body: Stack(
          children: [
            SingleChildScrollView(
              physics: ScrollPhysics(),
              scrollDirection: Axis.vertical,
              child: Stack(
                children: [
                  Column(
                    children: [
                      Container(
```

```
                          height: 220.h,
                          width: double.infinity,
                          color: color_1,
                          child: Column(
                            children: [
                              SizedBox(
                                height: 40.h,
                              ),
                              Row(
                                children: [
                                  SizedBox(
                                    width: 20.w,
                                  ),
                                  // InkWell(
                                  //   onTap: () {
                                  //     // Navigator
.pop(context);
                                  //   },
                                  //   child:
Container(
                                  //     color:
Colors.transparent,
                                  //     padding:
EdgeInsets.all(5),
                                  //     child:
SvgPicture.asset(
                                  //       "assets/
images/arrow.svg",
                                  //       height: 16
.h,
                                  //       width: 8.w
,
                                  //     ),
                                  //   ),
                                  // ),
                                  SizedBox(
                                    width: 11.w,
                                  ),
                                  Text(
                                    "Account",
                                    style:
```

```dart
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/material.dart';
import 'package:flutter_screenutil/
flutter_screenutil.dart';
import 'package:get_storage/get_storage.dart';
import 'package:hader/screens/admin/
admin_profile_screen.dart';
import 'package:hader/screens/instructor/
instructor_profile_screen.dart';
import 'package:hader/screens/invigilator/
profile_screen.dart';
import 'package:hader/screens/invigilator/
student_scanner_screen.dart';
import 'package:hader/screens/splash_screen.dart';
import 'package:hader/screens/student/
student_upload_excuse.dart';
import 'package:hader/screens/student/
students_profile_screen.dart';
import 'package:hader/screens/students_add_form.
dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp();
  await GetStorage.init();
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return ScreenUtilInit(
      designSize: Size(428, 926),
      minTextAdapt: true,
      splitScreenMode: true,
      builder: () => MaterialApp(
        debugShowCheckedModeBanner: false,
        //... other code
```

```dart
import 'package:flutter/material.dart';
import 'package:flutter_screenutil/
flutter_screenutil.dart';
import 'package:flutter_svg/svg.dart';
import 'package:google_fonts/google_fonts.dart';
import 'package:qr_flutter/qr_flutter.dart';

import '../../auth/authentication.dart';
import '../../constants/colors.dart';
import '../auth_screens/change_pass_screen.dart';
import '../auth_screens/sign_in_screen.dart';
import 'instructor_notification_screen.dart';

class InstructorProfileScreen extends
StatefulWidget {
  const InstructorProfileScreen({Key? key}) : super
(key: key);

  @override
  _InstructorProfileScreenState createState() =>
      _InstructorProfileScreenState();
}

class _InstructorProfileScreenState extends State<
InstructorProfileScreen> {
  @override
  Widget build(BuildContext context) {
    return SafeArea(
      child: Scaffold(
        resizeToAvoidBottomInset: false,
        backgroundColor: color_1,
        body: Stack(
          children: [
            SingleChildScrollView(
              physics: ScrollPhysics(),
              scrollDirection: Axis.vertical,
              child: Stack(
                children: [
                  Column(
                    children: [
                      Container(
```

```
                                    ),
                                ),
                                SizedBox(
                                  height: 19.h,
                                ),
                                Text(
                                  "Lecturer",
                                  style: GoogleFonts.
    roboto(
                                    fontWeight:
    FontWeight.w400,
                                    fontSize: 17.sp,
                                    color: color_4,
                                  ),
                                ),
                                SizedBox(
                                  height: 32.h,
                                ),
                                Container(
                                  height: 310.h,
                                  width: 310.h,
                                  padding: EdgeInsets.
    all(14),
                                  decoration:
    BoxDecoration(
                                      color: color_1,
                                      borderRadius:
    BorderRadius.circular(50.r)),
                                  child: QrImage(
                                    data: '1234567890'
    ,
                                    version:
    QrVersions.auto,
                                    size: 254.w,
                                    gapless: false,
                                    errorStateBuilder
    : (cxt, err) {
                                      return Container
    (
                                        child: Center(
                                          child: Text(
```

```
                                        "Uh oh! Something went wrong...",
                                        textAlign: TextAlign.center,
                                      ),
                                    ),
                                  );
                                },
                              ),
                            ),
                            SizedBox(
                              height: 32.h,
                            ),
                            InkWell(
                              onTap: () {
                                Navigator.push(
                                    context,
                                    MaterialPageRoute(
                                        builder: (context) => ChangePassScreen()));
                              },
                              child: Row(
                                mainAxisAlignment: MainAxisAlignment.center,
                                children: [
                                  Container(
                                    color: Colors.transparent,
                                    child: SvgPicture.asset(
                                      "assets/images/key.svg",
                                      height: 17.w,
                                      width: 17.w,
                                    ),
                                  ),
                                  SizedBox(
```

File - /Users/ruduawesome/Desktop/nader-master/lib/screens/instructor/instructor_profile_screen.dart

```
165                                       width: 13.w,
166                                     ),
167                                     Text(
168                                       "Reset
    Password",
169                                       style:
    GoogleFonts.roboto(
170                                         fontWeight:
    FontWeight.w400,
171                                         fontSize: 17
    .sp,
172                                         color:
    color_3,
173                                       ),
174                                     ),
175                                   ],
176                                 ),
177                               ),
178                               SizedBox(
179                                 height: 15.h,
180                               ),
181                               InkWell(
182                                 onTap: () {
183
    AuthenticationHelper().signOut();
184                                   Navigator.
    pushReplacement(
185                                       context,
186
    MaterialPageRoute(
187                                           builder: (
    context) => SigninScreen()));
188                                   ScaffoldMessenger.
    of(context).showSnackBar(
189                                     SnackBar(content
    : Text("Signed Out")),
190                                   );
191                                 },
192                                 child: Row(
193                                   mainAxisAlignment
    : MainAxisAlignment.center,
```

```
194                                     children: [
195                                       Container(
196                                         color: Colors.
    transparent,
197                                         child:
    SvgPicture.asset(
198                                           "assets/
    images/exit.svg",
199                                           height: 16.h
    ,
200                                           width: 17.w,
201                                         ),
202                                       ),
203                                       SizedBox(
204                                         width: 14.w,
205                                       ),
206                                       Text(
207                                         "Sign Out",
208                                         style:
    GoogleFonts.roboto(
209                                           fontWeight:
    FontWeight.w400,
210                                           fontSize: 17
    .sp,
211                                           color:
    color_3,
212                                         ),
213                                       ),
214                                     ],
215                                   ),
216                                 ),
217                               ],
218                             ),
219                           ),
220                           // Container(
221                           //   height: 56.h,
222                           //   color: color_5,
223                           // ),
224                         ],
225                       ),
226                       Positioned(
```

```dart
import 'package:flutter/material.dart';
import 'package:flutter_screenutil/
flutter_screenutil.dart';
import 'package:flutter_svg/svg.dart';
import 'package:google_fonts/google_fonts.dart';

import '../../constants/colors.dart';
import 'instructor_profile_screen.dart';

class InstructorNotificationScreen extends
StatefulWidget {
  const InstructorNotificationScreen({Key? key}) :
super(key: key);

  @override
  _InstructorNotificationScreenState createState
() =>
      _InstructorNotificationScreenState();
}

class _InstructorNotificationScreenState
    extends State<InstructorNotificationScreen> {
  @override
  Widget build(BuildContext context) {
    return SafeArea(
        child: Scaffold(
      resizeToAvoidBottomInset: false,
      backgroundColor: color_1,
      body: Stack(
        children: [
          Column(
            children: [
              Container(
                height: 220.h,
                width: double.infinity,
                color: color_1,
                child: Column(
                  children: [
                    SizedBox(
                      height: 40.h,
                    ),
```

```
36 MyConfig.STUDENT_DETAILS));
37
38   @override
39   Widget build(BuildContext context) {
40     return SafeArea(
41         child: Scaffold(
42       resizeToAvoidBottomInset: false,
43       backgroundColor: color_1,
44       body: Column(
45         children: [
46           Container(
47             height: 220.h,
48             width: double.infinity,
49             color: color_1,
50             child: Column(
51               children: [
52                 SizedBox(
53                   height: 40.h,
54                 ),
55                 Row(
56                   children: [
57                     SizedBox(
58                       width: 20.w,
59                     ),
60                     InkWell(
61                       onTap: () {
62                         Navigator.pop(context);
63                       },
64                       child: Container(
65                         color: Colors.transparent,
66                         padding: EdgeInsets.all(5),
67                         child: SvgPicture.asset(
68                           "assets/images/arrow.svg"
   ,
69                           height: 16.h,
70                           width: 8.w,
71                         ),
72                       ),
73                     ),
74                     SizedBox(
75                       width: 11.w,
```

Appendix F

The accounts to access HADER Application:

User	Email	Password
Student	name.student@iau.edu.sa	12345678
Invigilator	name.invigilator@iau.edu.sa	12345678
Admin	name.admin@iau.edu.sa	12345678
Instructor	name.instructor@iau.edu.sa	12345678

Appendix G

MOBILE APP USER MANUAL

available for

iOS / Android

Table of Contents

HADER MOBILE APP

Hader's application solves a variety of exam attendance issues at CCSIT and mainly aims to streamline the process of taking attendance in exam halls.

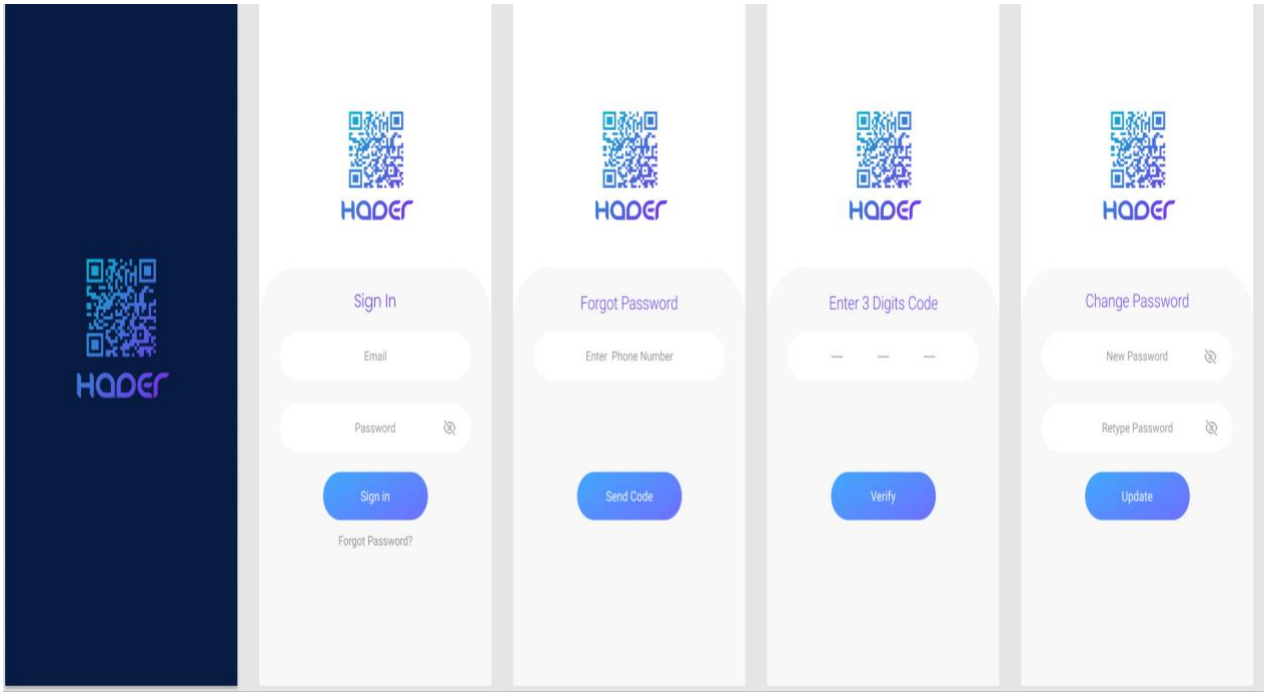
This manual provides the functions that are directly accessible from user smartphone device (iOS/Android).

- To use HADER:

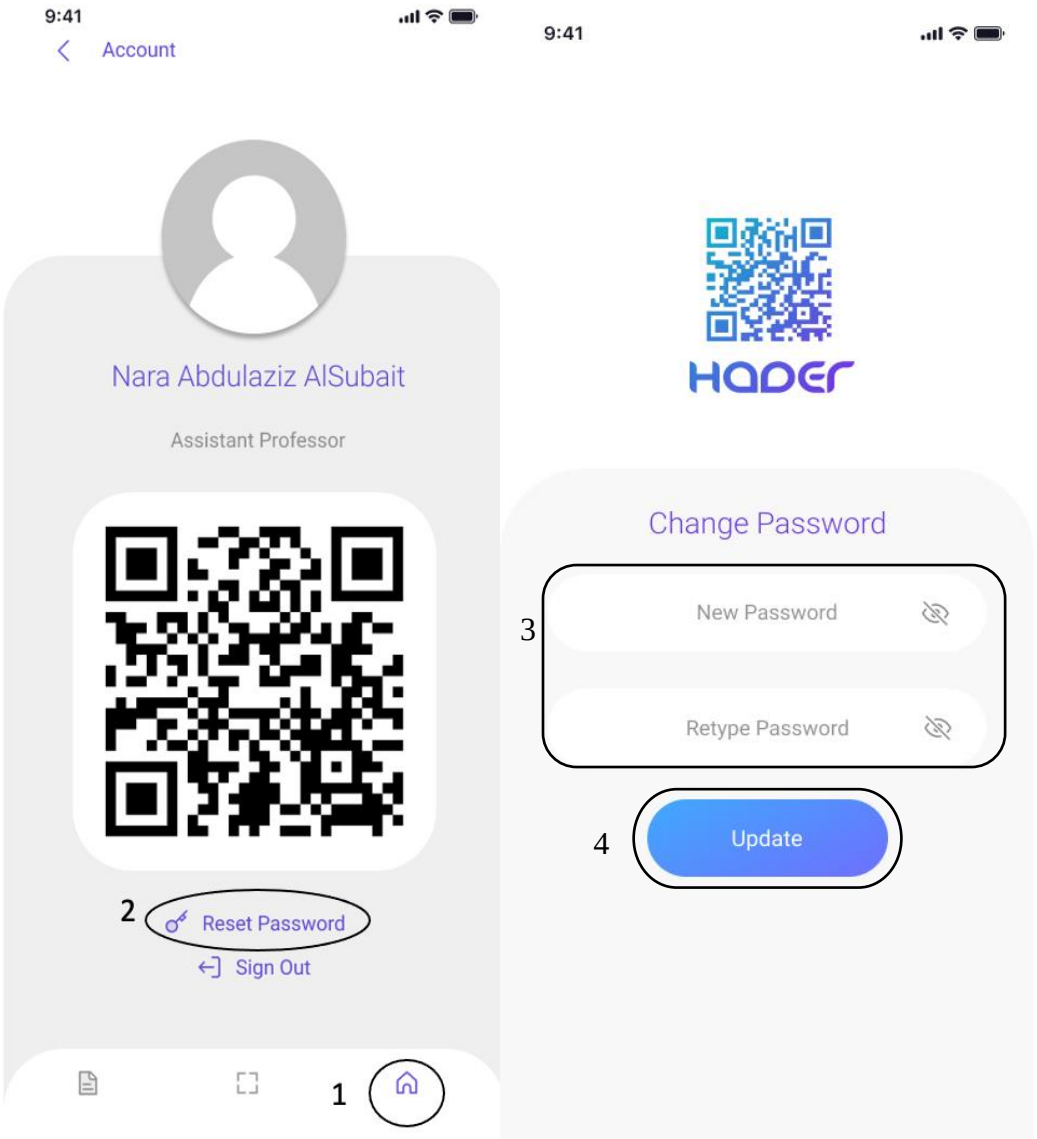
1. Download the free "**HADER"** Mobile App

   • App iOS available on App Store
   • Android available on Play Store

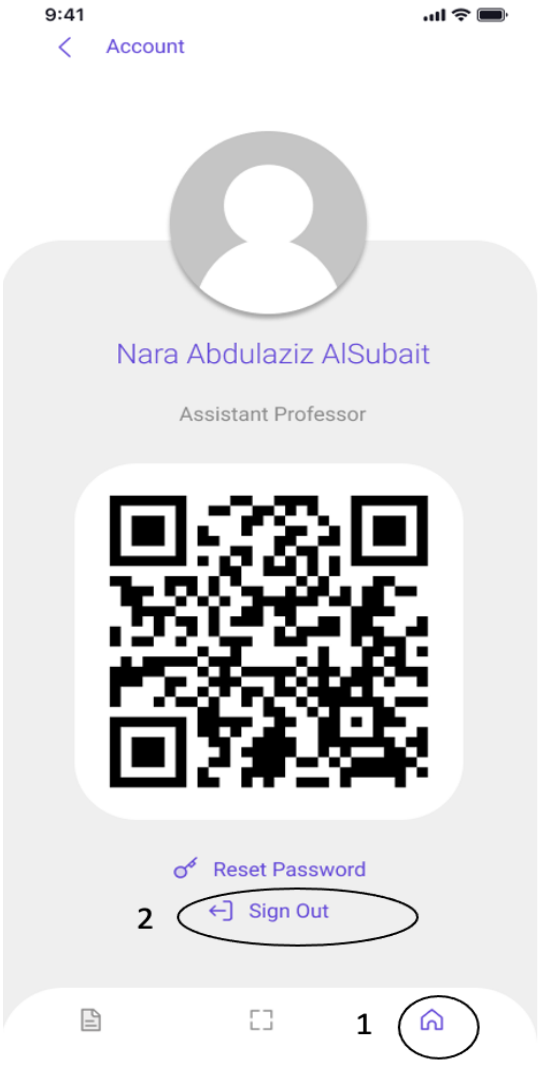
2. Open the application and sign in by entering user **email** and **Password.**
3. Users can reset their passwords by, clicking on forgot password and entering their number and then they'll receive a SMS with a 3-digit code then they can reset the password.
4. Once user have completed the steps above, user enter the App.



- Reset Password
  1. Enter Home icon.
  2. Click Reset Password.
  3. Type the new password then retype password.
  4. Click update and it should be updated.



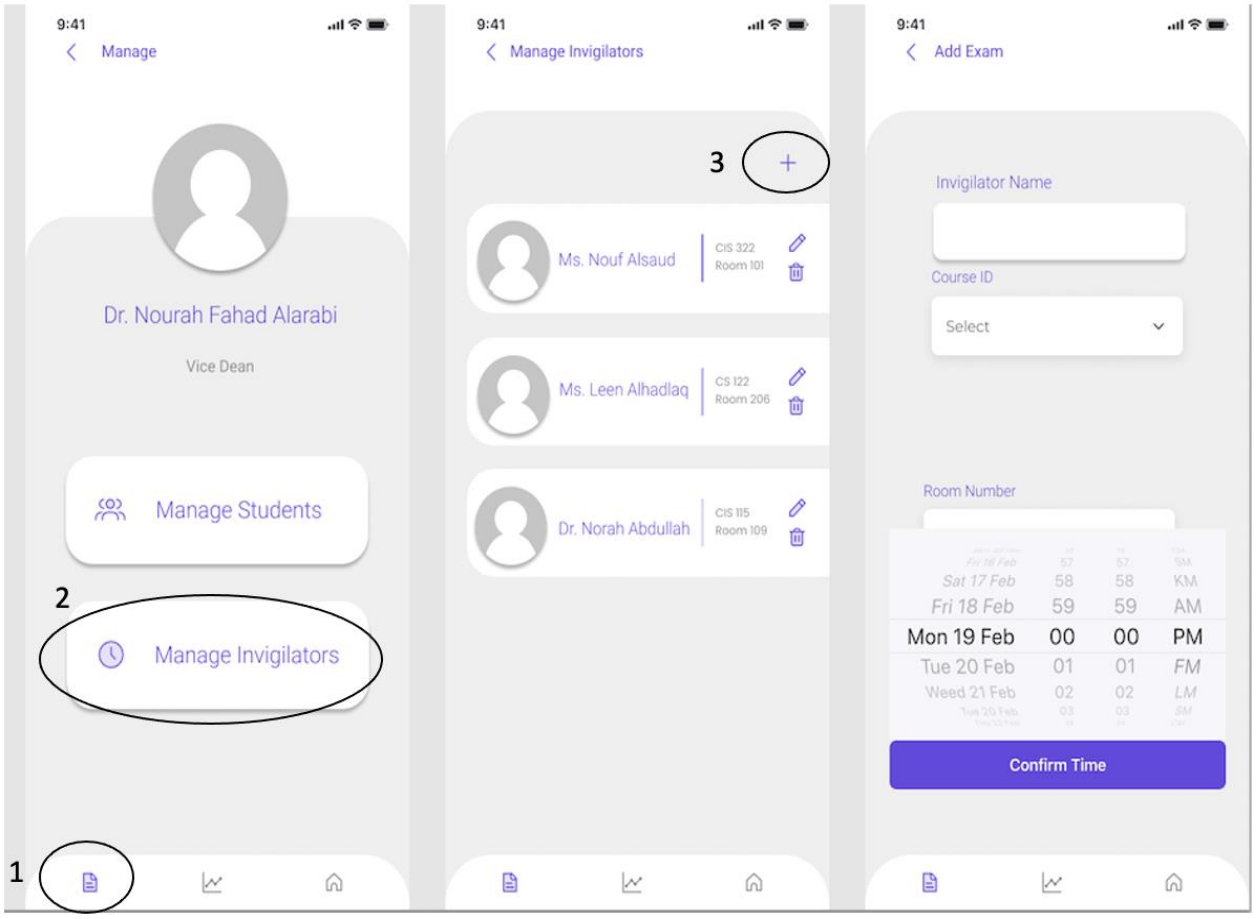
- Sign Out
  1. Enter Home icon.
  2. Click Sign Out.

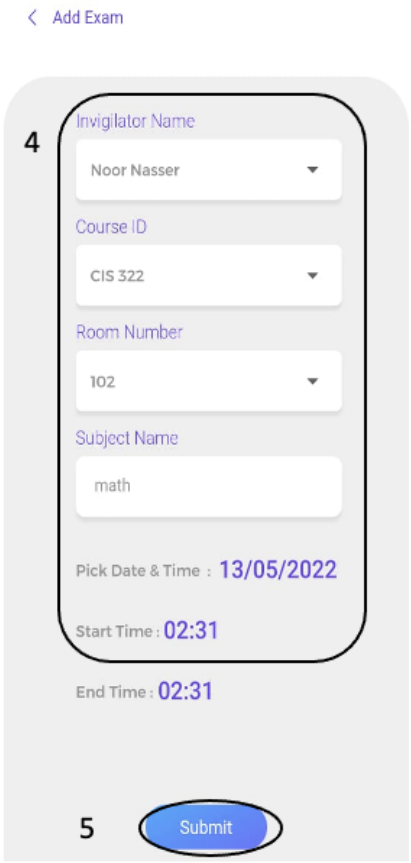


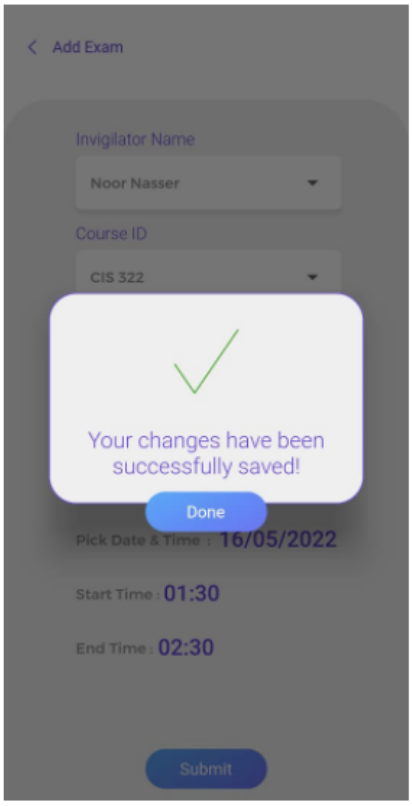
ADMIN USER

After sign the Admin can:

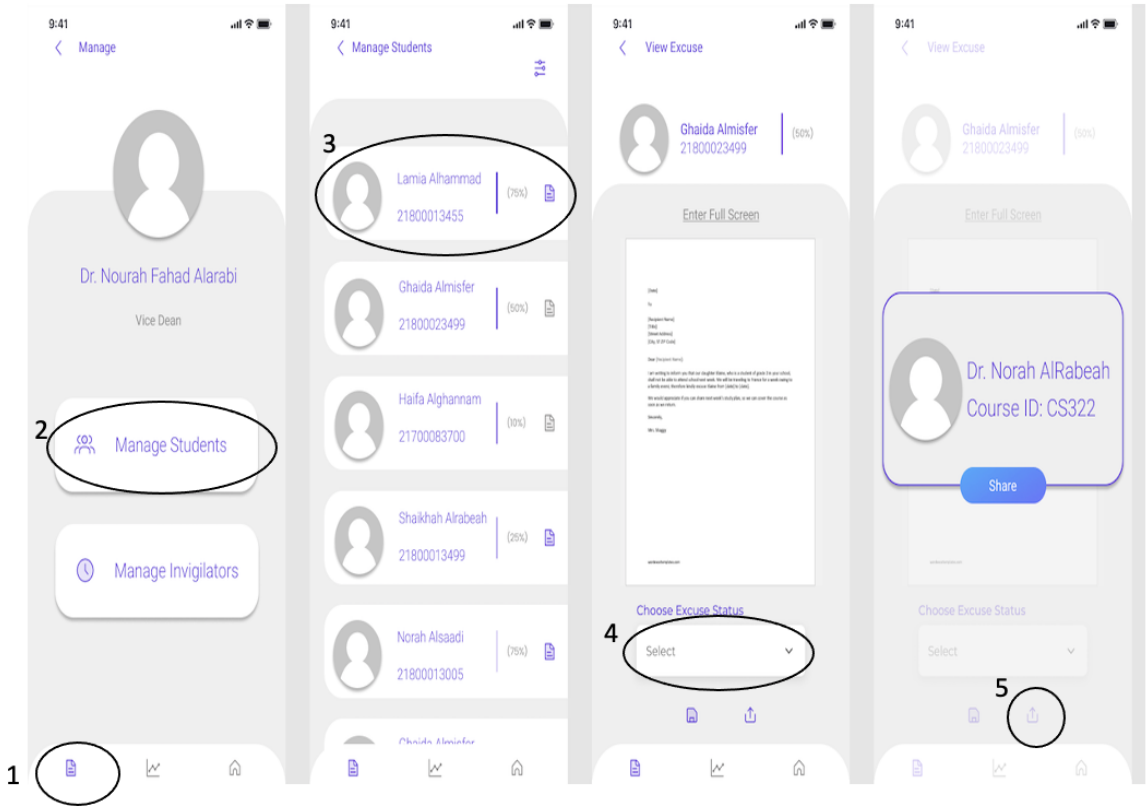
- Manage Exams:
  1. Enter Manage icon.
  2. Enter Manage Invigilators.
  3. Click the + sign.
  4. Choose the Invigilator name, Course ID, and Room Number, write subject Name, Pick date, start Time, and End Time, then click Submit.
  5. Admin will receive pop up message says, “Your changes have been successfully saved!”
  6. After adding the exam, it will show up in the exams list.

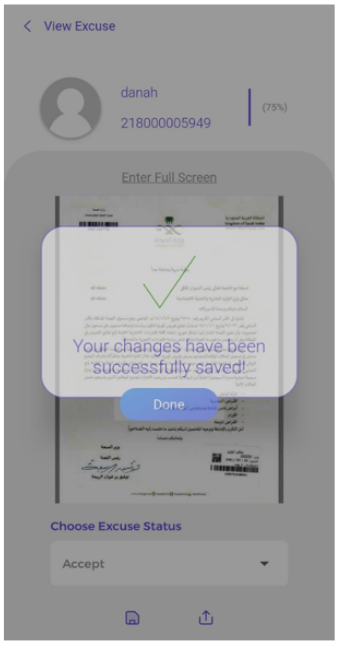






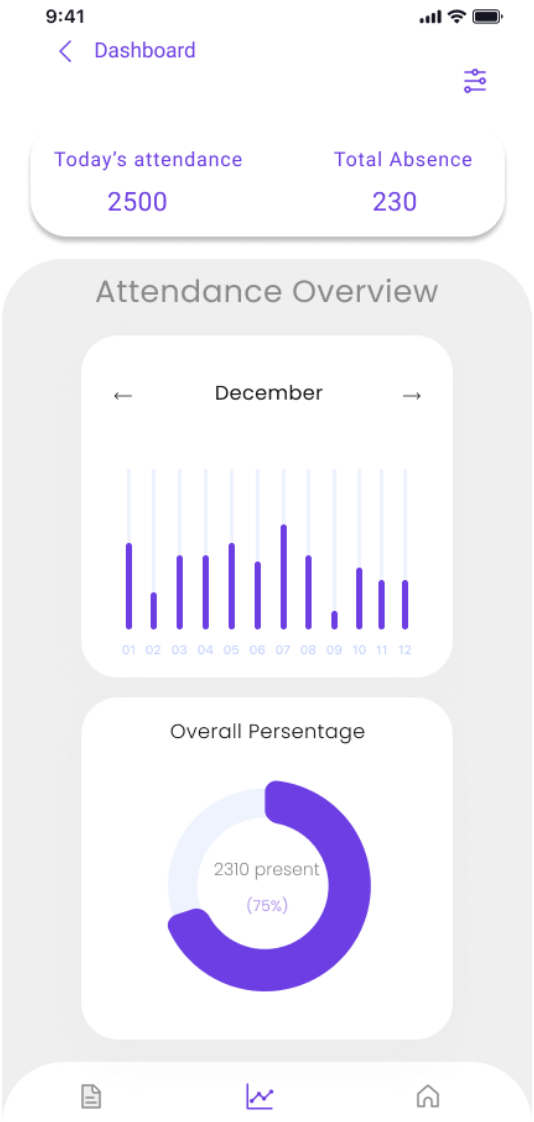
- Manage students:
  1. Enter Manage icon.
  2. Enter Manage students.
  3. Admin will see the absent students list who upload the excuse, select name.
  4. The Excuse will appear as PDF, then Choose Excuse Status.
  5. Admin can share the excuse to the instructor course.
  6. Admin will receive pop up message says, “Your changes have been successfully saved!”


9:41
Manage
Dr. Nourah Fahad Alarabi
Vice Dean
2
Manage Students
Manage Invigilators
1
Manage Students
3
Lamia Alhammad
21800013455
(75%)
Ghaida Almisfer
21800023499
(50%)
Haifa Alghannam
21700083700
(10%)
Shaikhah Alrabeah
21800013499
(25%)
Norah Alsaadi
21800013005
(75%)
View Excuse
Ghaida Almisfer
21800023499
(50%)
Enter Full Screen
Choose Excuse Status
4
Select
Dr. Norah AlRabeah
Course ID: CS322
Share
5


View Excuse
danah
218000005949
(75%)
Enter Full Screen
Your changes have been successfully saved!
Done
Choose Excuse Status
Accept

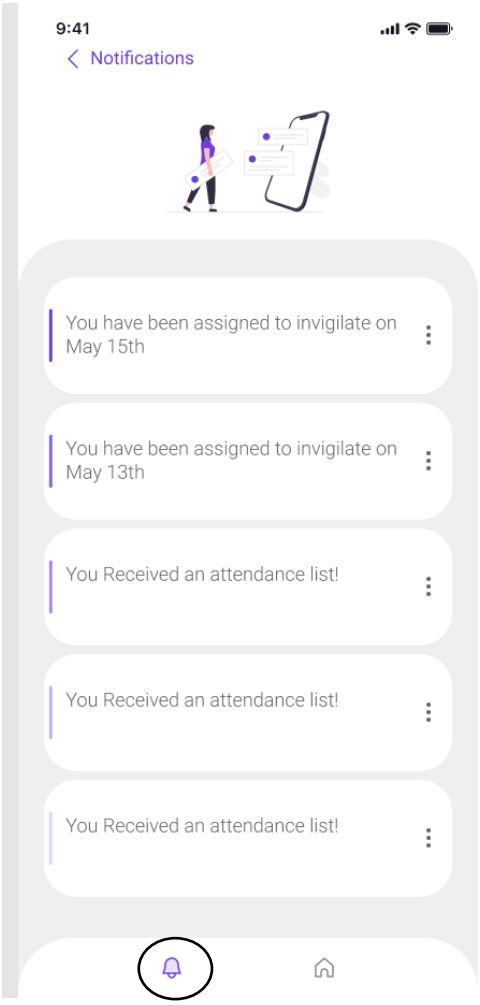
- Dashboard

In the dashboard will display today's attendance total attendance and attendance overview.



INSTRUCTOR USER

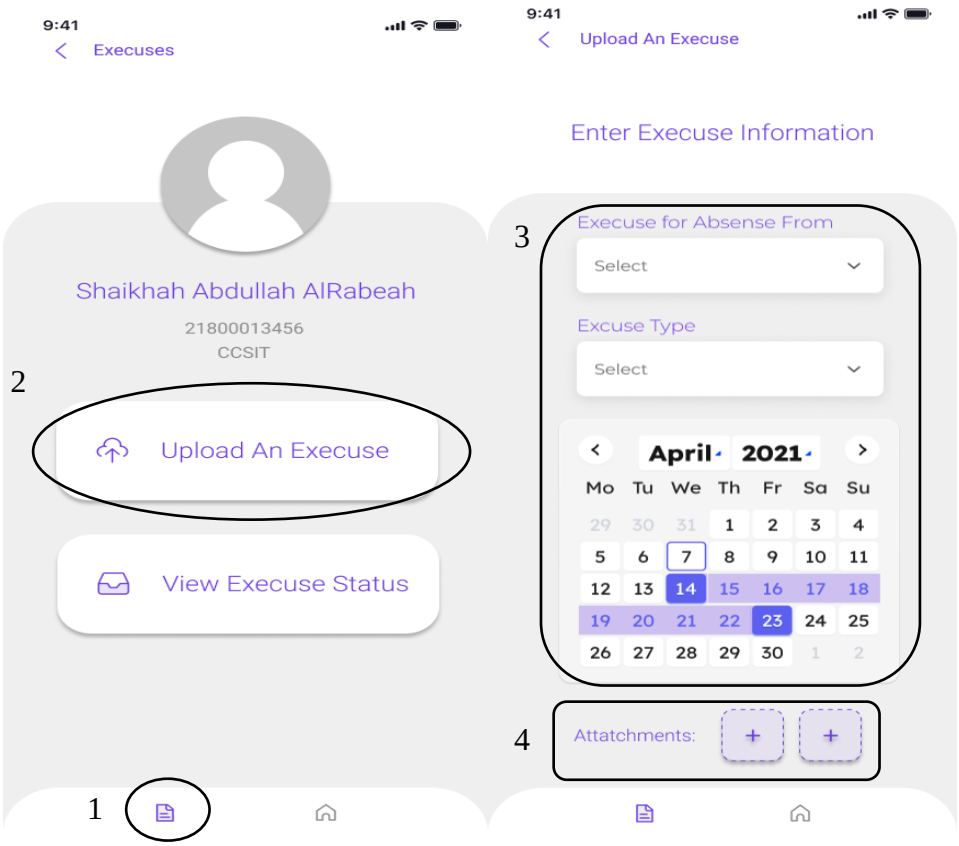
After sign the Instructor can:
View Notification: by Enter Notification icon.



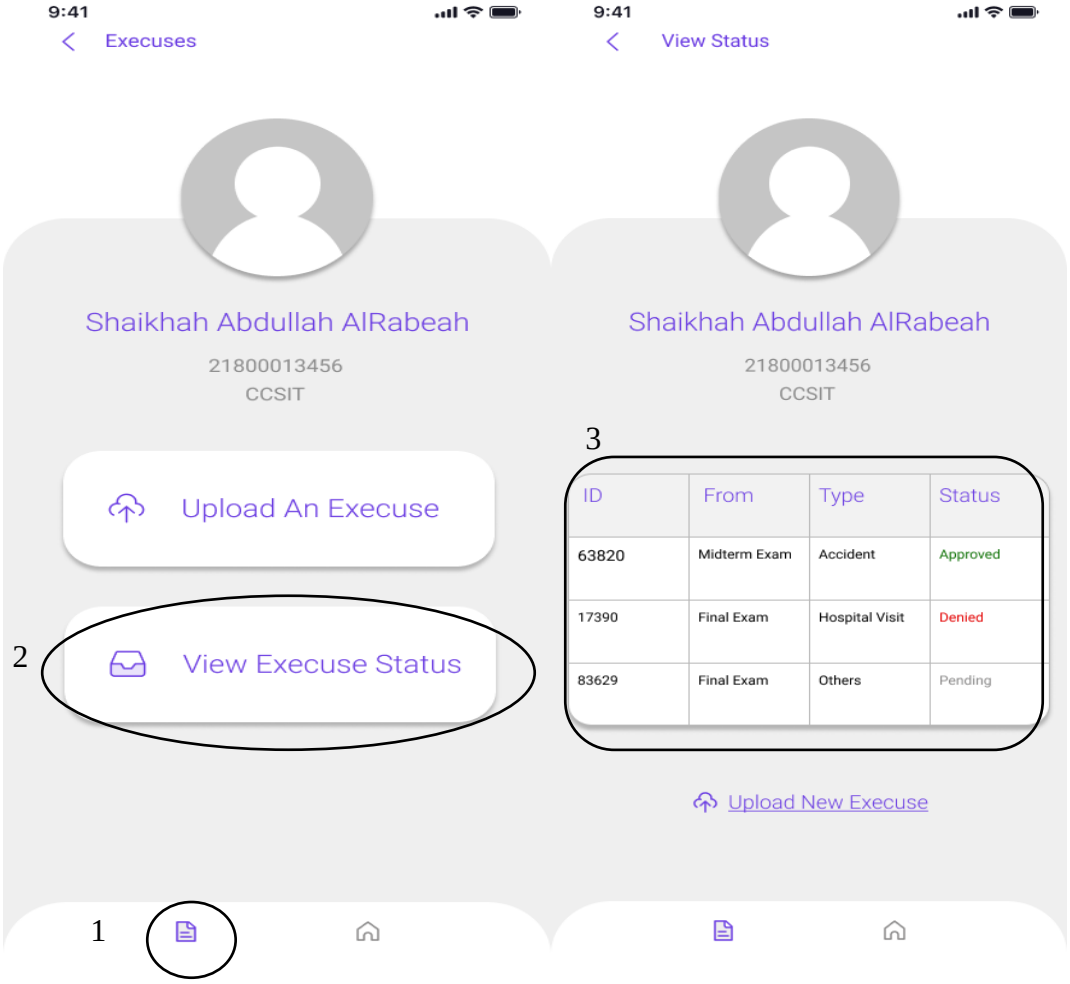
STUDENT USER

After sign the Student can:

- Upload An Excuse.
    1. Enter Excuse icon.
    2. Click Upload An Excuse.
    3. Select Excuse for Absent From, Excuse Type, and Start date and End date.
    4. Attach the Excuse in the form of a PDF.
    5. Click Submit.
    6. Student will receive pop up message says, “Your changes have been successfully saved!”



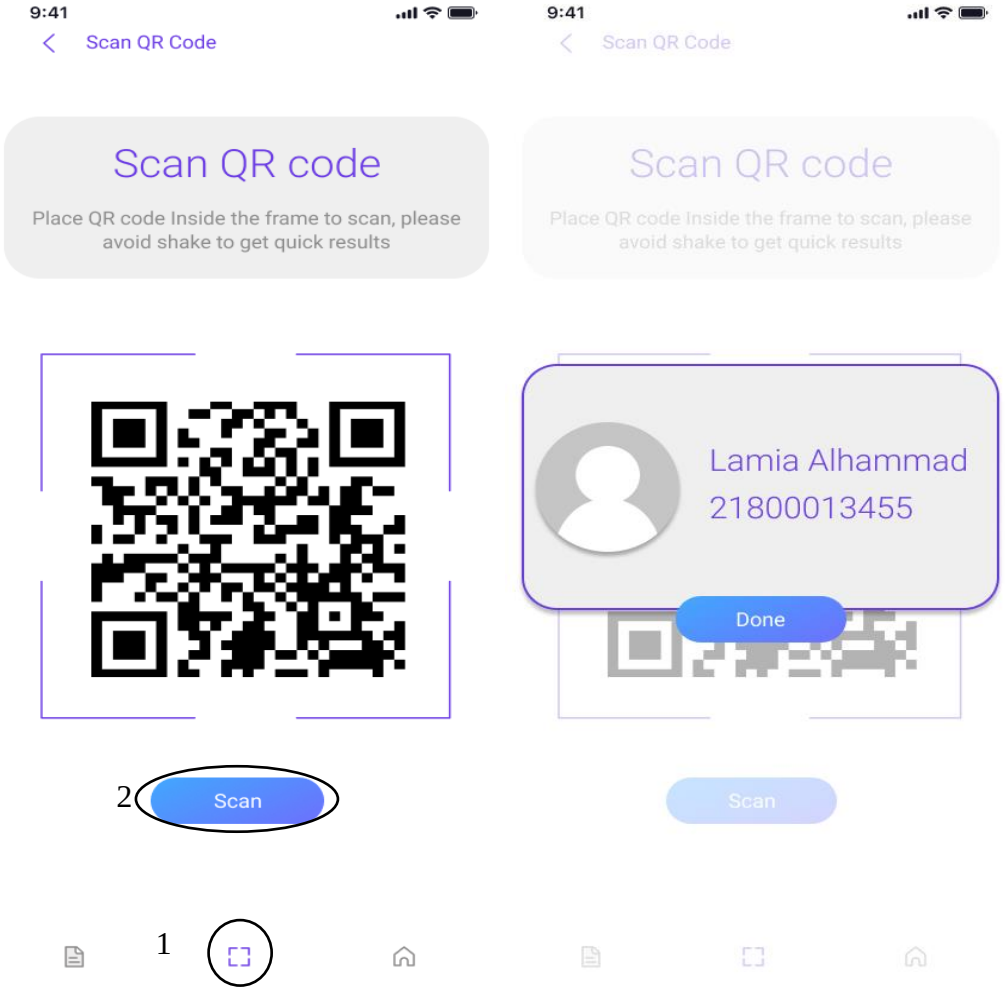
- View Excuse Status
    1. Enter Excuse icon.
    2. Click View Excuse Status.
    3. Excuse Status will be shown, and Student can also Upload a New Excuse.

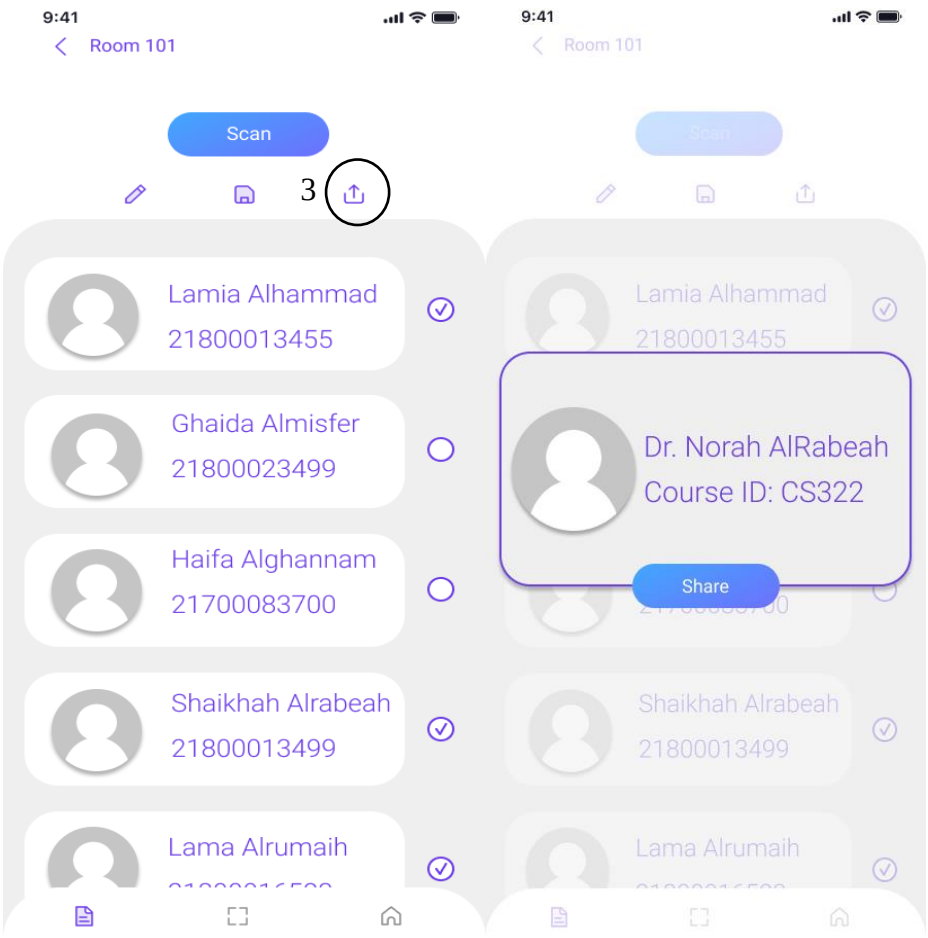


INVIGILATOR USER

After sign the Invigilator can:

- Scan Student QR code.
  1. Click Scan QR code icon.
  2. Scan Student QR code.
  3. Invigilator will receive pop up message says, “Your changes have been successfully saved!”
  4. Invigilator can edit and share the list to the instructor course.




9:41
Room 101
Scan
3
Lamia Alhammad
21800013455
Ghaida Almisfer
21800023499
Haifa Alghannam
21700083700
Shaikhah Alrabeah
21800013499
Lama Alrumaih
9:41
Room 101
Lamia Alhammad
21800013455
Dr. Norah AlRabeah
Course ID: CS322
Share
Shaikhah Alrabeah
21800013499
Lama Alrumaih

- Search for an exam date.
  1. Click Schedule icon.
  2. Select the date that the Invigilator are looking for.
  3. Click search.
  4. exam list will be shown.