University of East London

SCHOOL OF ARCHITECTURE, COMPUTING & ENGINEERING

Department of Computer Science and Digital Technologies – CDT

CN7021 – Advanced Software Engineering

Hamro Tuition Management System

Group: 1.**D**

Students Name & ID: Abdul Azeem - u2776658
Students Name & ID: Dipin Bhandari - u2784783
Students Name & ID: Amandeep - u2776903
Students Name & ID: Sai Manish Gandra - u2778998

Tutor: **Reena Popat**

Module Leader: **Dr Hisham AbouGrad**

Month Year

23 December 2024

Table of Contents

Table of Figures.

Table of Tables.

1.Introduction.

The "Hamro Tuition management system" paper lists the hardware, software, and major features that a Hamro Tuition management system must have. Functional needs include managing student and teacher information, user registration, login features, creating random questions for practice exams, creating test results and PDF downloads, accessing online books, bookmarking, email notifications, and a fee payment mechanism are all covered. Web browser compatibility and internet access are among the hardware requirements, while Python, Django, HTML/CSS (Bootstrap), JavaScript, and Visual Studio Code are among the software requirements. Additionally, the document outlines requirements for security, interoperability, maintainability, scalability, performance, usability, and dependability to make that the system satisfies the demands of administrators and users. Furthermore, a thorough list of actors and use cases is supplied for every system function, including roles for administrators, instructors, and students. (smartclasses.in, 2022)

1.1 Purpose.

The goal of the "Hamro Tuition Management System" is to create an organized and efficient framework for managing the core functions of a Hamro Tuition Centre. This includes features including fee processing, access to online resources, user registration, the ability to take sample tests, management of students and teachers, and notifications. The system is designed to enhance user experience and operational effectiveness with exacting hardware and software specs. The system distributes tasks and responsibilities to users, including administrators, teachers, and students, in order to handle tuition-related tasks in a reliable, secure, and scalable way.

1.2 Document Conventions.

When developing and documenting this Hamro Tuition management system project, the following guidelines have been followed for uniformity and clarity:

- Bold Text
- Arial Fonts
- Versioning: The project and document revision history will be preserved in the GitHub repository.

- Priority Notation: Utilizing the agile technique. We designate tasks as completed, in progress, and not completed on the scrum desk.

1.3 Intended Audience and Reading Suggestions

The following groups are targeted by the "Hamro Tuition Management System" document:

a) Project Managers and Stakeholders.

purpose: To comprehend the goals, requirements, and extent of the project.
Reading suggestions: To comprehend the project's goals, needs, and scope Pay attention to the system's introduction, purpose, high-level needs, and general objectives.

b) Developers and Engineers.

purpose: To fully comprehend hardware/software specifications, system architecture, and functional and non-functional requirements.
Reading suggestions: Pay close attention to the system's introduction, purpose, high-level needs, and general objectives.

c) QA/Testers.

purpose: To comprehend the system's anticipated performance and functionality requirements, allowing for efficient testing and validation.
Reading suggestions: Pay close attention to the use cases, performance and reliability requirements, and functional needs.

d) System Administrators.

Purpose: To comprehend the system's intended performance and functionality standards, allowing for efficient testing and validation.
Reading suggestions: Pay close attention to the use cases, performance and reliability requirements, and functional needs.

e) End Users (Teachers, Students, and Administrators).

Purpose: To comprehend the maintenance and operating needs of the system.

Reading suggestions: Look through the sections on compatibility, security, hardware requirements, and maintainability.

1.4 Project Scope.

A complete platform for managing the main functions of a Hamro Tuition center, such as managing teachers and students, creating and evaluating mock exams, registering and authenticating users, providing resource access, sending out alerts, and processing fee payments, is what the "Hamro Tuition Management System" seeks to offer. Web browsers will be able to access the system, which will be made to be user-friendly, scalable, and secure to meet the demands of educators, administrators, and students.

Purpose: To improve productivity, accessibility, and the user experience for administrators, teachers, and students by digitizing and streamlining Hamro Tuition center operations.

Goal: To provide Hamro Tuition centers with a dependable, safe, and expandable management system that streamlines administrative duties, boosts user involvement, and meets their operational and academic requirements.

1.5 Requirements References Documents.

1. “Hamro Tuition Management System using Django Framework” Repository

Link: https://github.com/Ashishdeuja/Hamro-Tuition

Author: Ashish Deuja

Date: 2022

2.Overall Descriptions.

2.1 Software Architecture and methodology.

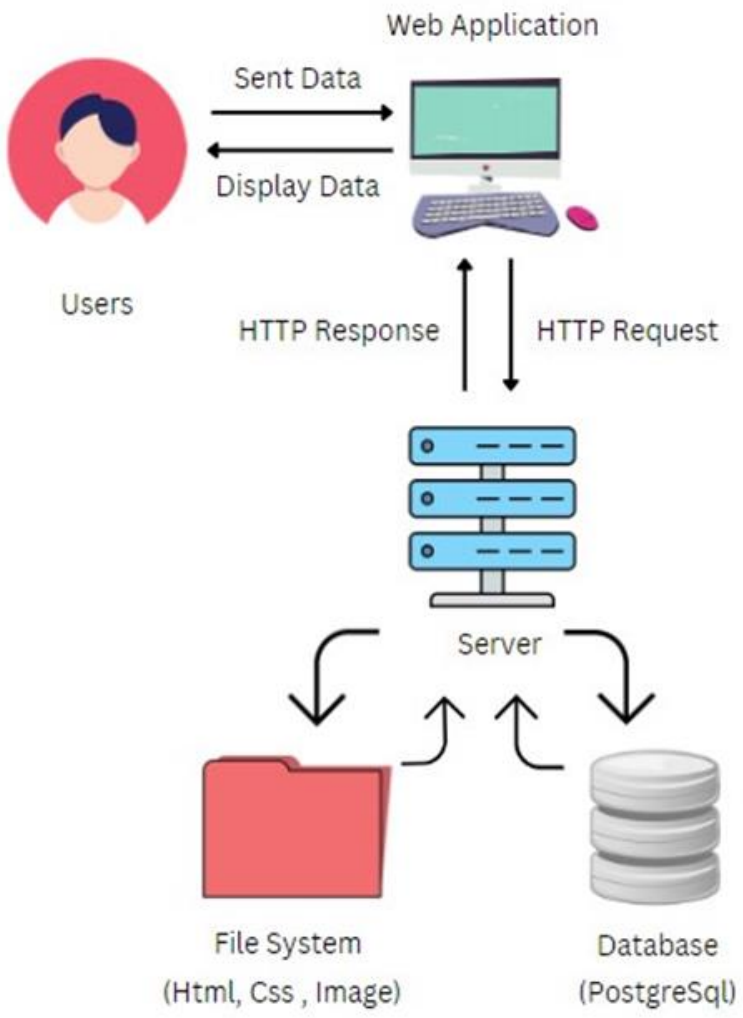


Figure 1: Software Architecture.

User hits the websites for the particular action and the website sent http request to the server and the server looks the database and file system and fetch the data as per required for user. Then the http responses to the server and server display the information at web and the user get required data as per their requirements. (Pragy Agarwal, 2023)

Methodology (Agile-Scrum).

Scrum is a preferred Agile methodology for handling and finishing challenging projects. It places a strong emphasis on collaboration, responsibility, and iterative advancement toward clear objectives. (scrum.org, 2024)

(scrum.org, 2024)

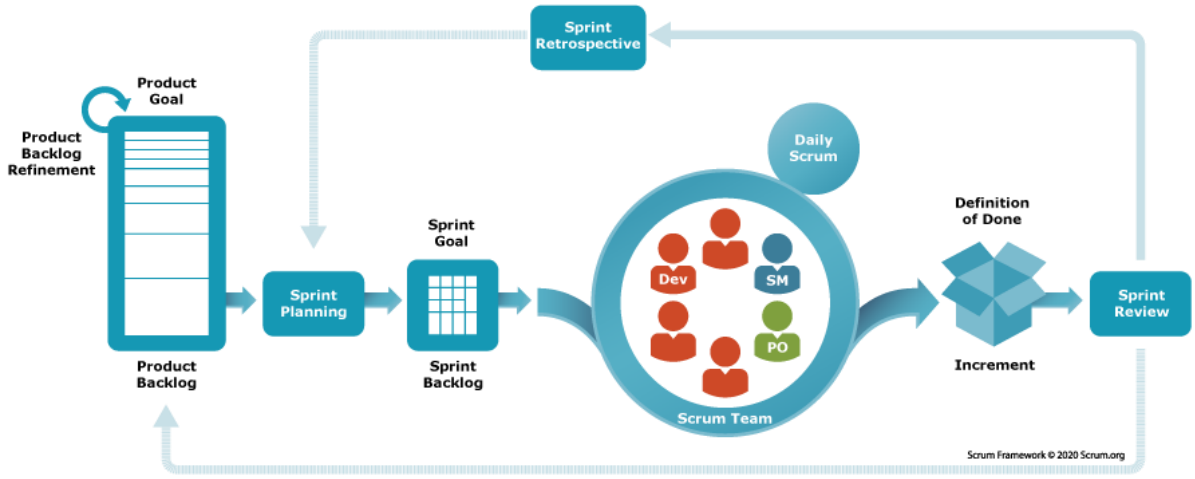


Figure 2: scrum methodology.

Reasons to Select Agile Scrum for projects.

- **Adaptability to changing Requirements:** Enables backlog modifications to correspond with changing requirements, such as the addition of new functionality.
- **Incremental Delivery:** enables early testing and feedback by delivering usable features at the conclusion of each sprint.
- **Enhanced Collaboration**: promotes collaboration and frequent input in order to satisfy stakeholders.
- **Transparency and Visibility**: aligns progress with goals by providing transparent updates through sprint reviews and daily standups.
- **Risk Mitigation**: Early risk identification and mitigation is achieved by frequent testing and gradual delivery.

(agilest.org, n.d.)

Phases of Scrum Agile Methodology's.

- **Project Initiation:** Establish a preliminary product backlog, define the product vision and identify stakeholders.
- **Sprint Planning:** Decide which backlog activities are most important and establish objectives for sprint planning.
- **Sprint Execution:** During the sprint, team members develop, test, and deploy features.
- **Daily Standups:** Organize brief meetings to go over accomplishments, difficulties, and future work.
- **Sprint Review**: Show stakeholders the finished product and get their input.
- **Sprint Retrospective**: To determine the sprint's achievements and potential areas for development, reflect on it.
- **Product Increment Delivery**: After every sprint, provide a workable system increment for stakeholders to use or assess.
- **Project Completion**: Complete development, verify the system, and put it into production.

(Rehkopf, 2024)

2.1.1 Class Diagram

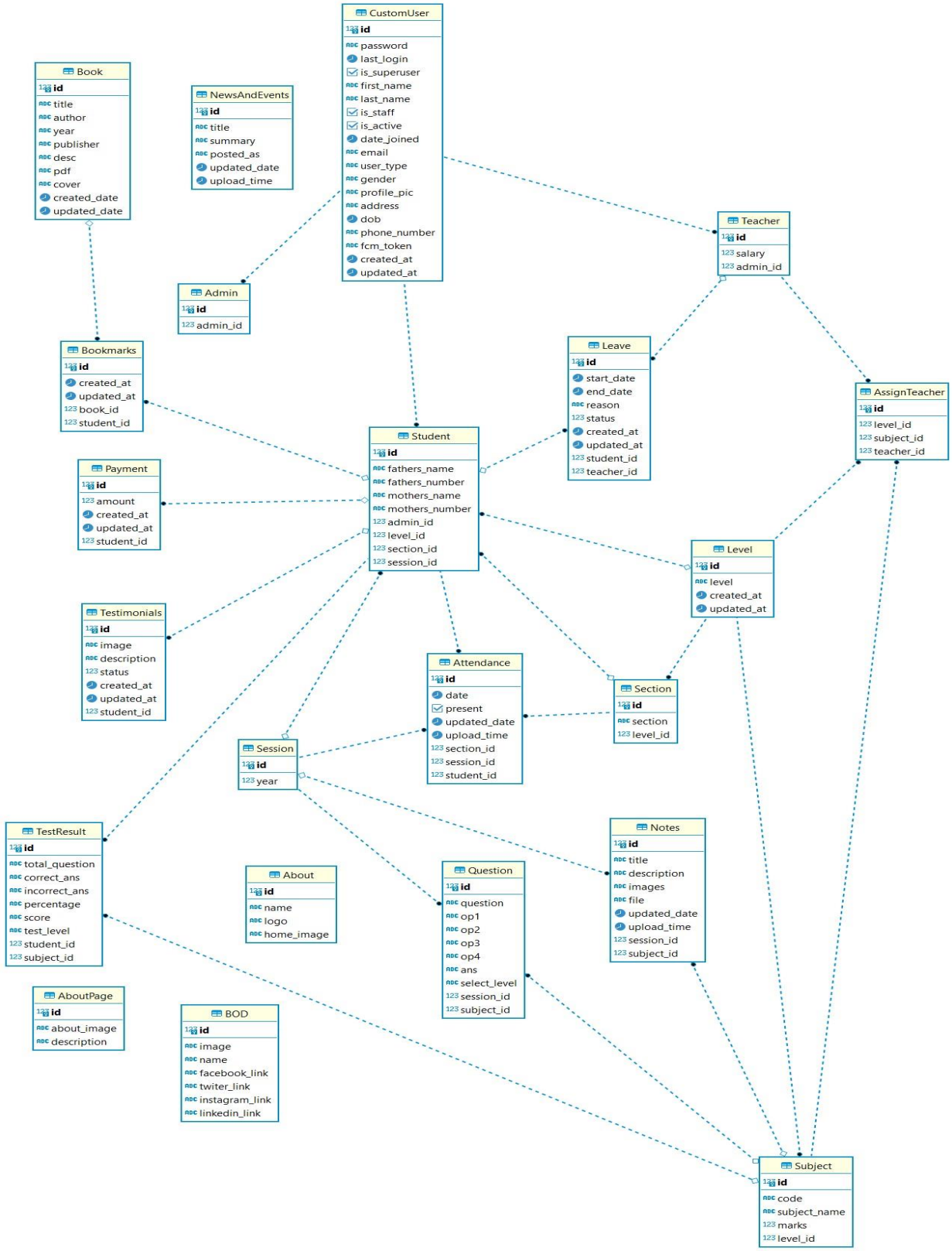


Figure 3: Class Diagram.

2.2 Product Functions and Use case.

1) Student Function.

a) Account Management:

- Register as a user.
- Log in to the system.

b) Academic Resources:

- View notes and books.
- Add bookmarks for quick reference.

c) Testing and Evaluation:

- Take mock tests.
- View and generate mock test results.

d)Attendance and Leave:

- View attendance records.
- Apply for leave.

e) Payment:

- Make payments for Tuition fees.

f) Notifications:

- View notifications for updates or announcements.

g) Error Handling:

- Receive error messages for invalid categories or actions.

2) Teacher Functions.

a) Content Management:

- Add and manage notes for students.

b) Testing Support:

- Add and manage questions for mock tests.

c) Attendance Management:

- Take and record attendance for students.

3) Admin Functions.

a) User Management:

- Enroll and manage students and teachers.
- Academic Resource Management:
- Add and manage classes, sections, and subjects.
- Add and manage books.

b) Communication and Notifications:

- Post notices and announcements.
- Push notifications to users.

c) Feedback and Testimonials:

- Manage feedback and testimonials.

d) Attendance and Leave:

- Generate attendance reports.
- Handle leave management for students and teachers.

e) Error Handling and Alerts:

- Display error messages and send alerts.

Use Case Diagram.

Figure 4: Use Case Diagram.

The actors and use cases of the tuition management system.

S.N.	Use Case	Actors
1	Login	Admin, Student, Teacher
2	View Notes	Student
3	View Book	Student
4	Give Mock Test	Student
5	Add Bookmarks	Student
6	Provide Testimonials	Student
7	View Mock Test Result	Student
8	Make Payment	Student
9	View Attendance	Student, Teacher, Admin
10	Apply for Leave	Student, Teacher
11	View Notification	Student, Teacher
12	Add Notes	Teacher
13	Manage Notes	Teacher
14	Take attendance	Teacher
15	Add Question	Teacher
16	Manage Question	Teacher
17	Add Class	Admin
18	Manage Class	Admin
19	Add Section	Admin
20	Manage Section	Admin
21	Add Subject	Admin
22	Manage Subject	Admin
23	Enroll Teacher	Admin
24	Manage Teacher	Admin
25	Enroll Student	Admin
26	Manage Student	Admin
27	Add Book	Admin, Teacher
28	Manage Book	Admin, Teacher
29	Generate Attendance Report	Admin, Teacher, Student
30	Push Notification	Admin
31	Manage Leave	Admin
32	Push Notices and Announcements	Admin

Figure 5: Actor of Use Case Diagram.

2.3 User and Stakeholder.

The complete platform known as the Hamro Tuition Management System was created to simplify and automate a Hamro Tuition center's fundamental activity. This system seeks to increase the center's operational effectiveness while improving the user experience for administrators, teachers, and students.

1. Administrators.

Role: The administrator is responsible for managing and organizing every operation while making ensuring they run smoothly. They are able to access and control any part of the system.

Characteristics:

a. complete control of all system features.
b. the ability for creating and control user roles.
c. announcements, and monitor system-wide operations.

2. Teacher.

Role: Teachers create and manage courses, upload learning resources, and design mock tests. They track students' progress, provide feedback, and communicate with students about course updates, test schedules, and other important information.

Characteristics:

a. They manage attendance and collaborate with administrators.
b. To ensure the smooth operation of the courses and resources.
c. They track students' progress, provide feedback, and communicate with students.

3. Student.

Role: Student is responsible for engaging with the platform to access learning materials, participate in mock tests, track their progress, and manage their academic.

Characteristics:

a. View and download study materials, resources, and lectures.
b. Take and complete mock tests or quizzes assigned by teachers.
c. Monitor their performance in tests, assignments, and overall course progress.

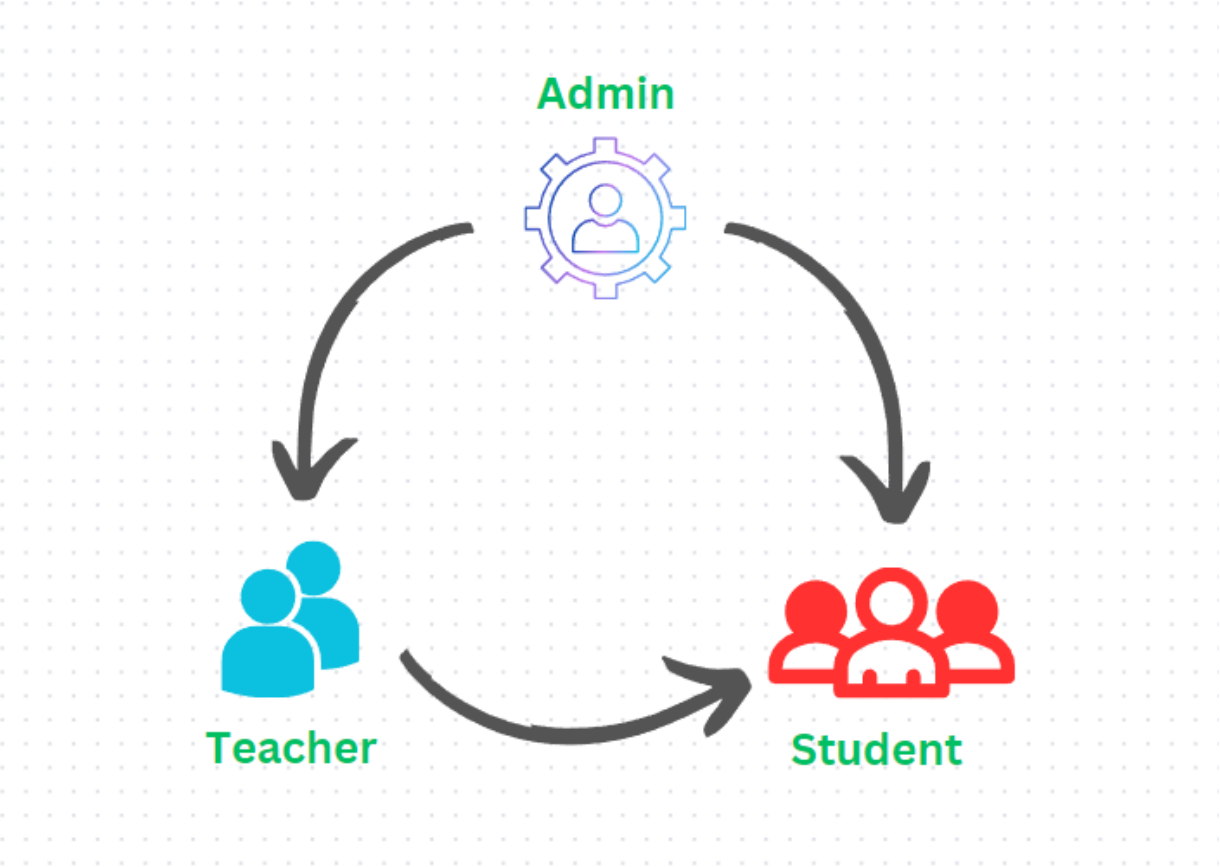


Figure 6: User and Stakeholder.

2.4 Operating Environment.

Hardware Platform

A hardware system for a Hamro Tuition management system consists of the physical devices required to support the operation, data processing, and communication of the system. (Rouse, 2017) Here's a breakdown of the hardware components typically used in such a system:

1. **Server Hardware.**

- Requires sufficient processing power (multi-core CPUs) and high RAM to handle multiple concurrent users.
- High storage capacity, fast processing power, and robust networking capabilities.

2. **Client Devices.**

- Devices like desktops, laptops, tablets, or smartphones used by administrators, tutors, and students.
- Basic processing capabilities, moderate RAM (4GB–16GB).

Operating System.

The fundamental software that controls computer hardware and software resources is called an operating system (OS). The OS plays a vital role in ensuring smooth and efficient system operations. (Shacklett, 2024)

1. **Server Operating System.**

- Window server, it is user-friendly and integrates well with Windows-based environments.
- Linux Distributions such as Centos, kali Linux, parrot, Ubuntu for hosting the Hamro Tuition management system on Django Application.

2. **Client Operating System.**

- Windows operating system like 7, 10, 11 for laptop and desktop.
- Any web-browser such as Google Chrome, Brave, Mozilla Firefox, Opera Mini.

Software Components.

Several software components are necessary for a Hamro Tuition management system to operate well.

1. **Django Framework.**

- Version used for this project is **Django==4.2.1** (docs.djangoproject.com/, 2024)

2. **Database Management System.**

- SQLite for data storage and retrieval data. (docs.djangoproject, 2024)

3. **Programming Language.**

- Python (**24.3.1 version**) for backend development system. (docs.python.org, 2024)

4. **Frontend Technologies.**

- Html, CSS, and JavaScript is used for frontend libraries framework for development. (w3schools, 2024)

2.5 System Design and Implementation Constraints.

Based on the Hamro Tuition Management System scenario, the following is a thorough analysis that covers institutional policies, data protection laws, hardware constraints, software technologies, maintenance, and support: (Al-Sarayreh, 2010)

1. **Data Privacy Regulations.**

- GDPR protects personal data of EU citizens with rights like consent and deletion.
- CCPA ensures transparency and user control over data for California residents.
- FERPA safeguards educational records of students in the U.S.
- Implementation includes encryption, privacy policy, and user consent.

2. **Institutional Policies.**

- Role-based access ensures only authorized users handle specific data.
- Strong authentication methods secure user accounts.
- Data retention policies specify how long records are stored.
- Incident response outlines steps for breaches or system failures.

3. **Hardware Limitation.**

- Server CPU, RAM, and storage must handle the system's workload.
- Reliable network infrastructure ensures fast system performance.
- Compatibility with older devices broadens accessibility for users.

4. **Software Technologies.**

- Django handles back-end logic effectively.
- SQLite store structured data offers flexibility.
- HTML, CSS, and JavaScript provide interactive user interfaces.
- APIs like Stripe for payments and Twilio for communications.

5. **Maintenance and Support.**

- Quickly resolve bugs for uninterrupted service.
- Update software and frameworks to fix vulnerabilities.
- Schedule regular data backups to avoid loss.
- Offer helpdesk and tutorials for user support.
- Monitor system health with tools like Nagios or Zabbix.

2.6 User Documentation.

The user manual for Hamro Tuition management system with the extracurricular activities function to help users comprehend and make efficient use of the system. The lists that follow:

Steps to Run the Project

1. **Install Python**

   Download and install Python from the official Python website. Ensure Python is added to the system PATH.

2. **Install Visual Studio Code (VS Code)**

   Download and install VS Code from the official VS Code website.

3. **Open Git Bash in VS Code**

   Launch VS Code and open the Git Bash terminal integrated into VS Code.

4. **Install Required Dependencies**

   In the project folder, locate the requirements.txt file.

   Use the following commands in Git Bash to install the dependencies:

bash

Copy code

./env/Scripts/activate

pip install -r requirements.txt

5. **Activate Virtual Environment**

   To activate the virtual environment, enter the following command:

bash

Copy code

./env/Scripts/activate

6. **Navigate to the Project Folder**

   Use the cd command to move into the project folder:

bash

Copy code

cd Hamro_Tuition

7. **Run the Django Server**

   Use the following command to start the server:

bash

Copy code

py manage.py runserver

This will display a localhost URL (e.g., http://127.0.0.1:8000/).

8. **Access the Tuition Management System**

   Open a web browser and navigate to the provided localhost URL.

   You will see the "Hamro Tuition Management System" login screen.

9. **Log in to the System**

   Enter your credentials to log in and start using the system.

2.7 Assumptions and Dependencies.

Assumptions	Description.
User Roles	Users (administrators, teachers, and students) will have access to the internet and devices like desktops, laptops, or smartphones.
Internet Access	The system assumes that all users have reliable internet access to use the online features
Supported Browsers	The system will work on modern web browsers like Chrome, Firefox, Edge, Brave and Safari.
Skill Level	Users are expected to have basic knowledge of using web-based applications.
Payment Gateway	The payment gateway provider ensures secure and seamless transaction processing.
Data Volume	The initial deployment assumes manageable data volumes with the potential for future scaling.
Development Tools	Developers will use the stated software stack (Python, Django, HTML/CSS, JavaScript, etc.) for implementation.
Email Services	The email notification feature assumes access to a reliable email service provider.

Dependencies	Description.
Hardware Requirements	The system depends on compatible hardware like servers, user devices, and reliable networking equipment.
Software Stack	The project relies on the integration of Python, Django, JavaScript, and Bootstrap for functionality.
Database Systems	A compatible relational database SQLite must be set up and maintained.
Third-Party Services	External services like payment gateways, email servers, and cloud storage are crucial for certain functionalities.
Security Protocols	The system depends on SSL/TLS protocols for secure communication and data encryption.
Hosting Environment	A reliable hosting platform with adequate server resources is required for deployment.
Legal Compliance	The system relies on adherence to data protection laws (e.g., GDPR) for handling user data.
User Feedback	Continuous user feedback is essential for maintaining and improving system performance.

3. External Interface Requirements.

3.1 User Interfaces.

The interfaces of the Hamro Tuition management system should be easy to use and straightforward for administrators, teachers, and students. According to the use case diagram, these interfaces will allow for smooth interaction with the system's features.

Admin Interfaces:

1. Admin Login Screen.

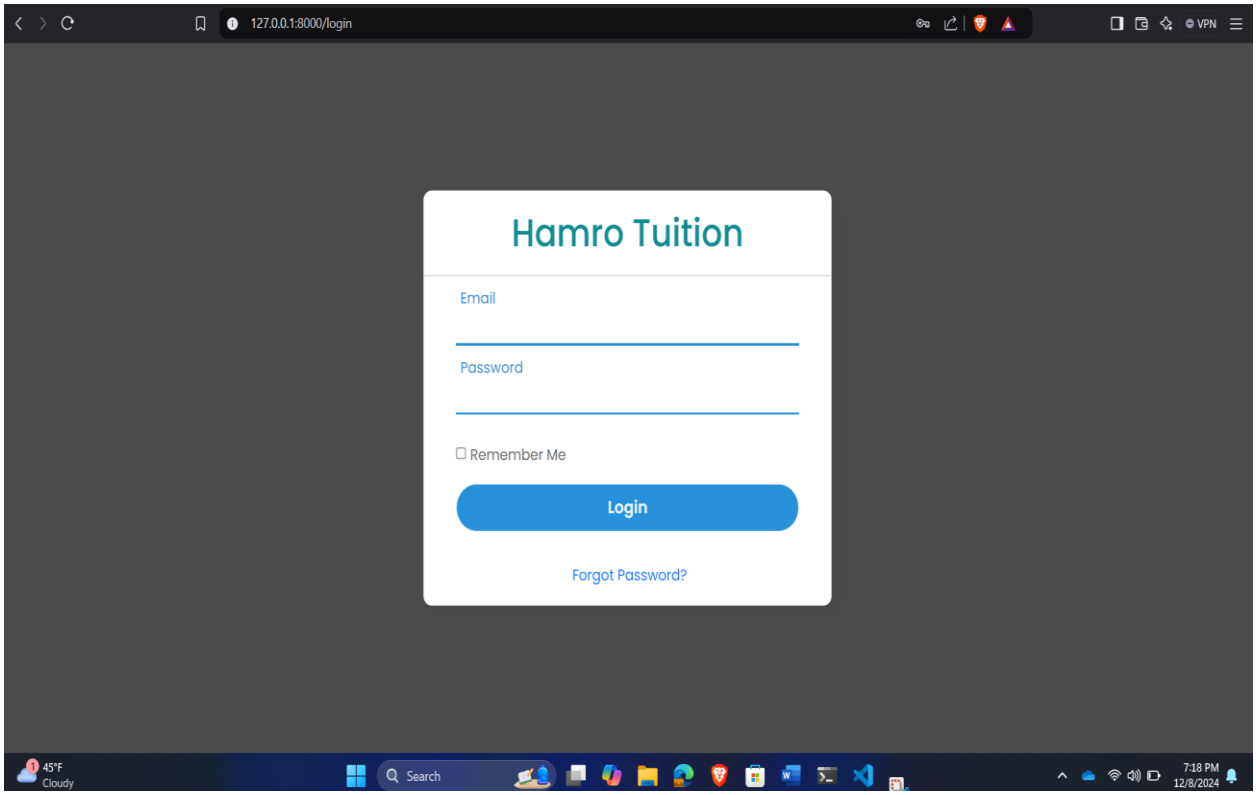


Figure 7: Admin Login Screen.

2. Admin Dashboard.

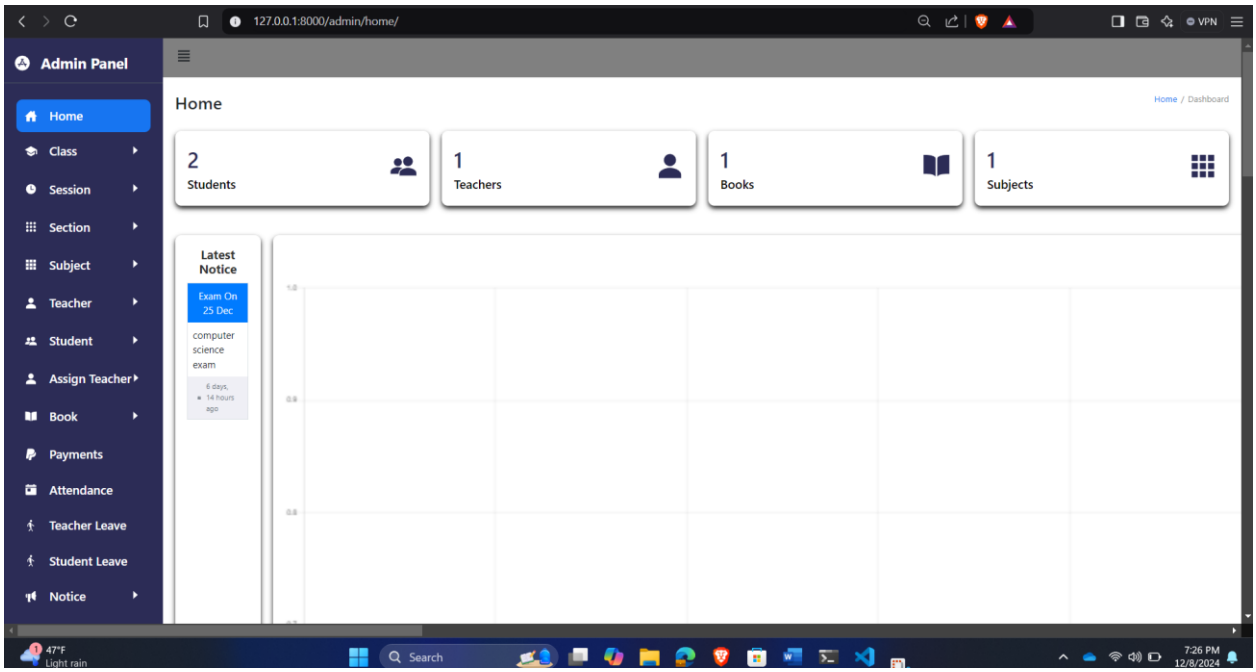


Figure 8: Admin Dashboard.

Teacher Interfaces:

1.Teacher Dashboard.

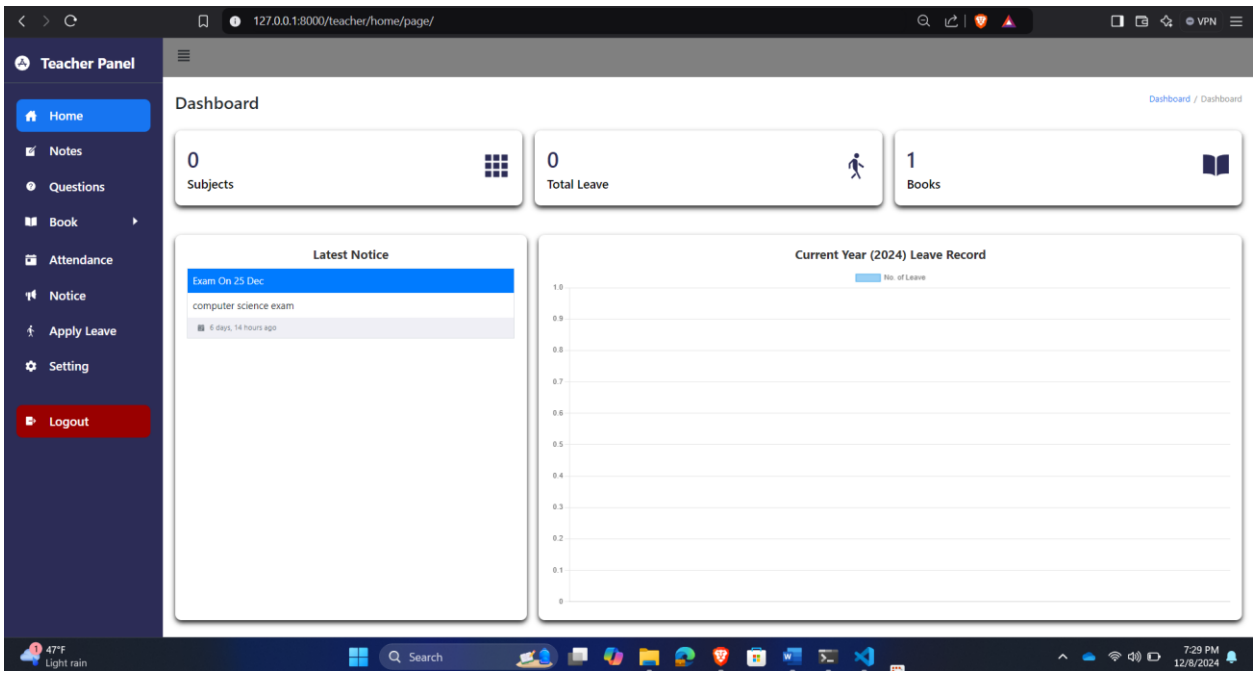


Figure 9: Teacher Dashboard.

Student Interfaces:

1. Student Dashboard.

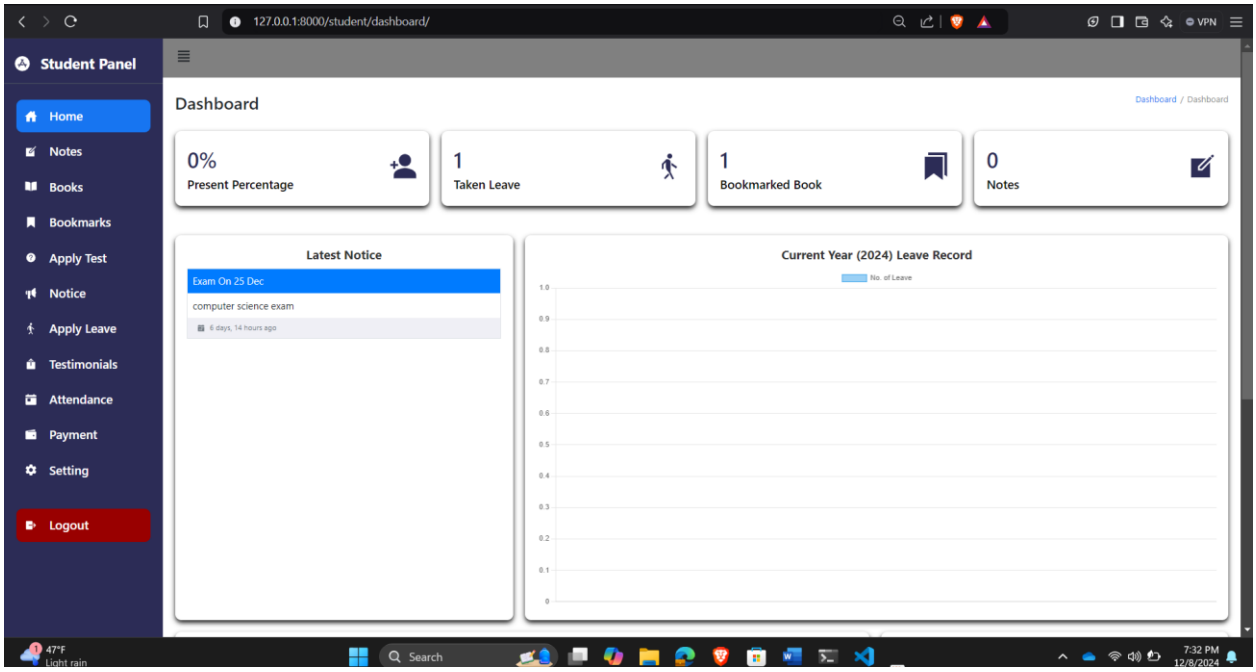


Figure 10: Student Dashboard.

3.2 Hardware Interfaces.

The Hamro Tuition Management System is a web-based platform that needs simple hardware setups to work smoothly. The main hardware interfaces that the system requires are listed-below.

Hardware	Minimum	Recommended
Processor	Intel Core i3	Intel Core i5
Ram	8GB	16GB
solid-state drive (SSD)	128GB	512GB
Graphics Card	Intel UHD Graphics 620/630	NVIDIA GTX 1650

3.3 Software Interfaces.

Below is a detailed breakdown of the required software interfaces based on the system's functionalities.

Software	Version
IDE	Visual Studio
Backend	Python 24.3.1
Django Framework	4.2.1
Frontend	Html, CSS, & JavaScript
Database	SQLite

3.4 Communication Interfaces.

To facilitate data transmission and interaction between users, system components, and external services, the Hamro Tuition Management System needs effective communication interfaces. (Woolf, 2009)

1. **Web-Based Communication:** Students, teachers, and administrators can communicate securely online with the server using HTTP/HTTPS.
2. **Database Communication:** Django ORM (Object-Relational Mapper) for database queries and operations.
3. **Network server communication:** Transmission control protocol/internet protocol (TCP/IP) enables dependable communication and effective data transfer between client machines and servers.

4. System Requirements.

4.1 Requirement F1.

Table 1:F1: Create Superuser (Administrator).

F1: Create Super User (Administrator)	
Priority	Must
Description	Enable the creation of a super-user who will have complete administrative control over the system.
Input/Outputs Sequences	Utilize the Django framework to generate a super-user. **Input:** Python commands: py manage.py createsuper user. Enter Email, Enter Password. **Output:** Success message 'superuser created successfully'.
Functional Requirements	F1.1: Users must input a valid email address. F1.2: Users must enter and confirm the password. F1.3: System provides confirmation of successful creation.

4.2 Requirement F2.

Table 2: F2: Attendance Management by Teacher

F2: Attendance Management by Teacher	
Priority	Must
Description	Enable teachers to view, mark, edit, and generate reports of attendance records for a selected class or section.
Input/outputs	**Input:** Search for a student, select date range, status (Present/Absent), or class section. Mark attendance or edit existing records. **Output:** Updated attendance status, filtered attendance records, or exported attendance reports.
Functional Requirements	**F1.1:** Teachers must input valid details (student name, section, or date range) for search or filters. **F1.2:** The system must allow marking attendance for all students in a selected section for a given date. **F1.3:** The system must enable editing existing attendance records to correct errors. **F1.4**: Attendance reports must be exportable in formats like PDF or Excel.

4.3 Requirement F3.

Table 3: F3: Apply for Leave Student.

F3: Apply For Leave Student	
Priority	Must
Description	Enable students to apply for leave and view the status of their leave requests.
Input/outputs	**Input:** Student selects start date, end date, and enters a leave message. **Output:** The leave request is submitted and appears in the Leave History table with a status
Functional Requirements	**F3.1:** Students must input valid start and end dates for the leave period. **F3.2:** Students must provide a message or reason for the leave request. **F3.3:** The system must store and display leave requests in the Leave History table. **F3.4:** Students should be able to filter or search leave requests by date or status. **F3.5:** The system must notify the student when the leave request status changes.

4.4 Requirement F4.

Table 4: F4: Payment Khalti is integrated for student Tuition Fee.

F4: Payment for Student Tuition Fee.	
Priority	Must
Description	Allow students to pay their tuition fees securely using the Khalti wallet integration.
Input/outputs	**input:** Students select the amount to pay, confirm the payment details, and complete the transaction using their Khalti wallet credentials. **Output:** Payment confirmation is displayed, and the transaction is recorded in the system.
Functional Requirements	**F4.1:** The system must integrate with the Khalti payment gateway for secure transactions. **F4.2:** Students must provide valid Khalti credentials for authentication. **F4.3:** The system must calculate the total dues and display them before payment. **F4.4**: A payment receipt or confirmation must be generated and stored for future reference. **F4.5:** The payment history should be updated to reflect successful transactions. **F4.6:** Student receive notifications (email/SMS) for successful payments. **F4.7:** The system must handle payment failures and provide error messages with retry options.

4.5 Requirement F5

Table 5: F5: Student can apply for Test.

F5: Student can apply for Test.	
Priority	Must
Description	Allow students to apply for tests, choose difficulty levels (Easy or Difficult), and take tests with multiple-choice questions (MCQs).
Input/outputs	**Input:** Students select a test, choose difficulty level (Easy/Difficult), and answer MCQs within a time limit. **Output:** The system evaluates the answers, displays the score, and stores the result in the student's test history.
Functional Requirements	**F5.1:** Students must be able to register for a test through the system. **F5.2:** The system must allow students to select the difficulty level (Easy or Difficult). **F5.3:** Tests must be in MCQ format, with a timer to limit the test duration. **F5.4:** The system must evaluate the answers automatically and calculate the score. **F5.5:** A summary of the test, including the score and correct/incorrect answers, must be displayed after submission. **F5.6:** Students should be able to view their test history, including scores and details of past tests.

5.User Stories and Scenarios.

User Story 1: Create Superuser (Requirement F1).

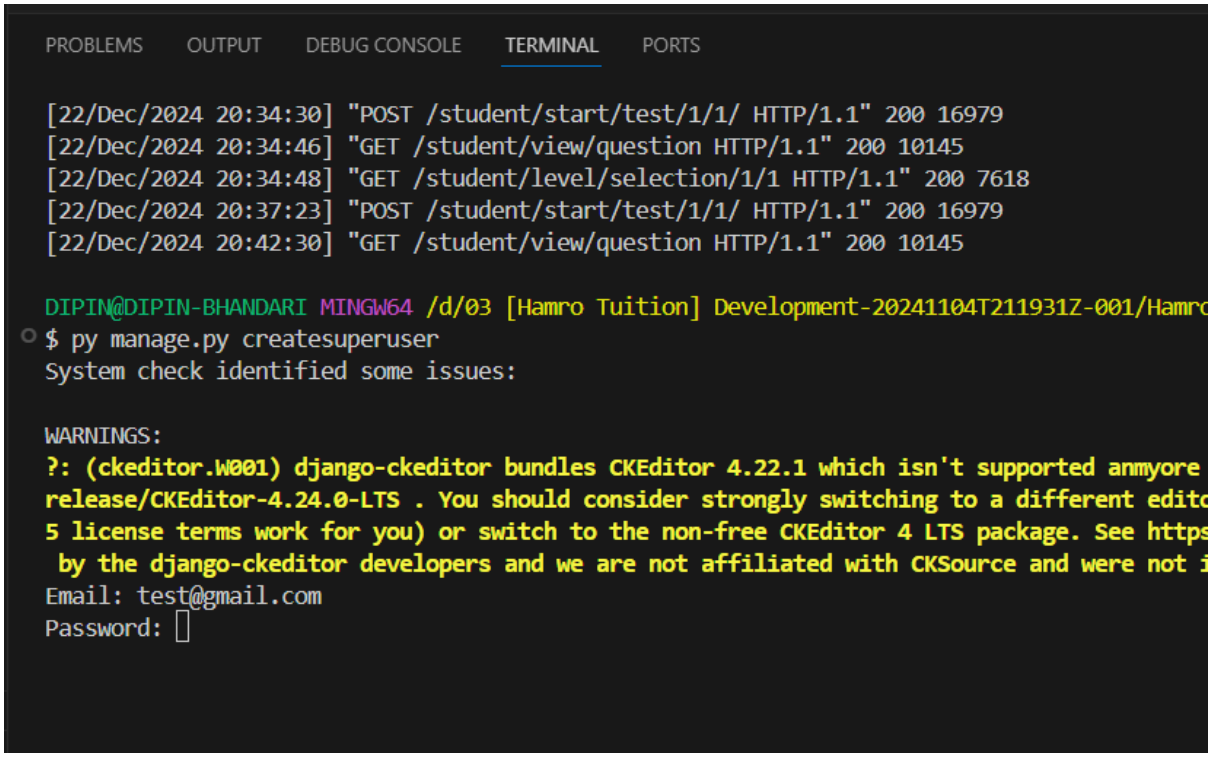


Figure 11: Create Superuser.

User Story 2: Attendance Management by Teacher (Requirement F2).

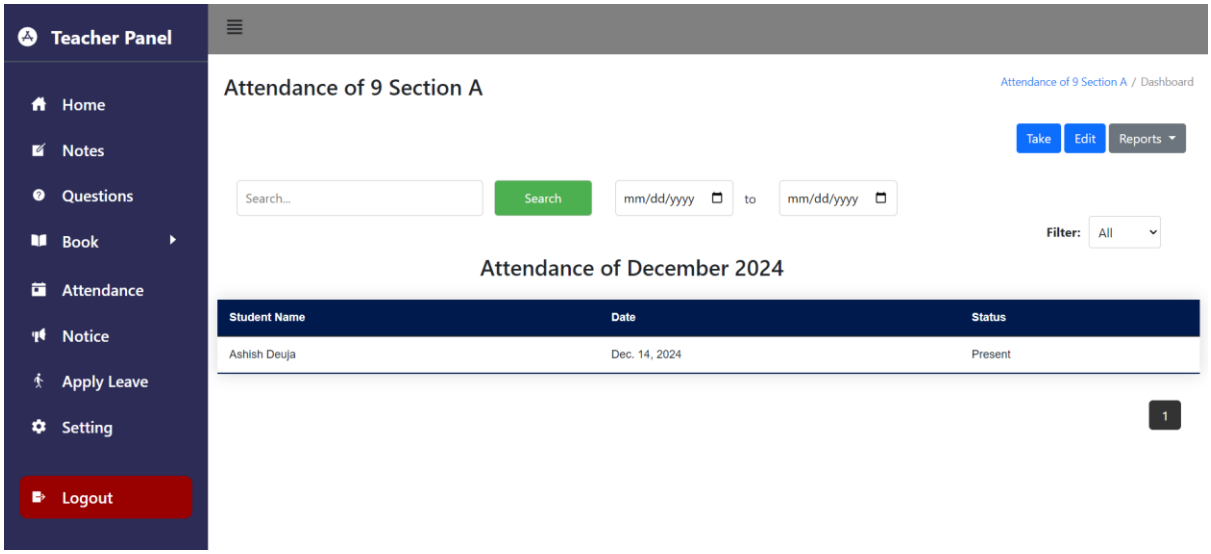


Figure 12: Attendance Management by Teacher (Requirement F2).

User Story 3: Apply For Leave Student (Requirement F3).

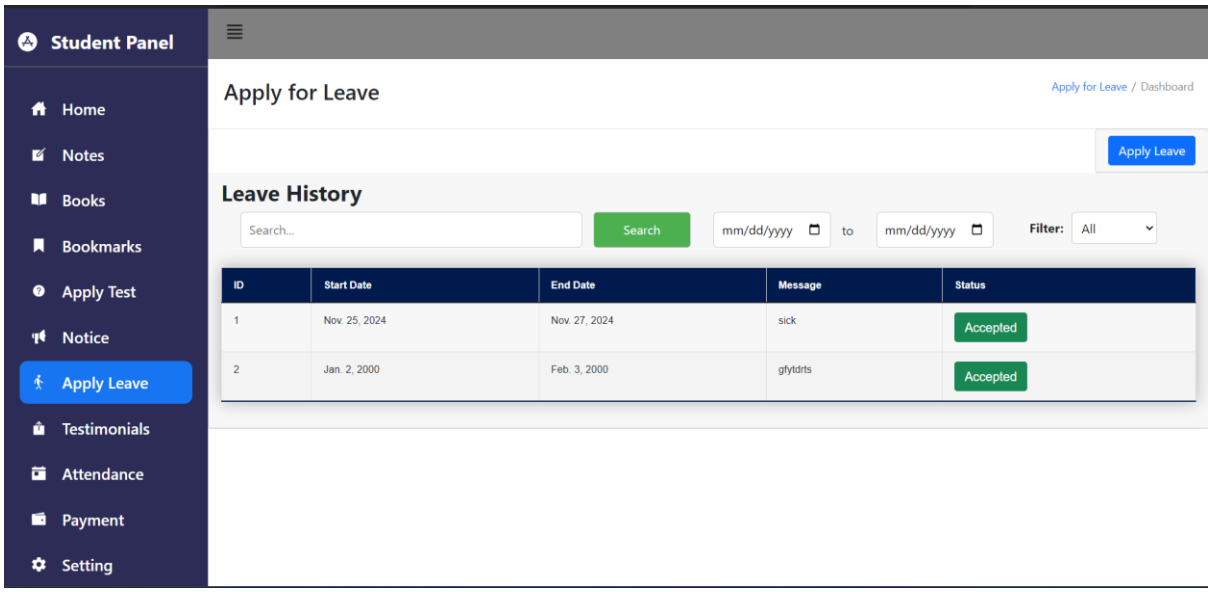


Figure 13: Apply for Leave Student (Requirement F3).

User Story 4: Payment Khalti is integrated for student Tuition Fee (Requirement F4).

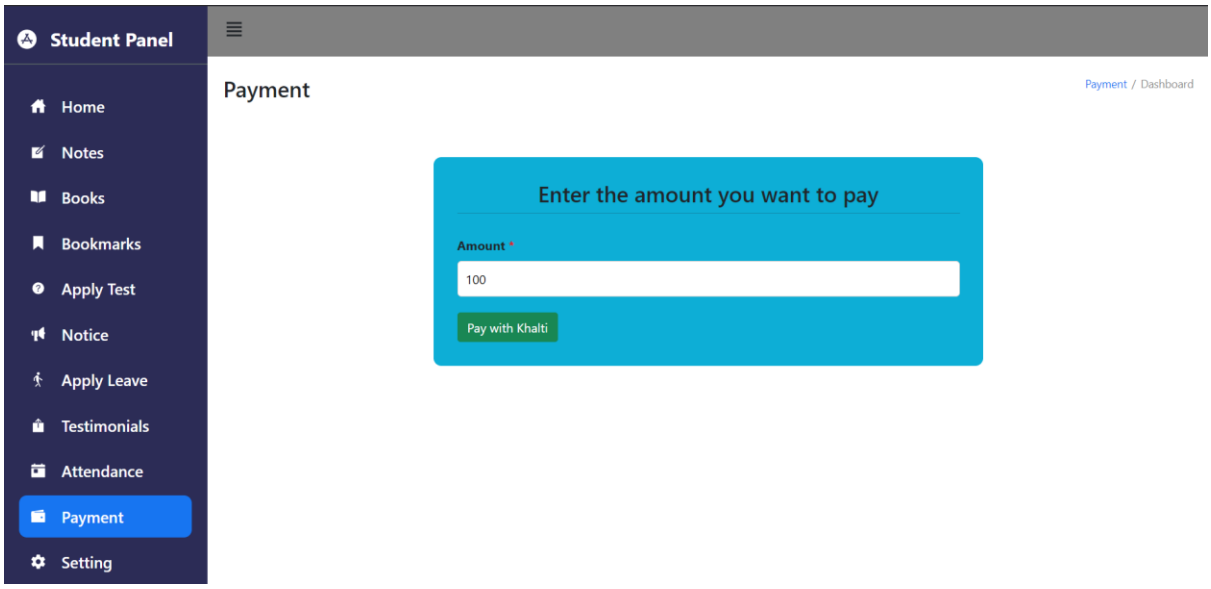


Figure 14: Payment Khalti is integrated for student Tuition Fee (Requirement F5).

User Story 5: Student can apply for Test (Requirement F5).

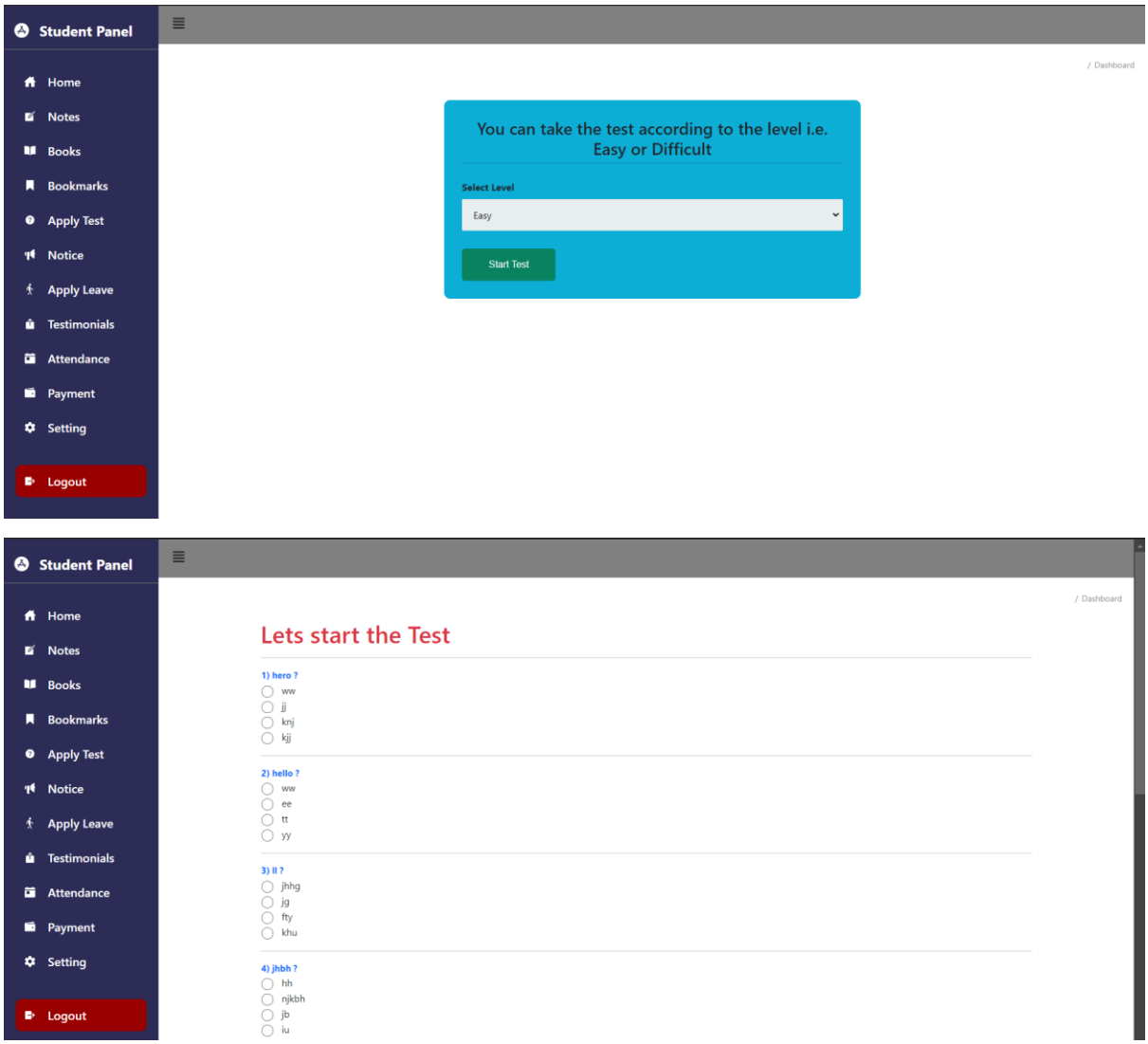


Figure 15: Student can apply for Test (Requirement F5).

6.Nonfunctional Requirements.

6.1 Performance Requirements.

System Response Time: The online application should not crash when it opens, and the graphical user interface should not include any errors. It should be a responsive and fast system. Instantaneous system reaction is required. Response times for every operation carried out on the system with less than 50% memory must be fewer than 10 seconds. (ibm.com, 2024)

6.2 Safety Requirements.

Safety standards guarantee that users, data, and the infrastructure are not harmed during system operation. The main goals of the Hamro Tuition Management System's safety regulations are to safeguard users, maintain data integrity, and stop system abuse or malfunction. (paula, 2024)

6.3 Security Requirements.

It protects all data within the system from intrusions like viruses and illegal access to the framework. It is not necessary to implement it in the product, but it is decoded in accordance with the desire for practical requirements as well. (Ivanov, 2023)

6.4 Software Quality Attributes.

Software quality attributes: ensure that the system meets user and organizational needs effectively and efficiently. It must support scaling to accommodate increased users, data, and functionalities. (Priyanka, 2024)

6.5 Other External Requirements.

Database Compatibility: Database compatibility ensures scalability, flexibility, and performance by enabling the system's database to operate without interruption across a range of settings, tools, and configurations. (Krüger, 2024)

6.6 Business Rule.

Permission: Admins have full control over the system, including managing users, payments, notifications, and system settings.

7.Test Plan.

7.1 Test Suite T1.

7.1.1 Test Case 1.

Test Description of Login

Objective	To verify that users can successfully log in to the system with valid credentials.
Action	The valid email and the password are entered.
Expected Result	The user should be redirected to the home page.
Actual Result	The user was redirected to home page
Conclusion	Test was Successful.

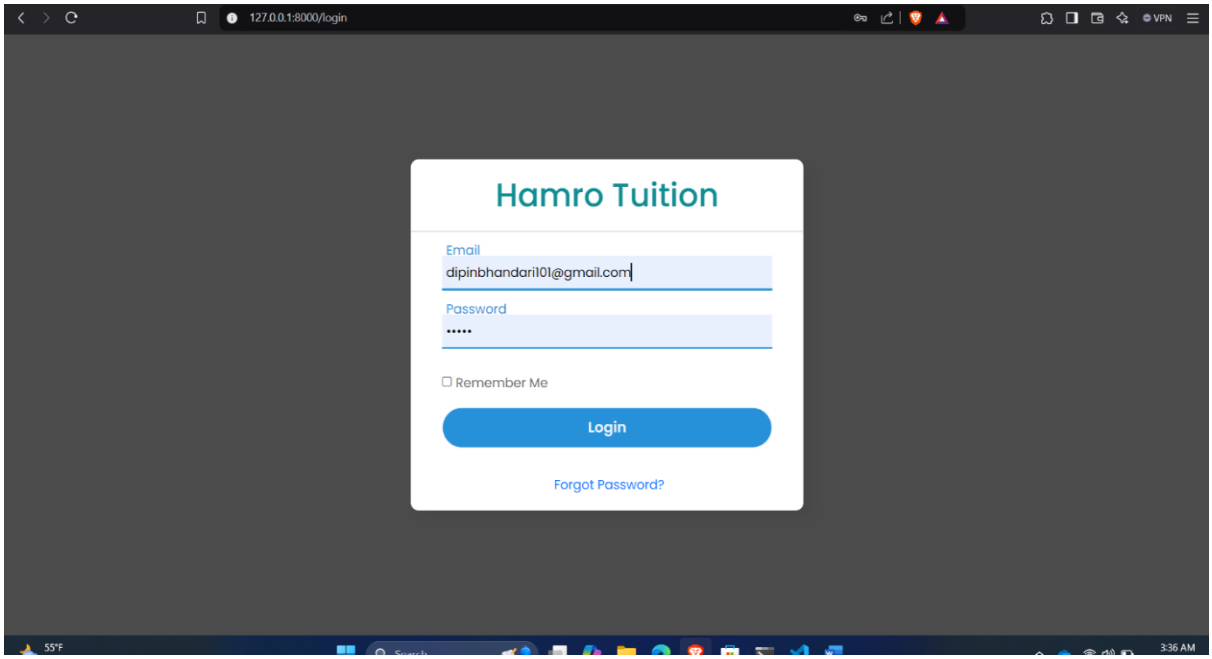
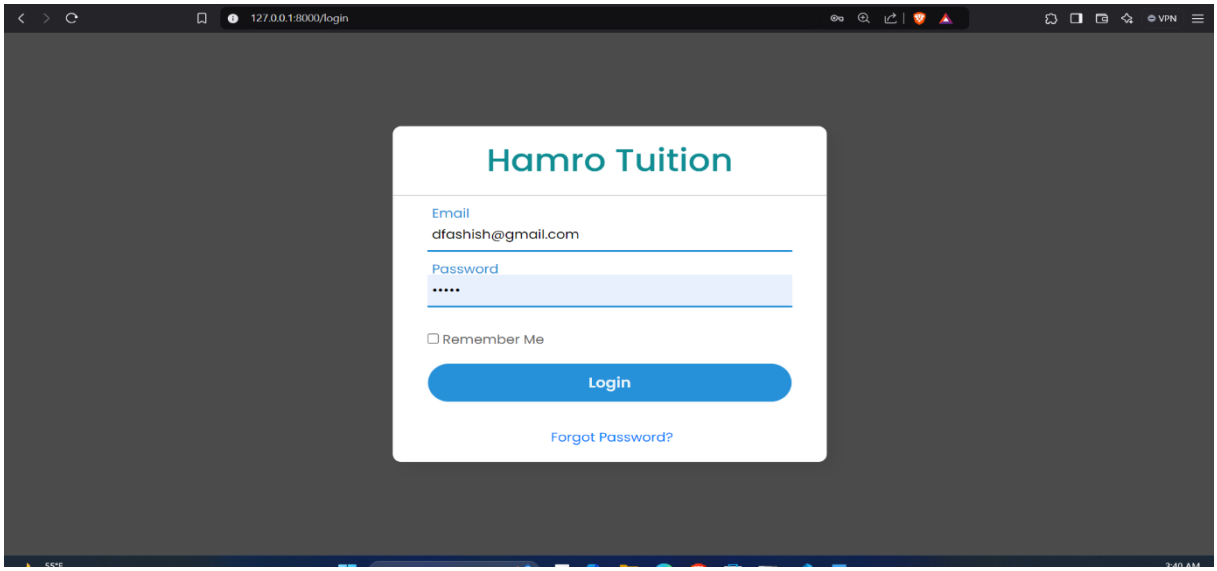


Figure 16: Login Testing.

7.1.2 Test Case 2.

Table 6: Login with invalid details.

Objective	To verify that users cannot login in the system with invalid credentials.
Action	The invalid password is entered.
Expected Result	The user should not be redirected to the home page.
Actual Result	The user was not redirected to the home page.
Conclusion	Test was Successful.



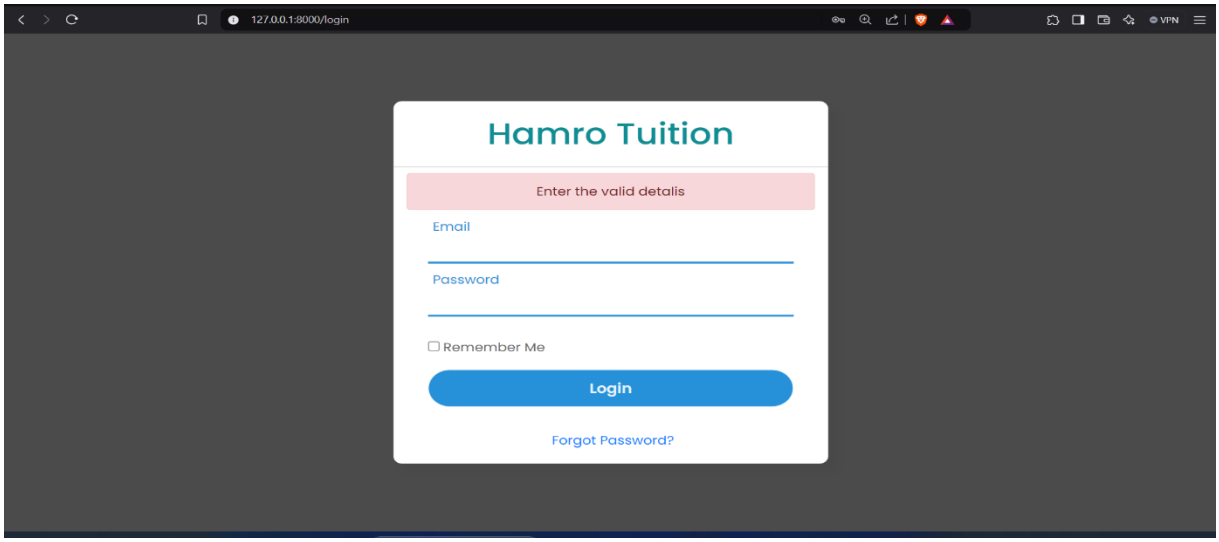


Figure 17: Login with invalid details.

7.1.3 Test Case 3.

Table 7:Test of Add Level (Class).

Objective	To add the class in the system with valid data.
Action	The valid details are entered.
Expected Result	The class should be added, and the added successful message should be displayed.
Actual Result	The class was added, and the added successful message should be displayed.
Conclusion	The test was successful.

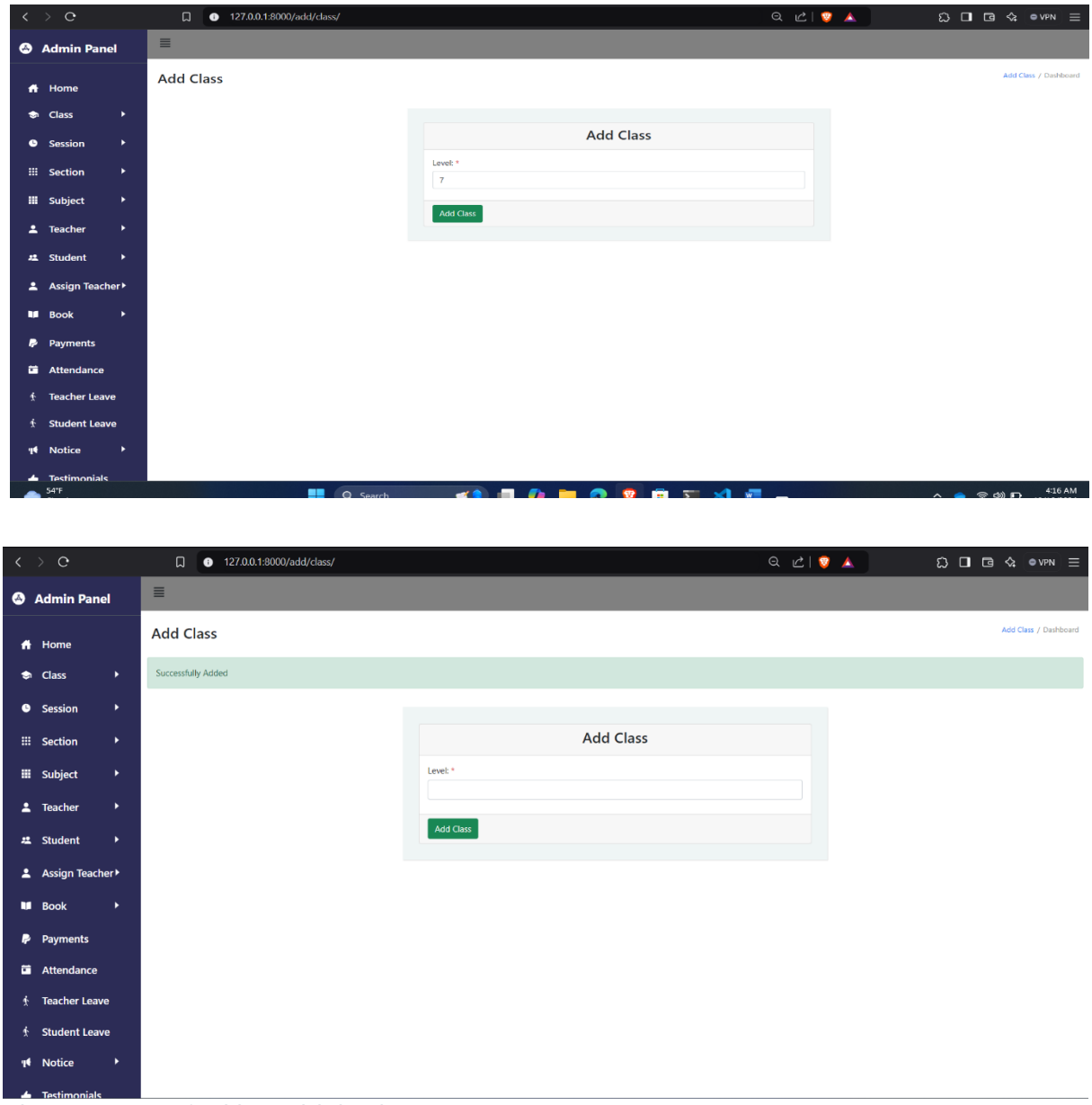


Figure 18: Test of Add Level (Class).

7.2 Test NF Requirement NF1.

Table 8: Responsiveness Testing on various devices.

Objective	Check the Tuition Management System's responsiveness on various devices.
Action	Test the system's layout, navigation, and interactive features by accessing it on devices with varying screen sizes.
Expected Result	All information is displayed, the layout changes constantly, and interactive features function flawlessly across all platforms.
Actual Result	After testing is completed, it will be documented.
Conclusion	The test was successful.

8. Project Management.

8.1 Cost Estimation.

A model for estimating software costs, the Constructive Cost Model (COCOMO) aids in forecasting the time, money, and effort needed for a software development project. (geeksforgeeks.org, 2024)

Basic COCOMO Model:

The formula used by the Basic COCOMO model to determine effort is $E = a * (KLOC)^b$

Where;

E = Effort in person-months.

(a, b) variables are depending on the kind of project (organic, semi-detached, or embedded)

KLOC = The project's size expressed in Kilo Lines of Code (1 KLOC is synonymous with 1,000 lines of code).

COCOMO Constants:

For an organic Project, the constants are as follows:

a = 2.4

b = 1.05

Given Data Information:

a) **Project Mode:** Organic
b) **Project Size:** Medium (Hence, according to our assumption, equals KLOC=27.56)
c) **Development Team Experience**: Highly Experienced.

COCOMO Calculations:

1) Effort Calculation (E) = $a * (KLOC)^b$

Substituting the values,

E = $2.4 * (27.56)^{1.05}$ = 69.12 person-months.

2)Development Time (T) = $3.0 * (E)^{0.33}$

Substituting the values,

T = $3.0 * (69.12)^{0.33}$ = 12.25 months.

3) Number of People (P) = E / T

Substituting the values, **P** = 69.12 / 12.25 = 5.64 People.

Findings of our Cost estimating calculations, which include maintenance, using the **COCOMO III** model are as follows:

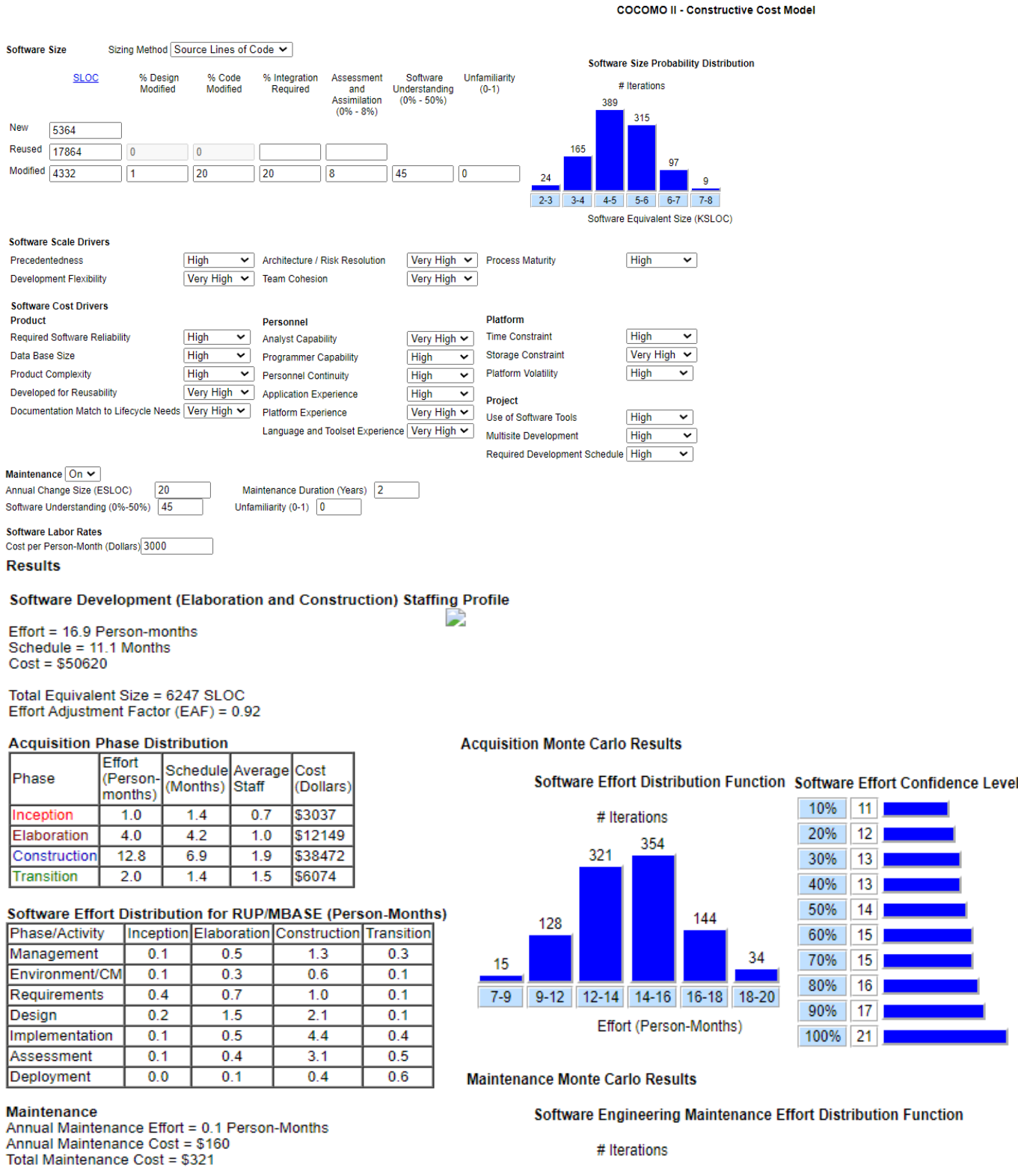


Figure 19: Cost Calculation COCOMO model.

8.2 Scrum Desk.

A project management application called Scrum Desk was created especially for teams using Agile and Scrum approaches. It offers an all-inclusive framework for managing product backlogs, organizing sprints, monitoring advancement, and enhancing teamwork. (scrumdesk, 2024)

The main features of Scrum desk are shown in the below figure.

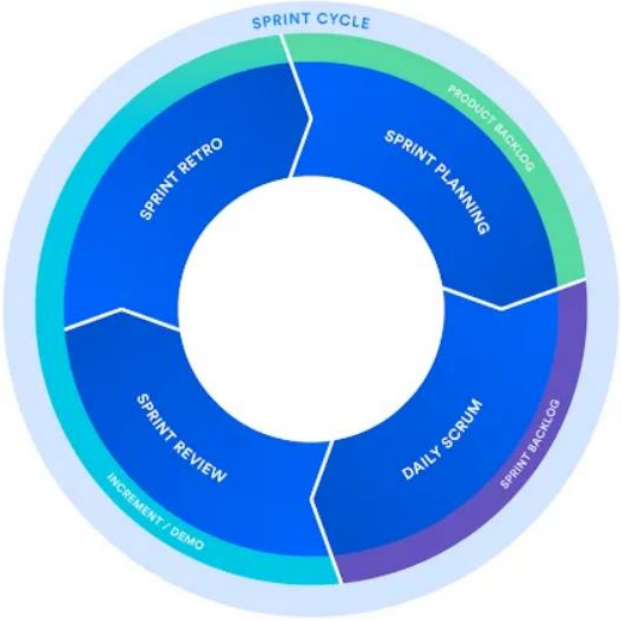


Figure 20: Scrum Desk.

The roadmap to build our system has been divided into 3 sprints of 2 weeks each as follows:

Table 9: Sprint table.

Sprint	Sprint Name	Start Date	End Date
Sprint 1	Admin	16th oct 2024	30th oct 2024
Sprint 2	Teacher	2nd nov 2024	19th nov 2024
Sprint 3	Student	21st nov 2024	3rd dec 2024

Sprint Board.

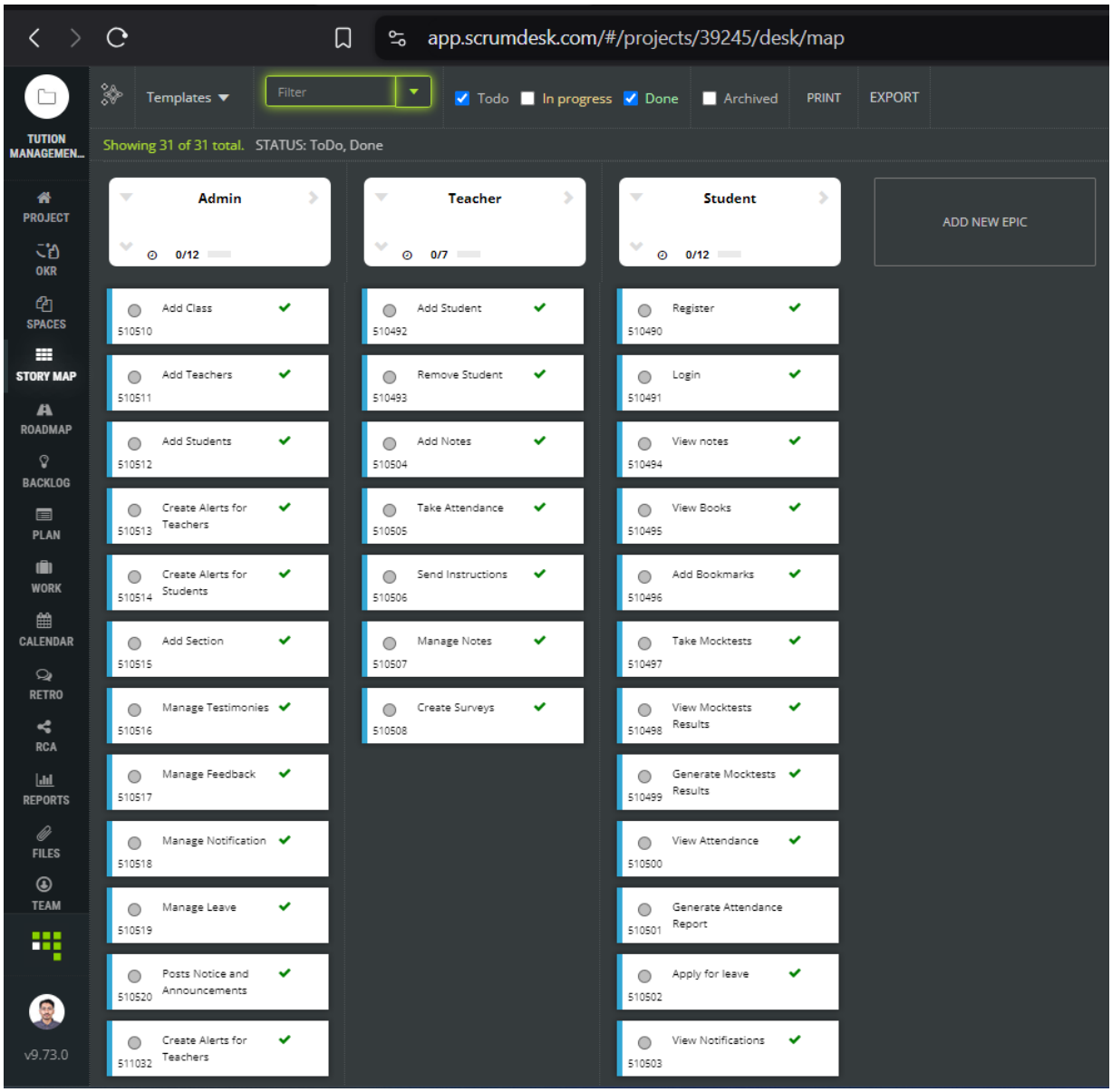


Figure 21: Sprint Board.

Scrum Board view from sprint.

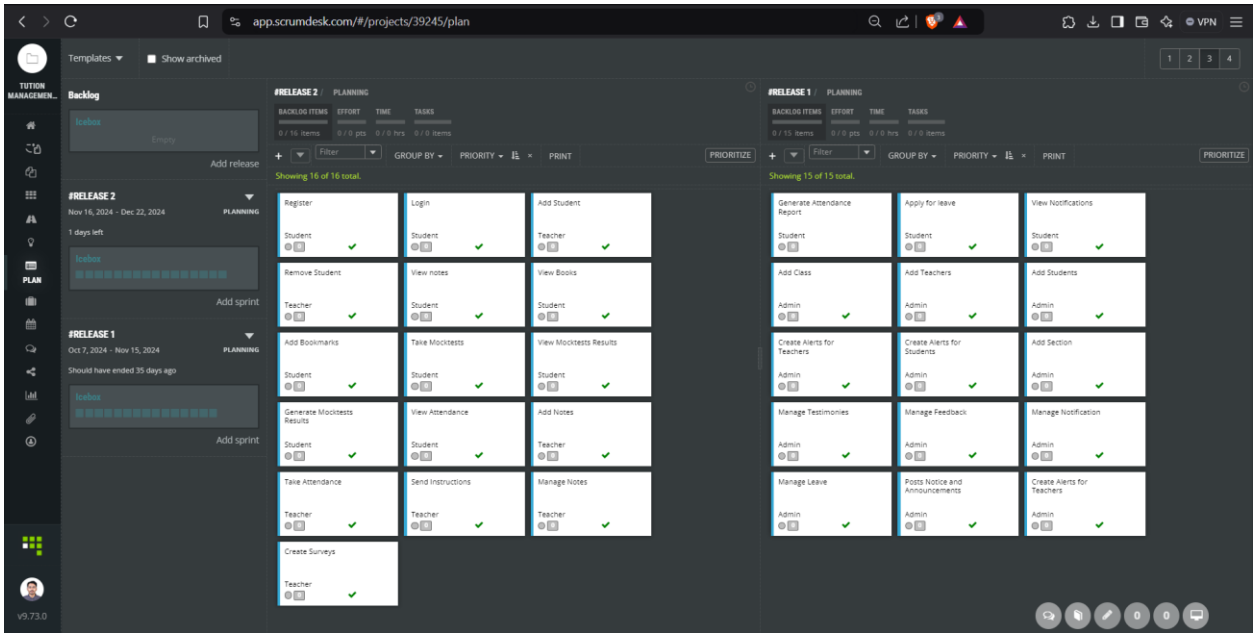


Figure 22: Scrum Board view from sprint.

Sprint Release 1.

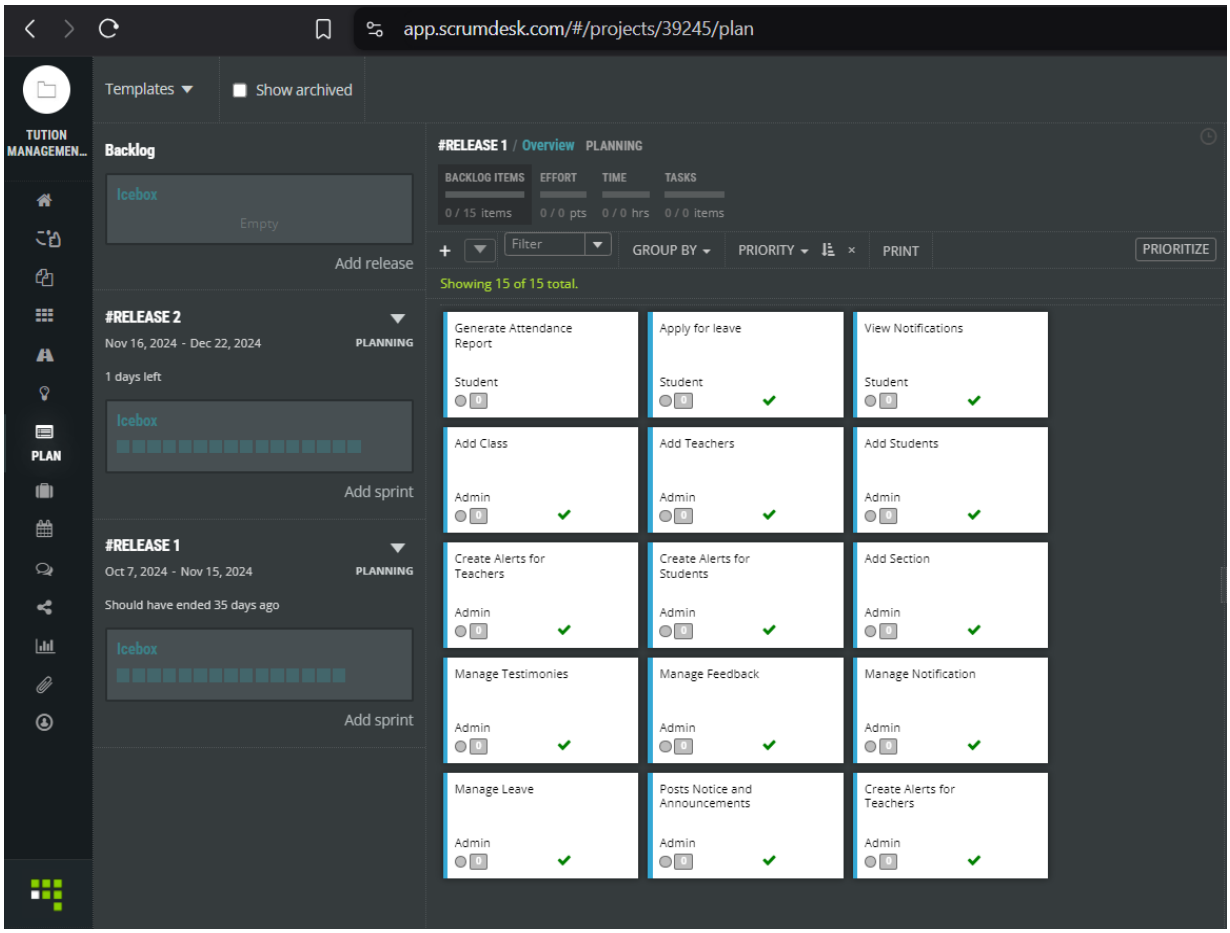


Figure 23: Sprint Release 1.

Sprint Release 2.

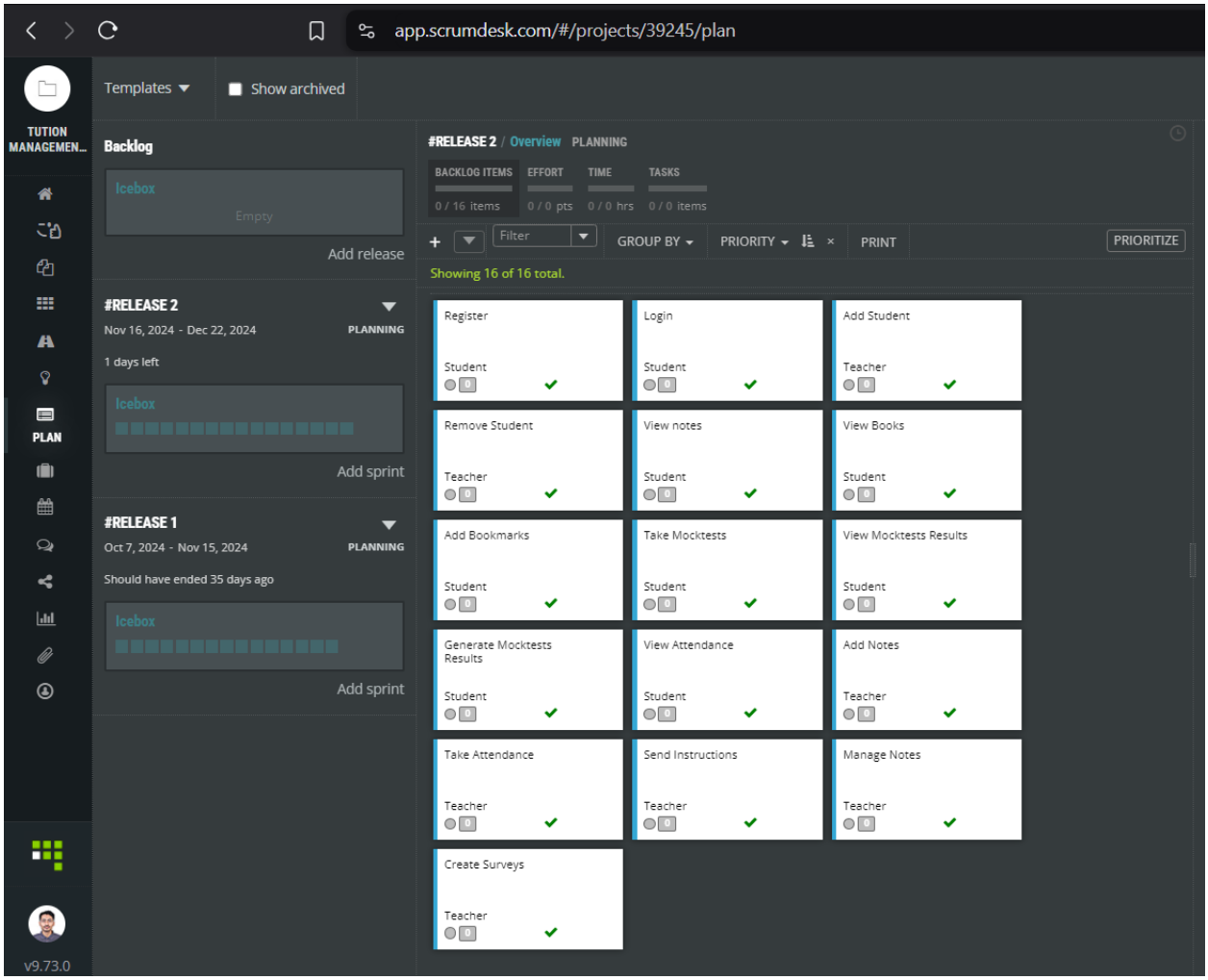


Figure 24: Sprint Release 2.

9. References

agilest.org, n.d. [Online]
Available at: https://www.agilest.org/scrum/why-does-scrum-work/

Al-Sarayreh, K. T., 2010. *A Standards-Based Model for the Specification of System Design and Implementation Constraints,* France: s.n.

docs.djangoproject.com/, 2024. [Online]
Available at: https://docs.djangoproject.com/en/5.1/intro/tutorial01/

docs.djangoproject, 2024. [Online]
Available at: https://docs.djangoproject.com/en/5.1/ref/databases/#sqlite-notes

docs.python.org, 2024. [Online]
Available at: https://docs.python.org/3/tutorial/interpreter.html#argument-passing [Accessed 5 dec 2024].

geeksforgeeks.org, 2024. *geeksforgeeks.org.* [Online]
Available at: https://www.geeksforgeeks.org/software-engineering-cocomo-model/

ibm.com, 2024. *ibm.com.* [Online]
Available at: https://www.ibm.com/docs/en/cics-ts/6.x?topic=constraints-observing-response-time

Ivanov, K., 2023. *medium.com.* [Online]
Available at: https://medium.com/@svosh2/non-functional-requirements-ce6e55305900

Krüger, G., 2024. [Online]
Available at: https://www.perforce.com/blog/alm/what-are-non-functional-requirements-examples

paula, 2024. s.l.: https://premieragile.com/what-are-safe-non-functional-requirements/.

Pragy Agarwal, 2023. [Online]
Available at: https://www.interviewbit.com/blog/system-architecture/

Priyanka, 2024. [Online]
Available at: https://testsigma.com/blog/software-quality-attributes/

Rehkopf, M., 2024. phases of scrum. *scrum sprint,* p. 4.

Rouse, M., 2017. [Online]
Available at: https://www.techopedia.com/definition/7529/hardware-platform

scrum.org, 2024. [Online] Available at: https://scrumorg-website-prod.s3.amazonaws.com/drupal/inline-images/2023-09/scrum-framework-9.29.23.png

scrum.org, 2024. *scrum.org.* [Online] Available at: https://www.scrum.org/resources/what-scrum-module

scrumdesk, 2024. *scrumdesk.com.* [Online] Available at: https://www.scrumdesk.com/scrumdesk-blogs/

Shacklett, M. E., 2024. [Online] Available at: https://www.techtarget.com/whatis/definition/operating-system-OS

smartclasses.in, 2022. [Online] Available at: https://smartclasses.in/Tuition-Management-System-Complete-Guide# [Accessed 2024 nov 18].

w3schools, 2024. [Online] Available at: https://www.w3schools.com/howto/default_page4.asp [Accessed 5 dec 2024].

Woolf, B. P., 2009. *Communication Interface,* s.l.: s.n.

10. Appendices.

10.1 Appendix A: Contributions Table.

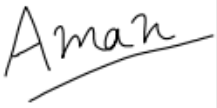
Project Task	Azeem	Amandeep	Dipin	Sai Manish	**Estimated Hours**
Requirement analysis	X	X	X	X	10
System Design and implementation		X	X	X	17
coding	X	X	X	X	29
Integration Testing	X		X		12
Documentation Writing	X	X	X	X	25

10.2 Appendix B: Agreement of Participation.

Agreement of Participation – Group Assignment CN7021

Please complete this agreement and keep a copy for each member of your group. The original of this agreement goes to your tutor, and the Electronic copy goes in your assignment.

We agree to work as a group (**a group of 4**) to complete the coursework for CN7021 and understand that the grade awarded will be the grade allocated to us individually as a result of our group work.

Student No.	Name (block letters) and eMail Address	Signature
u2778998	SAI MANISH GANDRA u2778998@uel.ac.uk	
u2776903	AMANDEEP u2776903@uel.ac.uk	
u2784783	DIPIN BHANDARI u2784783@uel.ac.uk	
u2776658	ABDUL AZEEM u2776658@uel.ac.uk	

Note: Students should form their groups (a group of 4) within the SAME Tutorial / Practical.

Tutorial / Practical Number: _____

Tutor's Name: Reena Popat

Date of agreement 25 October 2024

10.3 Appendix C: Glossary.

Html - HyperText Markup Language.

CSS - Cascading Style Sheets.

JS – JavaScript.

Srs - Software Requirements Specification.

Admin- Administrator.

Vs code – Visual Studio Code.

COCOMO - Constructive Cost Model.

10.4 Appendix D: Analysis and Design Models.

1) Context Level Diagram.

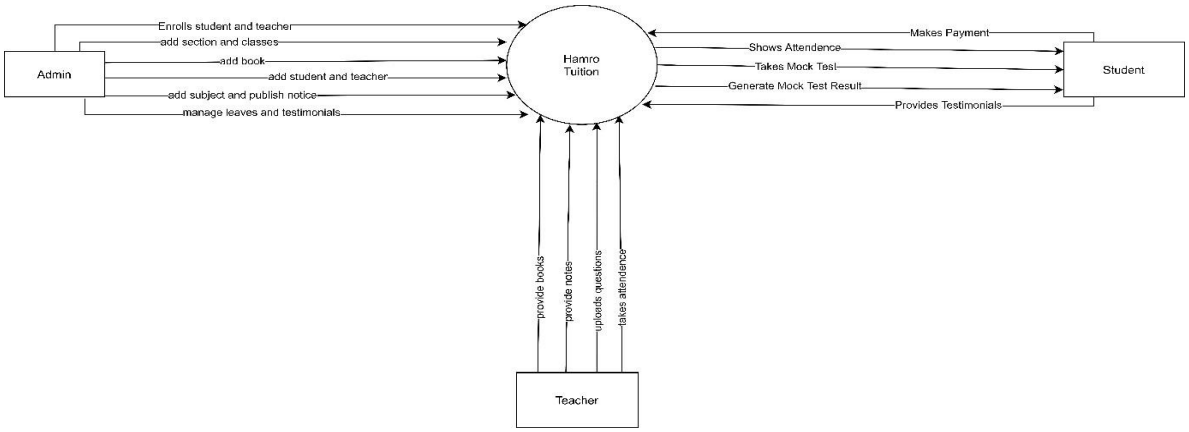


Figure 25: Context Level Diagram.

2) Admin Panel Collaboration Diagram.

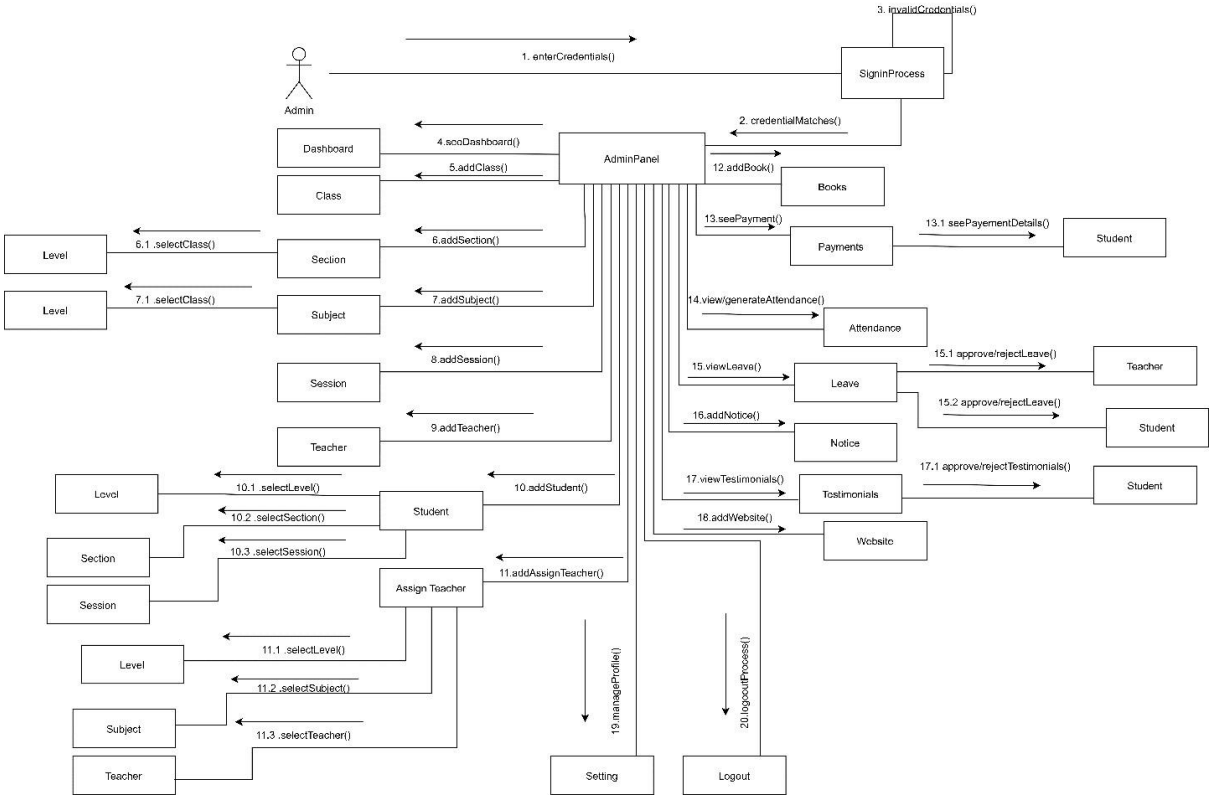


Figure 26: Admin Panel Collaboration Diagram.

3) Teacher Panel Collaboration Diagram.

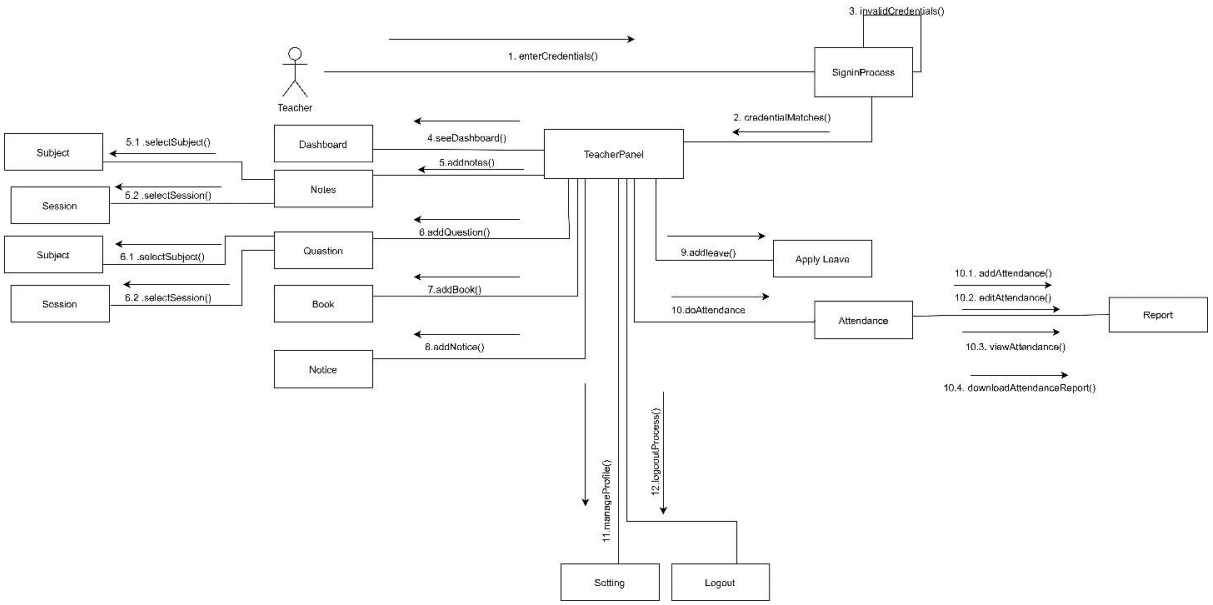


Figure 27: Teacher Panel Collaboration Diagram.

4) Student Panel Collaboration Diagram.

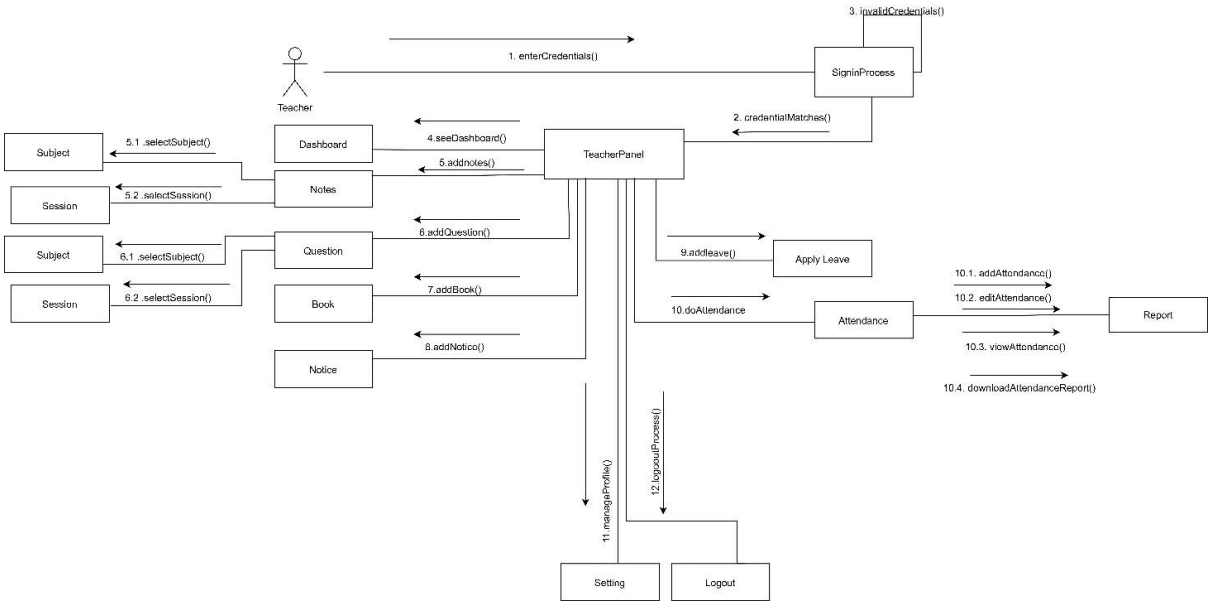


Figure 28: 4) Student Panel Collaboration Diagram.

10.5 Appendix E: To Do List.

1) online class integration.
2) In-app messaging integration.
3) Assignment Submission integration.
4) Language Support integration.
5) Calendar Integration.
6) Digital Library integration.

10.6 Appendix F: Source Code.

Drive Link for the source code:
https://drive.google.com/file/d/1hNd3aRLRDzpCQkzi1vKrxEpiJzlqhKX8/view?usp=drive_link