Automated Platform for Undergraduate Admission

Final Year Project Report

Submitted by:
Waleed Ahmed (1734-2021)
Dua Rahim (1597-2021)
Zoya Sayeed (1696-2021)

Supervisor
Sir Iqbal Uddin Khan

In partial fulfilment of the requirements for the degree of
Bachelor of Science in Computer Science 2021-25

Faculty of Engineering Sciences and Technology

Hamdard Institute of Engineering and Technology

Hamdard University, Main Campus, Karachi, Pakistan

Certificate of Approval

Faculty of Engineering Sciences and Technology

Hamdard Institute of Engineering and Technology
Hamdard University, Karachi, Pakistan

This project "**Automated Platform for Undergraduate Admission**" is presented by "Waleed Ahmed (1734-2021), Dua Rahim (1597-2021) & Zoya Sayeed (1696-2021)" under the supervision of their project advisor and approved by the project examination committee, and acknowledged by the Hamdard Institute of Engineering and Technology, in the fulfillment of the requirements for the Bachelor degree of Computer Sciences.

_____	_____
Mr. Iqbal Uddin Khan	In-charge FYP-Committee
(Project Supervisor)	
_____	_____
Chairman	(Dean, FEST)
(Department of Computing)	

Authors' Declaration

We declare that this project report was carried out in accordance with the rules and regulations of Hamdard University. The work is original except where indicated by special references in the text and no part of the report has been submitted for any other degree. The report has not been presented to any other University for examination.

Dated:

Authors Signatures:

_____

Waleed Ahmed

_____

Dua Rahim

_____

Zoya Sayeed

Plagiarism Undertaking

We, Waleed Ahmed, Dua Rahim, and Zoya Sayeed, solemnly declare that the work presented in the Final Year Project Report titled "Automated Platform for Undergraduate Admission" has been carried out solely by ourselves with no significant help from any other person except few of those which are duly acknowledged. We confirm that no portion of our report has been plagiarized and any material used in the report from other sources is properly referenced.

Dated:

Authors Signatures:

_____

Waleed Ahmed

_____

Dua Rahim

_____

Zoya Sayeed

Acknowledgments

All praises to Almighty" ALLAH", Who is the most generous, the most kind, and the wellspring of all human knowledge and wisdom. With all due respect, the Holy Prophet "Hazrat MUHAMMAD P.B.U.H "ﷺ

Acknowledgement due to Hamdard Institute of Engineering and Technology for their assistance with this project, which is a very valued accomplishment for us as undergraduates.

We owe it to Sir Iqbal Uddin Khan, our supervisor, who has served as our major advisor. We would like to thank them for their insightful advice, genuine assistance, and friendly appearance, all of which inspired us to complete the project successfully and make it a reality.

Several people have offered insightful comments and ideas on this proposal, particularly our classmates and team members themselves, which motivated us to make our project better. We would especially like to thanks Sir Jahangir Khan and everyone else who helped us finish our project, both directly and indirectly.

Document Information

Table 1: Document Information

Customer	
Project Title	Automated Platform for Undergraduate Admission
Document	Final Year Project Report
Document Version	1.0
Identifier	FYP-S001/SP24 Final Report
Status	Final
Author(s)	Waleed Ahmed, Dua Rahim , Zoya Sayeed
Approver(s)	Sir Iqbal Uddin Khan
Issue Date	12-01-2025

Definition of Terms, Acronyms, and Abbreviations

This section should provide the definitions of all terms, acronyms, and abbreviations required to interpret the terms used in the document properly.

Table 2: Definition of Terms, Acronyms, and Abbreviations

TERM	DESCRIPTION
Scrum	An agile project management methodology emphasizing iterative development and continuous feedback.
UI/UX	User Interface/User Experience, design principles focused on ease of use and accessibility.
MongoDB	A NoSQL database system used for scalable and flexible data storage.
Firebase	A platform for building web and mobile applications with backend support and hosting.
Frontend	The part of the application the user interacts with directly (HTML, CSS, JavaScript, React/Next.js).
Backend	The server side of the application responsible for logic and database operations (Node.js/Express.js).

Abstract

This project presents an automated platform for undergraduate admissions, designed to overcome challenges inherent in manual admission processes. It aims to improve reliability, flexibility, security, and data integrity while minimizing errors and enhancing overall efficiency. The initiative responds to the issues faced by the admissions team at HU, where manual workflows hinder productivity and accuracy.

To address these concerns, the proposed system will streamline and automate the admissions process. Developed using the Scrum methodology, it ensures iterative progress, continuous improvement, and active stakeholder involvement. Key goals include accurate data handling, consistent application of admission criteria, and a scalable infrastructure capable of managing high application volumes.

The platform will feature a user-friendly interface for seamless application submission and tracking, along with real-time status updates. Security measures such as vulnerability assessment, patch management, and compliance adherence are integral to the system's scope, ensuring a secure and transparent experience for applicants and staff alike.

Keywords:

- Automated Undergraduate Admissions
- Digital Application Workflow
- Real-Time Notifications
- Data Accuracy and Validation
- Payment Verification System
- Test Scheduling Automation
- Scalable Admissions Architecture
- Role-Based Access Control
- Regulatory Compliance and Transparency
- Agile Development (SCRUM Methodology)

Table of Contents

Certificate of Approval 01

Authors' Declaration 02

Acknowledgments 03

Document Information 04

Abstract 05

List of Figures 08

List of Tables 09

CHAPTER 1 INTRODUCTION 10

1.1 Motivation

1.2 Problem Statement

1.3 Goals and Objectives

1.4 Project Scope

CHAPTER 2 RELEVANT BACKGROUND & DEFINITIONS 12

CHAPTER 3 LITERATURE REVIEW & RELATED WORK

Literature Review 13

Related Work 14

Gap Analysis 15

CHAPTER 4

4.1 Software Engineering Methodology 17

4.2 Project Methodology 17

4.3 Phases of Project 18

4.4 Software/Tools that Used in Project 20

4.5 Hardware that Used in Project 20

Chapter 5

5.1 Proposed System Architecture/Design 21

5.2 Functional Specifications 25

5.3 Non-Functional Specifications 26

5.4 Testing 29

5.5 Purpose of Testing 33
5.6 Test Cases 35
Chapter 6 EXPERIMENTAL EVALUATIONS & RESULTS 41
Evaluation Testbed 41
Results and Discussion 43
CHAPTER 7 43
CONCLUSION AND DISCUSSION 43
7.1 Strength of this Project 43
7.2 Limitations and Future Work 45
7.3 Reasons for Failure – If Any 46
REFERENCES 47
APPENDICES 48
A0. COPY OF PROJECT REGISTRATION FORM 49
A1A. PROJECT PROPOSAL AND VISION DOCUMENT 50
A1B. COPY OF PROPOSAL EVALUATION COMMENTS BY JURY 58
A2. REQUIREMENT SPECIFICATIONS 59
A3. DESIGN SPECIFICATIONS 78
A4. OTHER TECHNICAL DETAIL DOCUMENTS 97
Test Cases Document 97
UI/UX Detail Document 105
Coding Standards Document
Project Policy Document
User Manual Document
A5. FLYER & POSTER DESIGN 105
A6. COPY OF EVALUATION COMMENTS 112
COPY OF EVALUATION COMMENTS BY SUPERVISOR FOR PROJECT – I MID SEMESTER
EVALUATION 112
COPY OF EVALUATION COMMENTS BY JURY FOR PROJECT – I END SEMESTER EVALUATION
112
COPY OF EVALUATION COMMENTS BY SUPERVISOR FOR PROJECT – II MID SEMESTER EVALUATION
A7. MEETINGS' MINUTES & Sign-Off Sheet 114

List of Tables

Table No.	Description	Page No.
TABLE 2.1	Comparison Table	6
TABLE 3.1	Phases of Project	7
TABLE 5.1	Test Case Set 1: User Registration & Login	109
TABLE 5.2	Test Case Set 2: Profile Management	110
TABLE 5.3	Test Case Set 3: Application Submission	111
TABLE 5.4	Test Case Set 4: Document Upload	111
TABLE 5.5	Test Case Set 5: Payment Gateway Integration	112
TABLE 5.6	Test Case Set 6: Test Scheduling System	113
TABLE 5.7	Test Case Set 7: Admin Dashboard	114
TABLE 5.8	Test Case Set 8: Real-Time Notifications	115
TABLE 5.9	Test Case Set 9: Security and Compliance	116
TABLE 5.10	Test Plan Schedule with Test Engineers	108

CHAPTER 1

INTRODUCTION

1.1 Motivation

Our team's motivation for this project is to address the inefficiencies and difficulties that candidates and educational institutions experience in order to transform the admissions process. Our goal is to use cutting-edge technologies to improve and automate the undergraduate admissions process, motivated by the need for transparent systems and efficient workflows.

The difficulties we have personally encountered navigating through numerous platforms for different services, which frequently lead to higher expenses and inefficiencies, are what motivate us as students in particular.

This experience has inspired us to develop a unified platform that streamlines processes, lessens administrative workloads, and offers applicants and admissions staff a smooth and accessible experience. Our ultimate goal is to provide a time-saving, cost-effective solution that improves universities' admissions processes but also establishes new benchmarks for usability, effectiveness, and accessibility.

1.2 Problem Statement

The current manual processes at FEST for handling admissions present several challenges. Manual data entry is a key issue, as admission staff manually collect and input student data into Excel sheets, leading to potential errors and data duplication. Additionally, there is no automated system in place to check for duplicate applications, resulting in multiple entries for the same student. Payment verification is another area of concern, as the process is manual, time-consuming, and prone to errors. Delays in verifying admission fee payments can lead to postponements in scheduling admission tests. Similarly, test scheduling is inefficient when done manually, often causing scheduling conflicts and miscommunication with students. Furthermore, communication delays arise from the manual notification of students about test dates, eligibility, and admission offers. The lack of an automated communication system not only results in delays but also leads to a poor experience for prospective students, as they are left uncertain about the status of their applications.

1.3 Goals and Objectives

Enhance Accuracy: Ensure correct data handling, reducing human errors in the admission process.

Consistency: Apply admission criteria uniformly across all applicants for fairness.

Efficiency: Reduce time and effort required to process applications, payments, and notifications. **User-Friendly Interface:** Provide an intuitive platform for applicants to submit and track applications.

Data Management & Scalability: Enable efficient storage and retrieval of data to handle large application volumes.

Transparency: Provide real-time updates and notifications about application status.

Compliance: Ensure adherence to institutional policies and regulations.

Improved Experience: Enhance overall satisfaction for applicants and administrative staff.

1.4 Project Scope

1-To automate the admissions process at FEST, reducing human error and delays.

2-To improve the security of sensitive applicant data through encryption and access control.

3-To enhance the overall user experience for both applicants and administrative staff.

4-To increase efficiency by providing real-time updates and better tracking of applications.

5-To ensure the system is scalable and flexible to accommodate future changes in admission requirements.

CHAPTER 2

RELEVANT BACKGROUND & DEFINITIONS

The project titled Automated Platform for Undergraduate Admissions was started because of a growing need to solve the difficulties in manual admissions procedures at universities including Hamdard University. Data entry, payment verification, test scheduling, and applicant communication are just a few of the operations that require a lot of human labor in the existing system. In addition to taking a lot of time, these procedures are subject to serious mistakes and discrepancies. For example, using manual data entry commonly leads to errors and duplicate information, and manually verifying payments causes delays that can prevent admission tests from being scheduled on time. The inefficiencies are further made worse by schedule issues and misunderstandings with candidates. Informing candidates of critical updates, like exam dates, eligibility status, and admission decisions, is delayed due to significant communication barriers caused by the absence of an automated notification system. All of these shortcomings add to a bad software experience and make staff

members' administrative workloads heavier. The existing system's scalability and reliability decrease as the number of applications increases.

This idea suggests creating a strong automated platform that uses technology to optimize the admissions process in order to overcome these obstacles. The suggested solution seeks to improve accuracy, efficiency, and transparency while greatly reducing the manual workload and related errors by automating data management, payment verification, test scheduling, and communication procedures. In accordance with current educational requirements, this solution guarantees scalability, compliance, and an easy-to-use interface for administrators and applicants alike.

Definitions

1. **Admissions Automation**: The process of using technology to streamline and manage all steps of the admissions process, including data entry, payment verification, test scheduling, and notifications.

2. **Data Integrity**: Ensuring the accuracy and consistency of data throughout its lifecycle to avoid duplication or errors.

3. **Scalability**: The system's ability to handle a growing number of applications without a corresponding increase in workload or resource consumption.

4. **Role-Based Access Control (RBAC)**: A security approach that restricts system access to authorized users based on their roles within the platform.

5. **SCRUM Methodology**: An Agile development framework used to ensure iterative progress through sprints, daily standups, and stakeholder reviews.

6. **Real-Time Notifications**: Immediate updates provided to applicants about their application status, test dates, or other critical information.

7. **Compliance Standards**: Adherence to institutional and regulatory guidelines to ensure transparency, fairness, and data security.

8. **Payment Verification System**: A digital mechanism to securely validate and track admission fee payments, minimizing errors and delays.

9. **User-Friendly Interface**: An intuitive platform design that allows applicants to submit and track their applications with ease.

10. **Vulnerability Assessment**: Evaluating the platform for potential risks or weak points to ensure data and system security.

CHAPTER 3

LITERATURE REVIEW & RELATED WORK

Literature Review

Multiple educational institutions have conducted detailed research on and implemented automation of admissions procedures, emphasizing its advantages in improving the applicant experience overall, reducing manual involvement, and streamlining workflows. A review of relevant frameworks and literature offers a strong foundation for understanding the importance of this task.

1. Challenges in Manual Admissions

Manual systems encounter challenges such as data duplication, inaccuracies, and inefficiency in managing increasing application quantities, rendering them unsustainable for modern educational institutions.

2. Benefits of Automation:

Automated systems offer user-friendly interfaces to speed workflows, improve accuracy, provide real-time tracking, and enhance the applicant experience.

3. Secure Data Management:

Role-Based Access Control (RBAC) protects data privacy and system integrity, addressing major security issues in admissions processes.

4. Efficient Payment Verification:

Automated payment channels eliminate delays and errors, allowing for a seamless transition to subsequent procedures such as test scheduling.

5. **Agile Development Practices:**

The SCRUM technique enables for iterative development with continuous feedback, guaranteeing that the system meets user expectations and adapts to institutional requirements.

6. **Scalability and Future Readiness:**

Scalability is ensured by cloud-based solutions and reliable databases like MongoDB, which enable systems to integrate cutting-edge features and manage growing application volumes.

Related Work

To solve issues with manual admissions procedures, a variety of automated admission platforms have been created. With real-time updates and online tracking, systems such as Creatrix Campus and HEIApply improve applicant experiences, simplify application operations, and increase data

accuracy. These platforms show how automation may effectively cut down on human mistake, processing lag, and communication breakdowns.

However, many existing systems have limits in scalability, flexibility, and responsiveness to specific institutional needs. For example, even if they offer strong features like data management and payment verification, they might not easily fit in with an institution's particular workflow or requirements for regulatory compliance. Furthermore, some platforms don't have a modular architecture that makes it simple to add new features or customize existing ones.

Using the SCRUM methodology, the proposed platform ensures iterative development, enabling continuous improvement and alignment with user feedback. This project takes inspiration from these existing systems while addressing their shortcomings, incorporating modern tools like ReactJS, MongoDB, and Firebase for scalability and a customized design for Hamdard University's admission needs.

Gap Analysis

Admissions at Hamdard University, for example, are still largely managed manually, resulting in a number of challenges that slow down the process and create a negative experience for both students and staff. Although some automated systems exist, they do not fully solve the problems or meet the institution's unique needs. Here's a breakdown of the issues and how this project addresses them.

Current Problems:

1. **Time-Consuming Manual Work**
   Manually entering information, confirming fees, and setting up tests takes a lot of time for admis sions teams.This frequently leads to errors, holdups, and redundant entries, making the procedure ineffective and annoying.

2. **Delays in Payment Processing**
   Verifying whether students have paid their fees is slow and prone to errors. This can cause delays in essential following stages, such as scheduling entrance tests or confirming eligibility.

3. **Lack of Communication**
   Students don’t obtain timely updates about their application status, exam dates, or admission results. Confusion and dissatisfaction result from this breakdown in communication.

4. **Scaling Challenges**
   The existing systems are unable to efficiently manage the load as the number of applications increases. This leads to additional errors, delays, and a requirement for more manual work.

5. **Limited Flexibility in Existing Systems**
   Some automation platforms are available, but they are often “one-size-fits-all” solutions. They can’t adapt to the unique processes or policies of specific institutions like Hamdard University.

What's missing in the Existing Solution

Some progress has been made by already-existing platforms such as Creatrix Campus and HEIApply, which provide features including online applications and simple tracking. But they:

- Do not completely eliminate manual work, particularly in areas such as payment verification and test scheduling.
- Are not made to adapt to the demands of a growing organization.
- Absence of options for customization to meet particular policies or workflows

How This Project Fills the Gaps

The goal of this project is to develop an automated undergraduate admissions platform that addresses the following issues:

- **Automation of Manual Tasks:**

  To cut down on errors and save time, the system will automatically manage data entry, payment verification, and test scheduling.

- **Real-Time Communication:**

  Students will be kept up to date at every stage with immediate updates regarding the progress of their applications, test dates, and results.

- **Scalable Design:**

  The platform can effectively manage an increasing number of apps thanks to its construction with contemporary technologies like ReactJS and MongoDB.

- **Customizable Features:**

  Because the platform was created especially for Hamdard University, it may be customized to meet their particular requirements and policies.

- **Secure Payment Processing:**

  Fast, accurate, and secure fee verification is guaranteed by an established payment platform

Why it Matters

In addition to addressing the present issues, this initiative will establish a new benchmark for the administration of admissions procedures. It makes the admissions process easier, quicker, and more dependable for both staff and students by filling in the gaps in automation, communication, and scalability.

CHAPTER 4
PROJECT DISCUSSION

1. Software Engineering Methodology

The decision to adopt an Agile approach for the development of this project is grounded in its alignment with the dynamic and iterative nature of Web based projects. The Agile methodology, with its core principles of flexibility, client collaboration, and incremental development, is particularly well-suited for endeavors where requirements are subject to change and continuous improvement is essential. In the realm of AI, which often involves intricate algorithms and evolving technologies, the Agile framework allows for regular adjustments based on emerging insights, changing priorities, and client feedback. the Agile approach is a strategic choice to navigate the complexities of content generation, fostering innovation, responsiveness, and a client-centric development process. The Agile approach is chosen for its adaptability, iterative nature, focus on user satisfaction, collaborative environment, risk mitigation, continuous improvement, and efficiency in delivering a product to the market. All these aspects make it well-suited for the development of this project

2. Project Methodology

- **Scrum**
  - The Scrum model is well-suited for the development of t this project due to its iterative and incremental nature, which aligns seamlessly with the characteristics of AI projects. In the context of content generation, where algorithms and models need fine-tuning and constant improvement, Scrum's iterative cycles, known as sprints, provide a structured framework for continuous refinement.
  - The choice to use the Scrum model in this project stems from its suitability for dynamic and iterative development processes, which aligns well with the nature of creating a content for Automated platform for Undergraduate Admission platform. Scrum is known for its flexibility and adaptability to changing requirements, which is crucial in the fast-evolving landscape of AI and content creation technologies.

- In a Scrum framework, development is organized into short, time-boxed iterations called sprints. This allows the team to regularly reassess priorities, incorporate feedback, and adjust development goals. Given the complexity of integrating AI models, developing a user-friendly interface, and meeting various content generation needs, Scrum's iterative nature helps in managing uncertainties effectively.

- The Scrum model is chosen for its adaptability, iterative approach, emphasis on collaboration, and its ability to deliver incremental, high-quality results—features that are particularly beneficial in a complex project like the development of an AI-Based Content Generator SaaS platform

3. Phases of Project

1. Planning Phase

- **Sprint Planning:** Define the goals and deliverables for the sprint. Break down the product backlog into sprint backlog items.
- **Requirement Gathering**: Collaborate with stakeholders to gather detailed requirements and user stories

2. Design Phase

- **System Design:** Create the system architecture, including front-end and back-end designs, data models, and AI integrations.
- **UI/UX Design:** Develop wireframes and prototypes using tools like Adobe XD or Figma to ensure a user-friendly interface.

3. Development Phase

- **Sprint Execution:** Develop features incrementally in sprints. Each sprint includes coding, testing, and integrating new features.
- **Daily Stand-ups:** Conduct daily meetings to discuss progress, challenges, and next steps

4. Testing Phase

- **Continuous Testing:** Perform unit testing, integration testing, and end-to-end testing within each sprint.
- **Feedback Integration:** Gather feedback from stakeholders and users at the end of each sprint and incorporate it into the next cycle.

5. Reviews and Retrospective Phase

- **Sprint Review**: Demonstrate the increment to stakeholders and gather feedback.
- **Sprint Retrospective:** Reflect on the sprint process to identify what went well and
  what can be improved in the next sprint.

6. Deployment Phase

- **Staging Environment:** Deploy the application to a staging environment for final testing.
- **Production Deployment:** After successful testing, deploy the application to the production environment.

7. Maintenance Phase

- **Monitoring:** Continuously monitor the application for performance and reliability.
- **Updates and Improvements:** Regularly update the application based on user feedback and emerging technologies.

4. Software/Tools that Used in Project

Visual Studio Code Frontend languages

Html
CSS
JavaScript
React JS / Next JS

Backend languages

Node JS / Express JS

UI Design

Figma Database
MongoDB / Firebase

Hardware that Used in Project

- Laptop (Core i5, 8GB RAM, 500GB SSD)
- Backup Hard drive (250 GB)
- USB
- Internet Device

Chapter 5
IMPLEMENTATION

5.1 Proposed System Architecture/Design

The proposed system for the Automated Platform for Undergraduate Admissions is a centralized web-based solution designed to streamline and automate all admission-related activities in the Department of Hamdard University. The system is built with a multi-user role structure consisting of Admin, Teacher, and Student, each having specific rights and responsibilities. The design focuses on usability, data accuracy, role-based access, and workflow efficiency. **5.1.1 System Overview**

The architecture follows a three-tier model:

- **Presentation Layer (Front-End):**
  - This is the user interface (UI) layer, accessible via a web browser.
  - Built using technologies such as HTML, CSS, JavaScript (or frameworks like React/Angular).
  - Provides dashboards and forms for each user role.
- **Application Layer (Back-End):**
  - Handles business logic, validation, and communication between the UI and database.
  - Developed using a server-side language like Python (Django/Flask), PHP (Laravel), or Node.js.
  - Implements security, session management, and workflow control.
- **Data Layer (Database):**
  - Manages all persistent data related to users, applications, departments, course enrolments, and statuses.
  - MySQL, PostgreSQL, or MongoDB can be used depending on relational or nonrelational needs.

5.1.2 Roles and Responsibilities

a. Admin Panel

- Full access to the entire platform.
- Manage user accounts (students and teachers).
- View admission progress and reports.
- Enroll students officially into the system.
- Activate or deactivate teacher/student accounts.
- Monitor department-wise and session-wise enrollment.

b. Teacher Panel

- Log in with verified credentials.
- Access a filtered list of students under their department.
- Assign or enroll students into specific courses/classes.
- Track academic progress and departmental strength.
- View and generate reports on class composition.

c. Student Panel

- Create and manage profile.
- Submit admission form with personal, academic, and program-related details.
- Upload required documents.
- Track application status (e.g., submitted, under review, accepted).
- Receive confirmation or rejection notifications.
- View their enrolled courses once assigned.

5.1.3 System Modules

1. User Authentication & Authorization

- Role-based login for Admin, Teacher, and Student.
- Password protection and session timeout for security.

2. Student Admission Module

- Admission form submission and document upload.
- Auto-generated application ID.
- Status updates with email/SMS notifications.

3. Teacher Class Enrollment Module

- List of students filtered by department/program.
- Manual or batch enrollment to classes/courses.

4. Admin Control Module

- Dashboard overview of total applicants, enrolled students, and pending tasks.
- CRUD (Create, Read, Update, Delete) operations on users and departments.

5. Reporting and Analytics Module

- Visual summaries of admission trends, enrollment numbers.
- Downloadable Excel/PDF reports for decision-making.

5.1.4 Database Design (Entity Relationships)

Key entities include:

- User Table (ID, name, email, role, status)
- Student Table (Student_ID, name, program, documents, application_status)
- Teacher Table (Teacher_ID, department, assigned_courses)
- Courses Table (Course_ID, course_name, department)
- Enrollment Table (Student_ID, Course_ID, Teacher_ID, date_enrolled) **Relationships:**
- One Admin manages many users.
- One Teacher can enroll many students.
- One Student can be enrolled in multiple courses.
- Courses are department-specific and linked to teachers.

5.1.5 Flow of System

- Student registers and submits admission form.
- Admin reviews and accepts/rejects applications.
- Teacher assigns accepted students to classes.
- Students receive confirmation of class/course enrollment.
- Admin oversees all activity and generates final reports.

5.1.6 Technology Stack

- **Front-End:** HTML, CSS, JavaScript (React.js or)
- **Back-End:** Node.js / Next.js
- **Database:** MySQL / MongoDB
- **Hosting:** Cloud-based (University server)
- **Security:** SSL encryption, role-based access control

5.1.7 Benefits of the Proposed Design

- Reduced manual workload for admin and teachers.
- Faster and more transparent admissions.
- Centralized data storage and easy retrieval.
- Reduced errors and improved tracking.
- Real-time updates and secure communication.

5.2 Functional Specifications

The functional specifications define the core features and capabilities that the system must provide for each type of user. These functions are designed to ensure a smooth, secure, and efficient admission process from application to class enrollment.

5.2.1 User Authentication

- Role-based login system (Admin, Teacher, Student).
- Secure sign-up and login with password encryption.
- Session timeout and logout functionality.

5.2.2 Student Module

- Fill and submit admission application form.
- Upload required academic documents (PDF, images).
- View application status (Pending, Accepted, Rejected).
- Receive email/SMS notifications for updates.
- View enrolled courses after approval.

5.2.3 Admin Module

- Create, view, edit, and delete student and teacher accounts.
- Accept or reject student applications.
- Monitor admission progress via dashboard.
- Assign roles and permissions to users.
- Generate summary reports and analytics.

5.2.4 Teacher Module

- View list of students in their department.
- Enroll students into department-specific classes/courses.
- View student academic history and details.
- Generate class rosters or enrollment lists

5.2.5 Admission Management

- Track and manage applications per semester/session.
- Department-wise student organization.
- Application deadlines and form control.

5.2.6 Course Enrollment Management

- View and manage available courses per department.
- Assign students to specific courses.
- Track total students enrolled in each course.

5.2.7 Notification System

- Email or SMS notifications for application status, course enrollment, or administrative actions.

5.2.8 Reporting

- Export reports for:
- Total applications
- Enrolled students
- Department-wise statistics
- Visual charts/graphs for decision-making.

5.3 Non-Functional Specifications

Non-functional specifications define the quality attributes of the system. These do not directly impact the core functionality but are essential for the performance, reliability, usability, and maintainability of the system. For the Automated Platform for Undergraduate Admissions, the following non-functional specifications are considered:

5.3.1 Performance Requirements

- The system should respond to any user request within 3 seconds under normal load.
- It must support multiple concurrent users, especially during peak admission times (e.g., 100–200 users at once).
- The database should be optimized to fetch application and enrollment data quickly with minimal lag.

5.3.2 Scalability

- The system should be scalable to support future expansion, such as:
- Additional departments or faculties.
- Increase in the number of users/applications.
- Architecture should support horizontal and vertical scaling.

5.3.3 Availability

- The system must be available 24/7, especially during the active admission period.
- Expected uptime should be 99.5% during peak operations.
- Scheduled maintenance windows should be minimal and preferably off-hours.

5.3.4 Usability

- The system must have a user-friendly interface for all roles (Admin, Teacher, Student).
- Clear instructions, tooltips, and field validations should be present.
- Interface should be designed to reduce user confusion and minimize the learning curve.

5.3.5 Security

- Role-based access control (RBAC) to restrict unauthorized access.
- Password encryption and secure login protocols (e.g., HTTPS, SSL).
- Protection against common vulnerabilities such as:
  - SQL injection
  - Cross-site scripting(XSS)
  - Cross-site request forgery (CSRF)
- Data should be stored securely in the database, with sensitive information properly encrypted.

5.3.6 Maintainability

- Codebase should be modular and well-documented for easy updates and debugging.
- Database schema should be normalized and adaptable for future changes.
- Regular logs and audit trails should be maintained for error tracking and monitoring.

5.3.7 Portability

- The system should be deployable on different environments, such as:
- Local university servers
- Cloud hosting services (AWS, Azure, etc.)
- Front-end should be accessible on multiple browsers (Chrome, Firefox, Edge).

5.3.8 Compatibility

The platform must be compatible with:

- Desktop, laptop, and tablet screens (responsive design).
- Major operating systems: Windows, macOS, Linux.
- Common browsers: Chrome, Firefox, Safari, Edge.

5.3.9 Backup and Recovery

- Daily automated database backups must be scheduled.
- Admins should have access to restore data in case of accidental deletion or corruption.
- Backups should be stored securely and off-site if possible.

5.3.10 Reliability

- The system should function consistently without failures during the admission season.
- Backup mechanisms must be in place to recover user and application data in case of unexpected crashes or data loss.

5.4 Testing

Testing is a critical phase in the development of the Automated Platform for Undergraduate Admissions as it ensures the system performs correctly, securely, and efficiently before deployment. It involves validating both functional and non-functional aspects of the system to confirm that the platform meets all specified requirements.

The testing process is divided into several levels and techniques to cover every component, role, and scenario.

5.3.1 Types of Testing Performed

a. Unit Testing

- Focuses on individual components such as:
  - Login authentication
  - Admission form submission
  - Enrollment logic
- Each function or module is tested separately to ensure correct output.
- Example: Testing if a student cannot submit the form without required fields.

b. Integration Testing

- Tests how different modules interact with each other.
- Ensures smooth flow between:

- Student form submit→Admin approval→Teacher enrollment
- Authentication→Role-based dashboard access

- Example: Ensuring data entered by a student is visible and editable by the admin.

c. System Testing

- End-to-end testing of the entire platform in a real-world scenario.
- Verifies whether the complete system meets the specified requirements.
- Example: Simulating the entire admission cycle from registration to final enrollment.

d. Acceptance Testing

- Done to validate whether the system is ready for delivery or deployment.
- Usually done with feedback from university staff or selected testers.
- Ensures the system is user-friendly and performs all expected tasks.

e. Regression Testing

- After fixing bugs or adding new features, regression testing ensures that existing functionality is not broken.
- Example: Testing login and form submission again after making changes to document upload logic.

5.4.2 Test Cases Examples

Test Case ID	Test Scenario	Expected Result	Status
TC01	Student login with valid credentials	Login successful and student dashboard loads	Pass
TC02	Submit admission form with missing fields	Error message shown and submission prevented	Pass
TC03	Admin rejects application	Student status updated to "Rejected"	Pass
TC04	Teacher enrolls student in wrong department	Enrollment blocked with error message	Pass
TC05	System handles 100 simultaneous users	No crash or slowdown observed	Pass

5.4.3 Tools Used

- Manual Testing using real browser environments.
- Postman for API testing and Backend testing.

5.4.4 Bug Tracking

A simple bug tracking sheet was maintained for logging:

- Bug ID
- Description
- Priority (Low, Medium, High)
- Assigned Developer
- Status (Open, Fixed, Retested)

5.4.5 User Testing / Feedback

- After initial testing, a demo version of the system was shared with selected faculty and students.
- Feedback was collected on:
- Ease of use

  Clarity of forms

  Pperformance and download speed
- Suggestions were implemented to improve UI and add tooltips and status indicators.

5.4.6 Final Testing Summary

- All critical modules passed testing successfully.
- System was functionally stable, secure, and responsive under normal and high load conditions.
- Approved for deployment in a controlled live environment.

5.5 Purpose of Testing

Testing is an essential part of the software development lifecycle. In the context of the Automated Platform for Undergraduate Admissions, the primary purpose of testing is to ensure that the system operates correctly, securely, and efficiently, while also meeting all defined functional and non-functional requirements. It helps in detecting defects early, verifying logic and workflows, and ensuring user satisfaction.

5.5.1 To Ensure Correct Functionality

The most fundamental purpose of testing is to confirm that each module and feature of the system performs as intended. For example:

- Students must be able to submit application forms.
- Admin must be able to approve or reject applications.
- Teachers must successfully enroll students into the correct department and courses.

Without thorough testing, these functions may behave incorrectly, leading to a flawed admission process.

5.5.2 To Identify and Fix Errors Early

Testing helps detect:

- Logical errors
- Validation failures
- Navigation issues
- Broken links or dead-end workflows
- Data processing bugs

Finding and fixing these issues early during development helps reduce future risks and avoids costly errors after deployment.

5.5.3 To Validate Workflow and User Roles

Since the system includes three distinct user roles (Admin, Teacher, and Student), testing ensures that each role:

- Has access only to permitted actions.
- Sees the correct interface and data.
- Experiences the system as per their responsibilities.

This prevents misuse, unauthorized access, and confusion

5.5.4 To Verify Security and Data Integrity

Testing confirms the platform's resistance to common threats like:

- Unauthorized access
- Data leaks
- Malicious form inputs
- Incorrect user access control

Security testing ensures the personal and academic data of students is protected, especially during form submission and document uploads.

5.5.5 To Ensure Performance Under Load

Admission systems are most active during specific months. Testing ensures the system:

- Performs well under high user load.
- Doesn't crash or slow down.
- Handles simultaneous logins, form submissions, and queries smoothly.

This is crucial to ensure a seamless experience during peak periods of student applications.

5.5.6 To Enhance User Experience

By identifying bugs, interface issues, or confusing workflows, testing helps:

- Improve form usability
- Ensure responsiveness across devices
- Provide clear notifications and instructions

User feedback testing (UAT) further helps in aligning the platform with real-world expectations of students and faculty**.**

5.5.7 To Maintain System Reliability and Stability

Testing proves that the system:

- Runs without crashing
- Handles unexpected inputs gracefully
- Recovers from minor failures without losing data

A stable system builds trust and confidence among users, especially in a critical process like university admissions.

5.5.8 To Ensure Compliance with Requirements

Testing verifies whether all the functional specifications and non-functional goals (such as speed, usability, and compatibility) outlined in the design documents have been achieved. It helps the development team confirm that everything planned has been delivered correctly and nothing critical is missing.

5.6 Test Cases

The test cases below are designed to cover the core features and user roles of the system: Admin, Teacher, and Student.

Test Case ID: TC-001
Title: Student Login with Valid Credentials
Objective: Verify that students can log in using correct credentials.

Steps:

1. Open the APFUA web portal.
2. Enter a registered student email and password.
3. Click the "Login" button.

   Expected Result: Student is successfully logged in and redirected to the student dashboard.

Test Case ID: TC-002
Title: Admin Login and Access Control
Objective: Ensure admin can log in and view admin-exclusive features.
Steps:

1. Log in with valid admin credentials.
2. Navigate to admin-only pages (e.g., Approve Applications, Generate Merit List).
   Expected Result: Admin is able to access authorized features and restricted access is enforced for other roles.

Test Case ID: TC-003

Title: Submit Admission Form with Incomplete Fields

Objective: Verify that form submission with missing fields triggers validation errors.

Steps:

1. Log in as a student.
2. Navigate to the admission form page.
3. Leave mandatory fields (e.g., CNIC or Academic History) blank.
4. Click "Submit."

Expected Result: Error messages displayed indicating the required fields must be filled.

Test Case ID: TC-004

Title: Upload Oversized Document File

Objective: Check file size restrictions during document upload.

Steps:

1. Navigate to the document upload section.
2. Select and attempt to upload a file greater than 5MB.
3. Click "Upload."

Expected Result: System displays error: "File size exceeds the 5MB limit."

Test Case ID: TC-005

Title: Apply Role-Based Access Restriction

Objective: Ensure users can only access features appropriate to their role.

Steps:

1. Log in as a student.
2. Attempt to access admin routes (e.g., /admin/panel).

Expected Result: Access denied message is shown or user is redirected to unauthorized page.

Test Case ID: TC-006

Title: Upload Valid Academic Document (PDF)

Objective: Verify successful upload of a valid academic document.

Steps:

1. Log in as a student.
2. Navigate to the document upload page.
3. Select a valid Matric PDF file under 5MB.
4. Click "Upload."

Expected Result: File is successfully uploaded with confirmation message.

Test Case ID: TC-007

Title: Registration with Existing Email

Objective: Ensure duplicate email registrations are rejected.

Steps:

1. Open the registration page.
2. Enter an email already in use.
3. Fill in all other valid details.
4. Click "Register."

Expected Result: Error message displayed: "Email already exists."

Test Case ID: TC-008

Title: Invalid CNIC Format on Registration

Objective: Validate format of national ID field during signup.

Steps:

1. Enter a CNIC in incorrect format (e.g., 12345).
2. Submit registration.

Expected Result: Message shown: "Please enter a valid CNIC number."

Test Case ID: TC-009

Title: Session Timeout After Inactivity

Objective: Check auto-logout after idle session.

Steps:

1. Log in as any user.
2. Stay inactive for 10 minutes.
3. Try performing an action.

Expected Result: User is logged out and redirected to login page.

Test Case ID: TC-010

Title: View and Update Academic Info

Objective: Ensure students can view and edit their academic history.

Steps:

1. Log in as student.
2. Navigate to Profile > Academic Info.
3. Edit fields and click "Save."

Expected Result: Academic details updated successfully.

Test Case ID: TC-011

Title: Submit Application Form with Attachments

Objective: Ensure students can upload documents while submitting the form.

Steps:

1. Fill out application form.
2. Upload Matric and Inter result cards.
3. Submit the form.

Expected Result: Application submitted successfully with documents.

Test Case ID: TC-012

T**itle:** Upload File in Unsupported Format Block uploads of non-image/PDF files.

Steps:

1.Try uploading a `.exe` or `.mp4` file.

Expected Result: Error: "Invalid file format."

Test Case ID: TC-013

Title: Upload More Than Max Allowed Documents
Obeective: Prevent over-uploading of files.
Steps:
1. Upload 5+ files in document upload section.

Expected Result: Only first 3 accepted; others blocked with message.

Test Case ID: TC-014

Title: Submit Payment with Expired Card **Objective:**
Ensure card validation is enforced.
Steps:
1. Enter valid Visa card number but expired date.
2. Click "Pay."

Expected Result: Message shown: "Payment failed. Expired card."

Test Case ID: TC-015

Title: Attempt Duplicate Fee Payment **Objective:**
Prevent duplicate transactions.
Steps:
1. Pay fee successfully.
2. Try paying again.

Expected Result: Message: "Payment already received."

Test Case ID: TC-016

Title: Book Entry Test Slot

Objective: Allow students to schedule entry test.

Steps:

1. Go to Test Schedule section.
2. Select a valid available date.
3. Click "Book Slot."

Expected Result: Slot booked successfully with confirmation message.

Test Case ID TC-017

Title :Select Already Full Test Slot Prevent overbooking of test slots.

Step:

1.Attempt to book a slot already at capacity.

Expected Result: Message: "Slot full. Please choose another."

Test Case ID: TC-018

Title: Approve Student Application (Admin)

Objective: Allow admins to approve submitted forms.

Steps:

1. Log in as admin.
2. Open pending applications.
3. Click "Approve" on a selected student.

Expected Result: Status changes to “Approved.”

Test Case ID: TC-019

Title: Generate Merit List

Objective: Ensure admin can generate sorted list based on scores.

Steps:

1. Navigate to Admin Panel.
2. Click “Generate Merit List.”

Expected Result: Merit list shown in descending order of marks.

Test Case ID: TC-020

Title: Send SMS/Email After Approval

Objective: Validate that student gets notified after application status changes.

Steps:

1. Approve an application as admin.
2. Check student's inbox/SMS.

Expected Result: Notification: “Your application has been approved.”

Test Case ID: TC-021

Title: SQL Injection Protection in Login Form **Objective:**
Prevent malicious input in login fields.

Steps:

1. Enter `' OR '1'='1` as email or password.
2. Click "Login."

Expected Result: Error: "Invalid credentials."

Test Case ID:TC-022

Title: Access Admin Panel via Direct URL as Student Prevent unauthorized access via manual URL typing.

Steps:

1.Log in as student.

2. Enter `/admin` in address bar.

Expected Result: “Access Denied” page displayed.

Test Case ID: TC-023

Title: Student Downloads Admit Card

Objective: Ensure admit card PDF can be downloaded after slot booking.

Steps:

1. Book test slot.
2. Click “Download Admit Card.”

Expected Result: PDF is downloaded with exam details.

Test Case ID: TC-024

Title: Logout from All Devices

Objective: Ensure student can logout from all sessions.

Steps:

1. Log in on two browsers.
2. Click “Logout from all devices.”

Expected Result: All sessions invalidated; return to login.

Chapter 6

EXPERIMENTAL EVALUATIONS & RESULTS

6.1 Evaluation Testbed

An evaluation testbed is the arrangement and setting used to test, validate, and assess the system's dependability, performance, and usability prior to its final deployment. The assessment testbed for the Automated Platform for Undergraduate Admissions was created to mimic real-world scenarios using sample data, representative users, and real workflows in order to guarantee accurate and effective system operation.

6.1.1 Purpose of Evaluation Testbed

- To simulate the actual environment in which the system will run.
- To perform controlled testing of functionalities, load, and security.
- To evaluate how the platform behaves when accessed by real users (students, teachers, admin).
- To collect feedback for final improvements.

6.1.2 Components of the Testbed

a. Hardware Environment

- **Client Machines**: Standard desktop/laptop systems with different operating systems (Windows, macOS).
- **Server**: Local or cloud-hosted server used to host the test version of the platform.
- **Network**: Connected through university intranet and external internet access for remote testing.

b. Software Environment

- **Operating Systems**:
  - Windows 10, Ubuntu Linux (for deployment tests)
- **Web Browsers**:
  - Google Chrome, Mozilla Firefox, Microsoft Edge (to test compatibility)
- **Database Server**:
  - MySQL or MongoDB (for backend storage)
- **Backend Framework**:
  - Node.js (JavaScript) for business logic handling
- **Testing Tools**:
  - Postman (for API testing) ○ Manual Testing

6.1.3 Test Data Used

To properly evaluate the system, dummy data was created to reflect actual conditions:

- **50+ student accounts** with realistic personal and academic details.
- **5+ teacher accounts** assigned to different departments.
- **Sample admission applications** in various statuses (Pending, Approved, Rejected).
- Uploaded sample documents (marksheets, CNIC, etc.).
- Course lists and department structures.

6.1.4 Evaluation Scenarios

The testbed was used to run the following practical scenarios:

- **Scenario 1**: 20 students simultaneously submit applications.
- **Scenario 2**: Admin processes multiple applications and views reports.
- **Scenario 3**: Teachers enroll students into courses and generate rosters.
- **Scenario 4**: Multiple users log in at the same time to test server load.
- **Scenario 5**: Attempt unauthorized access to restricted modules to test security.

6.1.5 Outcome of Evaluation

- System met performance and security expectations.
- Minor UI improvements and wording changes were suggested and implemented.
- No major functional bugs were found during testing.
- System marked **ready for final deployment** in the university department.

6.2 Results and Discussion

The automated admissions platform's main objective of integrating and simplifying the whole admissions procedure within the Department of Hamdard University was effectively accomplished. During the testing phase, every module functioned perfectly, from teacher-based class enrollments and admin approvals to student registration and form submission. It was simple for students to upload documents, submit applications, and monitor their admissions status. Even people with no technical background found the interface to be responsive and easy to use.

Administrative tasks like handling instructor and student accounts, evaluating applications, and keeping track of the admissions process as a whole were completed well. The administrator could easily approve or reject applications thanks to the centralized dashboard, and system automation reduced time and human error. The efficiency of the role-based access system and enrollment logic was further demonstrated by the teachers' successful access to their individual student lists and enrollment in department-specific classes.

Even with several users using it at once, the system demonstrated stability and responsiveness during testing. User feedback indicated that, in comparison to manual operations, the platform greatly decreased workload and confusion. The system's adaptability was demonstrated by the addition of tooltips and rapid filters, among other minor ideas. Overall, the findings demonstrated that the project had accomplished its goals and offered a dependable, open, and expandable undergraduate admissions management solution.

CHAPTER 7

CONCLUSION AND DISCUSSION

7.1 Strength of this Project

One of the most significant strengths of this project is that it provides a complete automation of the undergraduate admission process, removing the dependency on manual paperwork, in-person queues, or scattered records. The entire flow from student registration and form submission to admin approval and teacher-based enrollment happens within a centralized platform. This not only saves time but also ensures accuracy, proper tracking, and smooth flow of admission operations across the department.

Another major strength lies in the well-defined role-based system. Each user whether a student, teacher, or admin has specific access and functions according to their role. Admins have full control over the system and user management, teachers can enroll approved students into relevant classes, and students can track their own application progress in real time. This structure adds a layer of security, clarity, and accountability, reducing errors caused by unauthorized actions or confusion.

Lastly, the platform is designed to be simple, responsive, and user-friendly, making it accessible even for users with limited technical knowledge. Real-time notifications, guided forms, and clean dashboards make the user experience smooth and intuitive. Most importantly, the system is scalable, meaning it can easily be extended to other departments or integrated with future university-wide systems. This flexibility makes it a strong, long-term solution for improving the digital infrastructure of the admission process at Hamdard University

7.2 Limitations and Future Work

While the project successfully addresses the major challenges of manual admissions, there are still a few limitations that were observed during its development and testing. For instance, the current system focuses mainly on the admissions process within a single department, which means it may require structural adjustments before it can be scaled to work across the entire university. Also, while document upload and status tracking features are functional, there's room to improve how notifications are handled especially in terms of real-time alerts or SMS-based communication.

Another limitation is the manual intervention required in some areas, such as when teachers enroll students into classes. Though the process is digitized, it still depends on individual teacher input, which can be time-consuming during peak admission periods. Additionally, the absence of features like payment integration for application fees or auto-generated admission slips is something that can be considered for future versions of the system. Minor UI/UX improvements were also noted by users, such as adding visual progress bars or more guided tooltips.

Looking forward, there is great potential to enhance and expand this platform. In the future, the system could be connected with other university departments and databases to allow seamless communication across the academic ecosystem. Integrating features like **online** fee payment, interview scheduling, digital verification of documents, and mobile app support can take this project to the next level. These future improvements would not only make the system more robust but also more student-friendly, helping the university take a step closer to full digital transformation in education services.

7.3 Reasons for Failure – If Any

Although the project was overall a success, there were a few minor setbacks and challenges encountered during its development and implementation phases. One of the early difficulties was related to integrating all three user roles (Admin, Teacher, Student) in a seamless way. Initially, role-based navigation had bugs where users were being redirected incorrectly or accessing modules not meant for them. These issues were not due to failure in planning but rather small gaps in testing and early logic setup.

Another reason for minor delays was the lack of real-world data in the early development phase. Since real student records couldn't be used for privacy reasons, dummy data had to be created manually. This slowed down the testing process and sometimes didn’t fully represent the complexity of live admissions, which caused a few unexpected results when the system was exposed to more dynamic user behaviour later on.

Lastly, time constraints and limited team resources slightly restricted the system’s scope. Some features that were initially planned such as online fee payment, interview scheduling, and a mobile app version had to be postponed for future work. However, these are not considered complete failures, but rather functional compromises made to ensure that the core system was stable and ready for launch within the set timeline.

REFERENCES

1.Creatrix Campus.

Student Admission Management System. Available at https://www.creatrixcampus.com/studentadmission-management-system.

This resource explains features and benefits of automated admission management systems.

2.**HEIApply**.

Admissions Software for Higher Education. Available at https://heiapply.com/educationadmissions/admissions-software-higher-education. A comprehensive guide to features, functionalities, and use cases of automated platforms for admissions.

3.**Capterra**.

Best Admissions Software Tools & Reviews. Available at https://www.capterra.com/admissionssoftware/

A detailed review and comparison of various tools available for automating admission processes.

APPENDICES

List of Appendices

A0. Copy of Project Registration Form
A1a. Project Proposal and Vision Document
A1b. Copy of Proposal Evaluation Comments by Jury
A2. Requirement Specifications
A3. Design Specifications
A4. Other Technical Details
Test cases
UI/UX Details
Coding Standards
Project Policy
A5. Flyer & Poster Design
A6. Copy of Evaluation Comments
Copy of Evaluation Comments by Supervisor for Project – I Mid Semester Evaluation
Copy of Evaluation Comments by Jury for Project – I End Semester Evaluation
Copy of Evaluation Comments by Supervisor for Project – II Mid Semester Evaluation
Copy of Evaluation Comments by Jury for Project – II Mid Semester Evaluation
Copy of Evaluation Comments by Jury for Project – II End Semester Evaluation
A7. Meetings' Minutes
A8. Research Paper
A10. Any other

A0. COPY OF PROJECT REGISTRATION FORM

FYP -PSF-2024

Hamdard University
Faculty of Engineering Sciences and Technology

Department of Computing

FINAL YEAR PROJECT - PROPOSAL SUBMISSION FORM

Project Details: (to be filled-in by student)

Project Title: FEST Admission Flow Automation

Project Track: ☑ Product ☐ Service ☐ Research

Program of Study: Bachelor in Computer Science **Session:** Fall - 21

Expected Completion Date: June 2025 **Date:** 24th June 2024

Project Member(s): (to be filled-in by student; student #1 is the team lead)

S#	Name	CMS ID	Roll #	Cell #	E-mail ID	Signature
1	Waleed Ahmed	1734-2021		03122602807	waleed.techverk@gmail.com	
2	Dua Rahim	1597-2021		03118244494	duarahim4404@gmail.com	
3	Zoya Sayeed	1696-2021		03301432737	misszoyasayeed@gmail.com	

Supervisor Recommendation: (to be filled-in by the supervisor and co-supervisor, if any)

Any extra project-domain-specific course requirement: _____

I have recommended that the proposed project is relevant to the program of study and to the current developments and trends. The project will be beneficial for the students and can be completed within the given time and with mentioned resources. I furthermore verify that students have cleared all the pre-requisite courses and attained sufficient CGPA to be eligible for FYP. All the above students have completed at least 75 credit hours in their respective programs. Transcript, verified CGPA of each student & proposal report document of group are attached with this form.

Supervisor Name: Sir Iqbal Uddin Khan **Signature:** _____

Designation: Assistant Professor **Organization:** Hamdard University

Co-Supervisor Name: _____ **Signature:** _____

Designation: _____ **Organization:** _____

(For Office Use)

Convener FYP Committee:

☐ Approved ☐ Not Approved

Name: _____

Signature: _____

Comments: _____

Advisor FYP Committee:

☐ Approved ☐ Not Approved

Name: _____

Signature: _____

Comments: _____

A1A. PROJECT PROPOSAL AND VISION DOCUMENT

1 Introduction
1.1 Problem Statement
1.2 Project Motivation
1.3 Objectives
1.4 Literature Review
2 Project Vision
2.1 Business Case and SWOT Analysis
2.2 Background, Business Opportunity, and Customer Needs
2.3 Business Objectives and Success Criteria
2.4 Project Risks and Risk Mitigation Plan
2.5 Assumptions and Dependencies
3 Project Scope
3.1 In Scope
3.2 Out of Scope
4 Proposed Methodology
4.1 SDLC Approach (Waterfall/Agile/any model)
4.2 Team Role & responsibilities
4.3 Requirement Development
4.4 High-level Architecture / Design
4.6 Application (or Project) Testing
5 Project Planning
5.1 Gantt Chart
6 Project Requirements
6.1 Software tools requirements
6.2 Hardware requirements
7 Budget/Costing
7.1 Mention the budgeting cost of each item - required for this project
7.2 Estimated Budgeted Cost - of the Project
8 Project Deliverables
8.1 Phase I - Alpha Prototype
8.2 Phase II - Beta Prototype
8.3 Phase III - Release Candidate
8.4 Phase IV - Final Product
9 Proposed GUI (Disposable Prototype)
10 Meetings held with supervisor and/or client.
11 Reference

1. Introduction

This project aims to address a solution for Admissions Flow in FEST. The proposed solution focuses on Reliability, Flexibility , Security, Integrity of Admission Flow which has the potential to significantly improve the complication occurs as a Manual working for Admissions.

- **Identifying the problem**

  As HU_ Admission team faces difficulties in manual system managing the Team work for Admissions and Enrollment.

- **Propose a solution**

  Designing a software that can reduce barrier faced by Admission team.

- **Implement the solution**

  Build a functional software that is useful with scrum methodology to work on Admission Flow easier.

- **Evaluate the impact**

  Assess the effectiveness and benefits of the proposed solution

2. Objective

- **Enhance Accuracy**

  Ensures that all the data is correct and minimize human errors in data entry and processing also, Carefully review each application to verify completeness and accuracy.

- **Consistency**

  Apply admission criteria uniformly across all applicant to ensure fairness.

- **Efficiency**

  Reduce the time and efforts required to process applications and inquiries promptly to avoid any mishap**.**

- **User-Friendly Interface**

  Allowing applicants to submit and track their applications online from anywhere anytime.

- **Data Management and Scalability**

  Improve the organization, storage, and retrieval of application data. Handles larger volume of applications without a corresponding increase in workload.

- **Transparency**

  Provide real-time updates and notification to applicants about their application status. Keep all document and data organized and easily accessible.

- **Compliance and Services**

  Ensures that the process adheres to all relevant policies and regulations. Maintain clear communication with applicants, addressing any concerns or query they may have.

- **Experience and Evaluations**

  Enhance the overall experience for applicant as well as administrative staff.

3. Problem Description

The Admission Flow Automation project aims to streamline and enhance the efficiency of the admission process in educational institutions. This project will develop a comprehensive, userfriendly system that automates the various stages of admission, reducing manual intervention, minimizing errors, and improving the overall applicant experience. The Admission Flow Automation project aims to revolutionize the way educational institutions handle admissions by leveraging technology to create a streamlined, error-free, and applicant-friendly system. This project not only benefits the institution by optimizing its resources but also enhances the overall experience for applicants, making the admission process smooth and hassle-free.

4. Methodology

SCRUM _ Methodology

- **Sprints**

The project is divided into short iterations called sprints, typically lasting 2-4 weeks. • **Daily Stand-up Meetings**

The team gathers daily to discuss progress, obstacles, and plans for the day.

- **Sprint Planning**

At the beginning of each sprint, the team selects a set of tasks from the product backlog to complete.

- **Sprint Review**

At the end of each sprint, the team demonstrates the completed work and gathers feedback from stakeholders.

- **Sprint Retrospective**

The team reflects on the sprint, identifying what went well and what could be improved.

5. Project Scope

1. **Vulnerability Assessment:** Conduct thorough automated assessments to identify and prioritize vulnerabilities in the target systems, ensuring a comprehensive understanding of potential security risks.
2. **Patch Management**: Facilitate informed decision-making for administrators by providing insights into available patches and streamlining the patch application process, thereby addressing identified vulnerabilities promptly.
3. **Assessment Reporting**: Generate detailed and actionable assessment reports, highlighting vulnerabilities and providing recommendations for remediation, ensuring that stakeholders have a clear understanding of the security posture.

4. **Compliance Standards:** Implement compliance monitoring mechanisms aligned with industry standards to ensure that organizations meet regulatory requirements and adhere to best practices in cybersecurity.
5. **User Management:** Establish a system that logs and monitors user activities, configurations, and changes, maintaining an audit trail for oversight and compliance. Implement secure user management with role-based access control and authentication mechanisms to ensure proper access restrictions.

6. Feasibility Study

This project proposes a comprehensive solution to address a critical need within the Admission Team. The proposed system leverages technology to enhance the efficiency and accuracy of our admissions process. The outlined phases encompassing system design, development, integration, testing, and deployment are structured to align with the objectives and anticipated timelines.

Risks Involved:

1. **Automation Challenges:** The complexity of developing a system for automated paper generation may pose challenges.
2. **Data Availability:** The assumption of a diverse question bank relies on the availability of comprehensive and varied questions.

Mitigation Strategies:

1. **Iterative Development:** Implementing an iterative development approach allows for continuous refinement of system based on testing and feedback.
2. **Collaboration with Educational Institutions:** Partnering with educational institutions ensures access to a wide range of quality questions, mitigating potential data availability issues.

Resource Requirement:

1. **Development Servers:** Adequate servers for hosting and testing during the development phase.
2. **Cloud Services:** Utilization of cloud services for scalability and efficient resource management.

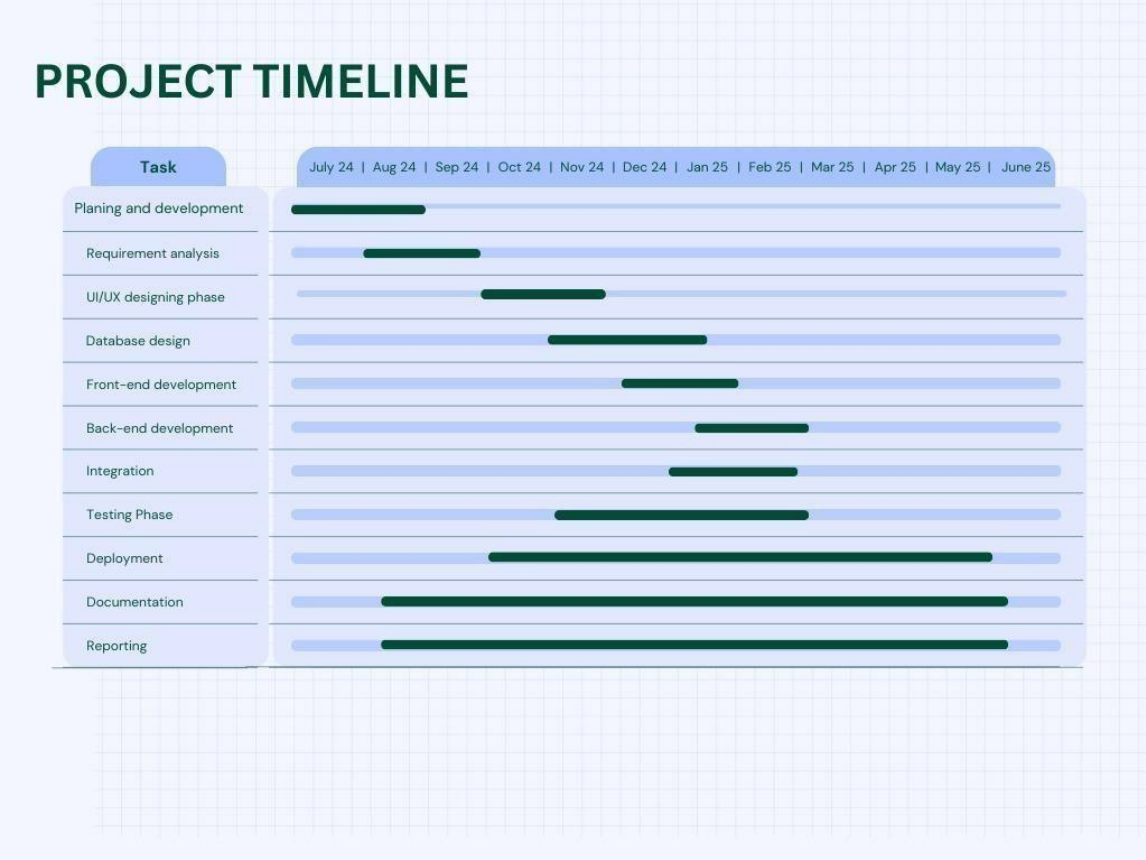
Human Resources:

1. **Development Team:** Skilled developers for system development and paper generation integration.
2. **Quality Assurance Team:** Testing and debugging require a dedicated quality assurance team.
3. **Educational Collaborations:** Collaborating with educational institutions for question bank access.

Time Resources:

1. **Project Timeline:** Adherence to a well-structured project timeline ensures efficient use of time resources.
2. **Timely Feedback:** Timely feedback loops during the development and testing phases facilitate swift issue resolution.

7. Planning GANTT CHART



8. Solution Application Area

The Automated Admission Flow System for higher education institutions offers significant value in the academic sector. Targeting universities, colleges, and training centers, the system enhances operational efficiency by automating the entire admission process. It ensures accuracy and fairness through objective data validation, adapts seamlessly to different program requirements, and provides valuable data-driven insights for administrators. The system's scalability caters to institutions of varying sizes, ultimately contributing to a more effective, transparent, and streamlined admission process.

9. Tools/Technology

- Visual Studio Code
- Html
- CSS
- JavaScript
- React JS / Next JS
- Node JS / Express JS
- Tailwind CSS
- Material UI
- MongoDB
- Firebase
- Figma

10.Responsibilities of the Team Members

R	Responsible
A	Accountable
C	Consulted
I	Informed

Task	Waleed Ahmed	Dua Rahim	Zoya Sayeed	Sir Iqbal Uddin
GUI Design	R	A	A	I,C
Integration	A	R,A	R	I,C
Testing	R	R	A	I,C
Documentation	A	R,A	R,A	I,C
Feedback	R	R	R	I,C

11.References

- https://www.creatrixcampus.com/student-admissionmanagement-system
- https://heiapply.com/education-admissions/admissionssoftware-higher-education/
- https://heiapply.com/education-admissions/admissionssoftware-higher-education

A1B. COPY OF PROPOSAL EVALUATION COMMENTS BY JURY

	Sir Asim Hussain	Can you complete your prototype's target in proper manner and tim

Sir Salman	Appreciated because they are developing their FYP for the betterment of our university.

Sir Jibran Rasheed	To upgrade the documentation

Dr Adnan Ahmed Siddiqui	To revise FYP name and to integrate chatbot

Supervisor's Comments

Name can be changed proposed by respected jury, Chatbot may be added as it will be an Extended Feature – We will see if we can do after completion of scope

Mr. Iqbal Uddin
(Project Supervisor)
2

Date: 22/07/2024

A2. REQUIREMENT SPECIFICATIONS

Any standard template may be used, as per project need approved by Project Coordinator & Supervisor. Following is a suggestive outline.

1. Introduction
1.1. Purpose of Document
1.2. Intended Audience
1.3. Abbreviations
2. Overall System Description
2.1. Project Background
2.2. Project Scope
2.3. Not In Scope
2.4. Project Objectives
2.5. Stakeholders
2.6. Operating Environment
2.7. System Constraints
2.8. Assumptions & Dependencies
3. External Interface Requirements
3.1. Hardware Interfaces
3.2. Software Interfaces
3.3. Communications Interfaces
4. Functional Requirements
4.1. Functional Hierarchy
4.2. Use Cases
5. Non-functional Requirements
5.1. Performance Requirements
5.2. Safety Requirements
5.3. Security Requirements
5.4. User Documentation
6. References

Introduction

The objective of this project is to offer a practical way to improve the admissions flow at FEST (Faculty of Engineering, Science, and Technology). The suggested system addresses the difficulties of manual operations by enhancing the admissions process's security, flexibility, reliability, and integrity. Data mishandling, errors, and delays are common outcomes of manual processing. By automating the admissions workflow, the suggested approach increases accuracy, decreases administrative burden, and improves efficiency. Through encryption and access control, it guarantees the safe handling of sensitive data and provides real-time updates and improved tracking. With the goal of making the process easier for both employees and candidates, the system is made to be scalable and flexible enough to accommodate future modifications. This solution guarantees a more dependable, safe, and efficient method for FEST by revolutionizing the admissions process.

1.1 Purpose of Document

The purpose of this Software Requirements Specification (SRS) document is to provide an overview of the design, architecture, and functionality of the Vulnerability Assessment and Penetration Testing (VAPT) application, developed using Scrum methodology. It serves as a blueprint for the development team, outlining the technical requirements and specifications necessary for successful implementation. This document ensures alignment with user needs, providing clear guidelines for the application’s development, while setting milestones and tracking progress throughout the project.

1.2 Intended Audience

Development Team**:** Developers, architects, and engineers responsible for the design, implementation, and testing of the VAPT application.

Security Team**:** Security analysts, VAPT specialists, and information security officers focused on assessing and ensuring the application's security.

Project Managers: Individuals overseeing the entire development lifecycle, including planning, execution, and delivery.

Stakeholders**:** Individuals or groups with an interest in the functionality, security, and performance of the application, including clients, investors, and regulatory bodies.

2. Overall System Description

2.1 Project Background:

The admissions process at FEST (Faculty of Engineering, Science, and Technology) has been handled manually, leading to inefficiencies, errors, and delays. With the increasing number of applicants each year, the limitations of the manual system have become more evident, including difficulties in tracking data, verifying payments, and scheduling tests. The lack of automation has resulted in duplication of applications, communication delays, and a poor applicant experience. To address these issues, there is a clear need for an automated admissions system that streamlines data entry, payment verification, test scheduling, and communication. The proposed solution will improve accuracy, reduce administrative burdens, and enhance efficiency, providing a better experience for both staff and applicants, while supporting the growing volume of applications.

2.2 Problem Statement

The current manual processes at FEST for handling admissions present several challenges. Manual data entry is a key issue, as admission staff manually collect and input student data into Excel sheets, leading to potential errors and data duplication. Additionally, there is no automated system in place to check for duplicate applications, resulting in multiple entries for the same student. Payment verification is another area of concern, as the process is manual, time-consuming, and prone to errors. Delays in verifying admission fee payments can lead to postponements in scheduling admission tests. Similarly, test scheduling is inefficient when done manually, often causing scheduling conflicts and miscommunication with students. Furthermore**,** communication delays arise from the manual notification of students about test dates, eligibility, and admission offers. The lack of an automated communication system not only results in delays but also leads to a poor experience for prospective students, as they are left uncertain about the status of their applications.

2.3 Project Scope

The main objectives of this project are:

1-To automate the admissions process at FEST, reducing human error and delays.

2-To improve the security of sensitive applicant data through encryption and access control.

3-To enhance the overall user experience for both applicants and administrative staff.

4-To increase efficiency by providing real-time updates and better tracking of applications.

5-To ensure the system is scalable and flexible to accommodate future changes in admission requirements.

2.4 Not in Scope

The scope of this project does not include manual processes beyond the initial migration of data into the system. Additionally, the system will not integrate with university-wide platforms or portals other than the admissions platform itself. Postadmission management, such as course registration and tracking academic progress, is also outside the project scope. Furthermore, advanced features such as AI-based predictive analytics or machine learning for forecasting admissions trends are excluded from the current development.

2.5 Project Objectives

Enhance Accuracy: Ensure correct data handling, reducing human errors in the admission process.

Consistency: Apply admission criteria uniformly across all applicants for fairness.

Efficiency: Reduce time and effort required to process applications, payments, and notifications.

User-Friendly Interface: Provide an intuitive platform for applicants to submit and track applications.

Data Management & Scalability: Enable efficient storage and retrieval of data to handle large application volumes.

Transparency: Provide real-time updates and notifications about application status.

Compliance: Ensure adherence to institutional policies and regulations.

Improved Experience: Enhance overall satisfaction for applicants and administrative staff.

2.6 Stakeholders & Affected Groups

Applicants: The primary users of the admissions system who will benefit from a faster, more secure application process.

Administrative Staff: Employees who manage the admission process and will experience reduced manual workload and improved system efficiency.

Project Managers: Individuals overseeing the project's execution and ensuring its alignment with timelines and objectives.

FEST Faculty and Department Heads: Groups that require accurate, timely data on admissions to support resource planning.

Security Analysts: Personnel responsible for ensuring the safe handling and storage of sensitive data throughout the admissions process.

2.7 Operating Environment

The platform will operate as a secure web-based system accessible via both desktop and mobile devices. The environment is designed to ensure high availability, as it must remain functional during peak admissions periods. Robust security measures will be implemented to safeguard sensitive student data. The backend will leverage MongoDB or Firebase for efficient data storage, while the frontend will use React/Next.js with Tailwind CSS for an intuitive and responsive user interface. The system will be hosted on cloud infrastructure to ensure scalability and reliability, accommodating large user volumes and evolving institutional needs.

2.8 System_Constraints

The project faces constraints including a limited timeframe for development, as it must be completed within the academic deadlines set for final year projects. Budget limitations restrict the purchase of additional hardware or premium software services. Institutional policies and regulatory standards may impose boundaries on system features and functionality. Resource limitations, such as a small team size and specific skillsets, also influence the project's scope and progress.

2.9 Assumptions & Dependencies

The success of the project assumes reliable internet connectivity for both applicants and university staff to access the system. Adequate hardware resources, such as development servers and devices for testing, are assumed to be available throughout the project lifecycle. The project's success also depends on active cooperation from Hamdard University for requirements gathering, system testing, and deployment. The system relies on third-party tools and frameworks, including Visual Studio Code, MongoDB, and Firebase, for development and hosting. Furthermore, regular engagement and constructive feedback from stakeholders are critical for ensuring the project aligns with their needs and expectations.

3. External Interface Requirements

3.1 Hardware Interfaces

The system requires specific hardware interfaces to support its operation effectively. On the user side, it requires devices capable of running modern web browsers, including desktops, laptops, tablets, and smartphones. For development and hosting, the system utilizes development servers equipped with adequate processing power and memory, such as machines with a minimum configuration of Intel Core i5 processors, 8GB RAM, and 500GB SSDs. Backup storage, such as external hard drives with at least 250GB capacity, will be used to ensure data redundancy and recovery. The physical addresses and logical structure for server hosting will align with cloud-based solutions for scalability and high availability. The expected behavior includes seamless data transfer, efficient processing of application requests, and reliable access across multiple hardware configurations.

3.2 Software Interfaces

The system interfaces with several applications to ensure seamless functionality. These include:

1. Database:
   - Name: MongoDB / Firebase
   - Owner: Project Team
   - Interface Details: The system interacts with MongoDB or Firebase for real-time data storage and retrieval, ensuring scalability and efficient data management.
2. Frontend Framework:
   - Name: React.js / Next.js
   - Owner: Project Team
   - Interface Details: The frontend application communicates with the backend through REST APIs to provide an interactive and responsive user experience.

3. Backend Framework:
   - Name: Node.js / Express.js
   - Owner: Project Team
   - Interface Details: The backend processes application logic, communicates with the database, and serves data to the frontend using secure API endpoints.

4. UI Design Tool:
   - Name: Figma
   - Owner: Third-party Provider
   - Interface Details: Used during the design phase for creating wireframes and mockups, ensuring consistency in UI/UX design.

3.3 Communications Interfaces

The system relies on several communication interfaces to facilitate connectivity and interaction with other systems and devices. A primary communication interface is the local area network (LAN) or internet connection, which ensures the system's web-based nature is accessible across devices and locations. Secure HTTP (HTTPS) protocols will be used for data transmission, ensuring encryption and secure interactions between clients and servers. Additionally, the system may interface with email servers for sending notifications and updates to applicants and staff. APIs will enable integration with payment gateways for secure fee processing and with potential university-wide systems for data exchange. These interfaces ensure the system remains reliable, secure, and capable of supporting real-time communication needs.

4. System Functions / Functional Requirements

4.1 System Functions

The system is designed to automate the undergraduate admissions process, ensuring accuracy, efficiency, and user satisfaction.

1. End-User Functions

- Applicants can create accounts, submit applications, make payments, and track their application status.
- Admission staff can review applications, schedule tests, and manage communications with applicants.

2. Operator Functions

- Manage user roles and access levels, ensuring secure role-based functionality for applicants, admins, and support staff.
- Oversee and modify application workflows, including deadlines, criteria, and admission stages.

  Generate summary reports on applications, payments, and test results for decision-making.
- Monitor system logs and user activity for auditing and compliance purposes.

3. Support Functions

- Assist applicants in account creation, application submission, and payment-related queries.
- Troubleshoot and resolve technical issues related to login, form submission, and notifications.
- Provide real-time support during admission test scheduling and result announcements.
- Facilitate communication between applicants and the admissions team for personalized assistance.

4. Integration Functions

- Connect securely with payment gateways for real-time fee verification and processing.
- Integrate with email and SMS services for automated notifications to applicants. Synchronize with third-party tools (e.g., MongoDB/Firebase) for storage and retrieval of application data.
- Enable API-based communication with university-wide systems for seamless data exchange.

4.2 Performance Requirements

- The system should handle up to 1,000 concurrent users without noticeable performance degradation.
- All operations, including form submission and status updates, should complete within 5 seconds

4.3 Design Constraints

- The system must adhere to web accessibility standards to ensure usability for all applicants.
- Secure encryption must be implemented for data transmission.

4.4 Programming Language

- Frontend: HTML, CSS, JavaScript, React.js, Next.js
- Backend: Node.js, Express.js
- Database: MongoDB or Firebase

4.5 Interface Requirements

- A user-friendly web interface for applicants and administrative staff.
- REST APIs for communication between the frontend and backend components.
- Secure integration with payment gateways for fee verification.

System Function Table

Ref#	Functions	Category	Attribute	Constraint
R1.1	Display admission portal forFall2024	Evident	Interface availability	Ensure the admission portal is accessible.
R1.2	Register new applicants	Evident	Input validation	Validate required fields (e.g., name, email, NIC).
R1.3	Allow program and campus selection	Evident	Dynamic dropdown	Options should be updated dynamically.
R1.4	Save applicant data to the database	Hidden	Data persistence	Ensure all submitted data is stored reliably.
R1.5	Display applicant information for editing	Evident	Editable fields	Allow users to update their info
R1.6	Manage concurrent user submissions	Hidden	Concurrent load handling	Support multiple users without delay.
R1.7	Provide password creation and confirmation	Evident	Password policy	Enforce strong password rules.
R1.8	Generate applicant ID	Hidden	Unique ID generation	Ensure each applicant has a unique identifier.
R1.9	Applicant General Info Page	Evident	Form Fields	Includes "Student Name," "Father Name," and other personal details.

R1.10	Program Preferences Selection Page	Evident	Program Options	Allows selection of campus, faculty, and preferences for programs.
R1.11	Qualification Details Page	Evident	Academic Records	Displays qualification details such as marks.
R1.12	Voucher Details Page	Evident	Payment Info	Displays voucher details like account title, amount, and status.
R1.13	Summary Pagefor Applicant	Evident	Review & Confirm	Provides a summary for final confirmation of application.
R1.14	Online Admission Portal Interface	Evident	Admin Options	Displays functions like "Print Offer Letter" and "Reset Password."
R1.15	Provisional Admission Offer Letter	Evident	Confirmation Letter	Shows details like name, father name, application ID, and semester fee.

System Attributes/ Nonfunctional Requirements

Attribute	Details and Boundary Constraints	Category
Response Time	The system must provide feedback like form submissions, and status updates within 5 seconds.	Mandatory
Concurrent User Load	The system should support at least 1,000 users connected simultaneously.	Mandatory
Interface Design	The interface must be browser-based, graphical, and intuitive for users.	Mandatory
Accessibility	The system must comply with web accessibility standards.	Mandatory
Scalability	The system must handle increased workloads without significant performance drops.	Optional

4.6 Use Cases

4.6.1 List of Actors

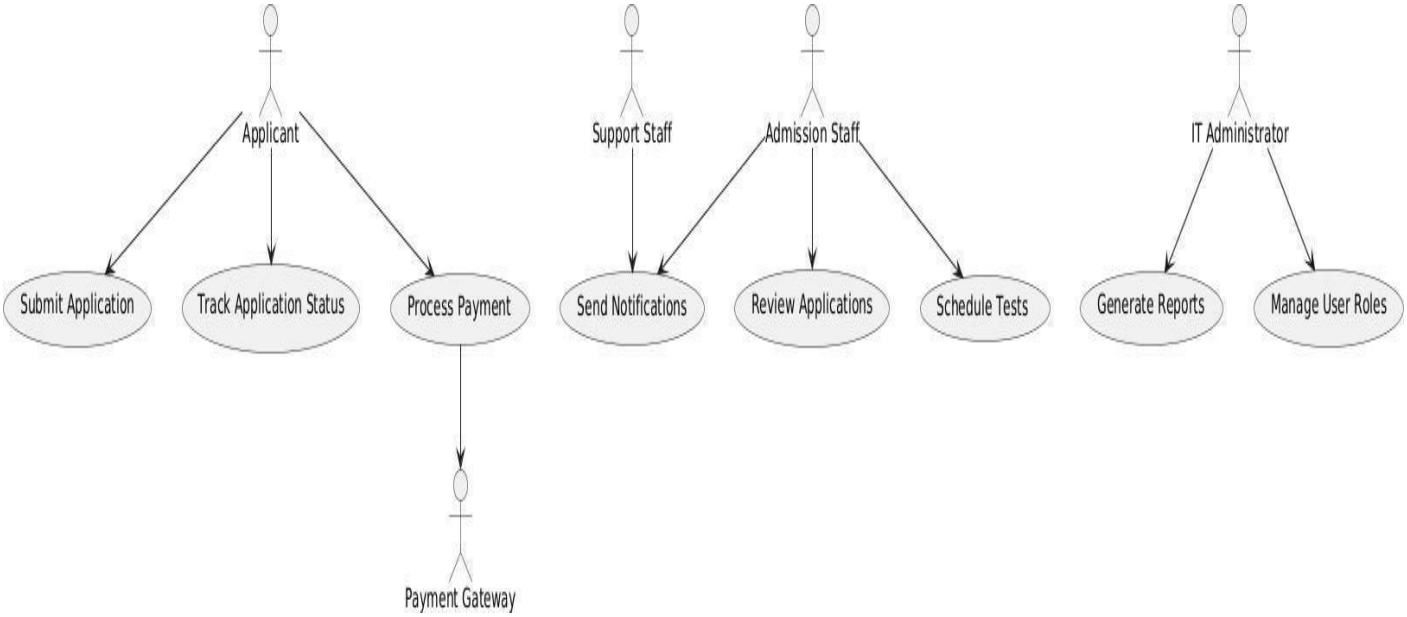
- Applicant: Submits applications, tracks their status, and makes payments.
- Admission Staff: Reviews applications, schedules tests, and communicates with applicants.
- IT Administrator: Manages user roles, monitors system health, and performs system maintenance.

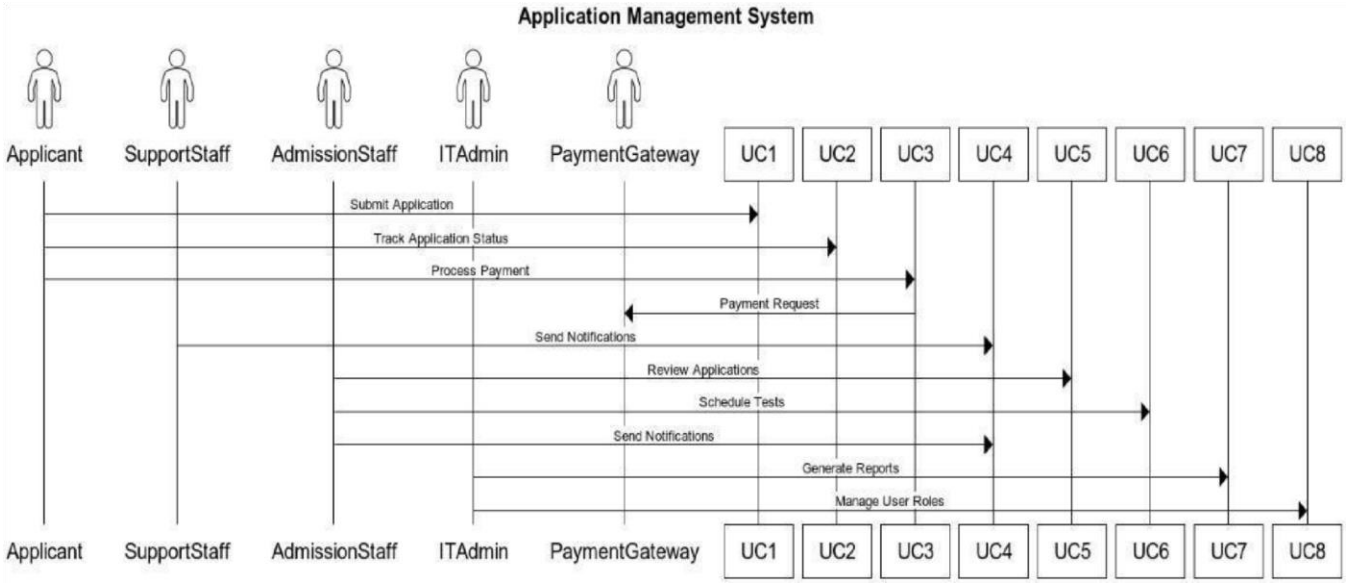
- Support Staff: Assists users with technical or process-related issues.
- Payment Gateway: Processes and verifies application fee transactions securely.

4.6.2 List of Use Case

- **Submit Application:**Applicants can submit their application forms online.
- **Track Application Status:** Applicants can view the progress and status of their applications
- **Process Payments:** The system verifies and processes application fee payments. 4-. 4- **Review Applications**: Admission staff reviews and validates submitted applications. 5- **Schedule Tests:**
- Admission staff schedules admission tests and communicates dates to applicants.
- **Generate Reports:** Administrators can create reports summarizing admission statistics.
- **User Role Management:** IT administrators assign and manage user roles for secure system access.

4.6.3 Use Case Diagram





4.6.4 Description of Use Cases

Section: Main	
Name:	Submit Application
Actors:	Applicant
Purpose:	Allow an applicant to complete and submit their application online.
Description:	An applicant creates an account, fills out the necessary forms, uploads documents, and submits the application. The system validates the information and provides a confirmation upon successful submission.
Cross References:	Functions: R1.1, R1.2 Use Cases: The applicant must have completed the "Create Account" use case.
Pre-Conditions	The system is online and accessible. The applicant has created an account and logged in.
Successful PostConditions	The application data is saved in the system. The applicant receives a confirmation email or notification.
Failure Post-Conditions	The application is not saved. The system displays validation errors for missing or incorrect fields.

Typical Course of Events		
Actor Action		**System Response**
1	The applicant logs into their account.	The system verifies credentials and grants access.
3	The applicant fills out the application form.	The system validates the entered data in real time.
4	The applicant uploads required documents.	The system verifies the documents for format and size compliance.
5	The applicant submits the application.	The system saves the data and sends a confirmation notification.
Alternative Course		
Step 3:		The applicant provides incomplete or invalid data. The system highlights the errors and prompts corrections.
Step 5:		The uploaded document format or size is invalid. The system rejects the file and requests a valid upload.
Step 7:		The system encounters a technical error during submission. The applicant receives an error message and is asked to retry.
Section: Track Application Status		
Actor Action		**System Response**
1	Applicant logs into the system.	System verifies credentials and grants access.
3	Applicant navigates to the status page.	System retrieves and displays the status.
Alternative Courses		

Step 3	The system denies access and prompts the applicant to reset their password.
Step 4	The system displays a “temporarily unavailable” message and logs the error for resolution.

Section: Process Payment			
Actor Action		**System Response**	
1	The applicant selects the "Pay Now" option.	2	The system redirects to the payment gateway.
3	The applicant enters payment details and confirms.	4	The gateway processes the payment and provides a success response.
5	The system updates the application status to fee paid.	6	A receipt is generated and sent to the applicant.

Alternative Courses	
Step 3	The payment gateway rejects the transaction and prompts for correction.
Step 4	The system notifies the applicant and allows retrying the transaction.
Step 5	A support ticket is generated for manual intervention to resolve the discrepancy.

Section: Review Applications		
Actor Action		**System Response**
1	Staff logs into the admin portal.	System verifies credentials.
3	Staff views the list of applications.	System retrieves and displays the data.

5	Staff reviews and updates application status.	System saves the changes.
Alternative Courses		
Step 3	The system highlights the issue and flags the application for further review.	
Step 5	The system logs the error and the staff is promoted to retry after a refresh	
Section: Schedule Tests		
Typical Course of Events		
Actor Action		**System Response**
1	Staff selects applicants for scheduling.	System retrieves eligible applicants.
3	Staff assigns test dates and times.	System updates the test schedule.
5	Notifications are sent to applicants.	Applicants receive test details.
Alternative Courses		
Step 3	The system suggests alternative dates or times based on availability.	
Step 5	The system logs the error and notifies staff for manual correction.	

Section: Send Notifications		
Actor Action		**System Response**
1	A triggering event occurs	System generates a notification message.
3	System sends the notification.	Confirmation is logged for delivery status.
Alternative Courses		
Step 3	The system skips the notification and alerts staff for updates.	
Step 4	The system retries delivery or schedules it for later.	

Non - Functional Requirements

Performance Requirements

- The system must handle up to 1,000 concurrent users without noticeable performance degradation.
- All operations, such as form submissions, payment processing, and status updates, should complete within 5 seconds.
- The database should efficiently handle large volumes of applications without impacting query times.
- A comprehensive user manual must be provided for applicants, detailing steps for account creation, application submission, and payment processing.
- Administrative documentation should include guides for managing user roles, reviewing applications, and troubleshooting common issues.

- FAQs and tutorial videos must be made available on the platform to assist users in resolving queries independently.

Safety Requirements

- The system must prevent data loss during critical operations, such as application submission or payment processing, using robust fail-safe mechanisms
- Regular backups of application data should be performed to mitigate the risk of loss due to hardware or software failure and the system must log errors for immediate resolution and notify administrators of system-level failures.

Security Requirements

- All data transmissions between users and the system must be encrypted using SSL/TLS protocols to prevent unauthorized access.
- User authentication must include secure login mechanisms, such as hashed passwords and session management.
- Role-based access control (RBAC) must ensure that users can only access features and data relevant to their roles.
- The system must detect and prevent unauthorized access attempts, including brute-force login attempts.

Reliability Requirements

- The system must maintain 99.9% uptime during the admissions cycle to support applicants and staff.
- Critical functions, such as payment processing and application submission, must operate without disruption under peak load conditions.

Usability Requirements

- The user interface must be intuitive and accessible, allowing applicants to navigate easily without prior training.
- The system must adhere to Web Content Accessibility Guidelines (WCAG) to ensure accessibility for users with disabilities.
- Real-time error messages and form validation must guide users in completing tasks without confusion.

Supportability Requirements

- system should allow for easy updates and patches without disrupting active user sessions.
- The modular architecture should enable future enhancements, such as integration with additional third-party tools.
- Detailed system logs should facilitate debugging and maintenance by the IT team.

User Documentation

- A comprehensive user manual must be provided for applicants, detailing steps for account creation, application submission, and payment processing.
- Administrative documentation should include guides for managing user roles, reviewing applications, and troubleshooting common issues.
- FAQs and tutorial videos must be made available on the platform to assist users in resolving queries independently.

6.References

1.Creatrix Campus. *Student Admission Management System*. Available at https://www.creatrixcampus.com/student-admission-management-system. This resource explains features and benefits of automated admission management systems.

2. **HEIApply**. *Admissions Software for Higher Education*. Available at https://heiapply.com/education-admissions/admissions-software-higher-education. A comprehensive guide to features, functionalities, and use cases of automated platforms for admissions.

3. **Capterra**. *Best Admissions Software Tools & Reviews*. Available at https://www.capterra.com/admissions-software/. A detailed review and comparison of various tools available for automating admission processes.

4. https://element451.com/blog/how-to-automate-student-admissions

A3. DESIGN SPECIFICATIONS

Any standard template may be used, as per project need approved by Project Coordinator & Supervisor. Following is a suggestive outline.

1 Introduction
1.1 Purpose of Document
1.2 Intended Audience
1.3 Project Overview
1.4 Scope
2 Design Considerations
2.1 Assumptions and Dependencies
2.2 Risks and Volatile Areas
3 System Architecture
3.1 System Level Architecture
3.2 Software Architecture
4 Design Strategy
5 Detailed System Design
5.1 Database Design
5.1.1 ER Diagram
5.1.2 Data Dictionary
5.1.2.2
Data 2
5.1.2.3 Data n
5.2 Application Design
5.2.1 Sequence Diagram
5.2.2 State Diagram

1. Introduction

This project **"Automated Platform for Undergraduate Admissions"** aims to streamline the admission process, replacing manual operations with an automated, secure, and efficient system. The system's primary focus is reliability, flexibility, integrity, and security, ensuring a seamless experience for both applicants and administrative staff. Scrum methodology is used, allowing iterative development and continuous stakeholder feedback.

1.1 Purpose of Document

The purpose of this document is to provide a detailed overview of the project, including its objectives, methodology, scope, and intended outcomes. It is designed for stakeholders, including developers, project supervisors, and end-users such as admissions teams. The document employs object-oriented design methodology to ensure scalability and modularity in system development.

1.2 Intended Audience

This document is intended for:

1- Project Supervisors – To oversee and guide project progress.
2- Development Team – For implementing and adhering to the design and technical requirements.
3- Admissions Staff – As end-users who will operate the automated system.
4- Applicants – Indirect stakeholders benefiting from improved processes.

1.3 Document Convention

Font Style: Arial

Font Size: 12pt for body text; headings may use larger sizes for emphasis.

1.4 Project Overview

The proposed software system automates undergraduate admissions at Hamdard

University. Its functionalities include:

Application Submission and Tracking: Applicants can apply online and track their progress.

Automated Payment Verification: Ensures timely and error-free fee verification.

Test Scheduling and Results: Manages scheduling for admission tests and sends automated notifications.

Transparency: Provides real-time updates to applicants regarding their status. The system leverages modern technologies like ReactJS, NodeJS, and MongoDB for development and follows Scrum methodology to ensure adaptability and timely delivery.

1.5 Scope

The system will:

1. Automate Data Entry: Eliminate duplicate entries with unique ID verification.
2. Payment Processing: Streamline admission fee verification securely.
3. Test Assignment: Generate credentials for online tests post-payment verification.
4. Admission Confirmation: Automate test result evaluation and offer letters.

Out of Scope:

- Management of programs beyond undergraduate admissions.

Integration with non-academic modules like financial aid.

2 Design Considerations

This section outlines the foundational elements necessary to create a comprehensive and efficient design for the "Automated Platform for Undergraduate Admissions." It addresses the assumptions, dependencies, and potential risks, ensuring the design remains adaptable to future changes.

2.1 Assumptions and Dependencies

The system design relies on the following assumptions and dependencies, which influence its architecture and functionality:

1. **Stable Internet Access**: Reliable internet connectivity is assumed for both applicants and the administrative staff to access the online platform.
2. **Software Environment**: The platform will be hosted on a scalable cloud environment (e.g., Firebase or MongoDB) to manage user load and ensure high availability.
3. **Technology Stack Compatibility**: Tools like ReactJS for frontend, NodeJS for backend, and MongoDB for database are assumed to integrate seamlessly without major technical issues.
4. **User Training**: Administrative staff are assumed to have basic technical skills or will undergo training to use the system effectively.
5. **Compliance Standards**: The system will adhere to relevant regulatory and institutional policies to ensure data security and integrity.

2.2 Risks and Volatile Areas

Several factors pose risks to the system design, and addressing these areas is crucial for a robust and flexible architecture:

-New Requirements: Changes in university admission policies or additional features requested by stakeholders can impact the design.

Mitigation: Incorporate a modular design approach with loosely coupled components, allowing easy updates or integration of new features.

-Technology Evolution: The rapid pace of technological change may render some selected tools or frameworks obsolete.

Mitigation: Choose widely supported, open-source technologies with active communities to ensure long-term viability. Also, maintain backward compatibility during upgrades.-Scalability Challenges: As the number of applicants increases, the system may face performance bottlenecks.

Mitigation: Use cloud-based solutions with dynamic scaling capabilities and optimize database queries to handle high user loads efficiently.

-Data Security and Privacy: Handling sensitive applicant data introduces risks of breaches or compliance violations.

Mitigation: Implement robust encryption for data at rest and in transit, along with secure authentication mechanisms like multi-factor authentication.

-Dependency on External Systems: Integration with payment gateways or external test scheduling platforms could face interruptions.

Mitigation: Design the system with fallback mechanisms, such as manual verification workflows, to handle external system failures.

-Team Collaboration and Resource Management: Miscommunication or delays in development phases could lead to inefficiencies.

Mitigation: Utilize the Scrum methodology with daily stand-ups, sprint reviews, and retrospectives to ensure continuous progress and issue resolution.

-System Adoption: Users may resist switching from manual processes to an automated system. Mitigation: Conduct user training sessions and provide a comprehensive user manual to ease the transition

3 System Architecture

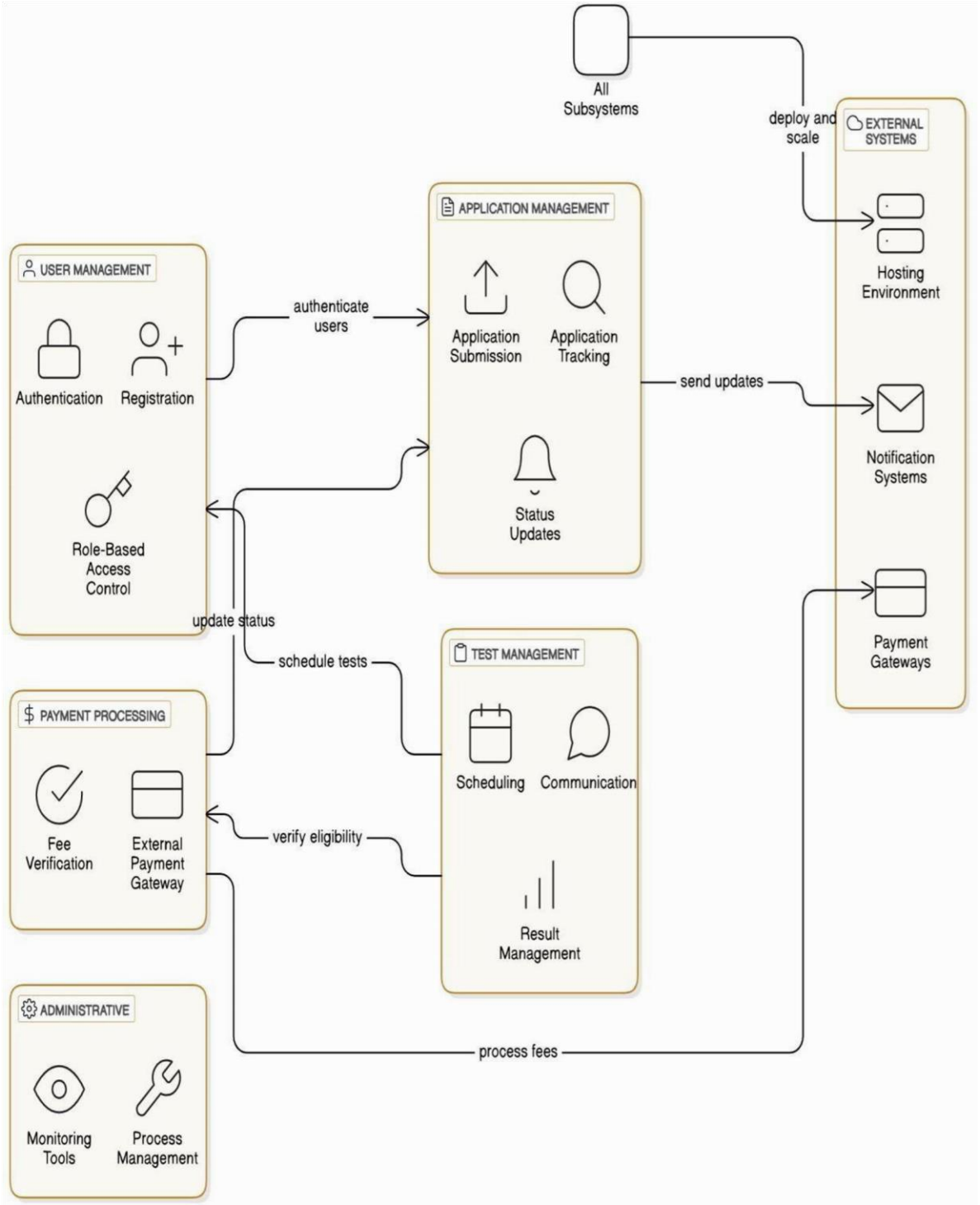
3.1 System Level Architecture

The system is decomposed into several subsystems to ensure modularity, scalability, and maintainability. These subsystems and their roles are described below: **System Decomposition**

- **User Management Subsystem**: Handles user authentication, registration, and rolebased access control.
- **Application Management Subsystem**: Facilitates application submission, tracking, and status updates.
- **Payment Processing Subsystem**: Manages fee verification and integrates with external payment gateways.
- **Test Management Subsystem**: Automates scheduling, communication, and result management for admission tests.
- **Administrative Subsystem**: Provides tools for the admissions team to monitor and manage processes.

Relationships Between Elements

- **User Management interacts with Application Management** to ensure only authenticated users can submit or modify applications.
- **Payment Processing communicates with Application Management** to update the application status upon fee verification.
- **Test Management integrates with User Management and Payment**
- **Processing** to schedule tests only for eligible applicants. **Interfaces to External Systems**
- **Payment Gateways**: Secure API integration for processing application fees.
- **Notification Systems**: Integration with email and SMS services for sending updates to applicants.
- **Hosting Environment**: Cloud-based infrastructure (e.g., Firebase or AWS) to deploy the application and manage scalability. **Global Design Strategies**
- **Error Handling**: Centralized logging and error-handling mechanisms to track and resolve issues across subsystems.
- **Security**: Role-based access control, data encryption, and compliance with data protection regulations.


All Subsystems
deploy and scale
EXTERNAL SYSTEMS
Hosting Environment
Notification Systems
Payment Gateways
APPLICATION MANAGEMENT
Application Submission
Application Tracking
Status Updates
send updates
USER MANAGEMENT
Authentication
Registration
Role-Based Access Control
authenticate users
update status
schedule tests
TEST MANAGEMENT
Scheduling
Communication
Result Management
verify eligibility
PAYMENT PROCESSING
Fee Verification
External Payment Gateway
process fees
ADMINISTRATIVE
Monitoring Tools
Process Management

3.2Software Architecture

The software architecture follows a layered architecture to ensure separation of concerns and scalability. It includes the following layers:

Layers

User Interface Layer:

- Provides the front-end experience for users (applicants and administrators). ○ Developed using modern frameworks like ReactJS or NextJS.
- Handles user input, application status viewing, and notifications.

Middle Tier:

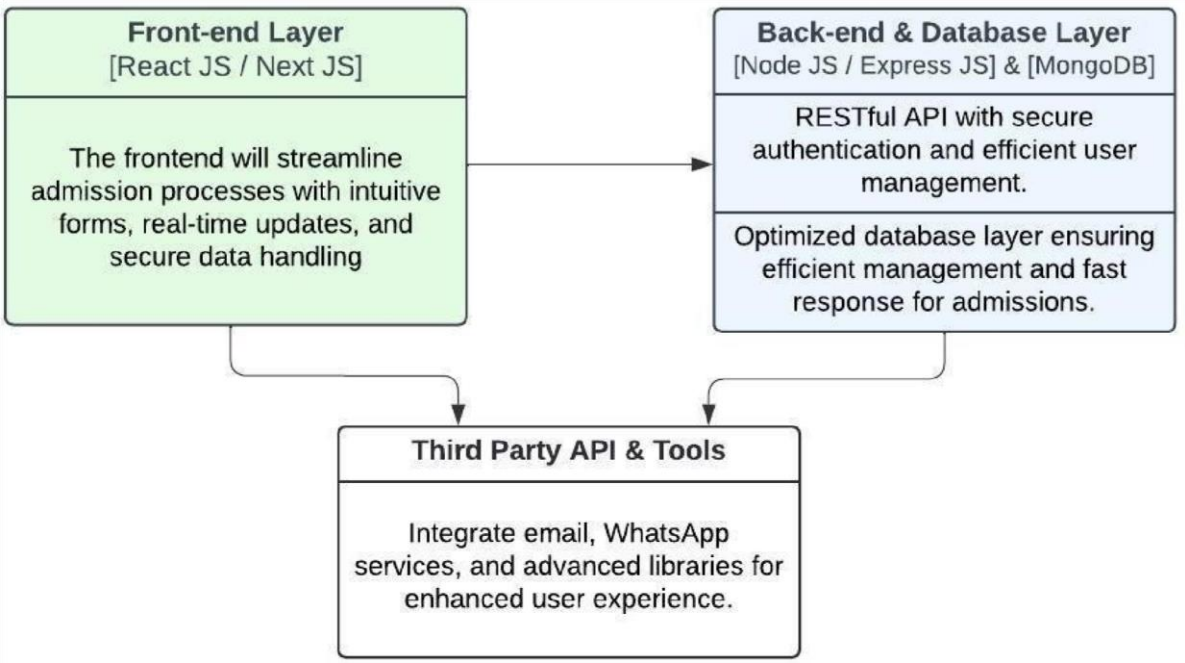
- Manages business logic and acts as a bridge between the user interface and the database.
- Built using NodeJS/ExpressJS for server-side processing. ○ Implements APIs for user authentication, data submission, and updates.

Data Access Layer:

- Handles interactions with the database. ○ Ensures secure and optimized queries for operations like application submission, fee verification, and test scheduling.
- Uses MongoDB for scalability and flexibility in handling structured and unstructured data.

Interactions Between Layers

- User Interface Layer interacts with Middle Tier via REST APIs to submit data and retrieve status updates.
- Middle Tier communicates with the Data Access Layer for CRUD operations on the database.
- Middle Tier integrates with external systems (e.g., payment gateways, notification services) through APIs.



4. Design Strategy

The design strategy for the "Automated Platform for Undergraduate Admissions" focuses on achieving scalability, modularity, security, and user-centric functionality. The strategy emphasizes creating a robust system architecture that supports current needs while allowing for future enhancements.

Future System Extension or Enhancement

- Modular Design: The system is decomposed into independent modules, such as User Management, Application Management, and Payment Processing, enabling easy addition or modification of features.
- Scalability: Cloud-based technologies (e.g., Firebase, MongoDB) ensure the system can handle increased workloads as the number of users and applications grows.
- Integration Capability: The architecture supports seamless integration with external systems, such as new payment gateways or notification services.

System Reuse

- Reusable Components: Core functionalities like user authentication, data validation, and API integrations are developed as reusable components, minimizing development effort for similar projects.
- Shared Libraries: The use of widely adopted frameworks (ReactJS, NodeJS) promotes reusability across different teams and applications.
- Database Models: Standardized database schemas ensure compatibility for future projects requiring similar data handling.

User Interface Paradigms

- Responsive Design: The system employs a responsive web design approach, ensuring compatibility across devices (desktops, tablets, and smartphones).
- User-Centric Interface: Built with modern frameworks like ReactJS, the interface provides an intuitive, user-friendly experience for both applicants and administrators.
- Role-Based Views: Different user roles (e.g., applicant, administrator) are assigned customized dashboards, displaying only relevant information and functionalities. **Data Management**

- Centralized Database: MongoDB serves as the primary database for storing application data, ensuring consistency and easy management.
- Data Security: Sensitive data is encrypted both at rest and in transit, adhering to compliance standards for data protection.
- Scalable Storage: The use of a NoSQL database (MongoDB) allows for dynamic scaling as the volume of data increases, supporting efficient query performance.
- Backup and Recovery: Regular data backups and recovery protocols are implemented to prevent data loss and ensure business continuity.

Concurrency and Synchronization

- Asynchronous Operations: APIs and database queries are designed to handle asynchronous operations, reducing system bottlenecks and improving performance.
- Queue Management: Task queues (e.g., for payment verification or test scheduling) are implemented to handle high-traffic scenarios efficiently.
- Concurrency Control: Optimistic concurrency control mechanisms are used to prevent data conflicts when multiple users interact with the system simultaneously.
- Scalable Resources: Load balancing is employed to manage concurrent user requests, ensuring consistent performance during peak usage.

Reasoning and Trade-Offs

- **Reasoning**: The system design reflects the project goals of accuracy, scalability, and user satisfaction. A modular architecture allows the system to evolve over time, while a layered approach ensures separation of concerns.

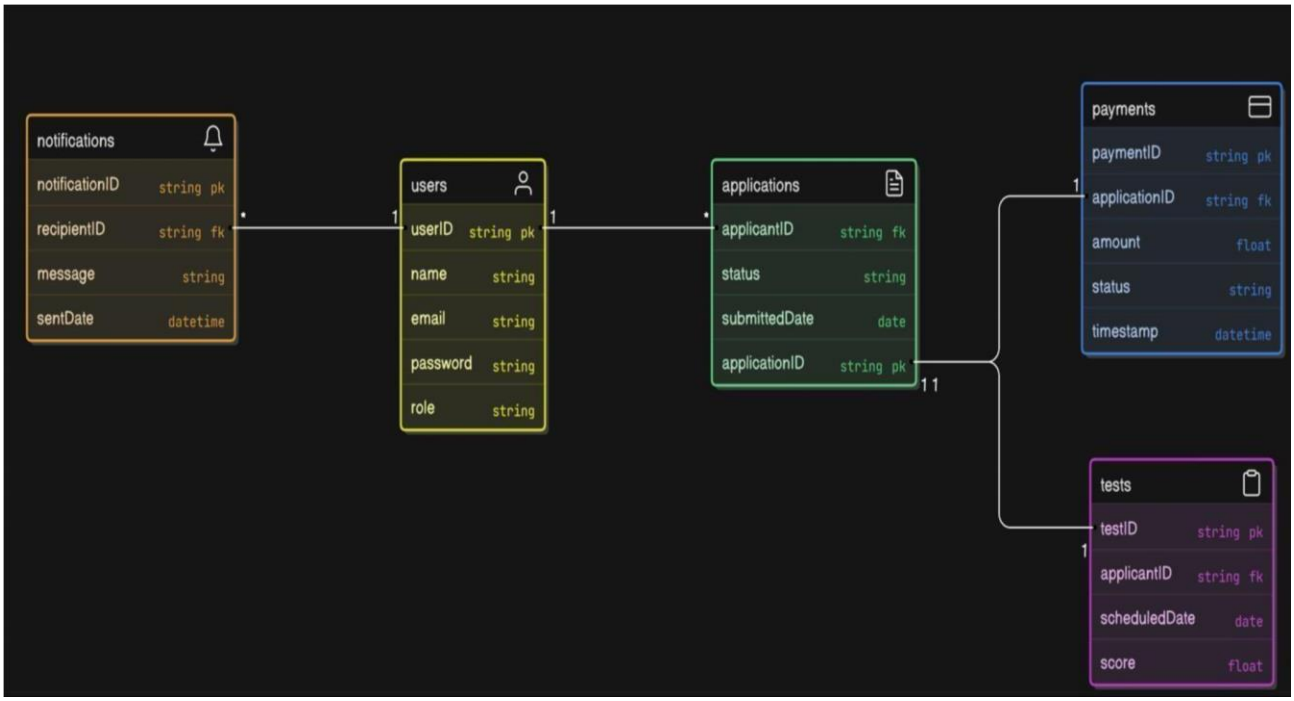
- **Trade-Offs**:

  - Using modern technologies like ReactJS and MongoDB prioritizes scalability and development speed but may require additional training for new developers.

  - Cloud hosting increases flexibility and performance but involves recurring costs compared to on-premises solutions.

6.1 Database Design

6.1.1 ER Diagram



4.1.2 Data Dictionary

4.1.2.1 Data 1

USER						
Name	User					
Alias	-					
Whereused/howused	Used by Login, Registration, Application Submission, Notification, Admin functions. Role determines access rights.					
Content description	Represents a person in the system who can either be an applicant or an admin. Stores personal information and credentials.					
Column Name	**Description**	**Type**	**Length**	**Nullable**	**Default Value**	**Key Type**

userID	Unique identifier for the user	Integer	10	No	None	PK
name	Full name of the user	Varchar	255	No	None	
email	User's email address for notifications.	Varchar	255	No	None	
password	Encrypted password for the user.	Varchar	255	No	None	
role	Role of the user (e.g., Applicant, Admin).	Varchar	50	No	None	

4.1.2.2 Data 2

Application	
Name	Application
Alias	App
Where-used/howused	Used by User for submission, Admin for status updates. Stored in the system after submission.
Content description	Represents a user's application for a program, containing status, submission date, and applicant information.

Column Name	**Description**	**Type**	**Length**	**Null**	**Default Value**	**Key Type**
applicationID	Unique identifier for	Integer	10	No	None	PK

	the application.					
applicantID	Reference to the user applying.	Integer	10	No	None	PK
status	Current status of theapplication (e.g., Pending).	Varchar	50	No	None	
Submitted Date	Date of the application submission.	Date	-	No	None	

4.1.2.3 Data 3

Payment						
Name	Payment					
Alias	-					
Whereused/howused	Used by Application Process to verify payments and handle refunds.					
Content description	Represents the payment made for an application, including the amount, payment status, and timestamp of the transaction.					
Column Name	**Description**	**Type**	**Length**	**Nullable**	**Default Value**	**Key Type**

paymentID	Unique identifier for the payment.	Integer	10	No	None	PK
applicationID	Reference to the linked application for payment.	Integer	10	No	None	PK
amount	Payment amount.	Decimal	10,2	No	None	
status	Payment status (e.g., Paid, Pending).	Varchar	50	No	None	
timestamp	Timestamp of the payment transaction	Timestamp	-	No	None	

4.1.2.4 Data 4

Test	
Name	Test
Alias	-
Where-used/howused	Used to schedule and record results for applicants. Linked with applications.
Content description	Represents a test scheduled for an applicant, including date, score, and test results.

Column Name	Description	Type	Length	Null able	Default Value	Key Type
testID	Unique identifier for the test.	Integer	10	No	None	PK
applicantID	Reference to the user taking the test.	Integer	10	No	None	FK
scheduledDate	Date of the test.	Date	-	No	None	
score	Test score for the applicant.	Decimal	5,2	Yes	NULL	

4.1.2.5 Data 5

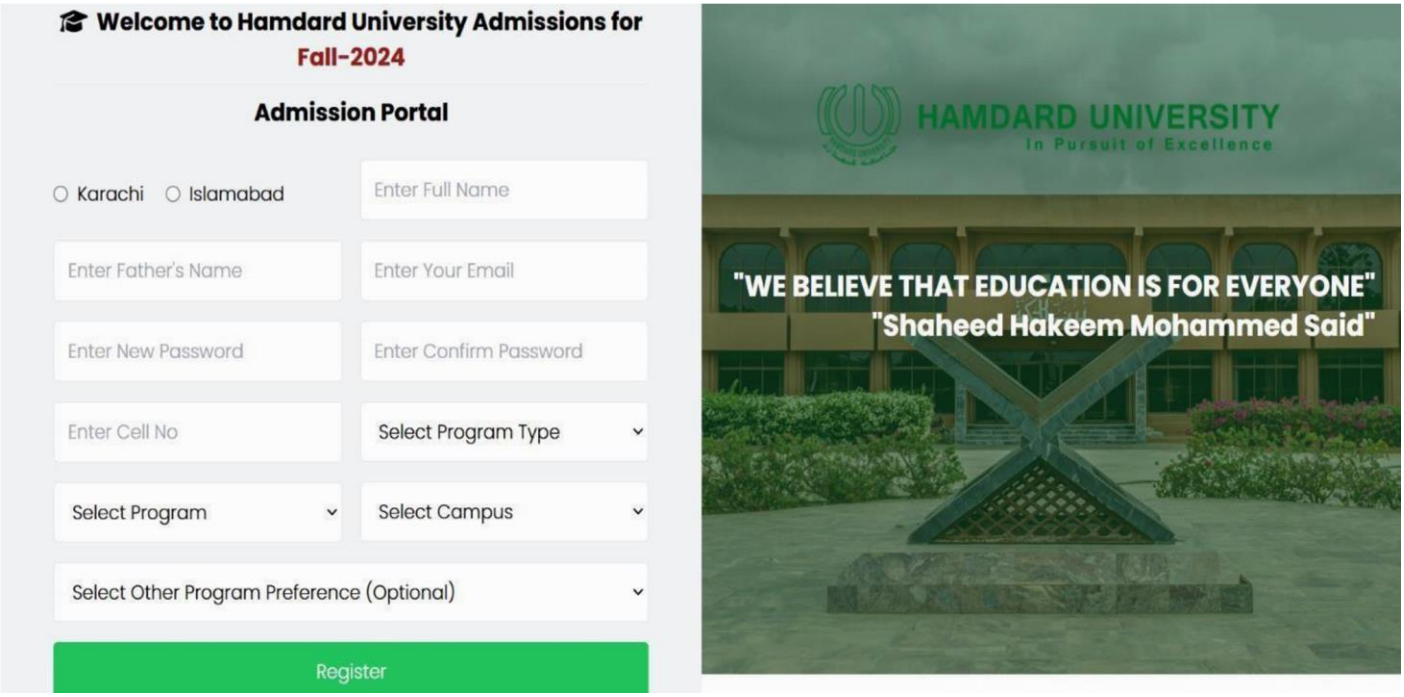
Notification	
Name	Notification
Alias	-
Where-used/howused	Used to send updates to users about application status, tests, payments, etc.
Content description	Represents a notification sent to a user, either by email or SMS, containing important updates.

Column Name	Description	Type	Length	Null able	Default Value	Key Type
notificationID	Unique identifier for the	Integer	10	No	None	PK
	notification.					

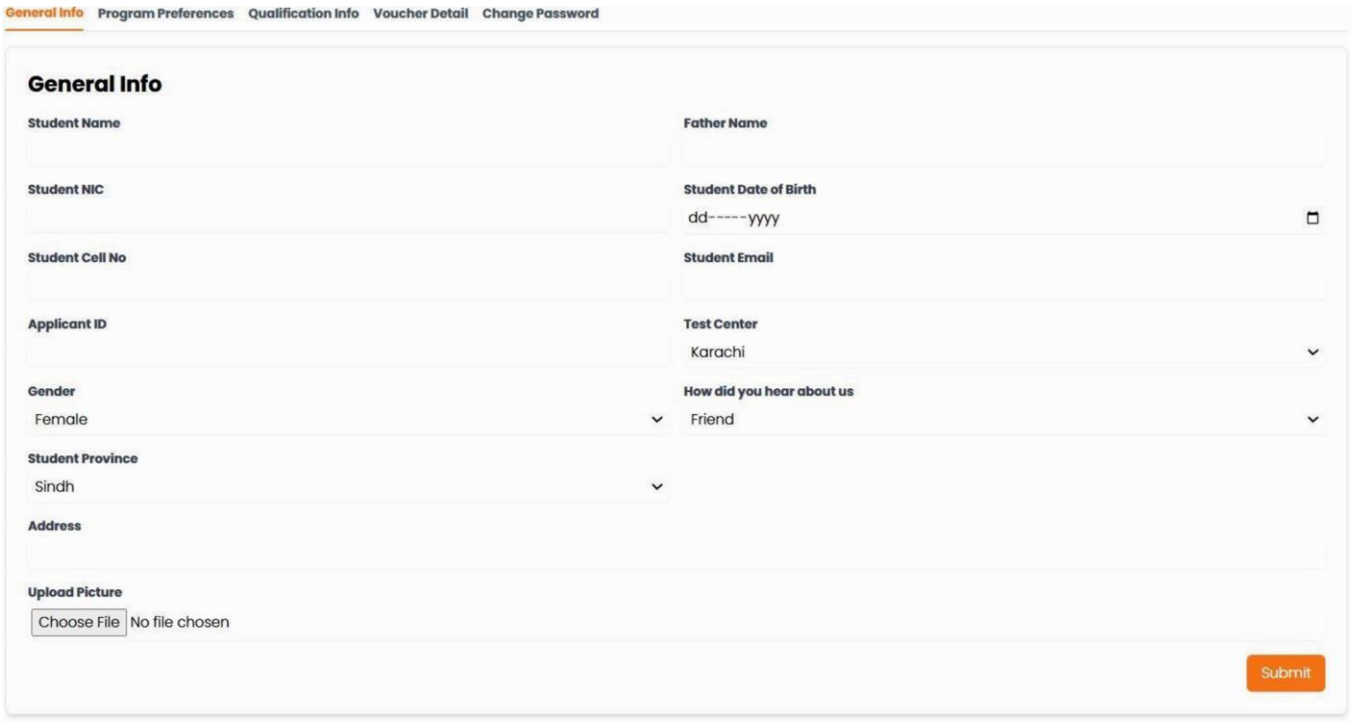
recipientID	Reference to the user receiving the notification.	Integer	10	No	None	FK
message	Content of the notification.	Text	-	No	None	
sentDate	Date and time the notification was sent.	Timestamp	-	No	None	

4.2 GUI Design

4.2.1



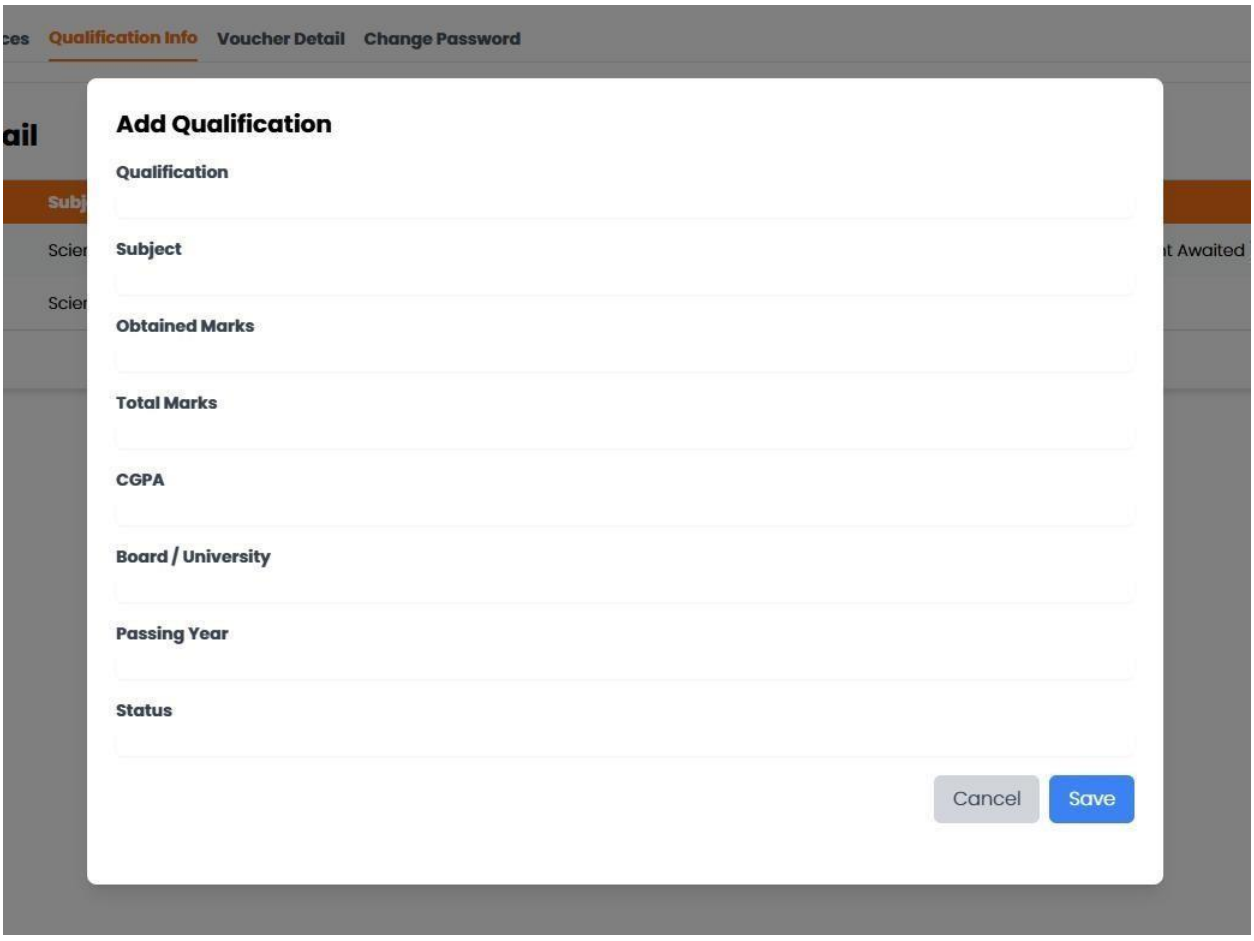
4.2.2



4.2.3


General Info
Program Preferences
Qualification Info
Voucher Detail
Change Password
Program Preferences
Campus
North Nazimabad KDA Campus
Faculty
Faculty of Engineering
1st Preference (Value Required)
BS (RIS)
2nd Preference
BS (DFCS)
3rd Preference
BS (RIS)
4th Preference
MSCS (SE)
5th Preference
MSCS
Submit

4.2.4


Qualification Info
Voucher Detail
Change Password
Add Qualification
Qualification
Subject
Obtained Marks
Total Marks
CGPA
Board / University
Passing Year
Status
Cancel
Save

4.2.5


General Info
Program Preferences
Qualification Info
Voucher Detail
Change Password
Qualification Detail
Add Qualification
Qualification
Subject
Obtained Marks
Total Marks
CGPA
Board / University
Passing Year
Status
Actions
Intermediate/Equivalent
Science
489
550
-
Hyderabad board
2024
Result/Document Awaited
Edit
Delete
MATRIC
Science
493
550
-
Hyderabad board
2021
Passed
Edit
Delete

4.2.6

4.2.7

4.2.8

App ID	Inquiry Date	Student Name	Status	Project	Faculty	Program	Offer Type	Cell No	Edit	Offer Letter Amount
129175	09-Dec-2024	Safia Ahmed	No Voucher	NNKDA	FEST	BS (RIS)	Spring-2025	03312321838	Edit	52,500
129162	08-Dec-2024	Rehman	Unpaid Voucher	NNKDA	FEST	BS (DFCS)	Spring-2025	03150212162	Edit	52,500
129162	08-Dec-2024	Abdul	Unpaid Voucher	NNKDA	FEST	BS (DFCS)	Spring-2025	03150212162	Edit	52,500
129162	08-Dec-2024	Zoya Saeed	Unpaid Voucher	NNKDA	FEST	BS (DFCS)	Spring-2025	03150212162	Edit	52,500
129162	08-Dec-2024	Ahmed	Unpaid Voucher	NNKDA	FEST	BS (DFCS)	Spring-2025	03150212162	Edit	52,500
129162	08-Dec-2024	Dua Rahim	Unpaid Voucher	NNKDA	FEST	BS (DFCS)	Spring-2025	03150212162	Edit	52,500

4.2.9

Offer Letter	Father Name	Student Email	Verification Code	Last Updated	Student Province	Print	Reset Password
First Enter Offer Letter Amount then Print Offer Letter	Nazim Ahmed	safia.a2005@gmail.com	9HRC	2024-12-09 (00:00)	6	Print	Reset Password
First Enter Offer Letter Amount then Print Offer Letter	Muhammad Saleem	firefoxbpo@gmail.com	MEU7	2024-12-08 (00:00)	6	Print	Reset Password
First Enter Offer Letter Amount then Print Offer Letter	Syed Shahid Hussain	ishfatabool2468@gmail.com	5KL4	2024-12-08 (00:00)	6	Print	Reset Password

References

1. https://www.creatrixcampus.com/student-admission-management-system
2. https://heiapply.com/education-admissions/admissions-software-highereducation 3. https://scrumguides.org
4. https://lucidchart.com/blog/sequence-diagram-tutorial
5. https://lucidchart.com/blog/sequence-diagram-tutorial

Appendices

Appendix A – User Roles and Permissions Matrix

Role	Permissions
Applicant	Register, Submit Application, Track Status, Make Payments, Receive Updates
Admin	Review Applications, Update Status, Schedule Tests, Generate Reports
Super admin	Full control over user management, system settings, and auditing functions

Endpoint	Method	Description
/api/register	POST	Register a new user
/api/login	POST	Authenticates user credentials
/api/applications	GET	Retrieves user's application details
/api/payment/verify	POST	Verifies payment made via gateway
/api/tests/schedule	POST	Schedules test for eligible applicants

Appendix B – Technologies Used

Technology	Purpose
ReactJS	Frontend development
NodeJS+Express	Backend services and API creation

MongoDB	Database management
Firebase Hosting	Web hosting and deployment
GitHub	Version control and collaboration

Appendix C – Sample Test Notification

To: applicant123@gmail.com

Subject: Admission Test Schedule Notification **Message:**

Dear Applicant,

Your admission test has been scheduled for **March 15, 2025** at **10:00 AM**. Please log in to your portal to access further instructions.

Thank you.

Hamdard University Admissions Team

A4. OTHER TECHNICAL DETAIL DOCUMENTS

Test Cases Document

Test plan:

S. No	Module Description	Test Engineer	Start Date	End Date
1	User Registration & Login	Zoya Sayeed	20-June-2025	21-June-2025
2	Profile Management	Dua Rahim	21-June-2025	22-June-2025
3	Application Submission	Dua Rahim	23-June-2025	24-June-2025
4	Document Upload	Zoya Sayeed	24-June-2025	25-June-2025
5	Payment Gateway Integration	Zoya Sayeed	25-June-2025	26-June-2025
6	Test Scheduling System	Dua Rahim	27-June-2025	28-June-2025
7	Admin Dashboard	Zoya Sayeed	29-June-2025	30-June-2025
8	Real-Time Notifications	Dua Rahim	31-June-2025	01-July-2025
9	Security and Compliance	Zoya Sayeed	01-July-2025	02-July-2025
10	Final Testing & Bug Fixing	All Members	02-July-2025	02-July-2025

Test cases:

Project Name: APFUA (Automated Platform for Undergraduate Admissions)
Test Engineer: Zoya Sayeed
Date: 20-06-2025
Test Scenario: User Registration & Login
Test Case Description: To verify the working of login and registration modules under different user roles securely and correctly.

Test Cases for User Registration & Login

TC ID	Steps	Input	Expected Result	Actual Result	Pass/Fail
TC-01	Register as student	Valid student info	Account created & confirmation email sent	As expected	Pass
TC-02	Register with existing email	Duplicate email	Error: Email already registered	As expected	Pass
TC-03	Register with weak password	Simple password	Error: Weak password warning	As expected	Pass
TC-04	Login as admin	Valid admin credentials	Redirect to Admin Dashboard	As expected	Pass
TC-05	Login with invalid password	Valid email, wrong pass	Error: Incorrect credentials	As expected	Pass
TC-06	Submit empty form	Blank fields	Error: Fields cannot be empty	As expected	Pass
TC-07	Session timeout on inactivity	Logged-in session	User logged out after 10 mins idle	As expected	Pass
TC-08	Remember me functionality	Checkbox enabled	User stays logged in	As expected	Pass

Test Engineer: Dua Rahim

Date: 21-06-2025

Test Scenario: Profile Management

Test Case Description: To validate if users can view, edit, and manage their personal and academic information.

Test Cases for Profile Management

TC ID	Steps	Input	Expected Result	Actual Result	Pass/Fail
TC-09	View profile	Logged-in student	All fields populated	As expected	Pass
TC-10	Edit phone number	New phone	Profile updated successfully	As expected	Pass
TC-11	Edit academic info	New marks entered	Saved successfully	As expected	Pass
TC-12	Upload profile picture	Valid image file	Image preview shown	As expected	Pass
TC-13	Remove uploaded photo	Click remove	Photo removed	As expected	Pass

Test Engineer: Dua Rahim
Date: 21-06-2025
Test Scenario: Application Submission
Test Case Description: To verify the proper working of admission form submission, ensuring validations and unique entry.

Test Cases for Application Submission

TC ID	Steps	Input	Expected Result	Actual Result	Pass/Fail
TC-14	Submit complete form	Valid academic/personal info	Application ID generated	As expected	Pass
TC-15	Leave academic details blank	Personal info only	Error: Incomplete form	As expected	Pass
TC-16	Duplicate submission attempt	Same user resubmits	Error: Already submitted	As expected	Pass
TC-17	Add optional field	Optional scholarship applied	Field accepted	As expected	Pass
TC-18	Submit form with symbols	Name field with symbols	Error: Invalid characters	As expected	Pass
TC-19	Back button after submission	Press browser back	Form resubmission prevented	As expected	Pass

Test Engineer: Zoya Sayeed
Date:24-06-2025
Test Scenario: Document Upload
Test Case Description: To verify upload, format validation, and removal functionality for supporting documents.

Test Cases for Document Upload

TC ID	Steps	Input	Expected Result	Actual Result	Pass/Fail
TC-20	Upload Matric PDF	Valid file	Success message	As expected	Pass
TC-21	Upload oversized file	>5MB	Error: File too large	As expected	Pass
TC-22	Upload unsupported format	.exe file	Error: Invalid file type	As expected	Pass
TC-23	Remove uploaded file	Click delete	File removed from system	As expected	Pass
TC-24	Upload multiple documents	Multiple files	All files uploaded successfully	As expected	Pass

Test Engineer: Zoya Saeed

Date: 25-06-2025

Test Scenario: Payment Gateway Integration

Test Case Description: To ensure secure payment transactions.

Test Cases for Payment Gateway Integration

TC ID	Steps	Input	Expected Result	Actual Result	Pass/Fail
TC-25	Pay using Visa card	Valid credentials	Payment success	As expected	Pass
TC-26	Expired card	Invalid expiry	Error: Payment declined	As expected	Pass
TC-27	Network failure	Disconnect midway	Error: Payment not completed	As expected	Pass
TC-28	Wrong CVV	Invalid CVV	Error: Payment authentication failed	As expected	Pass

Test Engineer: Dua Rahim

Date: 27-06-2025

Test Scenario: Test Scheduling System

Test Case Description: To validate entry test slot availability, booking, and rebooking logic.

Test Cases for Test Scheduling System

TC ID	Steps	Input	Expected Result	Actual Result	Pass/Fail
TC-29	View available slots	Logged-in student	Date/time slots displayed	As expected	Pass
TC-30	Book slot	Select slot	Confirmation message displayed	As expected	Pass
TC-31	Book full slot	Fully booked slot	Error: Slot not available	As expected	Pass
TC-32	Reschedule test	Select new date	Previous slot cleared	As expected	Pass

Test Engineer: Zoya Sayeed

Date: 29-06-2025

Test Scenario: Admin Dashboard

Test Case Description: To test application approval, merit generation, and student management actions.

Test Cases for Admin Dashboard

TC ID	Steps	Input	Expected Result	Actual Result	Pass/Fail
TC-33	Approve application	Student ID	Status updated to "Approved"	As expected	Pass
TC-34	Reject application	Student ID	Status updated to "Rejected"	As expected	Pass
TC-35	Generate merit list	Approved list	Merit list displayed	As expected	Pass
TC-36	Search student by ID	Student ID input	Student profile shown	As expected	Pass
TC-37	Delete a pending application	Click delete	Application removed	As expected	Pass

Test Engineer: Dua Rahim

Date: 31-06-2025

Test Scenario: Real-Time Notifications

Test Case Description: To validate that key actions trigger accurate student notifications.

Test Cases for Real-Time Notifications

TC ID	Steps	Input	Expected Result	Actual Result	Pass/Fail
TC-38	Application approved	Student approved	SMS/Email sent	As expected	Pass
TC-39	Payment done	Transaction ID	Confirmation email sent	As expected	Pass
TC-40	Test date scheduled	Slot selected	Notification sent	As expected	Pass

Test Engineer: Zoya Sayeed

Date: 1-07-2025

Test Scenario: Security and Compliance

Test Case Description: To verify that the system protects against common attacks and enforces access restrictions. **Test Cases for Security and Compliance**

TC ID	Steps	Input	Expected Result	Actual Result	Pass/Fail
TC-41	SQL Injection in login	' OR '1'='1	Input rejected	As expected	Pass
TC-42	Access admin as student	Student login	Access denied	As expected	Pass
TC-43	Access teacher panel as student	URL tampering	Access blocked	As expected	Pass
TC-44	XSS in form field	<script> tag	Input sanitized	As expected	Pass
TC-45	CSRF token missing	Invalid request	Form rejected	As expected	Pass
TC-46	Session hijack attempt	Stolen cookie	Session invalidated	As expected	Pass
TC-47	Brute force login	Rapid login tries	Account locked after 5 attempts	As expected	Pass
TC-48	File upload validation	.php file	Upload denied	As expected	Pass
TC-49	Password stored as plain text	DB check	Not visible/hashed	As expected	Pass
TC-50	Open redirect test	?redirect=evil.com	Redirect blocked	As expected	Pass

UI/UX Detail Document

Coding Standards Document

Automated Platform for Undergraduate Admissions (APFUA)

1. Introduction

This document outlines the coding conventions and best practices followed in the APFUA project to ensure readability, consistency, scalability, and maintainability of the codebase. A uniform coding standard is vital for teamwork and reduces time spent on debugging, understanding, and reviewing each other's code.

2. Technology Stack

- Frontend: HTML5, TailwindCSS, JavaScript (ES6+), ReactJS/Next.js
- Backend: Node.js with Express.js
- Database: MongoDB (Mongoose ORM)
- Authentication: JWT, bcryptjs
- Version Control:Git + GitHub
- Testing Tools: Manual testing

3. General Guidelines

- All code must be written in English with consistent case styling (camelCase for variables and functions, PascalCase for classes and components).
- Consistent indentation (2 or 4 spaces) must be applied across the codebase.
- Use meaningful and self-explanatory names for variables, functions, files, and folders.
- Avoid deeply nested logic; use helper functions or restructure control flow.
- All source files should include a header comment describing the file's purpose.

4. Commit Message Format

- Use conventional commit standards to maintain a readable version history:

<type>(<scope>): <subject>

Example:

feat(auth): implemented JWT-based login.

5. Frontend Coding Standards

- Only functional components should be used in React.
- All reusable UI logic should be abstracted into custom hooks.
- TailwindCSS must be used for styling; CSS files are restricted to global styles only.
- Components should follow this directory convention:

/components

/pages

/services
/assets

/contexts

- Form inputs must include built-in HTML validation and display user-friendly error messages.
- Use `axios` for API calls and store all URLs in a separate configuration file.

6. Backend Coding Standards

- API routes must be modularized using Express Router.
- All API endpoints must include request validation, error handling, and logging.
- Data must be validated using `Joi` or `express-validator`.
- Passwords must always be hashed using `bcryptjs`.
- Responses should follow a standard format: { success: true, data: {}, message:

"Operation successful."

}

- Organize server structure using MVC principles where applicable.

7. Security Practices

- Escape all user input to avoid injection attacks.
- Use Helmet to protect against known web vulnerabilities.
- Enable CORS with restricted domains.
- Environment variables (`.env`) must never be pushed to version control.
- Use HTTPS for secure data transmission.

8. Code Review Checklist

[ ] Variable and function names are descriptive.

[ ] Components and functions are reusable.

[ ] All inputs are validated and sanitized.

[ ] No hardcoded values or credentials.

[ ] No console logs in production.

[ ] Functions are modular and not overly long.

Project Policy Document

Automated Platform for Undergraduate Admissions (APFUA)

1. Purpose

This document defines the official policies to be followed during the development, testing, deployment, and maintenance of the APFUA project. It ensures that all team members are aligned with the institutional guidelines, ethical standards, and quality assurance measures.

2. Development Lifecycle Policy

- Methodology: Agile (SCRUM-based)
- Sprint Planning, Backlog Grooming, Daily Stand-ups, Sprint Review, and Retrospectives are conducted.
- Deliverables are reviewed at the end of each 2-week sprint.
- Development tasks are tracked via GitHub issues and Kanban boards.

3. Source Code Management Policy

- All development must occur on separate feature branches.
- Every feature must go through a pull request (PR) and peer review process.
- Main branch must always remain deployable and protected by CI.
- Use descriptive PR titles and link them to relevant issue numbers.

4. Documentation Policy

- All features must be accompanied by adequate documentation.
- Each module must include an overview, input/output specifications, and known limitations.
- Markdown format is preferred for all internal documentation.
- Final documents to be submitted include:
- Software Requirements Specification (SRS)
- Software Design Document (SDD)
- Extended Test Plan and Report
- Developer Documentation
- User Manual

5. Change Management Policy

- All new features or changes must go through approval from the Project Supervisor.
- Urgent bugs should be logged under “Hotfix” with priority status.
- No changes to production are allowed without UAT confirmation and supervisor sign-off.

6. Testing & Quality Assurance Policy

- Every major module must be manually tested against defined test cases.
- Peer reviews of test cases are mandatory before execution.
- Bugs must be logged using GitHub Issues.
- Final testing involves security testing, compliance checks, and performance assessment.

7. Data Privacy & Security Policy

- Applicant data is considered confidential.
- All personal identifiers (email, CNIC, address) must be encrypted at rest.
- Role-based access is strictly enforced.
- Developers must adhere to GDPR and local data protection guidelines.

8. Deployment & Release Policy

- Deployments are carried out only after supervisor approval.
- Firebase is used for production hosting with CI/CD scripts.
- Backups and release logs are maintained in a shared institutional drive.
- A rollback plan must be documented for every release.

User Manual Document

Automated Platform for Undergraduate Admissions (APFUA)

Target Users: Students, Administrators, Support Staff

1. Overview

The APFUA system simplifies the entire university admissions process. This manual guides students and staff in navigating the platform efficiently and securely. It covers user registration, form submission, document upload, test scheduling, and administrative features.

2. System Requirements

- Supported Browsers: Chrome, Firefox, Safari, Edge (latest versions)
- Device Compatibility: Desktop, Tablet, and Mobile
- Internet: Minimum 2 Mbps connection recommended

3. Student User Guide

3.1 Account Registration

- Open the platform URL and click on "Register".
- Provide details like Name, Email, CNIC, Password.
- Confirm via email link to activate your account.

3.2 Login & Dashboard

- Enter login credentials on the homepage.
- Dashboard displays status of application, tests, and payments.

3.3 Filling the Application Form

- Navigate to "Application" section.
- Enter personal, academic, and contact details.
- Choose desired degree programs and upload necessary documents.

3.4 Document Upload

- Accepted formats: PDF, JPEG, PNG
- File size limit: 5MB
- Each document should be clearly labeled before upload

3.5 Payment Process

- Open the “Payments” tab.
- Choose Visa/MasterCard, enter payment details securely.
- Confirmation and receipt are emailed to the user.

3.6 Entry Test Booking

- After payment, schedule your test slot under “Test Scheduling”.
- Select from available dates/times.
- A confirmation SMS and email will be sent.

3.7 Notification Center

- All updates such as approvals, payment status, and test info will be sent via email and displayed in the dashboard.

4. Admin User Guide

4.1 Admin Login

- Go to `/admin-login` URL. • Use authorized credentials.

4.2 Managing Applications

- Approve, reject, or return student applications with comments.
- Filter by program, date, or status.

4.3 Merit List Generation

- Use the “Generate Merit List” function after all approvals.
- Export list in PDF or Excel formats.

4.4 Notifications

- Admins can push mass notifications via dashboard.
- SMS gateway and email configurations are built-in.

4.5 Monitoring & Logs

- View login history, failed attempts, and audit trails.
- Download weekly reports.

5. Troubleshooting Guide

Issue	Likely Cause	Solution
Can't login	Wrong email/password	Use 'Forgot Password' link
File not uploading	Format/size error	Check if file is PDF/JPEG and <5MB
Payment not verified	Delay in transaction sync	Retry or contact support with ref. ID
Unable to select slot	No available test times	Contact admin or try a different date

UI/UX Design Document

Prepared By: UI/UX Design Team

1. Introduction

The purpose of this document is to describe the user interface and user experience (UI/UX) strategy for the APFUA project. This includes layout rationale, navigation flow, accessibility, responsiveness, and visual consistency across the platform.

2. Design Objectives

- Provide a modern, intuitive, and responsive user interface.
- Ensure seamless navigation across various modules (registration, application, test scheduling, payments).
- Support accessibility for all users (keyboard navigability, screen reader compatibility).
- Optimize for performance across devices (mobile, tablet, desktop).

3. Target Users

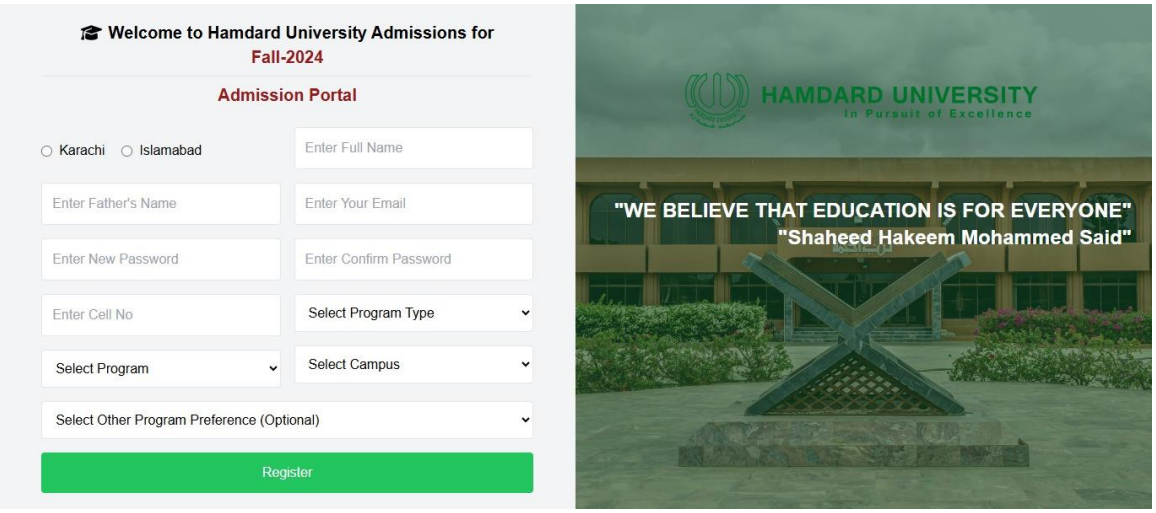
- Prospective Students – To apply for undergraduate admissions, upload documents, and book test slots.
- Administrators – To review applications, manage applicants, schedule tests, and publish merit lists.
- Support Staff – To provide user assistance and troubleshoot issues.

4. UI Design Overview

- Visual references are provided in the separate UI screenshots document.

4.1 Landing Page

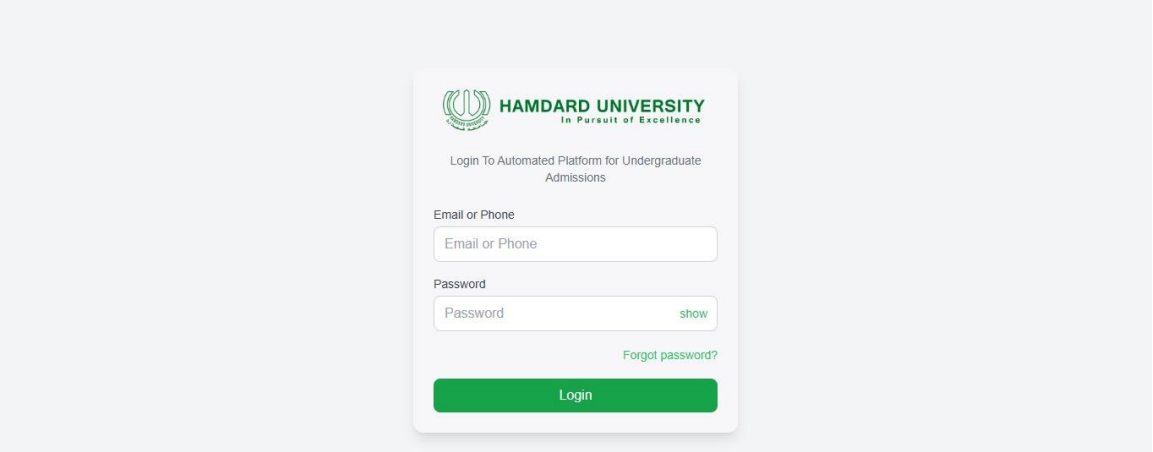
- Minimalistic design with institutional branding.



- CTA buttons for 'Apply Now', 'Login', and 'About Admissions'.

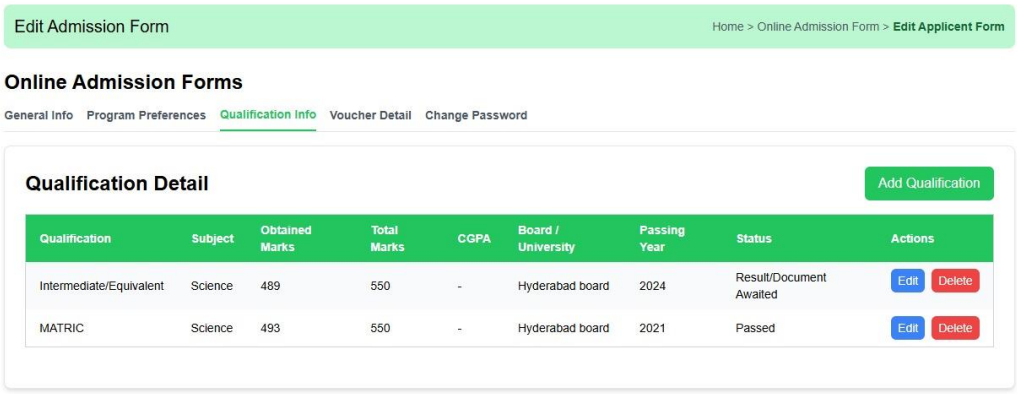
4.2 User Registration & Login

- Role-based login (Student/Admin).
- Password show/hide toggle, form validation, and friendly error messages.



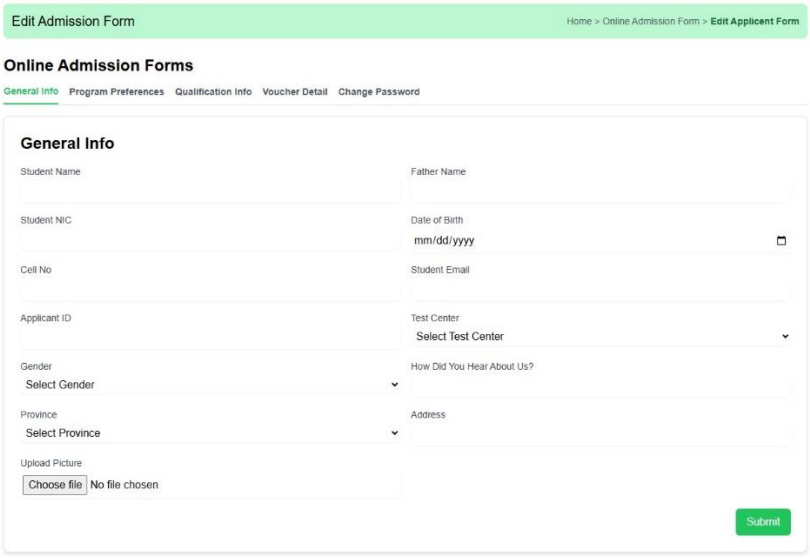
4.3 Dashboard (Student)

- Displays application status, notifications, and pending actions.
- Sidebar navigation for Profile, Documents, Test Scheduling, and Payments.

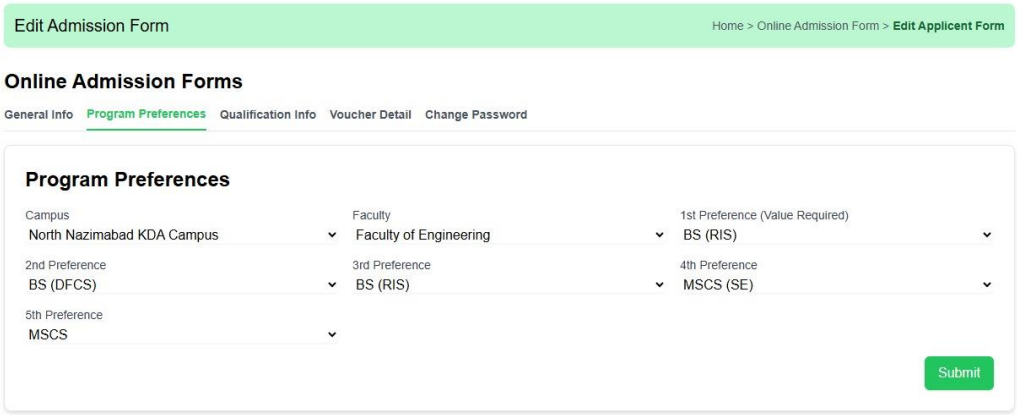


4.4 Application Form

- Multi-step form with field validations.
- Save and progress indicator.

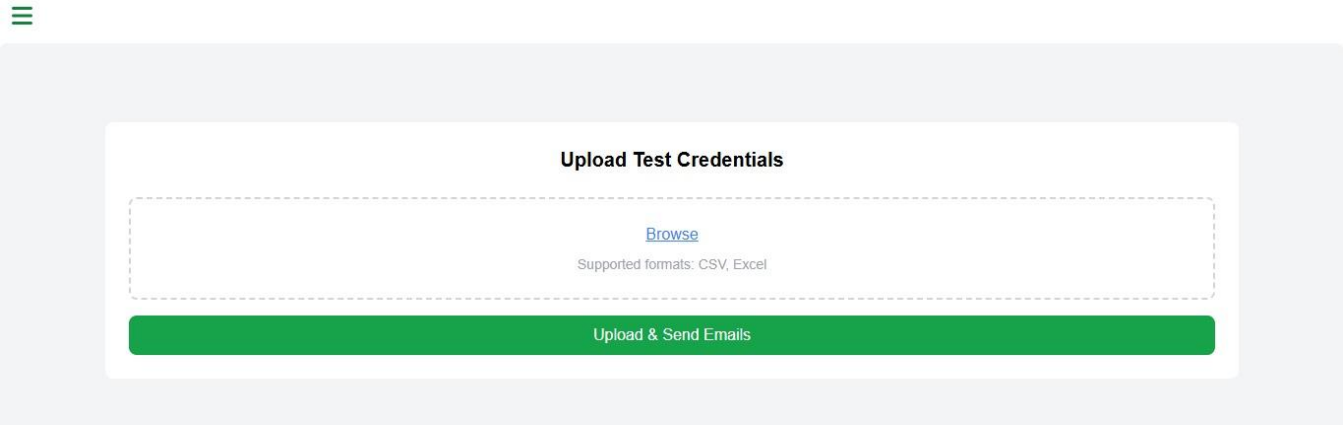


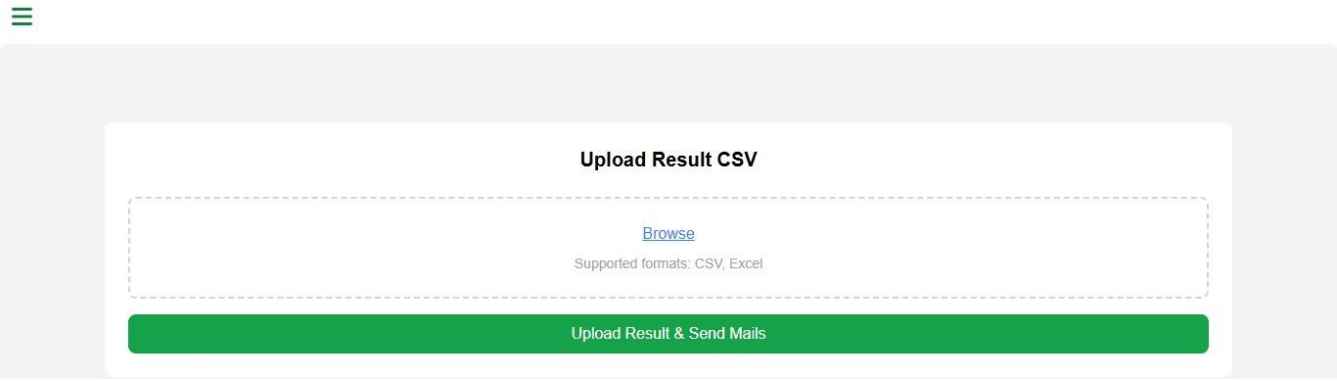
- Program selection form



4.5 Document Upload

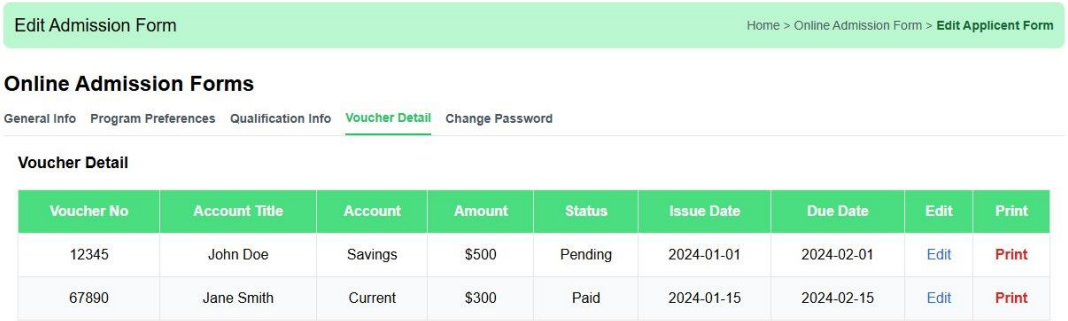
- Drag and drop area with progress feedback.
- File type and size validation.





4.6 Payment Module

- Secure payment gateway interface.
- Confirmation receipts and transaction logs.



4.7 Test Scheduling

- Calendar view for selecting slots.
- Real-time availability indicator.

4.8 Notifications

- Toast and badge-style alerts.
- Dashboard inbox for all messages.

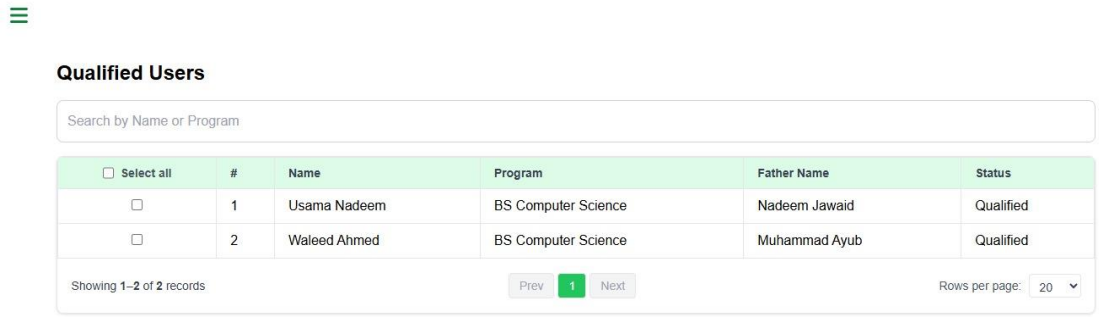
5. Admin Panel UI

5.1 Dashboard

- Visual metrics and direct management links.

5.2 Application Management

- Table view, status filters, and modals for details.



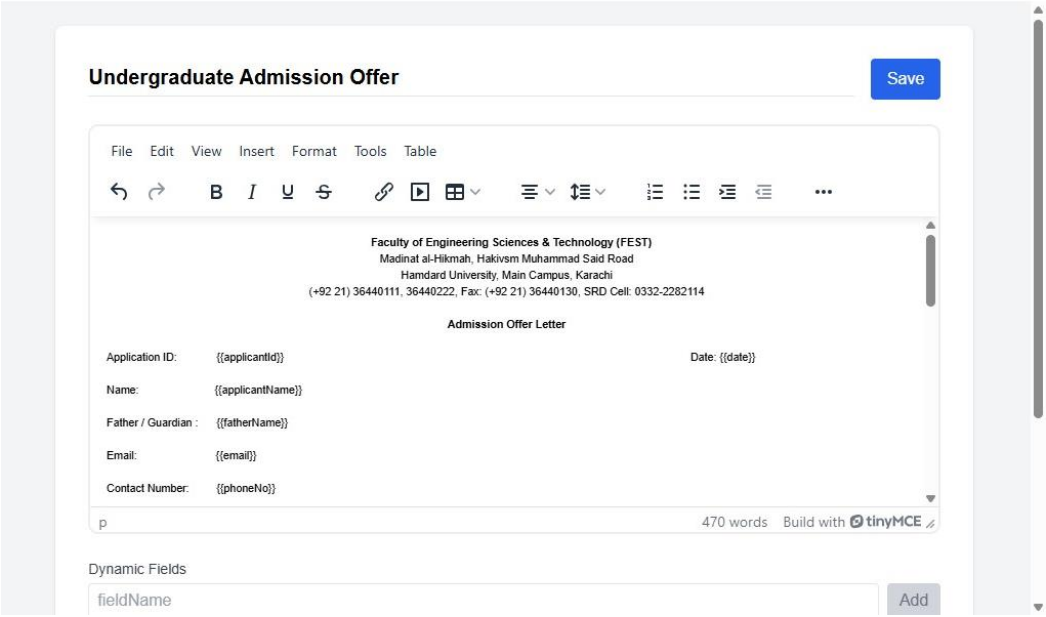
- Admins list and details

5.3 Merit List Generation

- One-click generation with PDF/Excel export.

Offer Letter	Father Name	Student Email	Verification Code	Last Updated	Student Province	Print	Reset Password
First Enter Offer Letter Amount then Print Offer Letter	Nazim Ahmed	safia.a2005@gmail.com	9HRC	2024-12-09 (00:00)	6	Print	Reset Password
First Enter Offer Letter Amount then Print Offer Letter	Muhammad Saleem	firefoxbpo@gmail.com	MEU7	2024-12-08 (00:00)	6	Print	Reset Password
First Enter Offer Letter Amount then Print Offer Letter	Syed Shahid Hussain	ishfatabool2468@gmail.com	5KL4	2024-12-08 (00:00)	6	Print	Reset Password

- Admission offer letter template design.



6. Design System

6.1 Color Scheme

Role	Color Name	HEX Code	Usage
Primary	Midnight Blue	#002B5B	Headers, primary buttons, active states
Secondary	Sky Blue	#00A6FB	Highlights, icons, link hovers
Background	Ghost White	#F8F9FA	Main background color
Text - Primary	Charcoal	#333333	Body text, form labels
Text - Muted	Cool Gray	#6C757D	Placeholders, helper text
Success	Emerald Green	#28A745	Submit buttons, success alerts
Danger	Crimson Red	#DC3545	Error messages, warnings
Warning	Amber Yellow	#FFC107	Info banners, tooltips

6.2 Typography

Property	Value
Font Family	Inter, Roboto, sans-serif
Font Weights	400 (normal), 600 (bold), 700 (headings)
Base Font Size	1rem (responsive)
Title Font Size	1.5rem
Scaling	Uses responsive rem units

6.3 Buttons

Type	Background	Text Color	Font Weight	Notes
Primary Button	#002B5B	`#FFFFFF`	600	Used for main actions
Success Button	#28A745	`#FFFFFF`	600	"Register", "Submit", etc.
Danger Button	#DC3545	`#FFFFFF`	600	For destructive actions

6.4 Components

- Buttons (Primary, Secondary, Disabled)
- Input Fields (Text, Select, Datepicker)
- Alerts (Toast, Modal)
- Cards (Application Status, Exam Slot, Notifications)

7. Responsiveness

- Mobile Devices: Hamburger menu, vertical stacking
- Tablets: Optimized grid layouts
- Desktops: Full-featured layout with side navigation

8. Accessibility Features

- All components are keyboard-navigable
- ARIA labels for assistive tech
- Color contrast meets WCAG 2.1
- Logical tab order and navigation

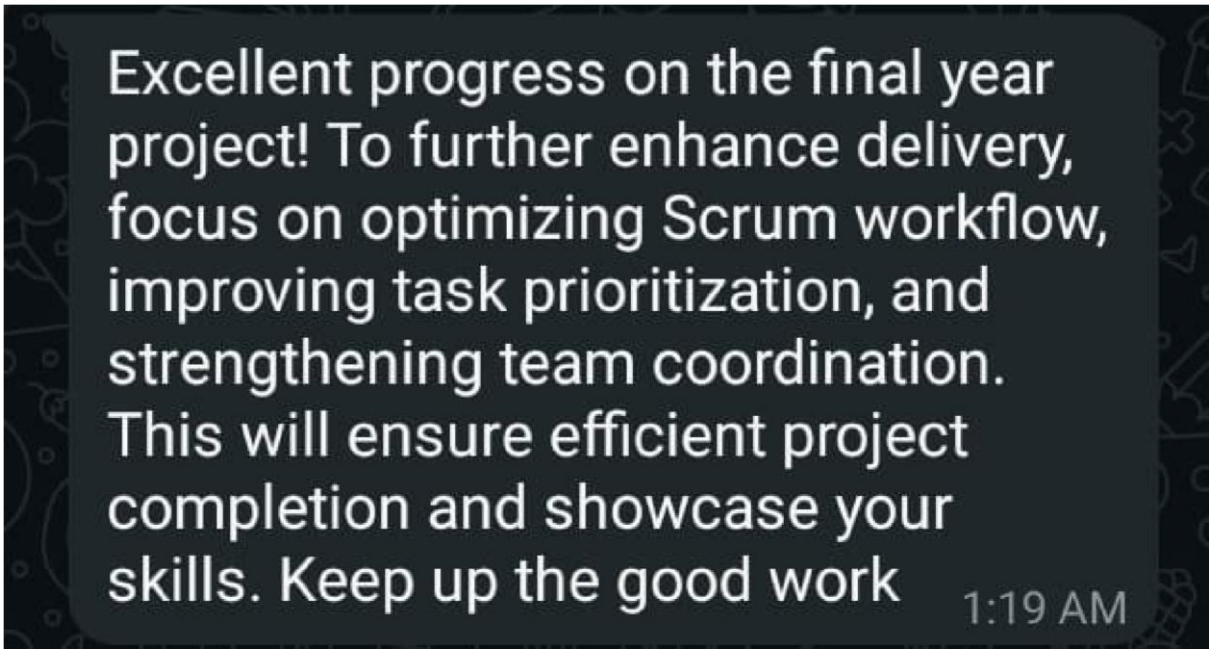
9. Tools Used

- Wireframing: Figma
- Prototyping: Adobe XD
- Development Framework: React + TailwindCSS
- Assets: SVGs and icons from Lucide

A5. FLYER & POSTER DESIGN

A6. COPY OF EVALUATION COMMENTS

COPY OF EVALUATION COMMENTS BY SUPERVISOR FOR PROJECT – I MID SEMESTER EVALUATION

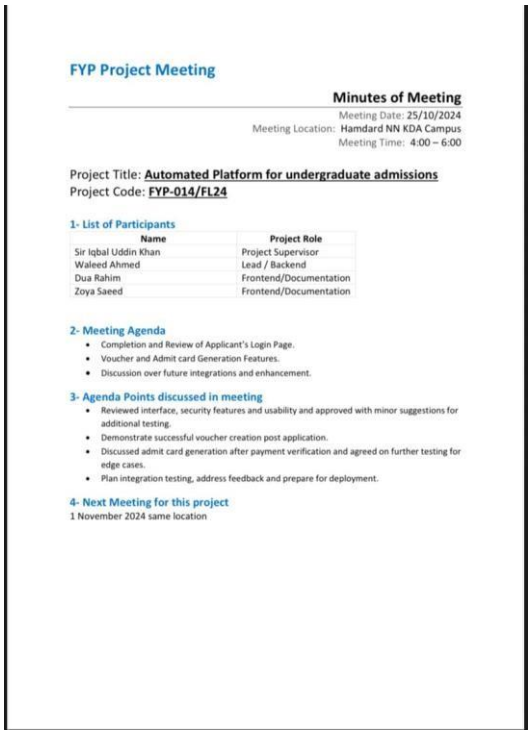


COPY OF EVALUATION COMMENTS BY JURY FOR PROJECT – I END SEMESTER EVALUATION

Dr. Taha Shabbir
Muhammad Salman
Ishmal Shahid
Engr. Farooq Iqbal
Dr. Khalid Charan

Bring value addition in your project
Overall is ok but need to enhance the scope
Overall, the project is okay, but enhancing the scope would provide more depth and increase its potential impact.
Ok but project scope is too narrow. Need to broaden the scope and add value/ novelty to the project.
above average

A7. MEETINGS' MINUTES & Sign-Off Sheet



FYP Project Meeting

Minutes of Meeting

Meeting Date: 25/10/2024
Meeting Location: Hamdard NN KDA Campus
Meeting Time: 4:00 – 6:00

Project Title: **Automated Platform for undergraduate admissions**
Project Code: **FYP-014/FL24**

1- List of Participants

Name	Project Role
Sir Iqbal Uddin Khan	Project Supervisor
Waleed Ahmed	Lead / Backend
Dua Rahim	Frontend/Documentation
Zoya Saeed	Frontend/Documentation

2- Meeting Agenda

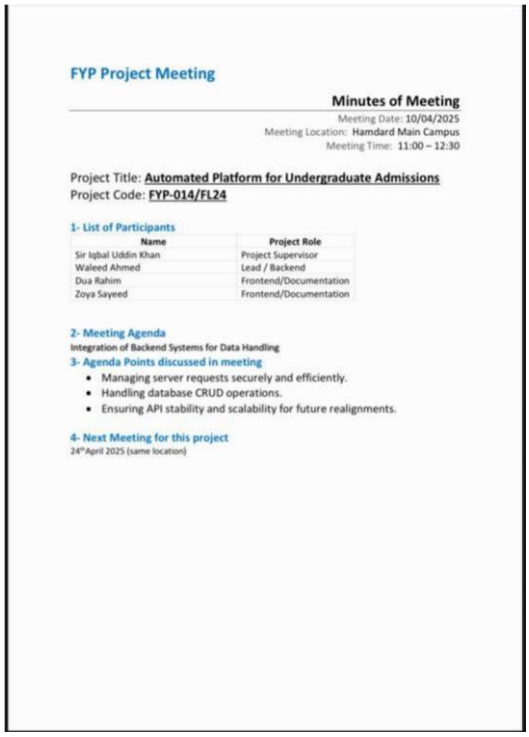
- Completion and Review of Applicant's Login Page.
- Voucher and Admit card Generation Features.
- Discussion over future integrations and enhancement.

3- Agenda Points discussed in meeting

- Reviewed interface, security features and usability and approved with minor suggestions for additional testing.
- Demonstrate successful voucher creation post application.
- Discussed admit card generation after payment verification and agreed on further testing for edge cases.
- Plan integration testing, address feedback and prepare for deployment.

4- Next Meeting for this project

1 November 2024 same location



FYP Project Meeting

Minutes of Meeting

Meeting Date: 10/04/2025
Meeting Location: Hamdard Main Campus
Meeting Time: 11:00 – 12:30

Project Title: **Automated Platform for Undergraduate Admissions**
Project Code: **FYP-014/FL24**

1- List of Participants

Name	Project Role
Sir Iqbal Uddin Khan	Project Supervisor
Waleed Ahmed	Lead / Backend
Dua Rahim	Frontend/Documentation
Zoya Sayeed	Frontend/Documentation

2- Meeting Agenda

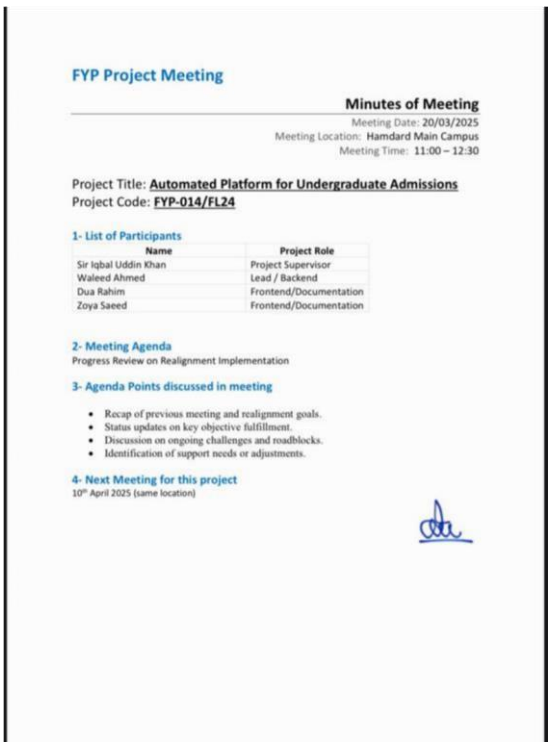
Integration of Backend Systems for Data Handling

3- Agenda Points discussed in meeting

- Managing server requests securely and efficiently.
- Handling database CRUD operations.
- Ensuring API stability and scalability for future realignments.

4- Next Meeting for this project

24th April 2025 (same location)



FYP Project Meeting

Minutes of Meeting

Meeting Date: 20/03/2025
Meeting Location: Hamdard Main Campus
Meeting Time: 11:00 – 12:30

Project Title: **Automated Platform for Undergraduate Admissions**
Project Code: **FYP-014/FL24**

1- List of Participants

Name	Project Role
Sir Iqbal Uddin Khan	Project Supervisor
Waleed Ahmed	Lead / Backend
Dua Rahim	Frontend/Documentation
Zoya Saeed	Frontend/Documentation

2- Meeting Agenda

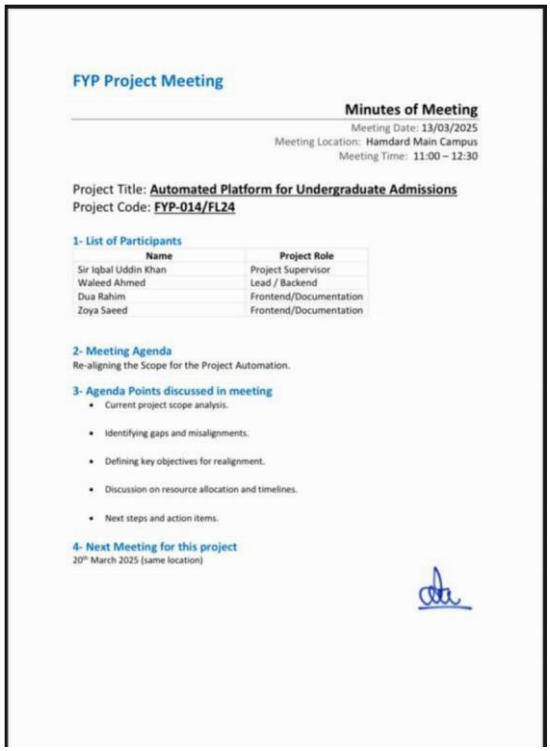
Progress Review on Realignment Implementation

3- Agenda Points discussed in meeting

- Recap of previous meeting and realignment goals.
- Status updates on key objective fulfillment.
- Discussion on ongoing challenges and roadblocks.
- Identification of support needs or adjustments.

4- Next Meeting for this project

10th April 2025 (same location)



FYP Project Meeting

Minutes of Meeting

Meeting Date: 13/03/2025
Meeting Location: Hamdard Main Campus
Meeting Time: 11:00 – 12:30

Project Title: **Automated Platform for Undergraduate Admissions**
Project Code: **FYP-014/FL24**

1- List of Participants

Name	Project Role
Sir Iqbal Uddin Khan	Project Supervisor
Waleed Ahmed	Lead / Backend
Dua Rahim	Frontend/Documentation
Zoya Saeed	Frontend/Documentation

2- Meeting Agenda

Re-aligning the Scope for the Project Automation.

3- Agenda Points discussed in meeting

- Current project scope analysis.
- Identifying gaps and misalignments.
- Defining key objectives for realignment.
- Discussion on resource allocation and timelines.
- Next steps and action items.

4- Next Meeting for this project

20th March 2025 (same location)

FYP Project Meeting

Minutes of Meeting

Meeting Date: 20/03/2025
Meeting Location: Hamdard Main Campus
Meeting Time: 11:00 – 12:30

Project Title: **Automated Platform for Undergraduate Admissions**
Project Code: **FYP-014/FL24**

1- List of Participants

Name	Project Role
Sir Iqbal Uddin Khan	Project Supervisor
Waleed Ahmed	Lead / Backend
Dua Rahim	Frontend/Documentation
Zoya Saeed	Frontend/Documentation

2- Meeting Agenda

Progress Review on Realignment Implementation

3- Agenda Points discussed in meeting

- Recap of previous meeting and realignment goals.
- Status updates on key objective fulfillment.
- Discussion on ongoing challenges and roadblocks.
- Identification of support needs or adjustments.

4- Next Meeting for this project

10th April 2025 (same location)

FYP Project Meeting

Minutes of Meeting

Meeting Date: 13/03/2025
Meeting Location: Hamdard Main Campus
Meeting Time: 11:00 – 12:30

Project Title: **Automated Platform for Undergraduate Admissions**
Project Code: **FYP-014/FL24**

1- List of Participants

Name	Project Role
Sir Iqbal Uddin Khan	Project Supervisor
Waleed Ahmed	Lead / Backend
Dua Rahim	Frontend/Documentation
Zoya Saeed	Frontend/Documentation

2- Meeting Agenda

Re-aligning the Scope for the Project Automation.

3- Agenda Points discussed in meeting

- Current project scope analysis.
- Identifying gaps and misalignments.
- Defining key objectives for realignment.
- Discussion on resource allocation and timelines.
- Next steps and action items.

4- Next Meeting for this project

20th March 2025 (same location)

FYP PROJECT MEETING

MINUTES OF MEETING

Meeting Date: 11/10/2024
Meeting Location: Hamdard NN KDA Campus
Meeting Time: 4:00 – 5:30

Project Title: **Automated Platform for Undergraduate Admissions**
Project Code: **FYP-014/FL24**

1- List of Participants

Name	Project Role
Sir Iqbal Uddin Khan	Project Supervisor
Waleed Ahmed	Lead / Backend
Dua Rahim	Frontend/Documentation
Zoya Saeed	Frontend/Documentation

2- Meeting Agenda

- Overview of current applicant login pages and forms.
- Analysis of strengths and weaknesses.
- Discussion on proposed improvements.
- Identifying action items and assigning responsibilities.

3- Agenda Points discussed in meeting

- Reviewed current login interfaces and form field and Discussed login methods
- user-friendly interface, basic validation checks
- Modernize layout, add progress indicators.

4- Next Meeting for this project

18th October 2024 (same location)

FYP Project Meeting

Minutes of Meeting

Meeting Date: 06/05/2025
Meeting Location: Hamdard Main Campus
Meeting Time: 11:00 – 12:30

Project Title: **Automated Platform for Undergraduate Admissions**
Project Code: **FYP-014/FL24**

1- List of Participants

Name	Project Role
Sir Iqbal Uddin Khan	Project Supervisor
Waleed Ahmed	Lead / Backend
Dua Rahim	Frontend/Documentation
Zoya Sayeed	Frontend/Documentation

2- Meeting Agenda

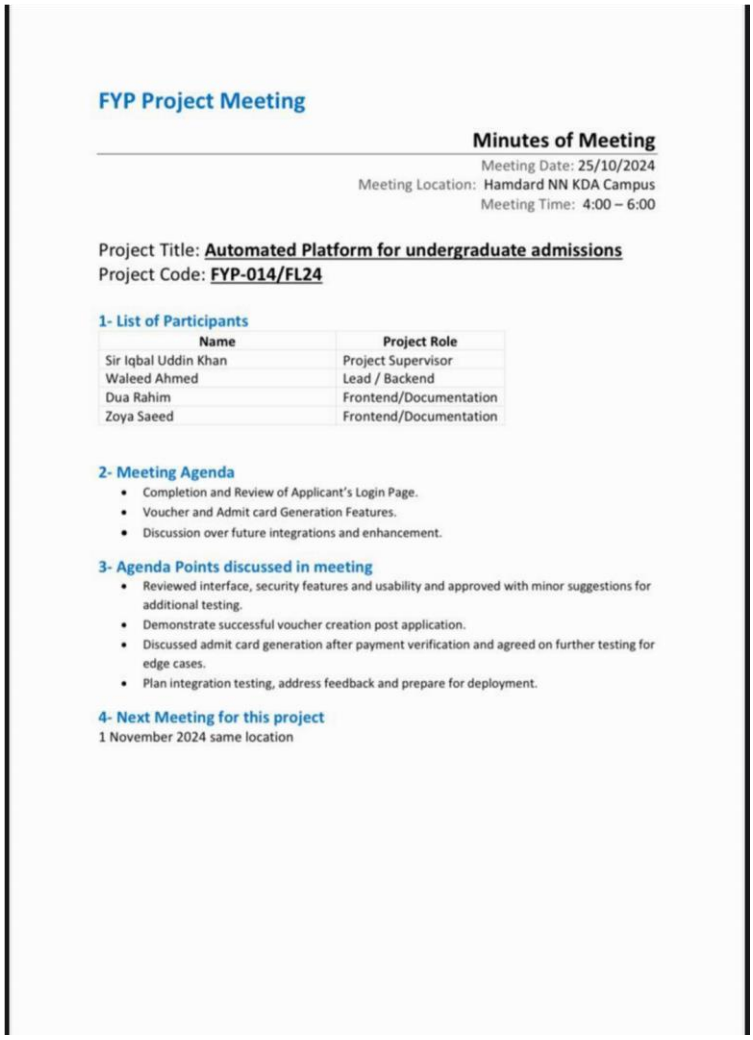
Frontend-Backend Communication & User Data Flow Validation

3- Agenda Points discussed in meeting

- Mapping backend endpoints to UI components
- Validating user input and secure data transmission
- Discussing potential bugs and error handling mechanisms

4- Next Meeting for this project

15th May 2025 (same location)



FYP Project Meeting

Minutes of Meeting

Meeting Date: 25/10/2024
Meeting Location: Hamdard NN KDA Campus
Meeting Time: 4:00 – 6:00

Project Title: **Automated Platform for undergraduate admissions**
Project Code: **FYP-014/FL24**

1- List of Participants

Name	Project Role
Sir Iqbal Uddin Khan	Project Supervisor
Waleed Ahmed	Lead / Backend
Dua Rahim	Frontend/Documentation
Zoya Saeed	Frontend/Documentation

2- Meeting Agenda

- Completion and Review of Applicant's Login Page.
- Voucher and Admit card Generation Features.
- Discussion over future integrations and enhancement.

3- Agenda Points discussed in meeting

- Reviewed interface, security features and usability and approved with minor suggestions for additional testing.
- Demonstrate successful voucher creation post application.
- Discussed admit card generation after payment verification and agreed on further testing for edge cases.
- Plan integration testing, address feedback and prepare for deployment.

4- Next Meeting for this project

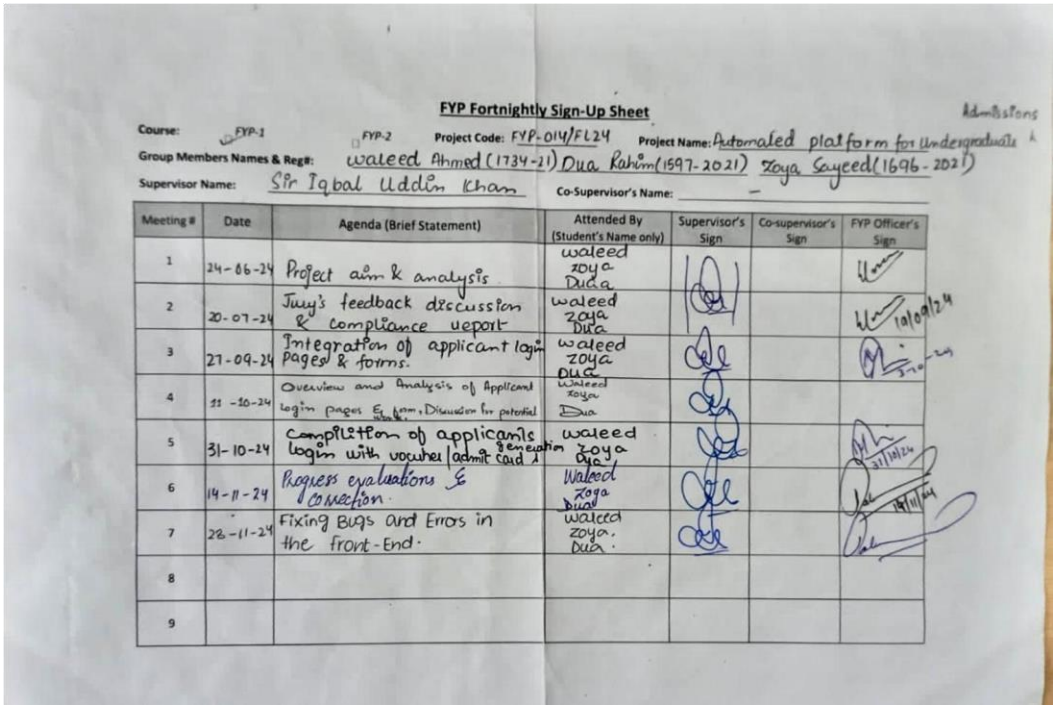
1 November 2024 same location

A8. DOCUMENT CHANGE RECORD

Date	Version	Author	Change Details
28/06/2025	01	Dua Rahim	Created the document with all relevant details according to the chapters.
29/06/2025	02	Zoya Sayeed	Added proposal, SRS & SDS
01/07/2025	03	Dua Rahim	Added all the snapshots and relevant media
03/07/2025	04	Dua Rahim Zoya Sayeed	Fixing typo, grammar, alignment & formatting

A9. PROJECT PROGRESS SIGN UP SHEET

FYP-01



FYP Fortnightly Sign-Up Sheet

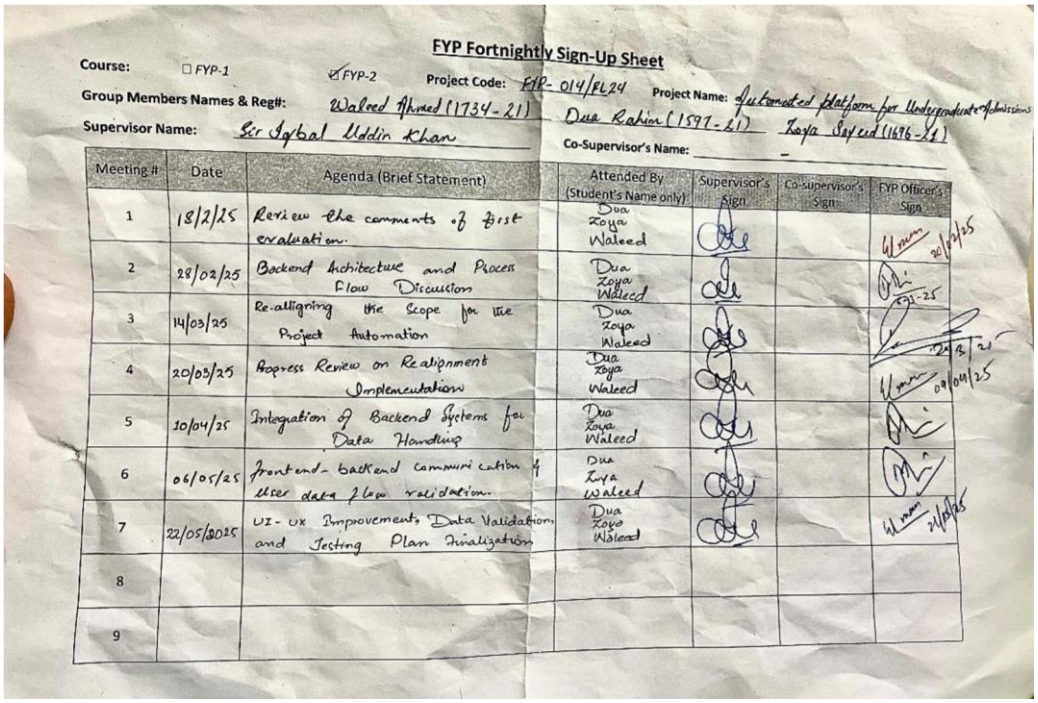
Course: ☐ FYP-1 ☐ FYP-2 Project Code: FYP-014/FL24 Project Name: Automated platform for Undergraduate Admissions

Group Members Names & Reg#: Waleed Ahmed (1734-21) Dua Rahim (1597-2021) Zoya Sayeed (1696-2021)

Supervisor Name: Sir Iqbal Uddin Khan Co-Supervisor's Name: –

Meeting #	Date	Agenda (Brief Statement)	Attended By (Student's Name only)	Supervisor's Sign	Co-supervisor's Sign	FYP Officer's Sign
1	24-06-24	Project aim & analysis	waleed zoya Dua			
2	20-07-24	Jury's feedback discussion & compliance report	waleed zoya Dua			10/08/24
3	27-09-24	Integration of applicant login pages & forms.	waleed zoya Dua			
4	11-10-24	Overview and Analysis of Applicant login pages & form, Discussion for potential	Waleed Zoya Dua			
5	31-10-24	Compilation of applicants login with voucher/admit card generation	waleed zoya Dua			31/10/24
6	14-11-24	Progress evaluations & correction	Waleed Zoya Dua			14/11/24
7	28-11-24	Fixing Bugs and Errors in the Front-End.	waleed zoya. Dua.			
8						
9						

FYP-02



FYP Fortnightly Sign-Up Sheet

Course: ☐ FYP-1 ☑ FYP-2 Project Code: FYP-014/FL24 Project Name: Automated Platform for Undergraduate Admissions

Group Members Names & Reg#: Waleed Ahmed (1734-21) Dua Rahim (1597-21) Zoya Sayeed (1696-21)

Supervisor Name: Sir Iqbal Uddin Khan Co-Supervisor's Name: –

Meeting #	Date	Agenda (Brief Statement)	Attended By (Student's Name only)	Supervisor's Sign	Co-supervisor's Sign	FYP Officer's Sign
1	18/2/25	Review the comments of first evaluation.	Dua Zoya Waleed			20/02/25
2	28/02/25	Backend Architecture and Process Flow Discussion	Dua Zoya Waleed			07-25
3	14/03/25	Re-aligning the Scope for the Project Automation	Dua Zoya Waleed			21/3/25
4	20/03/25	Progress Review on Realignment Implementation	Dua Zoya Waleed			09/04/25
5	10/04/25	Integration of Backend Systems for Data Handling	Dua Zoya Waleed			
6	06/05/25	Frontend- backend communication & user data flow validation.	Dua Zoya Waleed			
7	22/05/2025	UI - UX Improvements, Data Validation, and Testing Plan Finalization	Dua Zoya Waleed			21/05/25
8						
9						