A REPORT

ON

Scholarship Management System using SpringBoot

Submitted by,

Ms. Safia Rafi – 20211CAI0159
Adnan Talha Adil – 20211CDV0054
Dibbagalla Joy – 20211CDV0066
Mohammed Nihal A S – 20211CSE0367
Madhu Kumar V – 20211CSE0363
Mr. Hanfaa Khanum – 20211CSD0106

Under the guidance of,

Dr. Jayanthi Kamalasekaran

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

At

PRESIDENCY UNIVERSITY

BENGALURU

MAY 2025

PRESIDENCY UNIVERSITY

PRESIDENCY SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Internship report "**Scholarship Management System using SpringBoot"** being submitted by "SAFIA RAFI, ADNAN TALHA ADIL, DIBBAGALLA JOY, MOHAMMED NIHAL A S, MADHU KUMAR V, HANFAA KHANUM" bearing roll number "20211CAI0159, 20211CDV0054, 20211CDV0066, 20211CSE0367, 20211CSE0363, 20211CSD0106" in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in Computer Science and Engineering is a bonafide work carried out under my supervision.

Dr. JAYANTHI KAMALASEKARAN
Associate Professor
Presidency school of CSE(PSCS)
Presidency University

Dr. ASIF MOHAMED H B
Associate Professor & HoD
Presidency school of CSE(PSCS)
Presidency University

Dr. MYDHILI K NAIR
Professor & Associate Dean
Presidency school of CSE(PSCS)
Presidency University

Dr. MD. SAMEERUDDIN KHAN
Pro-Vice Chancellor - Engineering
Dean –PSCS & PSIS
Presidency University

PRESIDENCY UNIVERSITY

PRESIDENCY SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We hereby declare that the work, which is being presented in the report entitled "**Scholarship Management System using SpringBoot"** in partial fulfillment for the award of Degree of **Bachelor of Technology** in **Computer Science and Engineering**, is a record of our own investigations carried under the guidance of **Dr. Jayanthi Kamalasekaran , Associate Professor,** School of Computer Science Engineering & Information Science, Presidency University, Bengaluru.

We have not submitted the matter presented in this report anywhere for the award of any other Degree.

Name(s)	Roll No.	Signature(s)
Safia Rafi	20211CAI0159	
Adnan Talha Adil	20211CDV0054	
Dibbagalla Joy	20211CDV0066	
Mohammed Nihal A S	20211CSE0367	
Madhu Kumar V	20211CSE0363	
Hanfaa Khanum	20211CSD0106	

ABSTRACT

Managing scholarship applications manually is often inefficient and error-prone, especially with large volumes of data and diverse funding schemes. To address this, a **Scholarship Management System** was developed using **Spring Boot**, featuring role-based access for administrators and students. This web-based platform streamlines application workflows, enhances transparency, and supports real-time decision-making through dynamic data handling and visualization.

Authentication and authorization are secured using **Spring Security**, enforcing login functionality and clear role segregation. The login table dynamically stores user credentials and roles. Students access application-related features, while admins manage system controls, ensuring operational integrity and data privacy.

The student portal provides a user-friendly interface with a navigation bar linking to the homepage, scheme details, "About Us," contact form, scholarship tracking, and an application page. Students can apply for scholarships and monitor their status directly.

The admin portal features a dashboard displaying key statistics like application counts, approval rates, pending cases, and total funds disbursed. A sidebar enables navigation to manage scholarships and review, approve, or reject student applications. Admins can also view student messages submitted via the contact page.

The backend relies on four core tables managed by **Spring Boot's JPA** and **Hibernate**: login, messages, student_applications, and scholarships. Controllers are cleanly mapped to specific operations, ensuring modularity and maintainability. The architecture is scalable and ready for enhancements like automated eligibility checks or third-party integration.

By automating the scholarship process, the system offers a secure, efficient, and user-centric solution that improves both administrative workflows and student engagement.

ACKNOWLEDGEMENTS

First of all, we indebted to the **GOD ALMIGHTY** for giving me an opportunity to excel in our efforts to complete this project on time.

We express our sincere thanks to our respected dean **Dr. Md. Sameeruddin Khan**, Pro-VC - Engineering and Dean, Presidency School of Computer Science and Engineering & Presidency School of Information Science, Presidency University for getting us permission to undergo the project.

We express our heartfelt gratitude to our beloved Associate Dean **Dr. Mydhili K Nair,** Presidency School of Computer Science and Engineering, Presidency University, and Dr. "ASIF MOHAMED H B", Head of the Department, Presidency School of Computer Science and Engineering, Presidency University, for rendering timely help in completing this project successfully.

We are greatly indebted to our guide **Dr. Jayanthi Kamalasekaran** and Reviewer **Mr. Sunil Kumar Sahoo**, Presidency School of Computer Science and Engineering, Presidency University for his inspirational guidance, and valuable suggestions and for providing us a chance to express our technical capabilities in every respect for the completion of the internship work.

We would like to convey our gratitude and heartfelt thanks to the CSE7301 Internship/University Project Coordinator **Mr. Md Ziaur Rahman and Dr. Sampath A K,** department Project Coordinators and GitHub coordinator **Mr. Muthuraj.**

We thank our family and friends for the strong support and inspiration they have provided us in bringing out this project.

Safia Rafi(20211CAI0159)

Adnan Talha Adil(20211CDV0054)

Dibbagalla Joy(20211CDV0066)

Mohammed Nihal A S(20211CSE0367)

Madhu Kumar V(20211CSE0363)

Hanfaa Khanum(20211CSD0106)

LIST OF TABLES

LIST OF FIGURES

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

1.1 Background and Relevance

In today's digital age, the need for effective, transparent, and accessible public service delivery is more than ever before. One such imperative area that needs upgradation is scholarship management by educational institutions and government organizations. Manual systems of scholarship application handling are not only tedious but also inefficient, error-prone, prone to data loss, and result in undue delays in processing. With the progress of web technology and backend frameworks, it is apparent that there is a potential to digitize and automate these processes.

Web management systems based on contemporary frameworks like Spring Boot have emerged as sturdy, scalable, and secure processes of automation. Relying on functionalities like Spring Security for authentication and role handling, and Spring Data JPA for effortless database interaction, development in such systems helps institutions handle voluminous data in an efficient manner while providing secure as well as user-friendly access to various user roles.

1.1.1 Significance of Digital Scholarship Management Systems

Scholarship programs have an important role in financing students and facilitating access to education in an equitable way. An effective digital management system means that scholarship procedures—applications to approvals—are efficient, standardized, and transparent. Such systems enable real-time tracking, eliminate paperwork, and make administrative decisions faster. For students, it allows ease, improved visibility, and fewer intermediaries. For administrators, it enables streamlined dashboards, category-based insights, and effective fund management tools.

By using technologies such as Spring Boot, the system can provide RESTful APIs, role-based access, and dynamic content handling with adequate scalability and performance. Graphical dashboards, CRUD operations for scholarships, and automatic eligibility checking

significantly improve operational capabilities.

1.2 Motivation for the Project

The inspiration for this project comes from the fact that there is a necessity to automate and digitize the scholarship process in schools. Students tend to struggle with filling forms, submitting hard copy documents, and checking the status of their applications manually. Meanwhile, administrators are bogged down with sorting and authenticating high volumes of paper applications. The objective was to remove these inefficiencies by creating a secure, role-based web application that enables both students and administrators to interact with the system without any hassle.

1.3 Scope of the Project

This project is about creating a web-based Scholarship Management System with Spring Boot and Spring Security, having two major roles: Student and Admin. Students are allowed to register, apply for scholarships, view schemes, check status, and communicate with administrators. Admins are presented with a dashboard displaying analytical charts (applications by category, funds disbursed, pending approvals), and can manage scholarships, review applications, and reply to messages. The system utilizes four main dynamically generated tables—login, student_applications, scholarships, and messages—used to manage data flow.

Developed with modular controllers and dynamic database operations via Spring JPA and Hibernate, the application is built to be scalable, maintainable, and secure with future possibilities for add-ons like document verification automation, incorporation of financial systems, or multiple language support.

Chapter 2

LITERATURE SURVEY

2.1 Overview of Automation Tools

Contemporary scholarship management systems emphasize automating application, review, and award processes for greater efficiency and transparency. Most studies utilize web technologies for ease of access, while others investigate cloud-based options for scalability and management of data. Mobile apps are increasingly used for enhancing student accessibility and real-time feedback. The most common features across systems include document upload, tracking of status, and role-based access control. Yet some of these works are short of thorough security features and thorough performance analysis. Interoperability with current university infrastructures and fraud prevention through document authentication continue to be problem areas. Generally, scholarship management solutions are moving towards intuitive, scalable, and automated platforms that minimize human workload. Future studies ought to focus on strong security, easy integration, and thorough user trials.

2.2 Literature Review

- S. Gawas and S. D. Deshmukh, 2018 This paper illustrates a web-based scholarship management system with emphasis placed on automating application and review. The system streamlines scholarship application submission and assessment, cutting paperwork and human error. It utilizes web technologies for simplicity of access and real-time updates. The paper concentrates, however, on fundamental functionalities rather than on scalability or integration with other learning platforms. [1]

- S. Kumar, R. K. Sharma, and A. Sharma, 2018 The authors created a student-focused scholarship management system facilitating easy application and status tracking. The system promotes transparency and improves administrative efficiency. It employs a web-based interface using a database backend. The research points out increased user satisfaction but does not highlight security or mobile accessibility. [2]

- H. P. Singh and P. K. Mishra, 2019 This study suggests an online system to automate scholarship allocation with elements such as document upload and automated eligibility check. The system is expected to minimize manual processing and error. The conference paper illustrates prototype implementation but does not include intensive user testing and performance analysis. [3]

- Garg and R. K. Sharma, 2018 The research aims to create a completely web-based system of scholarship application, review, and distribution. It underscores user role management of applicant, reviewer, and administrator. The system enhances work process efficiency but does not include cloud deployment or backup strategies. [4]

- S. Ahmad, M. Khan, and A. Ahmed, 2018 This article presents a cloud-based

scholarship management system that is intended to cater for scalability and accessibility. Cloud infrastructure enables the centralized management of data and remote access. The system features automated notifications and document verification. The research highlights cloud advantages but does not have a comparison with the conventional systems. [5]

- M. Hossain, M. Rahman, and M. Islam, 2017 The authors created an easy-to-use web system that manages scholarship applications and database administration. The system aims to minimize manual processes and enhance user experience. Although the design is simple, security aspects like encryption and role-based access control are not mentioned in the paper. [6]

- J. Smith and K. Brown, 2019 This project automates the scholarship application process via a web portal, improving application submission, review, and approval workflows. The system provides real-time status visibility and minimizes administrative burden. The paper focuses on workflow automation but does not cover backend technology or integration with university ERP systems. [7]

- Jain and R. Singh, 2017 The case study investigates cloud computing's application in scholarship management to maximize system reliability and access. It is concerned with deployment on cloud platforms and the benefits of data redundancy and backup. The study reveals benefits in practice but fails to include in-depth performance benchmarks. [8]

- P. Kaur and A. Singh, 2017 This paper demonstrates a PHP and MySQL-based system for handling scholarship applications. It features registration of applicants, document uploading, and tracking application status. The use of open-source technologies reduces the cost of the system, but scalability and security issues are addressed to a limited extent. [9]

- S. M. Ali and T. R. Khan, 2019 The authors suggest using a mobile app-based scholarship system, allowing students to apply and monitor applications through smartphones. The app has notification and document scan functionalities aimed at enhanced accessibility. This research is mobile technology-focused but does not elaborate on backend infrastructure. [10]

Chapter 3

RESEARCH GAPS OF EXISTING METHODS

3.1 Introduction

While various scholarship management systems exist, they often fall short in key areas such as security, accessibility, and administrative control. Through a literature review and analysis of existing solutions, several gaps have been identified that hinder efficiency in managing scholarship applications

3.2 Identified Gaps in Existing Tools

- **Security Challenges in authentication**
  Many systems rely on outdated authentication mechanisms, making them prone to unauthorized access. Implementing Spring Security with JWT provides a more secure approach
- **Limited Personalization for scholarship recommendation**
  Most platforms lack a personalized recommendation system for students based on academic performance and eligibility.
- **Absence of a Centralized Dashboard for Administrators**
  Existing portals often lack a real-time dashboard where administrators can easily track applications, add/delete scholarships, and analyze student data efficiently.
- **User Experience & Accessibility Issues**
  Many systems have outdated and non-intuitive user interfaces, making it challenging for students to find relevant scholarship opportunities.

3.3 Key Research Findings

- **Students prefer intuitive interfaces** with easy navigation instead of complex portals that require unnecessary steps to access scholarship information.
- **Security vulnerabilities in authentication** continue to be a major concern, with many platforms failing to implement **robust encryption techniques** like JWT

authentication.

- **Administrative inefficiencies** arise due to a lack of **real-time tracking and data visualization**, making it hard for admins to monitor scholarship applications efficiently.
- **Personalized scholarship recommendations** are often missing in current systems, preventing students from easily discovering **relevant schemes** tailored to their academic and financial needs.

3.4 Solution Overview

The Scholarship Management System is designed to be a secure, user-friendly, and data-driven portal using Spring Boot, Spring Security, JWT authentication, and a structured admin dashboard . To improve the system, a contact page for student login and a messaging page for admin login should be added. This messaging page would allow administrators to view messages sent by students.

Chapter 4

PROPOSED METHODOLOGY

4.1 Overview

The proposed Scholarship Management System leverages Spring Boot, Spring Security, JWT, and a structured dashboard system to enhance security and streamline scholarship application processes.

4.2 System Architecture

The system is divided into two access modes:

- Student Access: Features tabs such as *Home*, *About*, *Schemes, Apply Now*, and *Contact*, offering a simplified scholarship discovery experience.
- Admin Access: A dashboard that allows administrators to add/delete scholarships, track applications, and analyze statistics.

4.3 Methodology Workflow

Phase 1: Secure Authentication

- Two login options: Admin & Student
- Spring Security & JWT ensure encrypted authentication

Phase 2: Scholarship discovery and application

- Home Tab: Displays new schemes dynamically
- Schemes Tab: Provides a filtered list based on eligibility criteria
- Apply Now: Allows students to submit applications

.

Phase 3: Administrative Control Dashboard

- Scholarship Management: Admins can add or remove scholarships
- Application Tracking: Displays the number of students who have applied

.

Phase 4: Technologies Used

The application uses Spring Boot as a backend framework, Spring Security and JSON Web Tokens (JWT) for secure authentication, Thymeleaf and HTML/CSS for UI development, and MySQL for database management.

1. Spring Boot: This acts as the foundation of the application, providing a simplified and streamlined approach to building Spring-based applications . It handles the configuration and setup of various components, making development easier and faster.

2. Spring Security & JWT: Spring Security is used to secure the application . It integrates seamlessly with JWT (JSON Web Tokens), a standard for creating access tokens used for authentication and authorization. JWTs are stateless, enhancing scalability and reducing the burden on the server . These tokens are typically passed in the Authorization header of HTTP requests . The system likely implements role-based authorization, controlling access based on user roles stored in the MySQL database .

3. Thymeleaf & HTML/CSS: Thymeleaf provides server-side templating capabilities for creating dynamic HTML pages . Combined with HTML and CSS, it forms the user interface, enabling interactions with the secured backend. The JWT token is likely managed within the UI using Thymeleaf, possibly allowing user login and logout functionality .

4. MySQL: This relational database system stores persistent data, including user credentials and potentially other application data . Spring Data JPA (Java Persistence API) likely simplifies interaction with the database .

Chapter 5

OBJECTIVES

5.1 Introduction

Automation and digitization are critical to improving the efficiency, transparency, and scalability of contemporary institutional systems. Scholarship Management System was implemented to substitute manual iterations and error-causing repetitive processes in scholarship management with a role-based secure web application. Built with Spring Boot and Spring Security, the application allows students to submit scholarship applications, view their status, and reach out to administrators, while admins can manage schemes, view applications, and monitor essential statistics via an easy-to-use dashboard. The aim was to craft a simple-to-use yet technologically sound platform that streamlines workflows for students and administrators alike. This chapter details the most important objectives that drove the system's development and operation.

5.2 Detailed Objectives

5.2.1. Automate Scholarship Application Process:

The Scholarship Management System should provide a fully digital platform that allows students to apply for scholarships online without the need for physical paperwork. This automation drastically reduces the time and effort involved in submitting applications, eliminating manual data entry errors and the risk of lost forms. Students can upload documents, fill out forms, and track their applications conveniently from any location. For the administrators, this eliminates redundant work and paperwork, making the workflow easier overall. Automation also facilitates quick processing and improved scalability, enabling institutions to process high volumes of applications more easily. Overall, application process digitization improves accessibility as well as user satisfaction.

5.2.2. Centralize Data Management:

One of the fundamental goals of the system is to consolidate all the scholarship-related

information, such as applicant information, scholarship information, eligibility, and disbursal history, in a single database. Centralization removes redundancy that normally takes place when data is being kept in separate silos. It also makes it easier for scholarship administrators and policymakers to retrieve and manage the data. Moreover, a centralized system improves collaboration among different departments and stakeholders, ensuring that everyone works with the most up-to-date information. This consolidated data repository also simplifies audits and compliance monitoring, supporting transparency and accountability within the scholarship program.

5.2.3. Improve Eligibility Verification:

The system should incorporate automated eligibility verification features to quickly and accurately evaluate whether applicants meet scholarship criteria. By cross-referencing submitted data against predefined eligibility rules—such as academic performance thresholds, income limits, or demographic requirements—the system reduces the need for time-consuming manual checks. Automated verification speeds up decision-making, prevents ineligible candidates from advancing unnecessarily, and minimizes human errors or biases in screening. This feature facilitates fairness and consistency in the choice of worthy candidates. It can also be configured to deal with more than one scholarship schemes, allowing for smooth processing for varied programs.

5.2.4. Improve Transparency:

Transparency is crucial to ensuring trust in scholarship administration. The mechanism must give applicants instant feedback on the status of their applications, such as confirmation of submission, verification of documents, approval, or rejection. To both administrators and funding organizations, thorough audit trails and process logs must be kept, allowing for thorough monitoring of every step of the scholarship life cycle. Transparent procedures eliminate misunderstandings and grievances by making it evident how decisions are reached and when actions take place. Transparency dissuades corruption and favouritism in favour of equity. In addition, transparent

reporting engenders accountability among the stakeholders, enhancing the overall integrity of the scholarship program.

5.2.5. Streamline Document Verification

Document verification is frequently a bottleneck in the processing of scholarships because certificates, proofs of income, and identification documents are manually scrutinized. The system will need to include technologies such as Optical Character Recognition (OCR) for automatically extracting and verifying data from uploaded documents. Automated verification decreases the amount of work for humans, makes the process of reviewing faster, and makes it more accurate by reducing errors introduced by manual transcription. In addition, the system can identify incomplete or suspicious documents for human verification, maintaining integrity. Multi-language support and support of many different document types facilitate ease of use for applicants from around the world. Such a simplified verification process eventually accelerates scholarship approval and disbursement of funds.

5.2.6. Facilitate Timely Fund Disbursement:

Once scholarship recipients are selected, the system must enable secure and prompt transfer of funds directly to their bank accounts. Integration with banking and government payment gateways ensures funds are disbursed efficiently through Direct Benefit Transfer (DBT) or similar mechanisms. Timely disbursement enhances student satisfaction and trust in the scholarship program. Additionally, automated tracking of disbursed amounts helps administrators monitor fund utilization and detect any discrepancies. The system should also support multiple disbursement schedules, accommodating one-time or installment-based scholarship payments. Ensuring secure, timely payments is critical to the program's success and beneficiary welfare.

5.2.7. Enable Role-Based Access:

A Scholarship Management System serves various users, including students, reviewers, and administrators, each requiring different access levels. Role-based

access control (RBAC) ensures that users can only view and modify data appropriate to their responsibilities. For example, applicants should only access their application details, while reviewers can assess applications without accessing sensitive administrative data. Administrators need full control for managing scholarships and generating reports. RBAC strengthens security by preventing unauthorized data access and reduces the risk of accidental or malicious modifications. This clear separation of duties also supports compliance with data privacy regulations and institutional policies.

5.2.8. Support for Mobile and Web Access:

Accessibility is important to allow all qualifying students to apply and monitor scholarships irrespective of location or device. The system must have a responsive web interface or specific mobile applications to ensure smooth usability on smartphones, tablets, and desktops. Because students in rural or far-flung places mostly use mobile phones to get access to the internet, mobile support makes the system more accessible. The user interface must be friendly and low-bandwidth optimized for handling different network qualities. Multi-platform support makes the system more inclusive and encourages level opportunities for scholarship applicants.

5.2.9. Ensure Data Security and Privacy:

Scholarship applications involve collecting sensitive personal and financial information, making security a top priority. The system must implement robust data protection measures, including encryption of data at rest and in transit, strong authentication protocols, and regular security audits. Compliance with data privacy laws such as the General Data Protection Regulation (GDPR) and local IT regulations is essential to avoid legal penalties and protect user rights. Secure handling of data builds trust among applicants and stakeholders. The system should also include mechanisms for secure data backup and recovery to prevent data loss due to system failures or cyberattacks, ensuring continuous and safe operation.

5.2.10. Endorse Multi-Scholarship and Multi-Institution Management:

The system should be multi-scholarship scheme-friendly, encompassing those schemes funded by various institutions, the government, or nongovernmental organizations. It should support varied eligibility criteria, application modes, and fund release timelines for every scheme under a single integrated system. This makes it possible for providers to manage their respective schemes efficiently without having multiple systems. Also, having multiple institutions under a single system makes it easier to monitor centrally, compare across, and optimize resources between programs. This goal improves scalability, lowers administrative burden, and gives applicants applying for multiple scholarships across different portals a smooth experience.

Chapter 6

SYSTEM DESIGN & IMPLEMENTATION

6.1 Introduction

This chapter outlines the software architecture and system-level implementation of the application. The project leverages a modular architecture to ensure scalability, maintainability, and future enhancements. Built using Spring Boot, the system integrates multiple components, including the user interface, authentication, database management, business logic, and security mechanisms.

6.2 System Architecture Overview

The application operates in key phases involving data processing, authentication, and application logic execution. The system is designed using Spring Boot and Thymeleaf to ensure an interactive and dynamic web application.

Core Modules:

- **GUI Module:** Provides a user-friendly interface using Thymeleaf for rendering views dynamically.
- **Authentication Module:** Implements Spring Security and JWT for secure login and role-based access control.
- **Database Module:** Manages application data using MySQL and Spring Data JPA.
- **Business Logic Module:** Handles core functionalities and data processing, ensuring smooth operations.
- **Error Handling & Logging Module:** Captures and manages system logs, debugging issues efficiently.

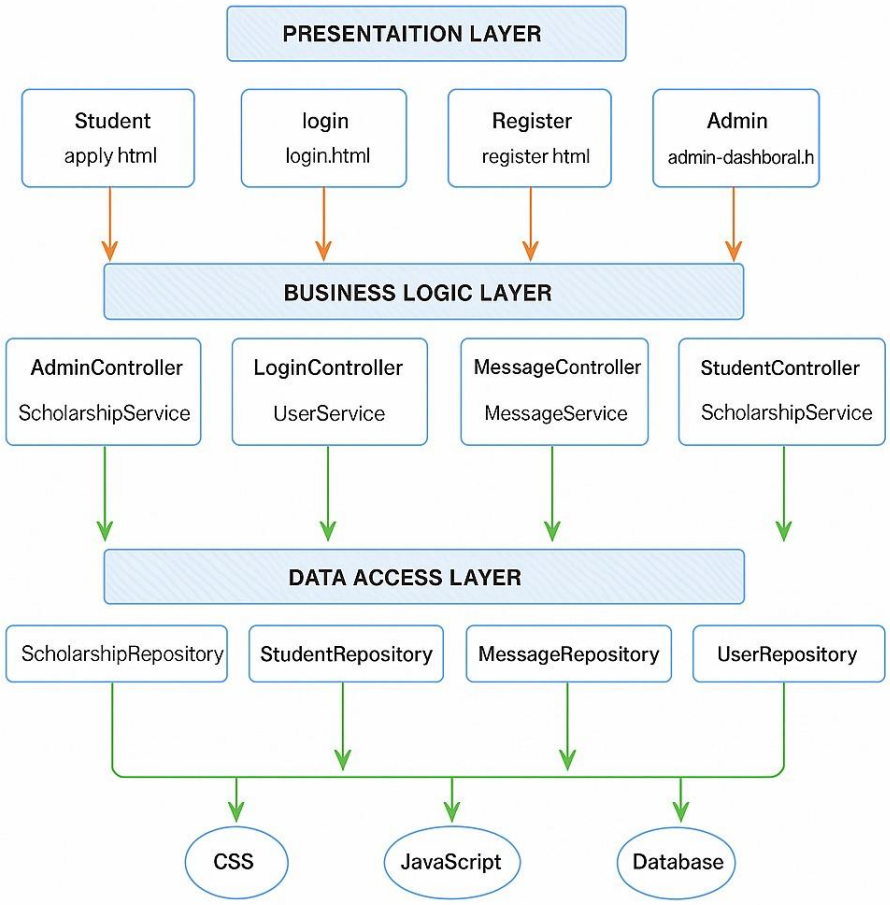


Figure 6.1: System Architecture

6.3 Workflow Description

- **User Interaction Phase:** Users interact with the application through the Thymeleaf-based interface.
- **Authentication & Authorization:** On login, Spring Security verifies credentials and assigns roles using JWT tokens.
- **Data Processing & Business Logic Execution:** The system processes requests and executes business logic through service layers.
- **Database Management & Logging:** Spring Data JPA interacts with MySQL, ensuring efficient data persistence while maintaining logs for monitoring.
- **Session Management:** Implements stateless authentication for secure and scalable user access.

6.4 Technologies and Tools Used

The main framework for backend development is called Spring Boot.

- JWT & Spring Security: Controls authorization and authentication.
- Thymeleaf: Manages front-end rendering that is dynamic.
- MySQL and Spring Data JPA: Control data retrieval and storage.
- Debugging & Logback Tools: Record logs to fix errors.

6.5 System Design Features

Layered Architecture: Assures update flexibility by dividing issues into modules. By establishing user roles and permissions, role-based access control offers security. JWT tokens are used in stateless session management to eliminate needless session tracking. Mechanisms for handling errors and logging them improve system dependability and debugging.

Chapter-7

TIMELINE FOR EXECUTION OF PROJECT

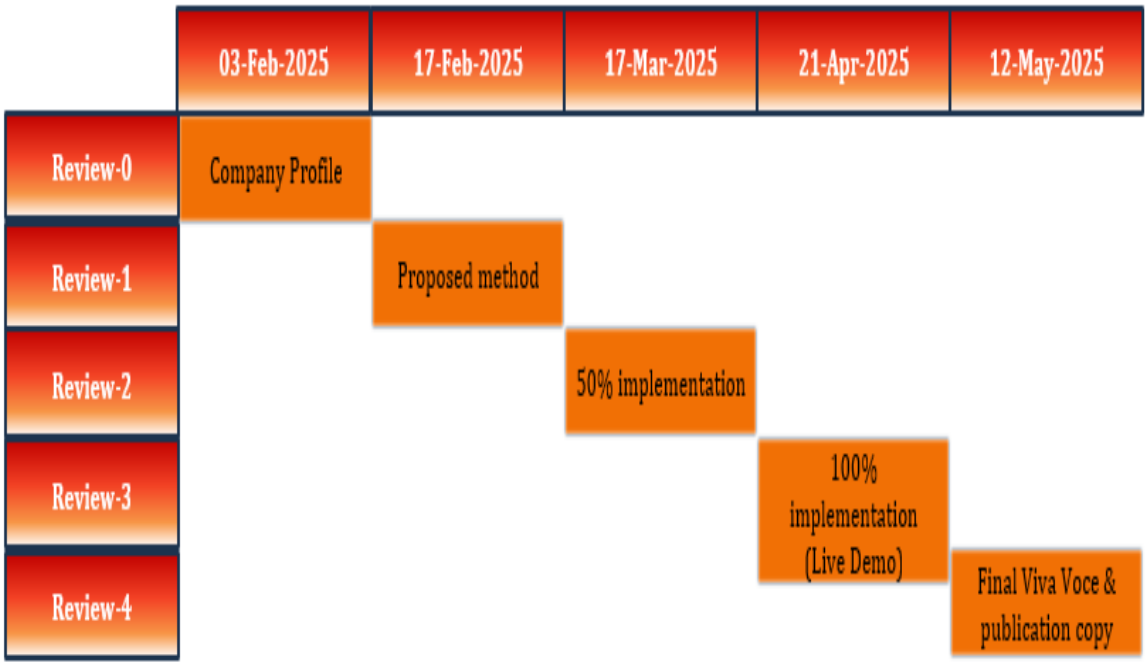
7.1 Introduction

The project was undertaken over a period of twelve weeks utilizing a structured and phased approach to ensure milestones were complied with and a working end product was delivered. Each stage of the project was planned toward a specific goal, beginning with requirement analysis and ending with final documentation and report submission. The following Gantt chart outlines the weekly distribution of tasks and the timeline for the execution of the project.

7.2 Gantt Chart and Timeline

The project was divided into several well-defined phases including research, design, implementation, testing, and documentation. The table below presents the Gantt chart showing the allocation of time for each phase during the internship period.

Table 7.1: Gantt Chart



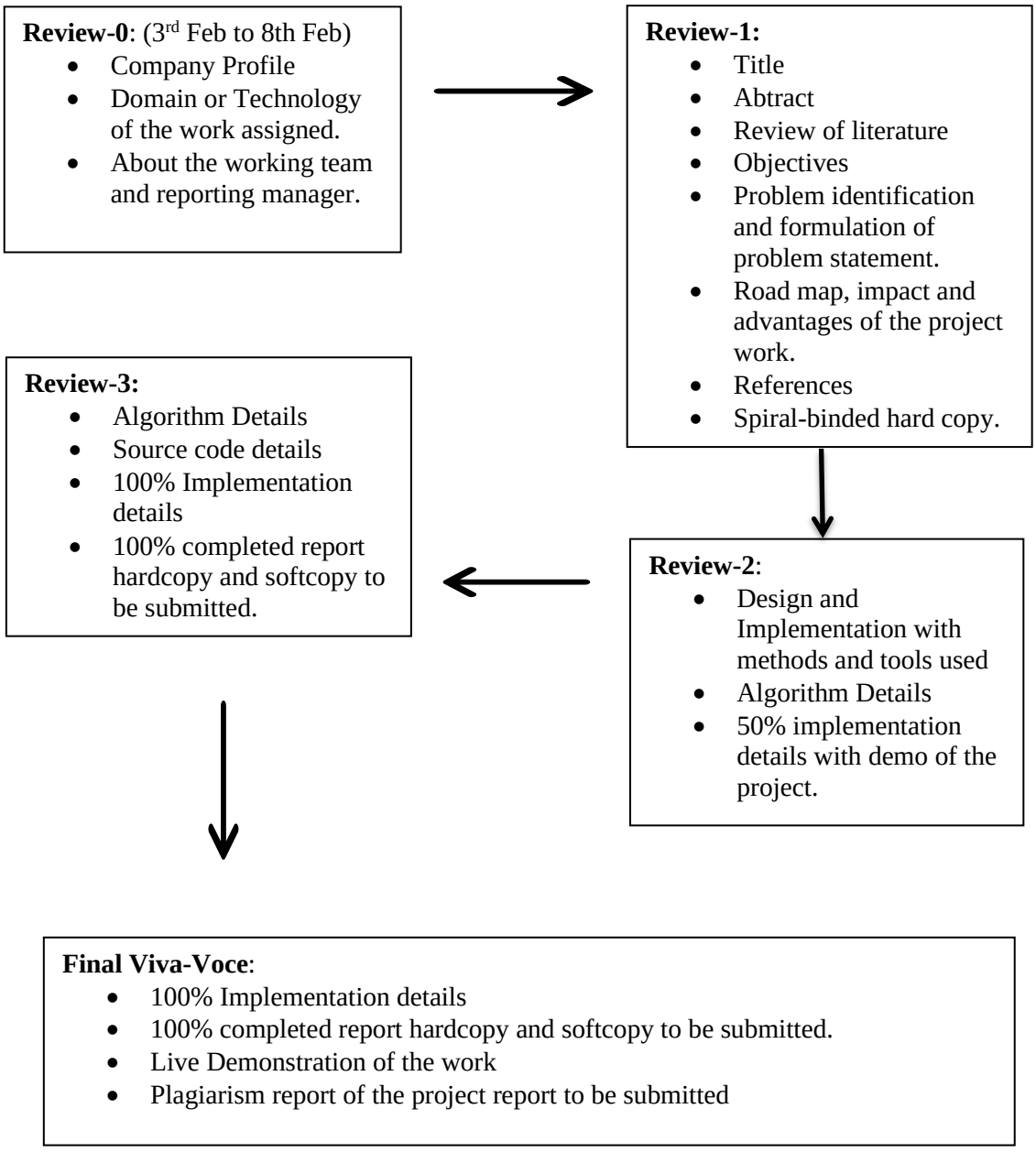


Figure 7.1: Timeline for Execution.

The execution followed a systematic development methodology with clearly defined tasks and deliverables in each phase. Through proper time management and team coordination, all phases of the project were completed within the stipulated duration. The successful implementation and demonstration of the scholarship management system, along with a detailed report, marked the conclusion of the internship project.

Chapter 8

OUTCOMES

8.1 Introduction

The successful development and implementation of this Spring Boot-based web application resulted in several impactful outcomes. These outcomes are measurable both technically and practically, reflecting improvements in system functionality, security, and efficiency. Additionally, the project strengthened development expertise, exposing developers to advanced backend techniques such as Spring Security, JWT authentication, error handling, and database management.

8.2 Functional Outcomes

The application achieved core objectives such as secure authentication, role-based access control, and smooth data processing. By integrating Spring Security and JWT, the system ensures stateless session management while preventing unauthorized access. Users can interact with the Thymeleaf-based GUI, allowing seamless navigation and intuitive data processing.

Key functional achievements:

- Secure Login & User Management: Implemented JWT-based authentication, ensuring encrypted, stateless access.
- Role-Based Access Control: Defined user roles and permissions, limiting unauthorized access to sensitive operations.
- Structured Data Handling: Utilized Spring Data JPA for efficient database management and MySQL for persistent data storage.
- Error Handling & Logging Mechanism: Established logging frameworks, improving debugging and system monitoring.

8.3 Technical Outcomes

From a technical perspective, the project leveraged Spring Boot's powerful ecosystem to deliver a scalable and modular backend framework. The development process emphasized

modularity, separating authentication, business logic, and database interactions into distinct service layers.

Key technical advancements:

- Spring Boot for Rapid Development: Simplified configuration using Spring Boot's auto-configuration and dependency injection features.
- JWT for Stateless Authentication: Elimi nated session-related complexities, ensuring scalable security.
- Spring Data JPA for Database Management: Enabled efficient CRUD operations with minimal boilerplate code.
- Exception Handling & Debugging: Integrated structured logging, improving the resolution of runtime errors.

8.4 Usability and User-Centric Outcomes

The project significantly improved the user experience by delivering a secure and responsive web interface. The Thymeleaf-based front-end ensures dynamic content rendering, allowing users to interact with data effortlessly.

User-centric outcomes:

- Improved Accessibility: Designed a clean and structured UI, ensuring smooth navigation.
- Enhanced Security Features: Users can operate securely within assigned roles, preventing unauthorized actions.
- Optimized Performance: Leveraged Spring Boot's built-in optimizations for fast data retrieval and efficient request handling.

8.5 Performance Outcomes

The system was tested across various scenarios, including high-concurrency authentication requests, large database queries, and error-prone API interactions. Results demonstrated high stability and accurate session handling, proving the effectiveness of Spring Security and JWT authentication.

Performance highlights:

- Fast Authentication Processing: JWT reduces latency by avoiding server-side session tracking.

- Smooth Data Transactions: Spring Data JPA optimized database queries, ensuring efficient data retrieval.
- Reliable Error Handling: Implemented custom exception handling, reducing system failures.

8.6 Educational and Developmental Outcomes

Beyond functionality, this project served as a valuable learning experience, deepening understanding of backend development, security protocols, debugging techniques, and database management. Developers gained exposure to industry-standard frameworks and best practices.

Key learning areas:

- Advanced Security Implementation: Gained expertise in Spring Security configurations, JWT, and role-based authorization.
- Structured Debugging Techniques: Understood exception handling methods, logging frameworks, and error resolution strategies.
- Database Optimization: Explored Spring Data JPA for efficient query execution and schema management.

Chapter 9

RESULTS AND DISCUSSIONS

9.1 Introduction

This chapter presents the results achieved through the design and development of the Scholarship Management Portal and offers a detailed discussion of its implementation and operational aspects. The system aims to provide an accessible, organized, and scalable platform for managing scholarship applications through a digital interface.

9.2 System Functionality and User Interaction

The Scholarship Management Portal was conceptualized to address the common challenges faced by academic institutions in managing large volumes of scholarship applications. The developed system comprises multiple static pages, including home.html, apply.html, admin.html, and login.html, which simulate core operations and user interaction flow.
Two major user roles are supported:

- **Students** can explore scholarship information, access the application form, and submit their academic and personal details.

- **Administrators** have access to a dashboard where they can review submitted applications and manage the portal's content, though dynamic data rendering is pending future back-end integration.

The login interface, though currently static, is designed to serve as a foundation for implementing secure role-based authentication in the future using technologies such as Java Spring Security.

9.3 User Interface Results

The front-end implementation achieved the following:

Feature	Outcome
Homepage Navigation	Functional and Responsive
Application Form	Static Form with Field Validation
Admin Dashboard (UI)	Structurally Designed
Dynamic Login Page	Placeholder for Authentication
Visual Design and Branding	Thematically Consistent

All pages feature consistent layout, thematic imagery, and intuitive navigation. The apply.html form captures key fields such as student name, email, educational background, and desired scholarship, enabling users to understand the kind of data that would typically be collected in a live application process.

9.4 User Experience and Feedback

The interface was evaluated on usability, clarity, and structure. The navigation between pages was smooth, with clear call-to-action elements guiding user interaction. The use of meaningful icons, buttons, and color palettes contributes positively to the overall user experience. Additionally, image assets and content were contextually appropriate, reinforcing the educational theme.

Despite these strengths, the form lacks advanced input validation or error handling, and dynamic interaction (such as field population or conditional logic) is not yet implemented. These are areas earmarked for enhancement during back-end development.

9.5 System Architecture Readiness

While the current version of the portal is front-end only, its structure is highly modular and aligned with back-end readiness. The HTML structure and form design are built to support RESTful API calls that can be integrated with a Java Spring Boot back-end. Database operations such as storing user data, retrieving applications, and updating scholarship statuses can be seamlessly added.

This separation of concerns adheres to good software engineering practices and ensures scalability. Future extensions may include:

- Dynamic form submissions using AJAX
- Login-based authentication using JWT
- Admin workflows with real-time dashboards and application filters

9.6 Challenges Encountered

Several challenges were observed during development:

1. Static Form Handling: Without a back-end, all data submissions remain local and non-functional. This limited the testing of complete form workflows.

2, Mobile Responsiveness: While partially responsive, the interface needs improvement for optimal rendering on smaller devices.

3, Validation and Security: The absence of client-side JavaScript validation and server-side protection presents data quality and security issues that will be resolved in the full stak version.

9.7 Future Scope

Several enhancements are planned to convert this prototype into a fully functional system:

- **Backend Integration**: Using Java Spring Boot to handle user authentication, data persistence, and scholarship logic.
- **Database Management**: Integration with MySQL for managing user profiles and applications.
- **Security Implementation**: Adding encrypted login credentials and data protection mechanisms.
- **Admin Features**: Incorporating filters, sorting, and decision-making workflows in the admin panel.
- **Notification System**: Sending updates to students via email/SMS regarding application status.

Chapter 10

CONCLUSION

The **Scholarship Management System**, designed with **Spring Boot** and **Spring Security**, effectively eliminates the inefficiencies and inconsistencies of conventional, manual scholarship application processes. By providing safe, role-based access for students as well as administrators, the system eliminates repetitive interactions, maintains data centrally, and enhances transparency in decision-making. The initiative started with the evaluation of current scholarship workflows, which found existing bottlenecks like absence of real-time tracking, manual document management, and inadequate communication between applicants and reviewers. These issues served as the basis for developing a highly scalable, easy-to-use web-based application that is designed according to educational and institutional requirements.

There were two main user types—**Student** and **Admin**. Students have been given an interface to view available schemes, apply for them, check their scholarship status, and post queries via a contact form. Administrators, on the contrary, are given a rich dashboard with charts and statistics like application breakup, approvals, pending cases, and total amount disbursed. Admins can deal with scholarship schemes using add/edit/delete functionalities, screen applications for eligibility, and reply to student messages—all via an integrated interface. The backend is based on four basic dynamic tables—login, students, scholarships, and messages—enabled by **Spring Data JPA** and **Hibernate** to guarantee smooth data persistence and retrieval.

Technically, the application of **Spring Boot** facilitated quick development with a clean and modular architecture, while **Spring Security** imposed stringent authentication and authorization rules. Clear controller mappings and optimized data handling logic helped ensure system stability and ease of maintenance. The easy navigation structure, responsive UI, and real-time updating also improved user experience, with less need for manual support and documentation.

User testing validated the platform functioned consistently with a variety of use cases, confirming such main features as role-based access, real-time status tracking, CRUD scholarship operations, and message communication. The modular approach will support future additions like automated eligibility checks, third-party funding API integration, or upload document and e-signature capabilities.

Overall, the Scholarship Management System achieves its goal of digitizing and automating the entire scholarship life cycle. It is a practical, secure, and scalable solution that enables institutions to more efficiently manage scholarships and enhance access and student experience. The project proves effective application of full-stack development principles, problem-solving, and user-centred design, and sets a solid foundation for further innovation in the area of educational process automation.

REFERENCES

[1] A. S. Gawas and S. D. Deshmukh, "Design and Implementation of Scholarship Management System Using Web Technology," *International Journal of Computer Applications*, vol. 179, no. 44, pp. 1–5, 2018.

[2] S. Kumar, R. K. Sharma, and A. Sharma, "Development of a Scholarship Management System for Students Using Web Application," *International Journal of Computer Science and Mobile Computing*, vol. 7, no. 5, pp. 78–84, May 2018.

[3] H. P. Singh and P. K. Mishra, "Online Scholarship Management System," *Proceedings of the 2019 International Conference on Computer Communication and Informatics (ICCCI)*, Coimbatore, India, Jan. 2019, pp. 1–6.

[4] Garg and R. K. Sharma, "Web Based Scholarship Management System," *International Journal of Engineering and Technology*, vol. 7, no. 3, pp. 1234–1238, 2018.

[5] S. Ahmad, M. Khan, and A. Ahmed, "A Comprehensive Scholarship Management System Using Cloud Computing," *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 3, pp. 321–326, 2018.

[6] M. Hossain, M. Rahman, and M. Islam, "Design and Development of Web Based Scholarship Management System," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 7, no. 1, pp. 78–85, Jan. 2017.

[7] J. Smith and K. Brown, "Automation of Scholarship Application Process Using Web Portal," *International Journal of Emerging Technologies in Learning*, vol. 14, no. 4, pp. 10–20, 2019.

[8] A. Jain and R. Singh, "Cloud Based Scholarship Management System: A Case Study," *International Conference on Cloud Computing and Big Data Analysis*, Chengdu, China, May 2017, pp. 110–115.

[9] P. Kaur and A. Singh, "Design and Implementation of Scholarship Management System Using PHP and MySQL," *International Journal of Computer Applications*, vol. 167, no. 6, pp. 1–7, 2017.

[10] S. M. Ali and T. R. Khan, "A Smart Scholarship Management System Using Mobile Application," *2019 International Conference on Computer, Communication, Control and Information Technology (C3IT)*, Bandung, Indonesia, Nov. 2019, pp. 233–238.

APPENDIX-A

PSEUDOCODE

a. BACK END

1.MODEL

STUDENT MODEL:

```
package com.admin.scholarship.model;

import jakarta.persistence.*;

@Entity
@Table(name = "students")
public class Student {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name= "id")
    private int id;

    @Column(name="studid")
    private String studid;

    @Column(name="name")
    private String name;

    @Column(name="income")
    private Double income;

    @Column(name="email", nullable = false, unique = true)
    private String email;

    @Column(name = "tenth_percentage")
    private Double tenthPercentage;

    @Column(name = "twelfth_percentage")
    private Double twelfthPercentage;

    @Column(name = "cgpa")
    private Double cgpa;

    @Column(name = "scholarshiptype")
    private String scholarshiptype;
```

```
public String getScholarshiptype() {
		return scholarshiptype;
	}

	public void setScholarshiptype(String scholarshiptype) {
		this.scholarshiptype = scholarshiptype;
	}


	@Column(name = "amount")
private double amount;

public double getAmount() {
		return amount;
	}

	public void setAmount(double amount) {
		this.amount = amount;
	}


	@Enumerated(EnumType.STRING)
private Status status = Status.PENDING;

public enum Status {
   PENDING, APPROVED, REJECTED
}

public int getId() {
   return id;
}

public void setId(int id) {
   this.id = id;
}

public String getStudid() {
		return studid;
}

public void setStudid(String studid) {
		this.studid=studid;
}
public String getName() {
   return name;
}
```

```
public void setName(String name) {
   this.name = name;
}

public Double getIncome() {
   return income;
}

public void setIncome(Double income) {
   this.income = income;
}

public String getEmail() {
   return email;
}

public void setEmail(String email) {
   this.email = email;
}

public Double getTenthPercentage() {
   return tenthPercentage;
}

public void setTenthPercentage(Double tenthPercentage) {
   this.tenthPercentage = tenthPercentage;
}

public Double getTwelfthPercentage() {
   return twelfthPercentage;
}

public void setTwelfthPercentage(Double twelfthPercentage) {
   this.twelfthPercentage = twelfthPercentage;
}

public Double getCgpa() {
   return cgpa;
}

public void setCgpa(Double cgpa) {
   this.cgpa = cgpa;
}

public Status getStatus() {
   return status;
```

```
    }

    public void setStatus(Status status) {
        this.status = status;
    }
}
```

SCHOLARSHIP MODEL:

```
package com.admin.scholarship.model;


import jakarta.persistence.*;

@Entity
@Table(name = "scholarships")
public class Scholarship {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name="id")
    private Long id;
    @Column(name="title")
    private String title;
    @Column(name="amount")
    private Double amount;
    @Column(name="egc1")
    private String egc1;
    @Column(name="egc2")
    private String egc2;
    @Column(name="egc3")
    private String egc3;
    @Column(name="egc4")
    private String egc4;
    @Column(name="egc5")
    private String egc5;

        public Long getId() {
                return id;
        }
        public void setId(Long id) {
                this.id = id;
        }
        public String getTitle() {
                return title;
        }
        public void setTitle(String title) {
                this.title = title;
```

```
	}
	public Double getAmount() {
		return amount;
	}
	public void setAmount(Double amount) {
		this.amount = amount;
	}
	public String getEgc1() {
		return egc1;
	}
	public void setEgc1(String egc1) {
		this.egc1 = egc1;
	}
	public String getEgc2() {
		return egc2;
	}
	public void setEgc2(String egc2) {
		this.egc2 = egc2;
	}
	public String getEgc3() {
		return egc3;
	}
	public void setEgc3(String egc3) {
		this.egc3 = egc3;
	}
	public String getEgc4() {
		return egc4;
	}
	public void setEgc4(String egc4) {
		this.egc4 = egc4;
	}
	public String getEgc5() {
		return egc5;
	}
	public void setEgc5(String egc5) {
		this.egc5 = egc5;
	}

}
```

MESSAGE MODEL:

```
package com.admin.scholarship.model;

import jakarta.persistence.*;

@Entity
@Table(name = "messages")
```

```
public class Message {

   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   private Long id;

   private String name;

   private String email;

   @Column(length = 1000)
   private String message;

        public String getName() {
                return name;
        }

        public void setName(String name) {
                this.name = name;
        }

        public String getEmail() {
                return email;
        }

        public void setEmail(String email) {
                this.email = email;
        }

        public String getMessage() {
                return message;
        }

        public void setMessage(String message) {
                this.message = message;
        }
   // Getters and Setters
}
```

2,CONTROLLER

STUDENT CONTROLLER:

```
package com.admin.scholarship.controller;

import com.admin.scholarship.model.Student;
import com.admin.scholarship.repository.ScholarshipRepository;
```

```
import com.admin.scholarship.repository.StudentRepository;
import com.admin.scholarship.service.StudentService;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.*;
import org.springframework.ui.Model;

@Controller
public class StudentController {

        @Autowired
        private StudentRepository studentRepository;

   @GetMapping("/apply")
   public String showApplyForm(Student student ,Model model) {
        model.addAttribute("student", new Student());
        model.addAttribute("scholarships", scholarshipRepository.findAll());
        student.setTenthPercentage(null);
      student.setTwelfthPercentage(null);
      student.setIncome(null);
      return "student/apply";
   }

   @GetMapping("/login")
   public String logoutAndRedirect() {

      return "/login";
   }

   @Autowired
   private StudentService studentService;


   @GetMapping("/student/searchStudent")
   @ResponseBody
   public String getStudentStatus(@RequestParam String email) {
      Student student = studentService.findByEmail(email);
      if (student != null) {
         return student.getStatus().toString();
      } else {
         return "Student not found";
      }
   }

   @PostMapping("/apply")
   public String submitForm(@ModelAttribute Student student, Model model) {
```

```
        student.setStatus(Student.Status.PENDING);
        studentRepository.save(student);
        model.addAttribute("message", "Application submitted successfully!");
        return "/student/home";
    }
    @Autowired
    private ScholarshipRepository scholarshipRepository;

    @GetMapping("/student/home")
    public String showStudentHome(Model model) {
        model.addAttribute("scholarships", scholarshipRepository.findAll());
        return "/student/home";
    }
}
```

ADMIN CONTROLLER:

```
package com.admin.scholarship.controller;

import com.admin.scholarship.model.Student;
import com.admin.scholarship.model.Scholarship;
import com.admin.scholarship.service.StudentService;
import com.admin.scholarship.service.ScholarshipService;

import java.util.Map;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;

@Controller
@RequestMapping("/admin")
public class AdminController {

    @Autowired
    private StudentService studentService;

    @Autowired
    private ScholarshipService scholarshipService;

    @GetMapping("/applications")
    public String viewApplications(Model model) {
        model.addAttribute("students", studentService.getAllStudents());
        model.addAttribute("totalAmount", studentService.getTotalSanctionedAmount());
        model.addAttribute("totalApplications", studentService.getAllStudents().size());
        return "admin/applications";
    }
```

```
    @PostMapping("/approve/{id}")
    public String approve(@PathVariable int id) {
        studentService.updateStatus(id, Student.Status.APPROVED);
        return "redirect:/admin/applications";
    }

    @PostMapping("/reject/{id}")
    public String reject(@PathVariable int id) {
        studentService.updateStatus(id, Student.Status.REJECTED);
        return "redirect:/admin/applications";
    }

    @GetMapping("/dashboard")
    public String viewDashboard(Model model) {
        double totalAmount = studentService.getTotalSanctionedAmount();
        int totalApplications = studentService.getAllStudents().size();
        int approvedCount = studentService.countByStatus(Student.Status.APPROVED);
        int pendingCount = studentService.countByStatus(Student.Status.PENDING);

        Map<String, Long> typeCountMap = studentService.getScholarshipTypeCounts();
        Map<String, Long> approvedTypeCountMap =
studentService.getApprovedScholarshipTypeCounts();

        model.addAttribute("totalAmount", totalAmount);
        model.addAttribute("totalApplications", totalApplications);
        model.addAttribute("approvedCount", approvedCount);
        model.addAttribute("pendingCount", pendingCount);
        model.addAttribute("typeCounts", typeCountMap);
        model.addAttribute("approvedTypeCounts", approvedTypeCountMap);

        return "/admin/dashboard";
    }

    @GetMapping("/logout")
    public String logout() {
        return "admin/logout";
    }


    @GetMapping("/scholarships")
    public String viewScholarships(Model model) {
        model.addAttribute("scholarships", scholarshipService.getAllScholarships());
        return "admin/scholarships";
    }
    @GetMapping("/add-scholarship")
    public String addScholarshipForm() {
```

```
        return "admin/addscholarship";
    }

    @PostMapping("/save-scholarship")
    public String saveScholarship(@ModelAttribute Scholarship scholarship) {
        scholarshipService.saveScholarship(scholarship);
        return "redirect:/admin/scholarships";
    }

    @PostMapping("/update-scholarship/{id}")
    public String updateScholarship(@PathVariable Long id, @ModelAttribute Scholarship
updated) {
        scholarshipService.updateScholarship(id, updated);
        return "redirect:/admin/scholarships";
    }

    @PostMapping("/delete-scholarship/{id}")
    public String deleteScholarship(@PathVariable Long id) {
        scholarshipService.deleteScholarship(id);
        return "redirect:/admin/scholarships";
    }
}
```

LOGIN CONTROLLER:

```
package com.admin.scholarship.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.*;

import com.admin.scholarship.util.JwtUtil;

import jakarta.servlet.http.HttpServletResponse;
import jakarta.servlet.http.HttpSession;


@Controller
@CrossOrigin(origins="*")
public class LoginController {

        @Autowired
        private JwtUtil jwtUtil;


    @GetMapping("/")
    public String showLoginPage() {
        return "login";
```

```
    }

    @PostMapping("/login-handler")
    public String loginHandler(@RequestParam String username,
                               @RequestParam String password,
                               @RequestParam String role,
                               HttpServletResponse response,
                               HttpSession session) {


        // Generate token
        String token = jwtUtil.generateToken(role.toUpperCase());

        session.setAttribute("JWT_TOKEN", token);
        session.setAttribute("ROLE", role.toUpperCase());

        if ("ADMIN".equalsIgnoreCase(role)) {
            return "redirect:/admin/dashboard";
        } else {
            return "redirect:/student/home";
        }
    }
}
```

MESSAGE CONTROLLER:

```
package com.admin.scholarship.controller;

import com.admin.scholarship.model.Message;
import com.admin.scholarship.service.MessageService;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;

@Controller
public class MessageController {

    private final MessageService service;

    public MessageController(MessageService service) {
        this.service = service;
    }

    @PostMapping("/contact")
    public String submitMessage(@RequestParam String name,
                                @RequestParam String email,
                                @RequestParam String message) {
        System.out.println("Received: " + name + ", " + email + ", " + message); // DEBUG
```

```
        Message msg = new Message();
        msg.setName(name);
        msg.setEmail(email);
        msg.setMessage(message);
        service.saveMessage(msg);
        return "thank-you";
    }

    @GetMapping("/admin/messages")
    public String viewMessages(Model model) {
        model.addAttribute("messages", service.getAllMessages());
        return "admin/admin-messages"; // redirect after successful submission
    }
}
```

3.REPOSITORY

STUDENT REPOSITORY:

```
package com.admin.scholarship.repository;

import com.admin.scholarship.model.Scholarship;
import com.admin.scholarship.model.Student;


import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;

public interface StudentRepository extends JpaRepository<Student, Integer> {

        Student findByName(String username);
        Student findByEmail(String email);
        int countByStatus(Student.Status status);
        @Query("SELECT SUM(s.amount) FROM Student s")
        Double getTotalSanctionedAmount();
}
```

SCHOLARSHIP REPOSITORY:

```
package com.admin.scholarship.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import com.admin.scholarship.model.Scholarship;

public interface ScholarshipRepository extends JpaRepository<Scholarship, Long> {

        Scholarship findByTitle(String title);

}
```

MESSAGE REPOSITORY:

```
package com.admin.scholarship.repository;

import com.admin.scholarship.model.Message;
import org.springframework.data.jpa.repository.JpaRepository;

public interface MessageRepository extends JpaRepository<Message, Long> {
}
```

4.SERVICE

STUDENT SERVICE:

```
package com.admin.scholarship.service;


import com.admin.scholarship.model.Scholarship;
import com.admin.scholarship.model.Student;
import com.admin.scholarship.repository.ScholarshipRepository;
import com.admin.scholarship.repository.StudentRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import java.util.List;
import java.util.Map;
import java.util.stream.Collectors;

@Service
public class StudentService {

   @Autowired
   private StudentRepository repo;

   public List<Student> getAllStudents() {
      return repo.findAll();
   }

   @Autowired
   private ScholarshipRepository scholarshipRepo;

   public void updateStatus(int id, Student.Status status) {
      Student student = repo.findById(id).orElse(null);
      if (student != null) {
         student.setStatus(status);

         if (status == Student.Status.APPROVED) {
            Scholarship scholarship =
```

```
scholarshipRepo.findByTitle(student.getScholarshiptype());
            if (scholarship != null) {
                student.setAmount(scholarship.getAmount());
            } else {
                student.setAmount(0); // fallback if not found
            }
        } else if (status == Student.Status.REJECTED) {
            student.setAmount(0);
        }

        repo.save(student);
    }
}

public double getTotalSanctionedAmount() {
    Double total = repo.getTotalSanctionedAmount();
    return total != null ? total : 0.0;
}



public String getStatusByName(String studentName) {
    Student student = repo.findByName(studentName);
    return (student != null) ? student.getStatus().toString() : "Student not found";
}

public Student findByName(String name) {
    return repo.findByName(name);
}

public String getStatusByEmail(String email) {
    Student student = repo.findByEmail(email);
    return (student != null) ? student.getStatus().toString() : "Student not found";
}

public Student findByEmail(String email) {
    return repo.findByEmail(email);
}
public int countByStatus(Student.Status status) {
    return repo.countByStatus(status);
}
public Map<String, Long> getScholarshipTypeCounts() {
    return repo.findAll().stream()
        .collect(Collectors.groupingBy(Student::getScholarshiptype, Collectors.counting()));
}

public Map<String, Long> getApprovedScholarshipTypeCounts() {
```

```
        return repo.findAll().stream()
            .filter(s -> s.getStatus() == Student.Status.APPROVED)
            .collect(Collectors.groupingBy(Student::getScholarshiptype, Collectors.counting()));
    }
}
```

SCHOLARSHIP SERVICE:

```
package com.admin.scholarship.service;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.admin.scholarship.model.Scholarship;
import com.admin.scholarship.repository.ScholarshipRepository;
@Service
public class ScholarshipService {
    @Autowired
    private ScholarshipRepository repo;

    public List<Scholarship> getAllScholarships() {
        return repo.findAll();
    }

    public void saveScholarship(Scholarship s) {
        repo.save(s);
    }

    public void updateScholarship(Long id, Scholarship updated) {
        Scholarship existing = repo.findById(id).orElseThrow();
        existing.setTitle(updated.getTitle());
        existing.setAmount(updated.getAmount());
        existing.setEgc1(updated.getEgc1());
        existing.setEgc2(updated.getEgc2());
        existing.setEgc3(updated.getEgc3());
        existing.setEgc4(updated.getEgc4());
        existing.setEgc5(updated.getEgc5());
        repo.save(existing);
    }
    public void deleteScholarship(Long id) {
        repo.deleteById(id);
    }
}
```

MESSAGE SERVICE:

```
package com.admin.scholarship.service;
```

```
import com.admin.scholarship.model.Message;
import com.admin.scholarship.repository.MessageRepository;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class MessageService {

    private final MessageRepository repo;

    public MessageService(MessageRepository repo) {
        this.repo = repo;
    }

    public void saveMessage(Message message) {
        repo.save(message);
    }

    public List<Message> getAllMessages() {
        return repo.findAll();
    }
}
```

5. CONFIG-SPRING SECURITY

SECURITY CONFIG:

```
package com.admin.scholarship.config;

import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.*;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.userdetails.*;
import org.springframework.security.provisioning.InMemoryUserDetailsManager;
import org.springframework.security.web.SecurityFilterChain;
import org.springframework.security.web.authentication.AuthenticationSuccessHandler;

import java.io.IOException;

@Configuration
@EnableWebSecurity
```

```
public class SecurityConfig {

        @Bean
        public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
            http
                .csrf().disable()
                .authorizeHttpRequests()
                    .requestMatchers("/", "/login", "/login-handler", "/css/**", "/js/**",
"/images/**").permitAll()
                    .requestMatchers("/admin/**").hasRole("ADMIN")
                    .requestMatchers("/student/**").hasRole("STUDENT")
                    .requestMatchers("/admin/searchStudent").hasAnyRole("ADMIN",
"STUDENT")
                    .anyRequest().authenticated()
                .and()
                .formLogin()
                    .loginPage("/login")
                    .loginProcessingUrl("/login-handler")
                    .successHandler(customAuthSuccessHandler()) //
                    .permitAll()
                .and()
                .logout()
                    .logoutSuccessUrl("/login?logout")
                    .permitAll();

            return http.build();
        }


    //handler for role-based redirection
    @Bean
    public AuthenticationSuccessHandler customAuthSuccessHandler() {
        return new AuthenticationSuccessHandler() {
            @Override
            public void onAuthenticationSuccess(HttpServletRequest request,
                                                HttpServletResponse response,
                                                Authentication authentication)
                    throws IOException, ServletException {
                if (authentication.getAuthorities().stream()
                        .anyMatch(auth -> auth.getAuthority().equals("ROLE_ADMIN"))) {
                    response.sendRedirect("/admin/dashboard");
                } else if (authentication.getAuthorities().stream()
                        .anyMatch(auth -> auth.getAuthority().equals("ROLE_STUDENT"))) {
                    response.sendRedirect("/student/home");
                } else {
                    response.sendRedirect("/login?error");
                }
```

```
        }
    };
}

@Bean
public UserDetailsService userDetailsService() {
    UserDetails admin = User.withDefaultPasswordEncoder()
        .username("admin")
        .password("admin000")
        .roles("ADMIN")
        .build();

    UserDetails student = User.withDefaultPasswordEncoder()
        .username("student")
        .password("student000")
        .roles("STUDENT")
        .build();

    return new InMemoryUserDetailsManager(admin, student);
}

protected void configure(HttpSecurity http) throws Exception {
    http
        .logout()
            .logoutUrl("/logout")
            .logoutSuccessUrl("/login") // redirects to login.html
            .invalidateHttpSession(true)
            .deleteCookies("JSESSIONID");
}
}
```

6.JWT-UTIL

JwtUtil.java:

```
package com.admin.scholarship.util;

import io.jsonwebtoken.*;
import org.springframework.stereotype.Component;

import java.util.Date;

@Component
public class JwtUtil {
    private final String SECRET_KEY = "mysecretkey";

    public String generateToken(String role) {
```

```
    return Jwts.builder()
            .claim("role", role)
            .setIssuedAt(new Date(System.currentTimeMillis()))
            .setExpiration(new Date(System.currentTimeMillis() + 1000 * 60 * 60)) // 1 hour
            .signWith(SignatureAlgorithm.HS256, SECRET_KEY)
            .compact();
  }

  public String extractRole(String token) {
    return Jwts.parser()
            .setSigningKey(SECRET_KEY)
            .parseClaimsJws(token)
            .getBody()
            .get("role", String.class);
  }

  public boolean isTokenValid(String token) {
    try {
      return !Jwts.parser()
              .setSigningKey(SECRET_KEY)
              .parseClaimsJws(token)
              .getBody()
              .getExpiration()
              .before(new Date());
    } catch (Exception e) {
      return false;
    }
  }
}
```

7.MAIN

SCHOLARSHIP DASHBOARD APPLICATION:

```
package com.admin.scholarship;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class ScholarshipDashboardApplication {

        public static void main(String[] args) {
                SpringApplication.run(ScholarshipDashboardApplication.class, args);
        }

}
```

b. FRONT END

1. home.html:

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  <title>Scholarship Management System</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet" />
  <link href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.10.5/font/bootstrap-icons.css"
rel="stylesheet">
  <style>
    .section { display: none; }
    .active-section { display: block; }
    .nav-link { cursor: pointer; }
    .main-headline {
      font-size: 2.5rem;
      font-weight: bold;
    }
          body {
            font-family: 'Segoe UI', sans-serif;
            background-color: #fff;
            min-height: 100vh;
            overflow-x: hidden;
          }

          .home-bg {
            background: url('/images/schol.jpg') no-repeat;
            background-position: center right -5px;
            background-size: cover;
          }


    .nav-link {
      color: #000;
      font-weight: 500;
    }

    .announcement-title {
      letter-spacing: 2px;
      font-weight: 700;
      margin-bottom: 10px;
      font-size: 1.2rem;
    }
```

```
.announcement-box {
  background: #ffffff;
  border-radius: 20px;
  box-shadow: 0 8px 18px rgba(0, 0, 0, 0.1);
  padding: 12px 20px;
  max-width: 270px;
  margin-top: 10px;
  display: flex;
  align-items: center;
  justify-content: space-between;
}
.btn-checknow, .btn-explore {
  background-color: #000;
  color: #fff;
  margin-top: 10px;
  border-radius: 20px;
  font-weight: 600;
  padding: 6px 16px;
  border-radius: 8px;
  font-size: 0.85rem;
}

.announcement-badge {
  width: 50px;
  height: 50px;
  border-radius: 50%;
  font-size: 1.2rem;
  font-weight: bold;
  display: flex;
  align-items: center;
  justify-content: center;
  margin-right: 10px;
  background-color: black;
  color: white;
  padding: 10px;
}
.announcement-text {
  line-height: 1.1;
  font-size: 0.85rem;
}

.announcement-text .title {
  font-weight: bold;
  font-size: 1rem;
}

 .student-img {
```

```
  opacity: 0.25;
  max-width: 300px;
  margin-bottom: -5px;
  width: 100%;
  border-radius: 12px;
  margin-top: 20px;
 }

 .main-headline {
  font-size: 2.8rem;
  font-weight: 600;
  line-height: 1.3;
 }

 .btn-explore {
  background-color: black;
  color: white;
  font-weight: 600;
  padding: 6px 16px;
  border-radius: 8px;
  font-size: 0.85rem;
 }

 @media (max-width: 768px) {
  .main-headline {
    font-size: 2rem;
  }

  .student-img {
    max-width: 200px;
  }
 }
 .navbar-nav .nav-link {
    position: relative;
    color: #000;
    transition: color 0.3s ease;
  }

  .navbar-nav .nav-link:hover {
    color: #0d6efd;
  }

  .navbar-nav .nav-link::after {
    content: '';
    position: absolute;
    width: 0%;
    height: 2px;
```

```
    left: 0;
    bottom: 0;
    background-color: #0d6efd;
    transition: width 0.3s ease;
  }

  .navbar-nav .nav-link:hover::after {
    width: 100%;
  }

.btn-apply {
  background-color: black;
  color: white;
  border-radius: 50px;
  padding: 8px 20px;
  font-weight: 500;
}

.bg-about {
  color: rgba(255, 255, 255, 0.845);
  text-shadow: 1px 1px 2px black;
}

.justified {
  text-align: justify;
}

#about {
  height: 100vh;
}


#about hr {
  width: 100%;
  border-top: 10px solid #333;
}

#schemes, #contact {
  height: 100vh;
}

#schemes {
  background: url('/images/Scheme_bg.jpg') center/cover no-repeat;
  padding: 0;
  color: white;
  background-position: center -100px;
}
```

```
#contact {
    position: relative;
    background-color: #ffffff;
    color: #000000;
    overflow: hidden;
  }

  .contact-card {
    background-color: #f8f9fa;
    padding: 20px;
    border-radius: 12px;
  }

  .contact-bg-img {
    position: absolute;
    bottom: 0;
    left: 0;
    width: 100%;
    height: 50vh;
    background: url('/images/contact_bg.jpg') no-repeat bottom center;
    background-size: cover;
    z-index: 1;
  }


#schemes h2, #contact h5 {
  z-index: 10;
  position: relative;
  text-align: center;
  padding-top: 30px;
  color: #000;
}

.bg-scheme {
  background: url('/images/Scheme_bg.jpg') no-repeat center center;
  background-size: cover;
  color: white;
  }

#schemes .schemes-section {
  padding: 40px 20px;
  text-align: center;
  color: #000;
  margin-top: 80px;
}
```

```
#schemes .title {
  font-size: 30px;
  color: #000;
  font-weight: 600;
  border-bottom: 2px solid #aa7c7c;
  display: inline-block;
  padding-bottom: 10px;
  margin-bottom: 40px;
}

#schemes .cards-container {
  display: flex;
  justify-content: center;
  flex-wrap: wrap;
  gap: 30px;
  position: relative;
  padding: 0 50px;
}

#schemes .card {
  background-color: rgba(255, 255, 255, 0.95);
  border-radius: 10px;
  width: 280px;
  box-shadow: 0 5px 15px rgba(0, 0, 0, 0.2);
  padding: 25px 20px;
  transition: transform 0.3s ease, box-shadow 0.3s ease;
}

#schemes .card:hover {
  transform: translateY(-10px);
  box-shadow: 0 10px 20px rgba(0, 0, 0, 0.3);
}

#schemes .card h3 {
  font-size: 18px;
  margin-bottom: 10px;
  display: flex;
  justify-content: space-between;
  align-items: center;
}

#schemes .amount {
  font-size: 20px;
  font-weight: 400;
  margin: 10px 0;
}
```

```
    #schemes .amount strong {
      font-size: 32px;
    }

    #schemes .amount span {
      font-size: 14px;
      font-weight: normal;
    }

    #schemes .card ul {
      text-align: left;
      font-size: 14px;
      line-height: 1.7;
      margin: 15px 0;
      padding-left: 20px;
    }

    #schemes .card button {
      background-color: #1a1a1a;
      color: white;
      border: none;
      padding: 10px 25px;
      border-radius: 5px;
      font-size: 15px;
      cursor: pointer;
      transition: background-color 0.3s, transform 0.3s;
    }

    #schemes .card button:hover {
      background-color: rgb(226, 36, 74);
      transform: scale(1.05);
    }
  </style>
</head>
<body class="home-bg">
  <nav class="navbar navbar-expand-lg bg-white py-3 px-4 shadow">
    <div class="container-fluid d-flex justify-content-between align-items-center">
      <div class="d-flex flex-column">
        <span class="fw-bold fs-5">SCHOLARSHIP</span>
        <span class="fw-light fs-6">MANAGEMENT SYSTEM</span>
      </div>
      <ul class="navbar-nav flex-row gap-3 align-items-center">
        <li class="nav-item"><a class="nav-link"
onclick="showSection('home')">Home</a></li>
        <li class="nav-item"><a class="nav-link"
onclick="showSection('about')">About</a></li>
        <li class="nav-item"><a class="nav-link"
```

```
onclick="showSection('schemes')">Schemes</a></li>
      <li class="nav-item"><a class="nav-link"
onclick="showSection('contact')">Contact</a></li>
                <li class="nav-item"><a class="nav-link"
onclick="showSection('status')">Track Status</a></li>
      <li><a th:href="@{/apply}" class="btn btn-apply">Apply Now!</a></li>
      <li class="nav-item dropdown position-relative">
        <a class="nav-link dropdown-toggle d-flex align-items-center" href="#"
id="userDropdown" role="button" data-bs-toggle="dropdown" aria-expanded="false">
          <i class="bi bi-person-circle fs-4"></i>
        </a>
        <ul class="dropdown-menu dropdown-menu-end mt-2 shadow user-dropdown" aria-
labelledby="userDropdown">
          <li><a class="dropdown-item text-danger"  th:href="@{/login}">Logout</a></li>
        </ul>
      </li>

    </ul>
  </div>
 </nav>

 <!-- Home Section -->
 <div id="home" class="section active-section container mt-4">
  <div class="row align-items-start">
   <div class="col-md-5 d-flex flex-column align-items-start">
    <div class="announcement-title">ANNOUNCEMENTS</div>
    <div class="announcement-box d-flex justify-content-between align-items-center">
     <div class="d-flex align-items-center">
      <div class="announcement-badge">M</div>
      <div class="announcement-text">
       <div class="title">Merit 2025</div>
       <div class="text-muted">Merit Scholarship for 2025</div>
      </div>
     </div>
     <button onclick="showSection('schemes')" class="btn btn-checknow">CHECK
NOW</button>
    </div>
    <img th:src="@{/images/stud.jpg}" alt="Students" class="student-img" />
   </div>
   <div class="col-md-7 text-center text-md-start mt-4 mt-md-0">
    <div class="main-headline">
     ALMOST 75% OF<br />
     STUDENTS GET<br />
     SCHOLARSHIPS EVERY<br />
     YEAR.
    </div>
    <button onclick="showSection('about')" class="btn btn-explore">Explore now</button
```

```
>
    </div>
   </div>
  </div>

  <!-- About Section -->
  <div id="about" class="section bg-light py-5">
   <div class="container d-flex flex-wrap align-items-center">
    <div class="col-md-6 justified">
     <h2 class="mb-3 fw-bold">ABOUT US</h2>
     <hr class="mb-4" />

      <p class="justified">Welcome to Scholarship Management System, a platform
dedicated to empowering students and unlocking their true potential. We believe that every
student deserves an opportunity to shine, regardless of their background.</p>
      <p class="justified">Our scholarship portal serves as a comprehensive resource for
merit-based and skill-driven scholarships, connecting talented individuals with opportunities
both within their institution and beyond. Whether you're an academic achiever, a gifted
innovator, or someone with remarkable skills, we are here to help you find scholarships
tailored to your strengths and ambitions.</p>

     <button onclick="showSection('schemes')" class="btn btn-outline-dark mt-3">Explore
More</button>
    </div>
   </div>
  </div>

  <!-- Schemes Section -->

<div id="schemes" class="section bg-scheme text-center">
   <div class="container">

    <!-- Cards Section -->
        <section class="schemes-section mt-5">
          <h2 class="title">SCHEMES</h2>
          <div class="d-flex flex-wrap justify-content-center gap-4">
            <div th:each="scholarship : ${scholarships}" class="card p-3" style="width:
300px;">
              <h3 th:text="${scholarship.title}">Scholarship Title</h3>
              <p class="amount">
                ₹<strong
th:text="${scholarship.amount}">15000</strong><span>/YR</span>
              </p>
              <ul class="text-start">
                <li th:text="${scholarship.egc1}">Eligibility 1</li>
                <li th:text="${scholarship.egc2}">Eligibility 2</li>
                <li th:text="${scholarship.egc3}">Eligibility 3</li>
```

```
                <li th:text="${scholarship.egc4}">Eligibility 4</li>
                <li th:text="${scholarship.egc5}">Eligibility 5</li>
              </ul>
              <button onclick="window.location.href='/apply'" class="btn btn-dark mt-
2">Apply</button>
            </div>
          </div>
        </section>
   </div>
  </div>


  <!-- Contact Section -->
<div id="contact" class="section position-relative">

   <div class="container py-5 position-relative" style="z-index: 2;">
    <h2 class="text-left fw-bold mb-4">CONTACT US</h2>
    <hr class="mb-5" />

    <div class="row justify-content-between">
     <!-- Contact Info -->
     <div class="col-md-5 mb-4">
      <div class="contact-card shadow">
       <h5><strong>Scholarship Management Committee</strong></h5><br>
       <p><i class="bi bi-telephone-fill me-2"></i>+91 9688500598</p>
       <p><i class="bi bi-envelope-fill me-2"></i>smc@gmail.com</p>
       <p><i class="bi bi-globe me-2"></i>www.smcmanagement.com</p>
      </div>
     </div>

     <!-- Contact Form -->
     <div class="col-md-5 col-lg-4">
      <div class="contact-card shadow">
       <form id="contactForm" th:action="@{/contact}" method="post">
        <div class="mb-3">
         <label class="form-label">Name</label>
         <input type="text" name="name" class="form-control" placeholder="Enter your
name" required/>
        </div>
        <div class="mb-3">
         <label class="form-label">Email</label>
         <input type="email" name="email" class="form-control" placeholder="Enter your
email" required/>
        </div>
        <div class="mb-3">
         <label class="form-label">Message</label>
         <textarea name="message" class="form-control" rows="3" placeholder="Type
```

```
your message" required></textarea>
            </div>
            <button type="submit" class="btn btn-dark w-100">Send</button>
          </form>
        </div>
      </div>
    </div>
  </div>

  <div class="contact-bg-img"></div>
 </div>

 <div id="status" class="section bg-light pt-3 pb-2">
    <div class="container d-flex flex-wrap align-items-center">
      <div class="col-md-6 justified">
        <h2 class="mb-3 fw-bold">Track Scholarship Status</h2>
        <hr class="mb-4" />

                    <form id="statusForm">
                       <label for="email">Enter Student Email:</label>
                       <input type="text" id="email" name="email" placeholder="Enter Email"
required />
                       <button type="submit">Search</button>
                    </form>
                    <div id="statusResult">
                       <!-- Status will be displayed here -->
                    </div>


      </div>
    </div>
  </div>
        <footer class="bg-dark text-white text-center py-3 mt-5">
          &copy; 2025 Scholarship Management System. All rights reserved.
        </footer>
 <script>
        function showSection(sectionId) {
            const sections = document.querySelectorAll('.section');
            sections.forEach(sec => {
              sec.classList.remove('active-section');
              sec.style.background = "none";
            });

            const activeSection = document.getElementById(sectionId);
            activeSection.classList.add('active-section');

                if (sectionId === 'home' || sectionId==='status') {
```

```
                    document.body.classList.add('home-bg');
                } else {
                    document.body.classList.remove('home-bg');
                }

            if (sectionId === 'about') {
              activeSection.style.background = 'url("/images/about1.jpg") no-repeat right
center';
              activeSection.style.backgroundSize = "50% auto";
            } else if (sectionId === 'schemes') {
              activeSection.style.background = 'url("/images/Scheme_bg.jpg") center/cover
no-repeat';
              activeSection.style.backgroundSize = "cover";
            }

            window.scrollTo(0, 0);
          }

              document.getElementById('statusForm').addEventListener('submit',
function(e) {
                  e.preventDefault(); // prevent full form submission
                  const email = document.getElementById('email').value; // Change
studentName to email

                  fetch(`/student/searchStudent?email=${encodeURIComponent(email)}`)
                    .then(response => response.text())
                    .then(data => {
                      document.getElementById('statusResult').innerHTML = `<h3>Status
of Application: ${data}</h3>`;
                    })
                    .catch(error => {
                      document.getElementById('statusResult').innerHTML = `<h3
style="color:red;">Error fetching status</h3>`;
                    });
                });



  </script>

  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```

2. dashboard.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org" lang="en">
<head>
  <meta charset="UTF-8">
  <title>Dashboard</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
  <style>
    * { margin: 0; padding: 0; box-sizing: border-box; }

    body {
      font-family: 'Segoe UI', sans-serif;
      background-color: #f4f7fc;
      display: flex;
    }

    .container {
      display: flex;
      width: 100%;
    }

    .sidebar {
      width: 270px;
      background: #2c3e50;
      min-height: 100vh;
      color: white;
      padding: 20px;
    }

    .sidebar h2 {
      text-align: center;
      margin-bottom: 30px;
      font-size: 24px;
    }

    .sidebar ul {
      list-style: none;
    }

    .sidebar ul li {
      margin: 2px 0;
    }

    .sidebar ul li a {
      color: white;
      text-decoration: none;
```

```
  display: block;
  padding: 10px;
  transition: 0.3s;
}

.sidebar ul li a:hover {
  background-color: #1a252f;
  padding-left: 15px;
}

.main-content {
  flex: 1;
  display: flex;
  flex-direction: column;
}

.topbar {
  background: white;
  padding: 15px 20px;
  display: flex;
  justify-content: space-between;
  align-items: center;
  box-shadow: 0 2px 4px rgba(0,0,0,0.1);
}

.topbar input {
  padding: 6px 12px;
  border: 1px solid #ccc;
  border-radius: 4px;
}

.content-area {
  padding: 20px;
  overflow-y: auto;
}

.cards {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(220px, 1fr));
  gap: 20px;
  margin-bottom: 30px;
}

.card {
  background: white;
  padding: 20px;
  border-radius: 8px;
```

```
  box-shadow: 0 2px 6px rgba(0,0,0,0.1);
}

.card h3 {
  color: #555;
  margin-bottom: 10px;
}

.card p {
  font-size: 24px;
  font-weight: bold;
}

.charts {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));
  gap: 20px;
  margin-bottom: 30px;
}

.chart-box {
  background: white;
  padding: 20px;
  border-radius: 8px;
  height: 350px;
}

.table-section {
  background: white;
  padding: 20px;
  border-radius: 8px;
  overflow-x: auto;
}

table {
  width: 100%;
  border-collapse: collapse;
}

th, td {
  text-align: left;
  padding: 10px;
  border-bottom: 1px solid #eee;
}

thead {
  background: #f0f0f0; }
```

```
    .status {
      font-weight: bold;
    }

    .pending { color: orange; }
    .approved { color: green; }
    .rejected { color: red; }
  </style>
</head>
<body>
  <div class="container">
    <!-- Sidebar -->
    <aside class="sidebar">
      <h2>Scholarship Admin</h2>
      <ul>
        <li><a href="/admin/dashboard">🏠 Dashboard</a></li>
        <li><a href="/admin/scholarships">🎓 Scholarships</a></li>
        <li><a href="/admin/applications">📝 Applications</a></li>
        <li><a href="/admin/messages">✉️ Messages</a></li>
        <li><a href="/login">🚪 Logout</a></li>
      </ul>
    </aside>

    <!-- Main content -->
    <div class="main-content">
      <div class="topbar">
        <div>Welcome, Admin □□</div>
        <input type="text" placeholder="Search...">
      </div>
      <div class="content-area">
        <section class="cards">
          <div class="card">
            <h3>Total Applications</h3>
            <p><span th:text="${totalApplications}">0</span></p>
          </div>
          <div class="card">
            <h3>Approved Applications</h3>
            <p><span th:text="${approvedCount}">0</span></p>
          </div>
          <div class="card">
            <h3>Pending Applications</h3>
            <p><span th:text="${pendingCount}">0</span></p>
          </div>
          <div class="card">
            <h3>Total Funds Disbursed</h3>
            <p>₹<span th:text="${#numbers.formatDecimal(totalAmount, 0, 'COMMA', 2,
```

```
'POINT')}">0.00</span></p>
        </div>
      </section>

      <section class="charts">
        <div class="chart-box">
          <h3> Applications by Scholarship Type</h3>
          <canvas id="amountChart"></canvas>
        </div>
        <div class="chart-box">
          <h3> Approved Scholarships by Category</h3>
          <canvas id="categoriesChart"></canvas>
        </div>
      </section>
    </div>
  </div>
</div>

<!-- THYMELEAF DATA BINDING FOR CHART -->
<script th:inline="javascript">
  let typeCounts = [[${typeCounts}]];
  let approvedTypeCounts = [[${approvedTypeCounts}]];
</script>

<script>
  const typeLabels = Object.keys(typeCounts);
  const typeData = Object.values(typeCounts);
  const approvedLabels = Object.keys(approvedTypeCounts);
  const approvedData = Object.values(approvedTypeCounts);

  // Line Chart - Applications by Type
  const ctx1 = document.getElementById('amountChart');
  if (ctx1) {
                  new Chart(ctx1, {
                    type: 'line',
                    data: {
                      labels: typeLabels,
                      datasets: [{
                        label: 'Applications',
                        data: typeData,
                        backgroundColor: 'rgba(54, 162, 235, 0.2)',
                        borderColor: 'rgba(54, 162, 235, 1)',
                        fill: true,
                        tension: 0.4
                      }]
                    },
                    options: {
```

```
                responsive: true,
                scales: {
                  x: {
                    title: {
                      display: true,
                      text: 'Scholarship Types',
                      font: { size: 14 }
                    }
                  },
                  y: {
                    title: {
                      display: true,
                      text: 'Number of Applications',
                      font: { size: 14 }
                    },
                    beginAtZero: true
                  }
                }
              }
            });
  }
  // Pie Chart - Approved by Type
  const ctx2 = document.getElementById('categoriesChart');
  if (ctx2) {
    new Chart(ctx2, {
      type: 'pie',
      data: {
        labels: approvedLabels,
        datasets: [{
          label: 'Approved Applications',
          data: approvedData,
          backgroundColor: [
            'rgba(75, 192, 192, 0.6)',
            'rgba(255, 206, 86, 0.6)',
            'rgba(255, 99, 132, 0.6)',
            'rgba(153, 102, 255, 0.6)',
            'rgba(54, 162, 235, 0.6)'
          ]
        }]
      },
      options: { responsive: true }
    });
  }
 </script>
</body>
</html>
```

APPENDIX-B

OUTPUT SCREENSHOTS

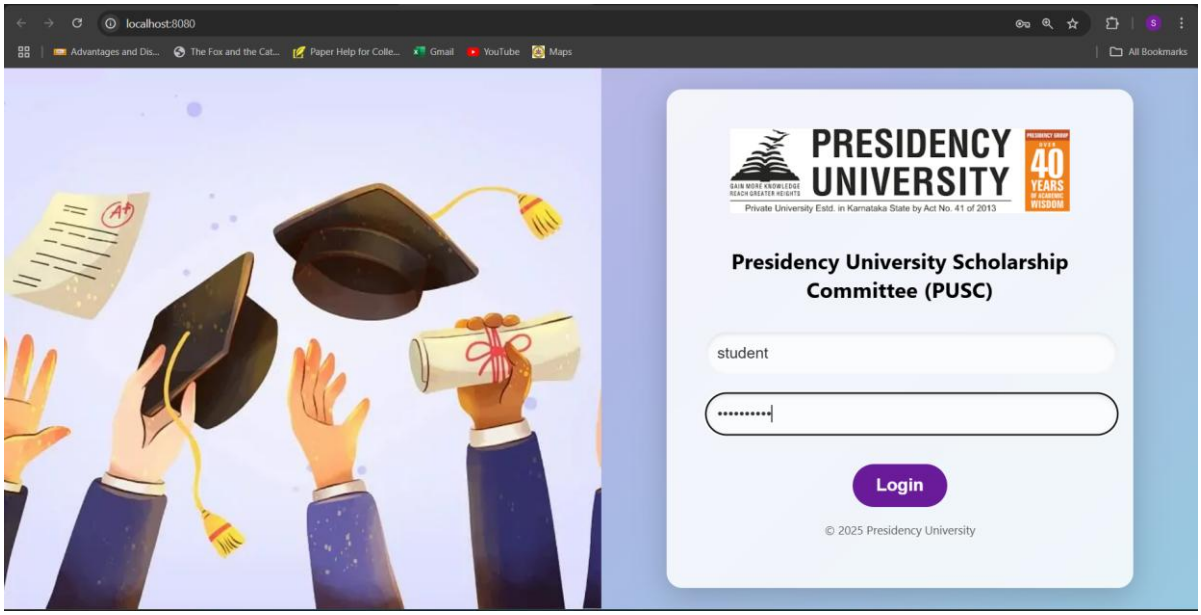


Fig. B (1): Login page

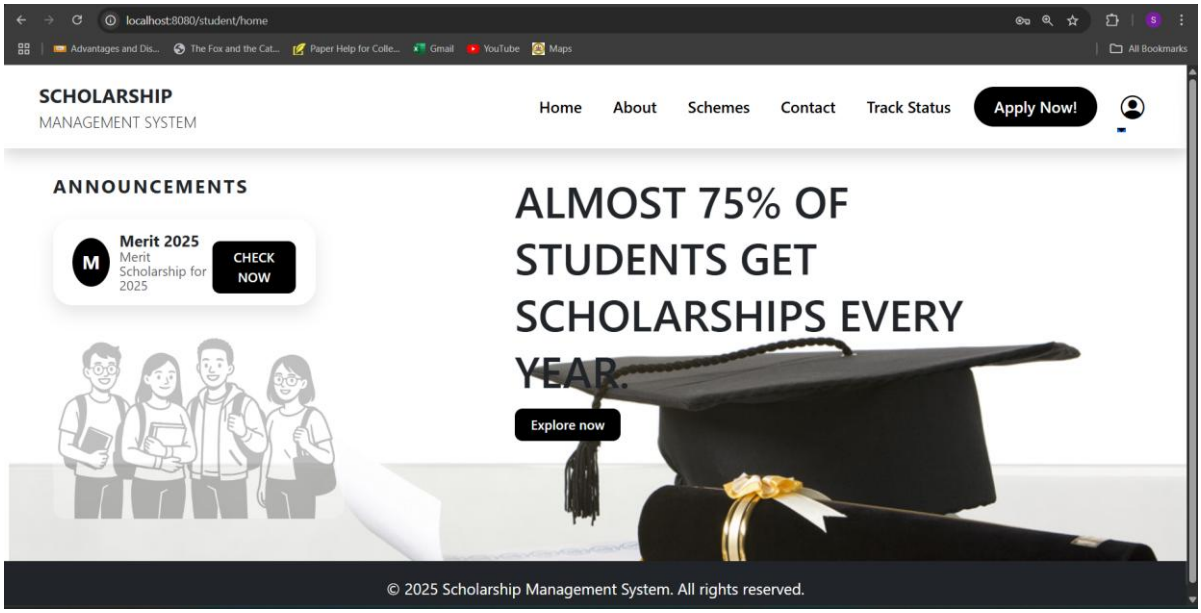


Fig B (2): Home page

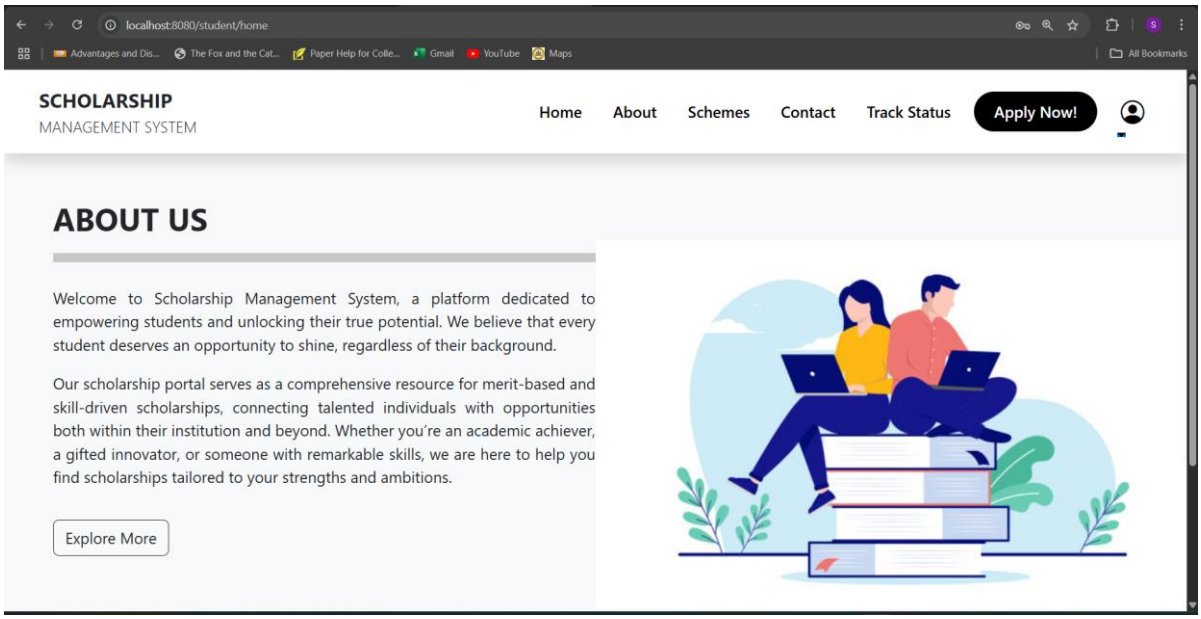


Fig B (3): About Us page

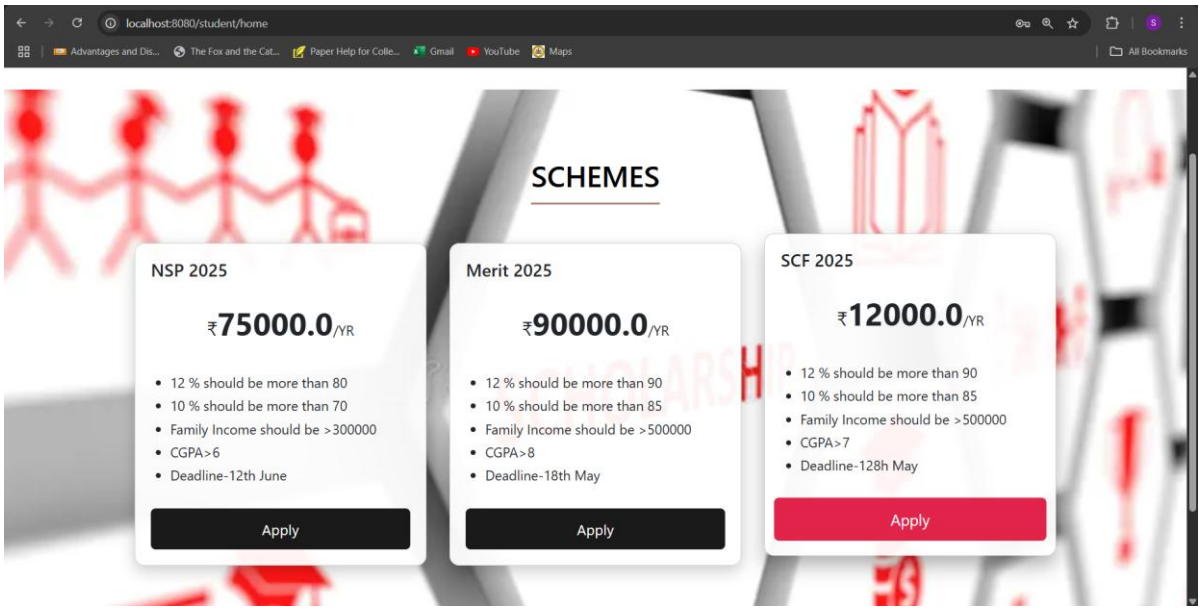


Fig B (4): Schemes page

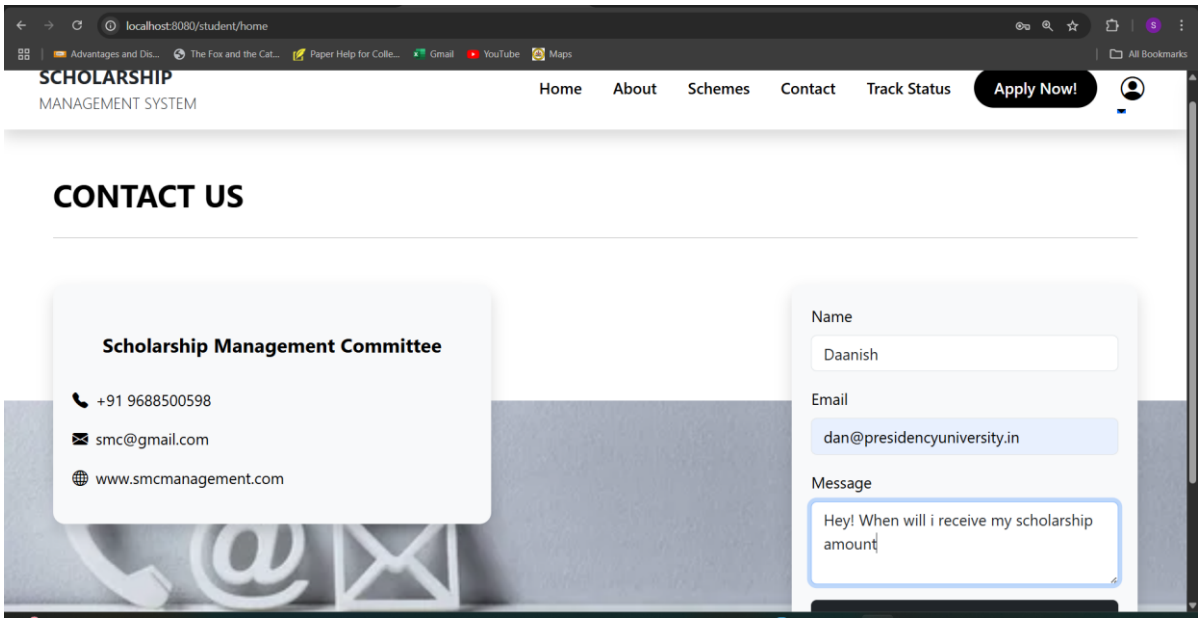


Fig B (5): Contact Us page

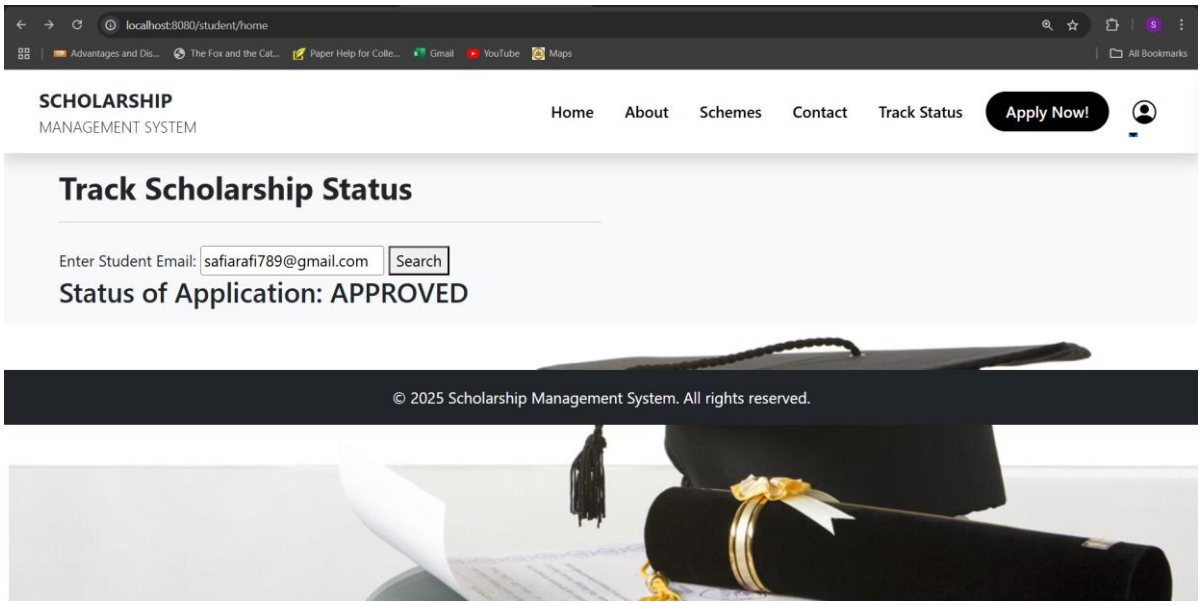


Fig B (6): Track Status page

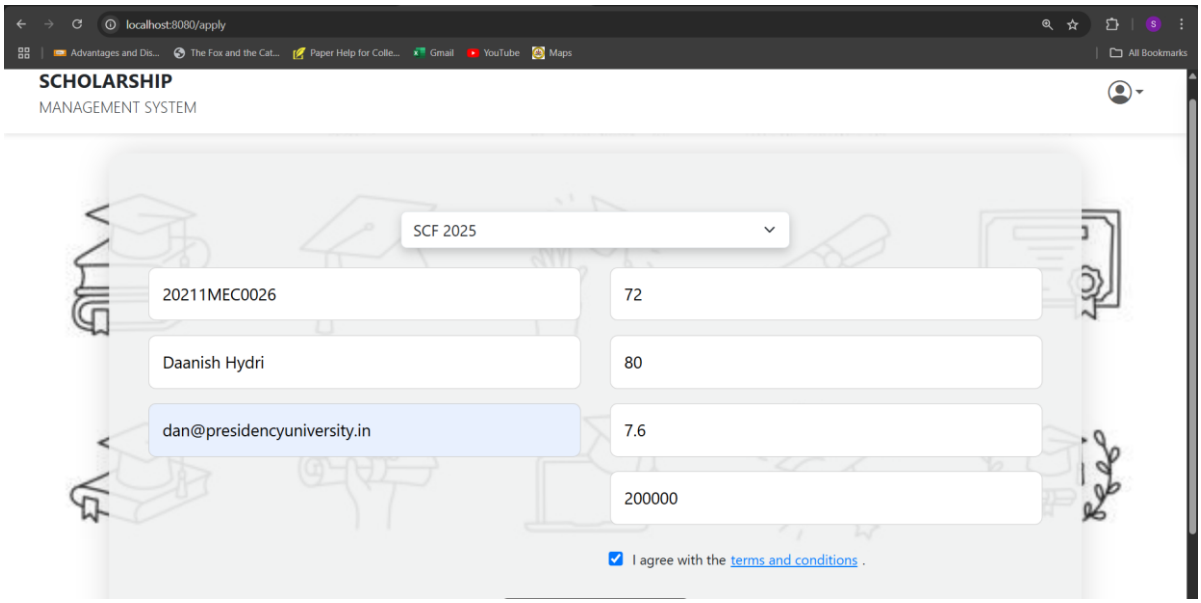


Fig B (7): Application form

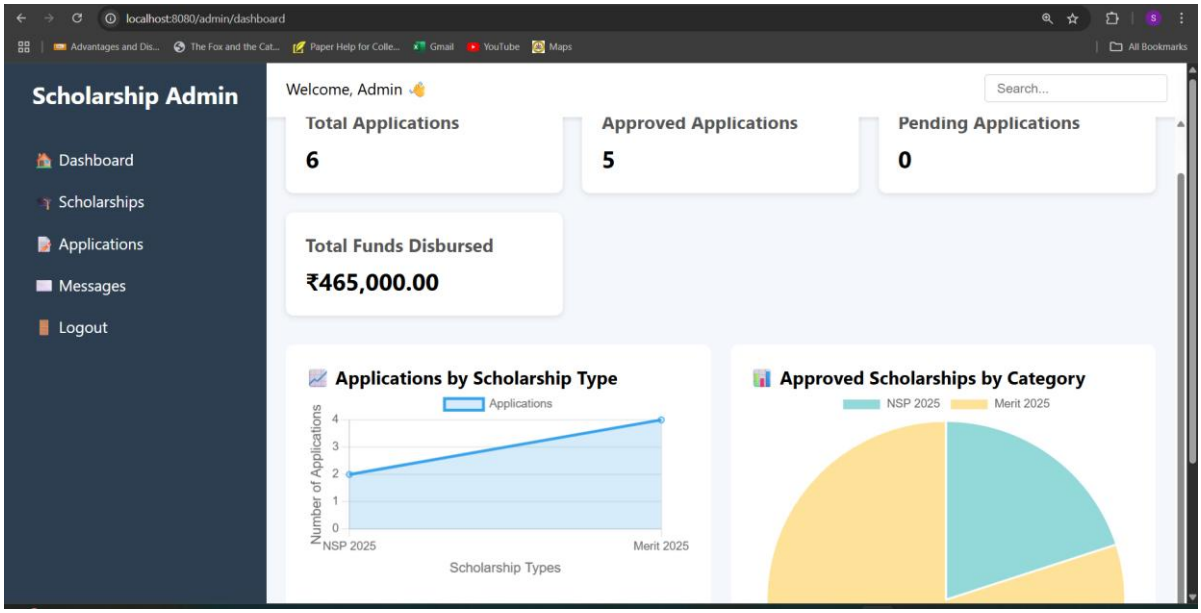


Fig B (8): Admin Dashboard

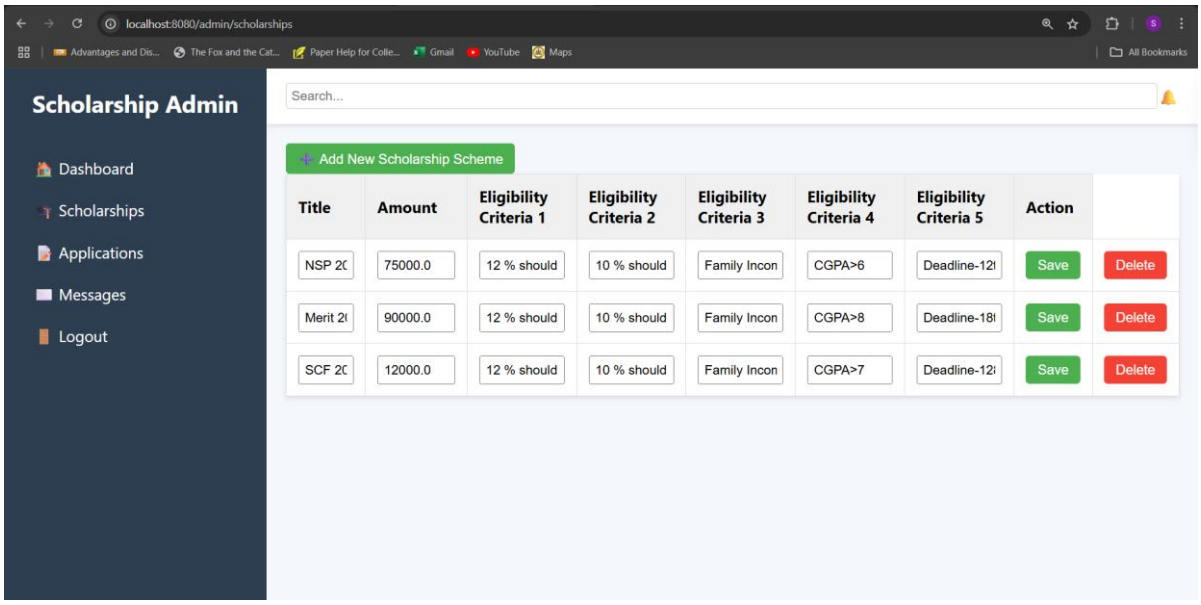


Fig B (9): Scholarship Management Page

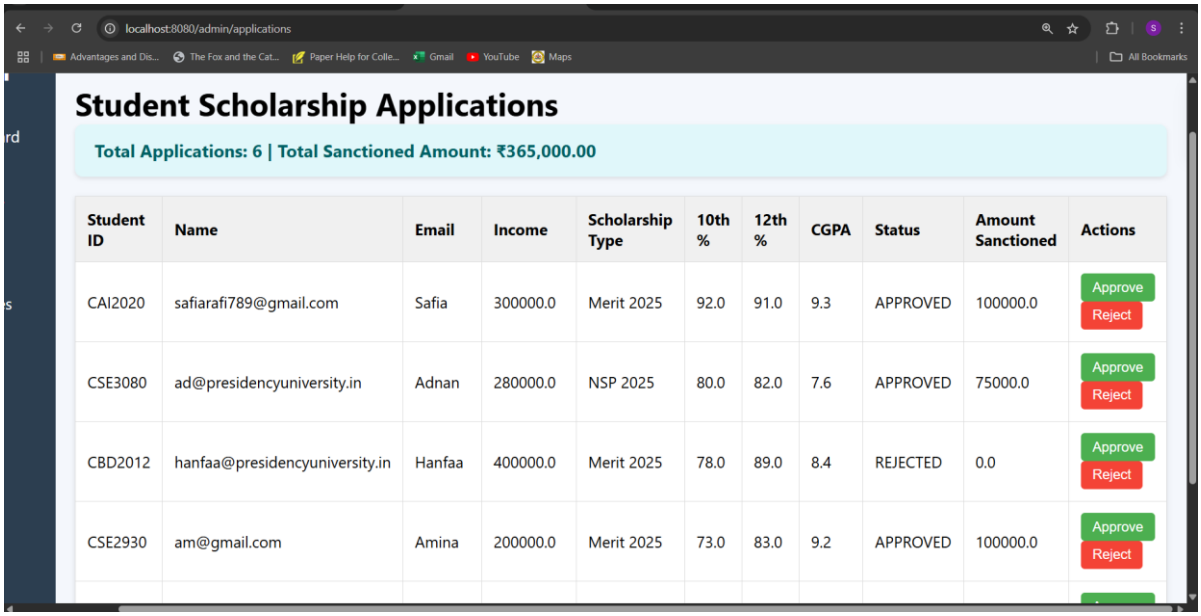


Fig B (10): Applications Management Page

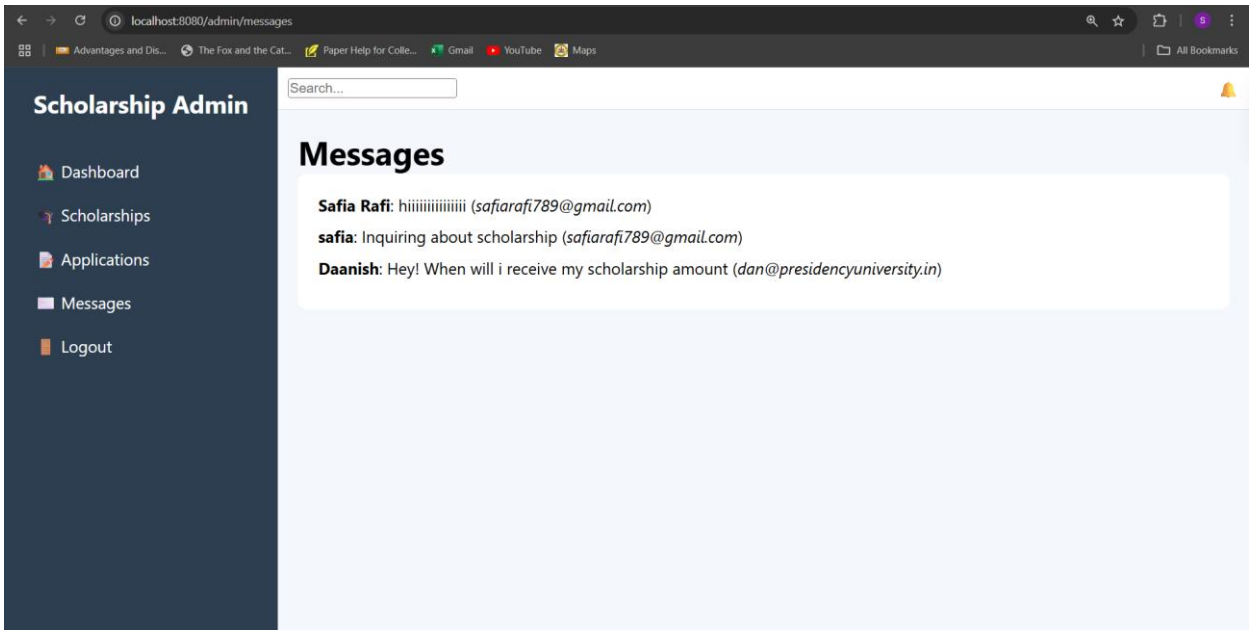
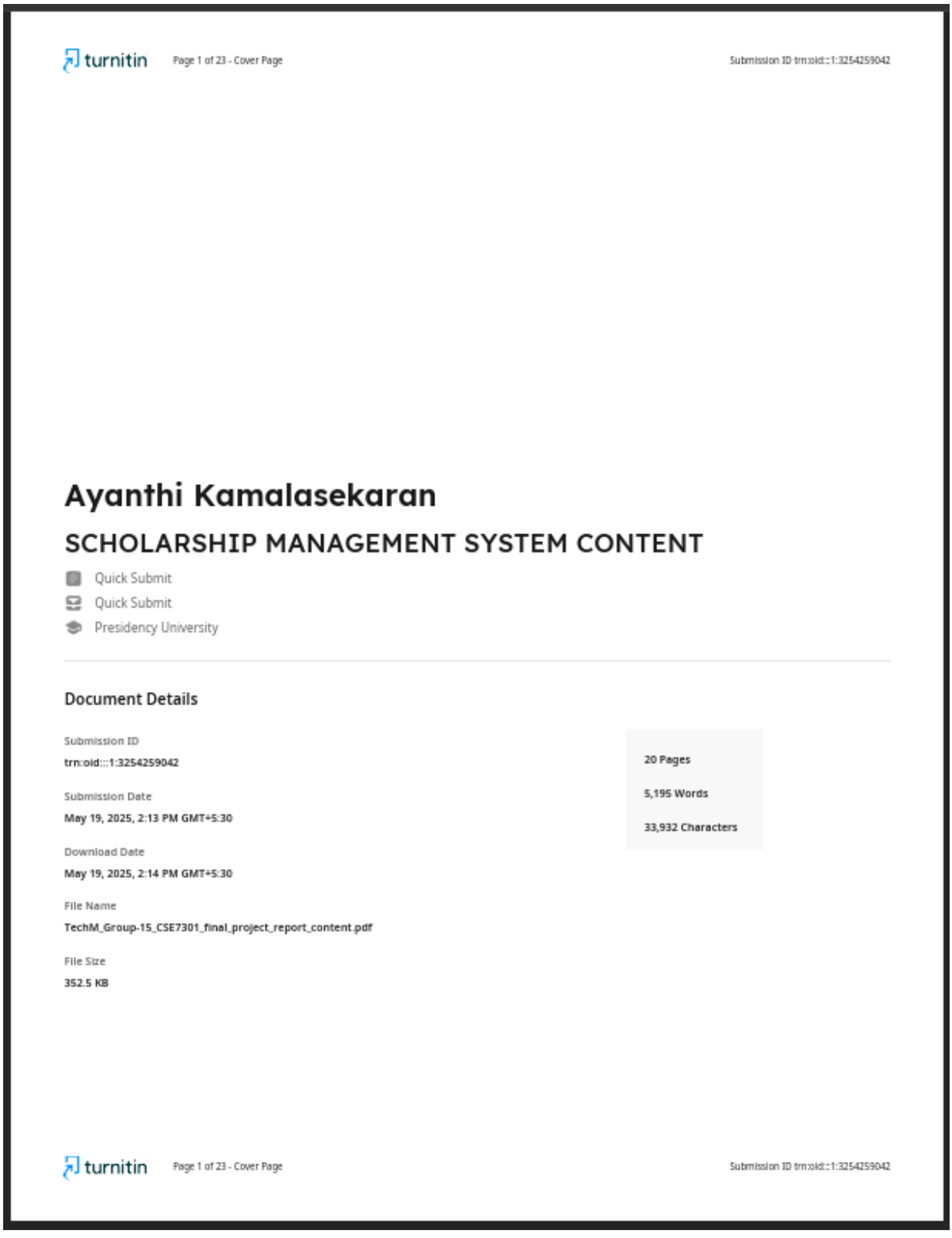


Fig B (11): Message Receiver Page

APPENDIX – C

ENCLOSURES



turnitin Page 1 of 23 - Cover Page Submission ID trn:oid:::1:3254259042

Ayanthi Kamalasekaran

SCHOLARSHIP MANAGEMENT SYSTEM CONTENT

Quick Submit
Quick Submit
Presidency University

Document Details

Submission ID
trn:oid:::1:3254259042

Submission Date
May 19, 2025, 2:13 PM GMT+5:30

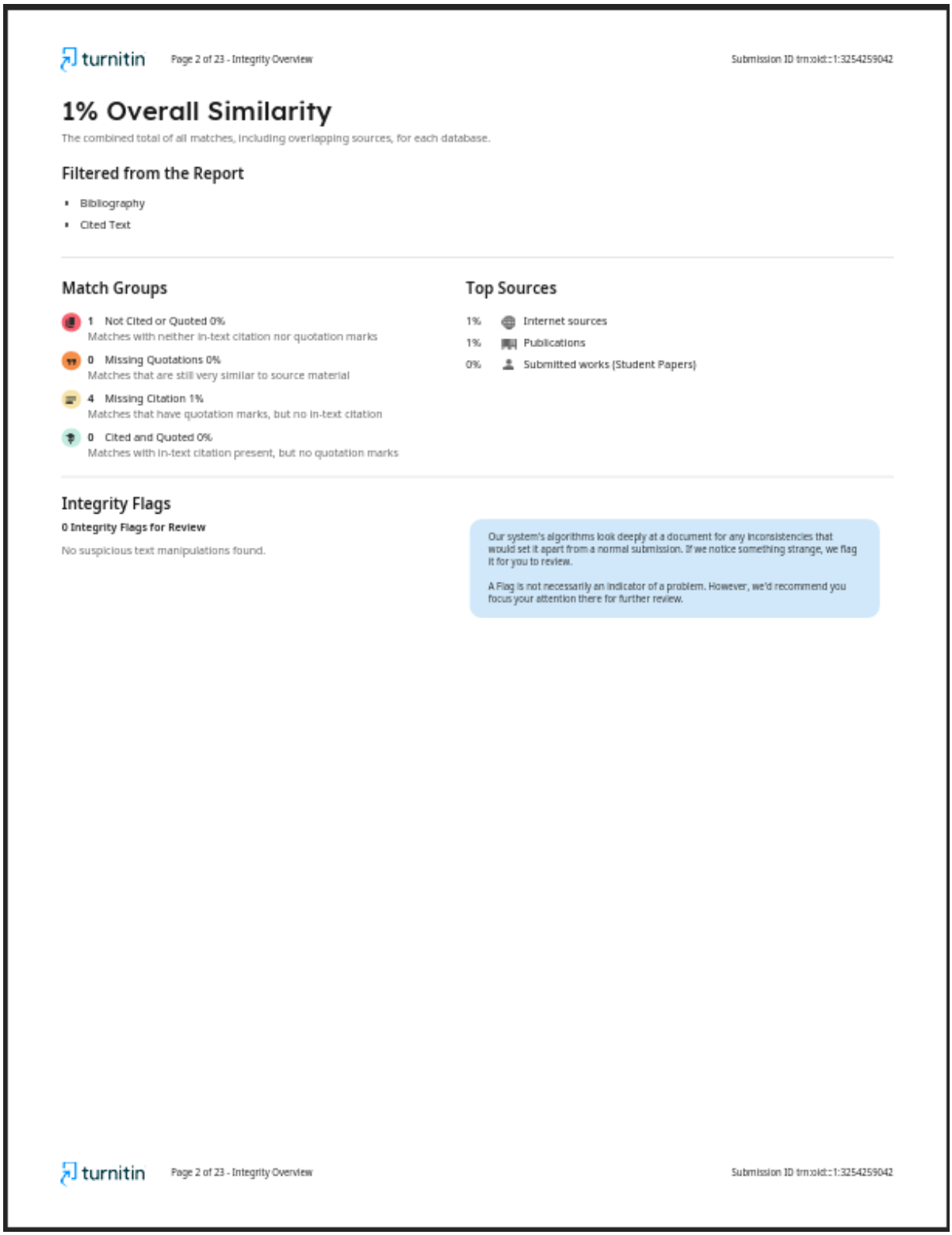
Download Date
May 19, 2025, 2:14 PM GMT+5:30

File Name
TechM_Group-15_CSE7301_final_project_report_content.pdf

File Size
352.5 KB

20 Pages
5,195 Words
33,932 Characters

turnitin Page 1 of 23 - Cover Page Submission ID trn:oid:::1:3254259042



turnitin Page 2 of 23 - Integrity Overview Submission ID trn:oid:::1:3254259042

1% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography
- Cited Text

Match Groups

1 Not Cited or Quoted 0%
Matches with neither in-text citation nor quotation marks

0 Missing Quotations 0%
Matches that are still very similar to source material

4 Missing Citation 1%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

1% Internet sources
1% Publications
0% Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

turnitin Page 2 of 23 - Integrity Overview Submission ID trn:oid:::1:3254259042

SUSTAINABLE DEVELOPMENT GOALS

~SDG 1: No Poverty

Provides low-income students access to scholarship information and application tools, helping reduce financial barriers to education.

~SDG 4: Quality Education

Directly addresses access to affordable education through digital scholarships; enables equitable participation via an inclusive online platform.

~SDG 9: Industry, Innovation, and Infrastructure

Promotes sustainable and resilient digital infrastructure in education

~SDG 10: Reduced Inequalities

Encourages inclusive access for students from rural, minority, or underrepresented groups, thereby reducing inequality in educational opportunities.