**Ehsas-Hub**

|  |
| --- |
|  |

**By:**

**Zain Muneer**

**35937**

**Abdullah shahid**

**35438**

**Hamza Ahmed**

**31967**

**Supervised by:**

**Tajamul Shahzad**

**Faculty of Computing**

**Riphah International University, Islamabad**

**Fall 2024**

**A Dissertation Submitted To**

**Faculty of Computing,**

**Riphah International University, Islamabad**

**As a Partial Fulfillment of the Requirement for the Award of the Degree of**

**Bachelors of Science in Computer Science**

**Faculty of Computing**

**Riphah International University, Islamabad**

Date: December, 2024

**Final Approval**

This is to certify that we have read the report submitted by ***Zain Muneer 35937 Abdullah Shahid 35438 Hamza Ahmed 31967*** for the partial fulfillment of the requirements for the degree of the Bachelors of Science in Computer Science (BSCS). It is our judgment that this report is of sufficient standard to warrant its acceptance by Riphah International University, Islamabad for the degree of Bachelors of Science in Computer Science (BSCS).

**Committee:**

|  |  |
| --- | --- |
| **1** | Tajamul Shahzad  (Supervisor) |
|  |  |
| **2** | Dr. Musharraf Ahmed  (Head of Department) |

**Declaration**

We hereby declare that this document “**Ehsas hub**” neither as a whole nor as a part has been copied out from any source. It is further declared that we have done this project with the accompanied report entirely on the basis of our personal efforts, under the proficient guidance of our teachers, especially our supervisor **Tajumal Shahzad,** if any part of the system is proved to be copied out from any source or found to be reproduction of any project from anywhere else, we shall stand by the consequences.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Zain Muneer**

**35937**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Abdullah Shahid**

**35438**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Hamza Ahmed**

**31967**

**Dedication**

Our project is dedicated to our parents, seniors, friends, and our supervisor "**Tajamul Shahzad**" who has been our continual source of inspiration and whose support has helped this project succeed. This project would not have been possible without their love and support.

**Acknowledgement**

First of all, we are obliged to Allah Almighty the Merciful, the Beneficent and the source of all Knowledge, for granting us the courage and knowledge to complete this Project.

We are deeply grateful to our friends who helped us along the way, our families for their support, and our supervisor, **Tajamul Shahzad**, for his direction.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Zain Muneer**

**35937**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Abdullah Shahid**

**35438**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Hamza Ahmed**

**31967**

**Abstract**

Ehsas Hub is a community-driven platform to connect donors, students, and volunteers toward a common cause: that is, making education accessible to the needy. Unlike most platforms focusing on money, Ehsas Hub is more interested in book-sharing. By doing this, it pairs each donated book with a student who actually needs the book on what they are interested in and what they aim to do in the future. Ehsas Hub, through smart technology, makes book recommendations to each learner to learn and grow, system takes student interest like (Favorite book, author, genres) and provide top rated books. It ensures that all that is done is open and honest so that trust may be built. This doesn't only get the right resources to the right students but empowers them to reach their full potential.

Table of Contents

[Table of Contents i](#_Toc184605584)

[List of Tables iii](#_Toc184605585)

[List of Figures iv](#_Toc184605586)

[Abstract 5](#_Toc184605587)

[Chapter 1: Introduction 6](#_Toc184605588)

[1.1 Goals and Objectives 6](#_Toc184605591)

[1.2 Scope of the Project 7](#_Toc184605594)

[Chapter 2: Literature Review 8](#_Toc184605595)

[2.1 Introduction 8](#_Toc184605596)

[2.2 Background and Problem Elaboration 8](#_Toc184605598)

[2.3 Detailed Literature Review 8](#_Toc184605599)

[2.3.1 Definitions 8](#_Toc184605600)

[2.3.2 Related Research Work 1 8](#_Toc184605601)

[2.3.3 Related Research Work 2 9](#_Toc184605602)

[2.4 Literature Review Summary Table 9](#_Toc184605604)

[2.5 Research Gap 9](#_Toc184605605)

[2.6 Problem Statement 10](#_Toc184605606)

[Chapter 3: Requirements and Design 11](#_Toc184605607)

[3.1 Requirements 11](#_Toc184605608)

[3.1.1 Functional Requirements 11](#_Toc184605609)

[3.1.2 Non-Functional Requirements 14](#_Toc184605610)

[3.1.3 Hardware and Software Requirements 14](#_Toc184605622)

[3.2 Proposed Methodology 15](#_Toc184605623)

[3.3 System Architecture 15](#_Toc184605624)

[3.4 Use Cases 16](#_Toc184605625)

[3.4.1 Login: 20](#_Toc184605626)

[3.4.2 Sign Up: 21](#_Toc184605627)

[3.4.3 Recover password: 21](#_Toc184605628)

[3.4.4 Donate Books: 23](#_Toc184605629)

[3.4.5 Show Donated Books: 24](#_Toc184605630)

[3.4.6 Show Donation Status: 24](#_Toc184605631)

[3.4.7 Manage Account: 25](#_Toc184605632)

[3.4.8 Approved Account: 26](#_Toc184605633)

[3.4.9 Freeze account: 27](#_Toc184605634)

[3.4.10 Active Account: 27](#_Toc184605635)

[3.4.11 Reject Account: 28](#_Toc184605636)

[3.4.12 Completed Request: 29](#_Toc184605637)

[3.4.13 Feedback: 30](#_Toc184605638)

[3.4.14 Edit Profile: 31](#_Toc184605639)

[3.4.15 Enter Interest: 32](#_Toc184605640)

[3.4.16 Request Books: 32](#_Toc184605641)

[3.4.17 Search Books: 33](#_Toc184605642)

[3.4.18 View Recommended Books: 34](#_Toc184605643)

[3.4.19 See book status: 35](#_Toc184605644)

[3.5 Database Design *(Optional)* 36](#_Toc184605645)

[3.6 Class Diagram (*Optional)* 36](#_Toc184605646)

[3.7 Sequence diagram *(Optional)* 36](#_Toc184605647)

[3.8 Any Other Artifact… 36](#_Toc184605648)

[3.9 GUI Graphical User Interfaces (*Optional)* 36](#_Toc184605649)

[Chapter 4: Implementation and Test Cases 37](#_Toc184605650)

[4.1 Implementation 37](#_Toc184605651)

[4.1.1 Implementation of First Component/Algorithm 37](#_Toc184605652)

[4.1.2 Prototype 37](#_Toc184605653)

[4.2 Test case Design and description 37](#_Toc184605654)

[4.2.1 Sample Test case No.1 38](#_Toc184605655)

[4.2.2 Sample Test case No.2 38](#_Toc184605656)

[4.3 Test Metrics 38](#_Toc184605657)

[4.3.1 Sample Test case Matric.No.1 38](#_Toc184605658)

[4.3.2 Sample Test case Metric.No.2 39](#_Toc184605659)

[4.3.3 Sample Test case Metric.No.3 39](#_Toc184605660)

[Chapter 5: Experimental Results and Analysis 40](#_Toc184605661)

[Chapter 6: Conclusion and Future Directions 41](#_Toc184605662)

[References 42](#_Toc184605663)

[Appendix 43](#_Toc184605664)

[Appendix A: Guidelines 43](#_Toc184605665)

[Appendix B: Heading of Sample Appendix B 43](#_Toc184605666)

[Formatting Guidelines 44](#_Toc184605667)

[Chapter 1: Heading 1 44](#_Toc184605668)

[1.1 Heading 2 44](#_Toc184605669)

[1.1.1 Heading 3 44](#_Toc184605670)

[Tables and Figures 45](#_Toc184605671)

[Equations 46](#_Toc184605672)

[Header/Footer 47](#_Toc184605673)

[Other Formatting Guidelines 47](#_Toc184605674)

[References 47](#_Toc184605675)

List of Tables

[Table 1: This is Sample table caption 46](#_Toc55559466)

[Table 2: This is Sample table caption 46](#_Toc55559467)

List of Figures

Figure 1: List of Styles 46

Figure 2: IEEE Reference style 47

# Abstract

Ehsas Hub is a community-driven platform to connect donors, students, and volunteers toward a common cause: that is, making education accessible to the needy. Unlike most platforms focusing on money, Ehsas Hub is more interested in book-sharing. By doing this, it pairs each donated book with a student who actually needs the book on what they are interested in and what they aim to do in the future. Ehsas Hub, through smart technology, makes book recommendations to each learner to learn and grow, system takes student interest like (Favorite book, author, genres) and provide top rated books. It ensures that all that is done is open and honest so that trust may be built. This doesn't only get the right resources to the right students but empowers them to reach their full potential.

# Introduction

Many students today who don't have much money and access to basic learning materials like books, which are important for both learning and personal growth. On the other hand, many people and groups are ready to donate books but don't know how to get in touch with people who need them. This gap can be filled by Ehsas Hub, a digital platform that makes it easy for donors, students, and volunteers to meet. The main goal of this project is to make it easier for people to donate books and make sure that they get to the right people by using a personalized recommendation system that is based on academic interests, and preferences.

Ehsas Hub is more than just a place to donate; it's a step towards making education available to everyone. The platform improves the process of matching given books with people who can use them by adding a recommendation system. This way, every book donated has the chance to improve someone's education. The platform uses technology to get around the problems that come with traditional book donation methods, like matching people with the right books.

## Goals and Objectives

### Goals:

1. **Bridge the Gap Between Donors and Needy:**
2. **Empower Volunteer Coordination:**
3. **Promote Transparency and Reliability:**
4. **Leverage Technology for Social Good:**

### Objectives:

1. **Develop a Comprehensive Donation and Volunteer Management System:**
2. **Provide Intelligent Recommendations for Book Donations:**
3. **Streamline Volunteer Engagement and Task Management:**
4. **Ensure Platform Adaptability and User-Centric Growth:**

## Scope of the Project

**1.2.1Students:** Students from low-income families, who live in orphanages or underprivileged areas for their studies, need educational materials.

**1.2.2 Donors:** People or groups eager to contribute books for the good of the community.

**1.2.3 Volunteers:** Those who want to help with logistics, such picking up and distributing books, in order to support the cause.

**Functional Features:**

* **User Registration and Authentication:** encompass secure login, multi-factor authentication, and the definition of user roles, including student, donor, and volunteer.
* **Profile Management:** Customizable profiles for students (academic interests), donors (donation listings), and volunteers (availability and locations) are provided.
* **Recommendation System:** Proposes books, articles, and novels aligned with students’ academic interests and the highest-rated books and authors.
* **Donation management enables the facilitation of book donations**: Offering options for categorization and presentation in a searchable catalog.
* **Request System:** Facilitates the process for students and institutions to request particular books or genres, aligned with available donations.

# Literature Review

## Introduction

In many areas, like e-commerce, entertainment, and education, recommendation systems are an important part of giving each user a personalized experience. This literature review is mostly about book recommendation systems, which try to match users with good books based on their likes, dislikes, and past actions. This chapter goes into definitions, linked research, and an analysis of methodologies. It then looks for research gaps and comes up with the Ehsas Hub project's problem statement.

## Background and Problem Elaboration

Book recommendation systems have evolved from simple content-based methods to sophisticated hybrid approaches. The challenges addressed by these systems include handling vast datasets, improving recommendation accuracy, and overcoming issues like cold-start problems and sparsity in user feedback. For Ehsas Hub, the aim is to integrate a recommendation engine specifically tailored to students' interests and academic goals, leveraging techniques like collaborative filtering and machine learning.2.3 Detailed Literature Review

## Detailed Literature Review

### Definitions

* **Content-Based Filtering**: Recommends items similar to those the user has liked based on item attributes (e.g., genre, author).
* **Collaborative Filtering**: Makes recommendations by finding similarities among users or items based on user ratings or interactions.
* **Hybrid Systems**: Combines content-based and collaborative methods to overcome the limitations of each technique.

### Related Research Work 1

A study by Gupta et al. (2020) explores the effectiveness of recommendation systems in e-commerce and library platforms. The research highlights the utility of content-based filtering for user-specific recommendations and discusses its limitation in handling new users (cold-start problem). Collaborative filtering, though powerful, requires extensive datasets to deliver accurate predictions.

### Related Research Work 2

## A personalized book recommendation system developed by Sarma et al. (2021) combines clustering techniques with cosine similarity to recommend books. The study uses datasets from Goodreads and applies machine learning models to improve recommendation accuracy. It effectively addresses sparsity and cold-start problems through clustering methods​.

## Literature Review Summary Table

Table 1: History of Computing Devices

The summary of various computing devices invented in the past from 1833-1901 is presented here.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Study | |  | | --- | |  |  |  | | --- | | Methodology | | Strengths | |  | | --- | |  |  |  | | --- | | Limitations | |
| Gupta et al. (2020) | Content-Based Filtering | Personalized recommendations | Struggles with cold-start problems |
| Sarma et al. (2021) | Clustering + Collaborative Filtering | |  | | --- | |  |  |  | | --- | | High accuracy and handles sparsity well | | Requires well-curated datasets |
| Rajpurkar et al. (2015) | Hybrid (Content + Collaborative) | Improves recommendation relevance | Computationally intensive for large datasets |

## Research Gap

Existing systems largely focus on generic book recommendations and often fail to align with specific user goals, such as academic interests. Moreover, while hybrid systems improve accuracy, they introduce higher computational complexity. There is a lack of scalable solutions tailored to nonprofit platforms like Ehsas Hub, which serve diverse user bases including students and donors.

## Problem Statement

The challenge is to design a scalable and efficient book recommendation system for Ehsas Hub that:

1. Personalizes recommendations based on user interests, academic goals, and ratings.
2. Effectively addresses cold-start problems and data sparsity.
3. Operates within the constraints of a nonprofit organization serving varied stakeholders.

# Requirements and Design

In this chapter, we have developed our functional requirements for our actors i.e. (**Needy**, **Donor, Admin** and **Volunteer**). The requirements are designed for especially for Ehsas-Hub platform.

**Ehsas-Hub** is a web-based platform designed to connect or interact with Needy and Donors easily with each other with help of volunteer.

The platform is user-friendly, easy to navigate and search, and provide a convenient and efficient way for both parties to connect and interact with each other.

We created our system **use cases** against each functional requirement and created use case diagrams, fully dressed use cases for our actors i.e. (User, Admin, Donor and Volunteer).

## Requirements

### Functional Requirements

**Needy:**

|  |  |
| --- | --- |
| **ID** | **Requirements** |
| FR-1.1 | User shall be able to Sign Up. |
| FR-1.2 | User shall be able to login to their account. |
| FR-1.3 | User shall be able to Forget/Recover their password. |
| FR-1.4 | User shall be able to complete there profile. |
| FR-1.5 | User shall be able to view profile |
| FR-1.6 | |  | | --- | | User shall be able to edit/update their profile. |  |  | | --- | |  | |
| FR-1.7 | |  | | --- | |  |   User shall be able to View Books Based on Recommendation with respect to their interest. |
| FR-1.8 | User shall be able to request specific books. |
| FR-1.9 | |  |  | | --- | --- | | User shall be able to add book to favourite | . |  |  | | --- | |  | |
| FR-1.10 | User shall be able to See favourite books. |
| FR-1.11 | User shall be able to view book stats. |
| FR-1.12 | User shall be able to view requested books. |
| FR-1.13 | User shall be able to search books. |
| FR-1.14 | User shall be able to give feedback. |
| FR-1.15 | User shall be able to Logout |

**Donor:**

|  |  |
| --- | --- |
| **ID** | **Requirements** |
| FR-2.1 | |  | | --- | | Donor shall be able to Sign Up. |  |  | | --- | |  | |
| FR-2.2 | Donor shall be able to Login. |
| FR-2.3 | Donor shall be able to recover their Password. |
| FR-2.4 | Donor shall be able to donate books |
| FR-2.5 | |  | | --- | |  |  |  | | --- | |  |   Donor shall be able to view donated book stats. |
| FR-2.6 | |  | | --- | | Donor shall be able to view donated book list. |  |  | | --- | |  | |
| FR-2.7 | Donor shall be able to Logout. |

**Volunteer:**

|  |  |
| --- | --- |
| **ID** | **Requirements** |
| FR-3.1 | Volunteer shall be able to sign up. |
| FR-3.2 | Volunteer shall be able to log in. |
| FR-3.3 | |  | | --- | | Volunteer shall be able to forget/recover password. |  |  | | --- | |  | |
| FR-3.4 | |  | | --- | | Volunteer shall be able to view profile. |  |  | | --- | |  | |
| FR-3.5 | Volunteer shall be able to edit profile. |
| FR-3.6 | Volunteer shall be able to check new request. |
| FR-3.7 | Volunteer shall be able to view accept request. |
| FR-3.8 | Volunteer shall be able to check request in process. |
| FR-3.9 | Volunteer shall be able to view completed request. |
| FR-3.10 | Volunteer shall be able to logout. |

**Admin:**

|  |  |
| --- | --- |
| **ID** | **Requirements** |
| FR-4.1 | |  | | --- | | Admin shall be able to sign up. |  |  | | --- | |  | |
| FR-4.2 | |  |  | | --- | --- | | Admin shall be able to login. |  |  |  | | --- | |  | |
| FR-4.3 | |  | | --- | | Admin shall be able to forget/recover password. |  |  | | --- | |  | |
| FR-4.4 | |  | | --- | | Admin shall be able to view profile. |  |  | | --- | |  | |
| FR-4.5 | |  | | --- | | Admin shall be able to edit profile. |  |  | | --- | |  | |
| FR-4.6 | Admin shall be able to manage accounts. |
| FR-4.7 | Admin shall be able to approve account. |
|  | Admin shall be able to reject account. |
|  | Admin shall be able to freeze account. |
|  | Admin shall be able to active account. |
|  | Admin shall be able to view donor request. |
|  | Admin shall be able to accept request. |
|  | Admin shall be able to reject request. |
|  | Admin shall be able to view approved request. |
|  | Admin shall be able to view in process request. |
|  | Admin shall be able to view completed request. |
|  | Admin shall be able to view needy request. |
|  | Admin shall be able to accept request. |
|  | Admin shall be able to reject request. |
|  | Admin shall be able to view needy approved request. |
|  | Admin shall be able to view in process request |
|  | Admin shall be able to view completed request. |
|  | Admin shall be able to view volunteer request. |
|  | Admin shall be able to view approved request. |
|  | Admin shall be able to view completed request. |
|  | Admin shall be able to view account statistics. |
|  | Admin shall be able to |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

### Non-Functional Requirements

# User-Friendly Interface:

# The platform must have a simple and intuitive interface, ensuring all users can navigate effortlessly.

# Security:

# Implement secure authentication mechanisms (e.g., hashed passwords, encrypted data).

# Protect sensitive user data and prevent unauthorized access.

# Performance:

# The platform should ensure a response time of less than 2 seconds for user actions.

# Reliability:

# The system should maintain 99.9% uptime and include robust backup mechanisms.

### Hardware and Software Requirements

**Hardware Requirements:**

**Server**: Dedicated or cloud-based server with at least 16GB RAM and 500GB SSD.

**Storage**: Sufficient storage for books metadata, user data, and logs.

**Processing Power**: Capable of handling concurrent user requests and machine learning tasks.

**Software Requirements:**

**Operating System**: Windows Server.

**Database**: MySQL for storing user profiles, book details, and donation records.

**Frontend**: React.js for building the user interface.

**Backend**: Node.js with Express.js for server-side logic.

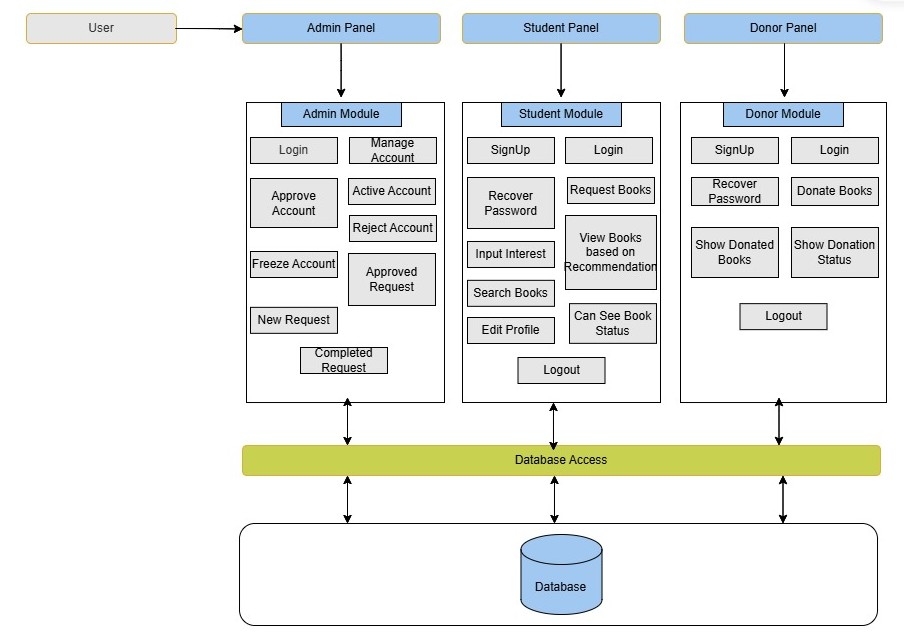
**Recommendation System:** Google Collab.

## Proposed Methodology

The project will follow the **agile methodology**, focusing on iterative development and user feedback:

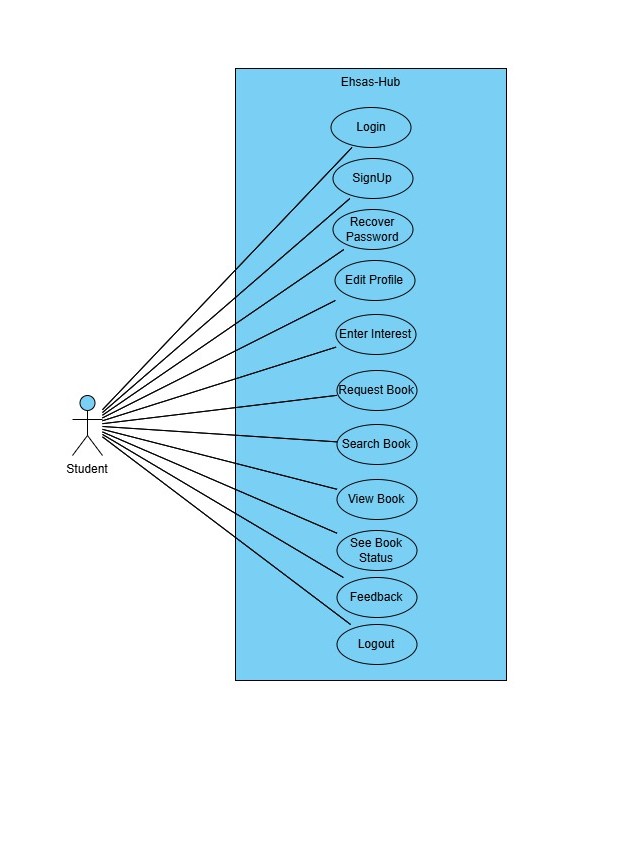
* **Requirement Gathering**: Identify user needs and define functionalities.
* **System Design**: Develop architecture and UI mockups.
* **Development**: Build core modules, including registration, donation management, and recommendation system.
* **Testing**: Validate functionality, performance, and security.

## System Architecture

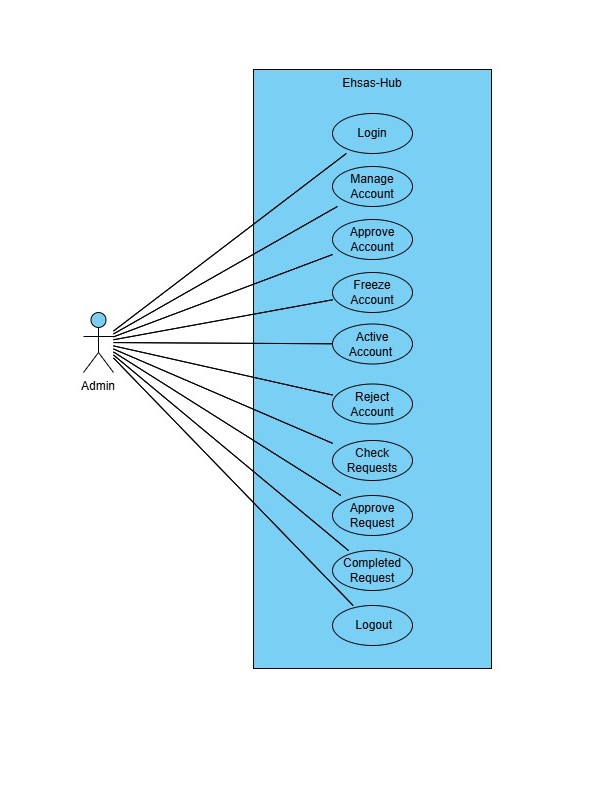


## Use Cases

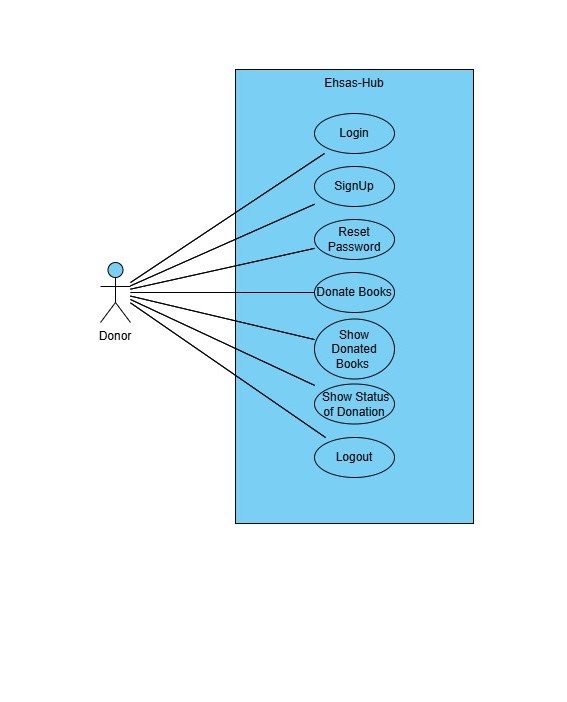
**Student Use-case Diagram:**

****

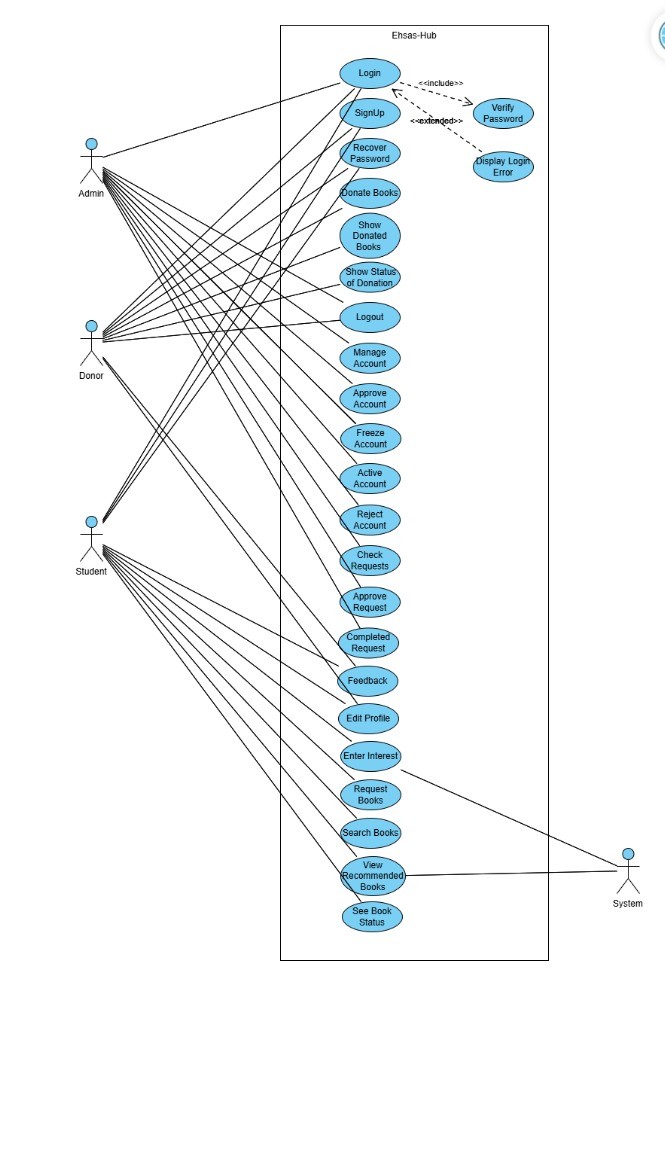
**Admin Use-Case Diagram:**

****

**Donor Use-Case Diagram:**

****

**Full System Use-Case Diagram:**

****

**Fully Dressed Use Cases:**

### Login:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | | Login | | |
| **Actors** | | Admin, Student, Donor | | |
| **Summary** | | The user shall provide their email and password on the login form and after successful verification, redirect the user to the home page. | | |
| **Pre-**  **Conditions** | | User must be registered on the System. | | |
| **Post-**  **Conditions** | | Users are Authenticated and redirected to their dashboard | | |
| **Special**  **Requirements** | | None | | |
| **Basic Flow** | | | | |
| **Actor Action** | | | **System Response** | |
| 1 | The user opens the login page. | | 2 | The login page is displayed asking for email and password. |
| 3 | The user enters valid email and password. | | 4 | The system verifies the email and password, establishes a session for the user and redirects the user to their dashboard |
| **Alternative Flow** | | | | |
| 3 | The user enters invalid email or password. | | 4-A | The system responds with an error **message**: *Incorrect email or password entered.* |

### Sign Up:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | | Sign up | | |
| **Actors** | | Donor Student, Admin | | |
| **Summary** | | The user creates a new account by providing necessary information. | | |
| **Pre-**  **Conditions** | | None | | |
| **Post-**  **Conditions** | | The user successfully creates a new account and can log in to the system. | | |
| **Special**  **Requirements** | | None | | |
| **Basic Flow** | | | | |
| **Actor Action** | | | **System Response** | |
| 1 | The user opens the sign-up page. | | 2 | The sign-up page is displayed, asking for the user's information such as name, email, and password. |
| 3 | The user enters their name, email, and password. | |  |  |
| 4 | The user submits the sign-up form. | | 5 | The system verifies the provided information and creates a new account for the user. |
|  |  | | 6 | The system displays a success message, informing the user that their account has been created. |
| **Alternative Flow** | | | | |
| 4.1 | If the user submits the sign-up form with incomplete or invalid information. | | 4.2 | The system responds with an *error message*, indicating the specific fields that need to be corrected. |

### Recover password:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | | Recover password | | |
| **Actors** | | Student, Donor, Admin | | |
| **Summary** | | This use case describes the process of recovering a password for a user who has forgotten or lost their login credentials. | | |
| **Pre-**  **Conditions** | | -The user must have a registered account on the Ehsas-Hub platform. -The user must have forgotten or lost their password. | | |
| **Post-**  **Conditions** | | The user's password is successfully reset, and they can log in to their account using the new password. | | |
| **Special**  **Requirements** | | None | | |
| **Basic Flow** | | | | |
| **Actor Action** | | | **System Response** | |
| 1 | User navigates to the Ehsas-Hub login page. | |  |  |
| 2 | User clicks on the "Forgot Password" link. | | 3 | The system presents a password recovery form. |
| 4 | User enters their registered email address in the provided field. | |  |  |
| 5 | User clicks on the "Submit" button. | | 6 | The system verifies the email address and generates a unique password reset code. |
|  |  | | 7 | The system sends an email to the user's registered email address containing a password reset code. |
| 8 | User checks their email inbox finds the password reset code. | | 9 | The user enters the password reset code and new password field occurs. |
| 10 | User enters a new password and confirms it in the provided fields. | |  |  |
| 11 | User submits the password reset form. | | 12 | The system verifies the new password and updates it in the user's account. |
|  |  | | 13 | The system displays a confirmation message indicating that the password has been successfully reset. |
|  |  | |  |  |
| **Alternative Flow** | | | | |
| 4.1 | If the user enters an invalid or unregistered email address. | | 4.2 | The system displays an error message and prompts the user to enter a valid email address. |
|  |  | | 4.3 | If the password reset code has expired or become invalid, the system prompts the user to initiate the password recovery process again. |

### Donate Books:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | | Donate Books | | |
| **Actors** | | Donor | | |
| **Summary** | | Donor can list books they wish to donate. | | |
| **Pre-**  **Conditions** | | The user must have access to the Ehsas-Hub website.  The user must be registered on the Ehsas-Hub platform. | | |
| **Post-**  **Conditions** | | Books are added to donation catalog. | | |
| **Special**  **Requirements** | | None | | |
| **Basic Flow** | | | | |
| **Actor Action** | | | **System Response** | |
| 1 | The Donor selects to “Donate Books”. | | 2 | The system displays the Donation form. |
| 3 | The Donor Provides Books details and submit. | | 4 | The system validates and save the donation. |
|  |  | | 5 | The System displays the “Donation Success Message”. |
| **Alternative Flow** | | | | |
| 3.1 | Incomplete details from donor. | | 3.2 | Missing fields message will be highlighted by the system. |

### Show Donated Books:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | | Show Donated Books | | |
| **Actors** | | Donor | | |
| **Summary** | | Donated books list can be viewed by the donor | | |
| **Pre-**  **Conditions** | | -The Donor must have a registered account on the Ehsas-Hub platform.  -The Donor must be logged in on the Ehsas-Hub platform. | | |
| **Post-**  **Conditions** | | Donated books list is displayed | | |
| **Special**  **Requirements** | | None | | |
| **Basic Flow** | | | | |
| **Actor Action** | | | **System Response** | |
| 1 | The Donor Selects “Show Donated Books”. | | 2 | Donated Book list is displayed by the System |
|  | **Alternative Flow** | | | |
|  |  | |  |  |

### Show Donation Status:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | | Donation Status | | |
| **Actors** | | Donor | | |
| **Summary** | | Donor can track the status of their donated books. | | |
| **Pre-**  **Conditions** | | * The user must have access to the Ehsas-Hub website. * The user must be logged in to their Ehsas-Hub account. | | |
| **Post-**  **Conditions** | | Donation status is displayed (e.g. pending, Delievered ). | | |
| **Special**  **Requirements** | | None | | |
| **Basic Flow** | | | | |
| **Actor Action** | | | **System Response** | |
| 1 | Donor selects “Donation status” | | 2 | Donation status is displayed by the system |
| **Alternative Flow** | | | | |
|  |  | |  |  |

### Manage Account:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | | Manage Account | | |
| **Actors** | | Admin | | |
| **Summary** | | Admin manage Users Account by Approving, Rejecting, Freezing or making them Active. | | |
| **Pre-**  **Conditions** | | The user must be logged in to their Ehsas-Hub Platform. | | |
| **Post-**  **Conditions** | | Account status is updated | | |
| **Special**  **Requirements** | | None | | |
| **Basic Flow** | | | | |
| **Actor Action** | | | **System Response** | |
| 1 | The admin selects Manage account | | 2 | User Accounts are displayed |
| 3 | The admin selects an account and updated its status | | 4 | The system performs the action and updates the account status. |
|  |  | | 5 | The System notifies the selected user about the account status changed. |
| **Alternative Flow** | | | | |
| 2.1 | The system finds no account for action | | 2.2 | “No accounts to manage” message is displayed by the system. |

### Approved Account:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | | Approved Account | | |
| **Actors** | | Admin | | |
| **Summary** | | Admin Approves the pending User account, allowing the user to access the system. | | |
| **Pre-**  **Conditions** | | The user must be logged in to their Ehsas-Hub account. | | |
| **Post-**  **Conditions** | | The Account is Approved and the user is notified of the approval. | | |
| **Special**  **Requirements** | | None | | |
| **Basic Flow** | | | | |
| **Actor Action** | | | **System Response** | |
| 1 | Admin selects “manage account” page. | | 2 | The system displays the list of pending books. |
| 3 | The admin selects the account and clicks approve. | | 4 | The system updates the account status to “Approved” and notifies the user. |
| **Alternative Flow** | | | | |
| 3.1 | The admin selects an invalid account. | | 3.2 | Error is displayed by the system. |

### Freeze account:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | | Freeze Account | | |
| **Actors** | | Admin | | |
| **Summary** | | Admin freezes a user account , temporarily block. | | |
| **Pre-**  **Conditions** | | Admin must be logged in. | | |
| **Post-**  **Conditions** | | The account is frozen and notified to user. | | |
| **Special**  **Requirements** | | None | | |
| **Basic Flow** | | | | |
| **Actor Action** | | | **System Response** | |
| 1 | Admin select “Manage account” page. | | 2 | Active Accounts are displayed by the system. |
| 3 | The admin selects an account and click freeze | | 4 | Account status is updated to frozen and notified to user by system |
| **Alternative Flow** | | | | |
| 3.1 | The admin selects an invalid account | | **3.2** | Error is displayed by the system |

### Active Account:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | | Active Account | | |
| **Actors** | | Admin | | |
| **Summary** | | Admin reactivates a previously frozen user account. | | |
| **Pre-Conditions** | | Admin must be logged in. | | |
| **Post-Conditions** | | The account is activated, and the user is notified of the activation. | | |
| **Special Requirements** | | None | | |
| **Basic Flow** | | | | |
| **Actor Action** | | | **System Response** | |
| 1 | The admin navigates to the "Manage Accounts" page. | | 2 | The system displays a list of frozen accounts. |
| 3 | The admin selects an account and clicks "Activate." | | 4 | The system updates the account status to "Active" and notifies the user. |
| **Alternative Flow** | | | | |
| 3.1 | The admin selects an invalid account. | | 3.2 | The system displays an error: "Unable to activate account." |

### Reject Account:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | | Reject Account | | |
| **Actors** | | Admin | | |
| **Summary** | | Admin rejects a pending user account, preventing the user from accessing the system. | | |
| **Pre-Conditions** | | Admin must be logged in. | | |
| **Post-Conditions** | | The account is rejected, and the user is notified of the rejection. | | |
| **Special Requirements** | | None | | |
| **Basic Flow** | | | | |
| **Actor Action** | | | **System Response** | |
| 1 | The admin navigates to the "Manage Accounts" page. | | 2 | The system displays a list of pending accounts. |
| 3 | The admin selects an account and clicks "Reject." | | 4 | The system updates the account status to "Rejected" and notifies the user. |
| **Alternative Flow** | | | | |
| 3.1 | The admin selects an invalid account. | | 3.2 | The system displays an error: "Unable to reject account." |

### Completed Request:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | | Completed Request | | |
| **Actors** | | Admin | | |
| **Summary** | | Admin marks a donation or request task as completed after it has been fulfilled. | | |
| **Pre-Conditions** | | The request must have a status of Approved or In Progress. | | |
| **Post-Conditions** | | - The request status is updated to Completed. | | |
| **Special Requirement** | | None | | |
|  | |  | | |
| **Basic Flow** | | | | |
| **Actor Action** | | | **System Response** | |
| 1 | The actor (Admin) navigates to the "Requests" page. | | 2 | The system displays a list of requests with their current statuses (e.g., Pending, Approved, In Progress). |
| 3 | The actor selects a specific request with a status of In Progress and clicks "Mark as Completed." | | 4 | The system updates the request status to Completed. |
|  |  | | 5 | The system notifies all relevant users (e.g., Donor) about the completion. |
|  | | **Alternative Flow** | | |
| 3.1 | The actor selects a request that cannot be completed (e.g., still Pending or already Completed). | | 3.2 | The system displays an error: "This request cannot be marked as completed at this stage." |

### Feedback:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | | Feedback | | | |
| **Actors** | | Student, Donor | | | |
| **Summary** | | Allows users (Student or Donor) to provide feedback about the request or donation process. | | | |
| **Pre-Conditions** | | The user must have an active account and be logged in. | | | |
| **Post-Conditions** | | The feedback is stored in the system and visible to the admin for review and analysis. | | | |
| **Special Requirement** | | None | | | |
| **Basic Flow** | | | | | |
| **Actor Action** | | | **System Response** | | |
| 1 | The user navigates to the "Feedback" section. | | | 2 | The system displays a feedback form. |
| 3 | The user fills out the feedback form, including a rating (e.g., 1–5 stars) and optional comments. | | | 4 | The system validates the input and saves the feedback |
|  |  | | | 5 | The system displays a confirmation message: "Thank you for your feedback." |
| **Alternative Flow** | | | | | |
| 3.1 | The user submits an incomplete or invalid feedback form. | | | 3.2 | The system highlights the missing fields or errors and prompts the user to correct them. |
|  |  | | |  |  |

### Edit Profile:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | | Edit Profile | | | |
| **Actors** | | Student, Donor | | | |
| **Summary** | | Allows users to update their profile information, such as name, email, or other details. | | | |
| **Pre-Conditions** | | The user must be logged in. | | | |
| **Post-Conditions** | | The updated profile information is saved in the system. | | | |
| **Special Requirements** | | None | | | |
| **Basic Flow** | | | | | |
| **Actor Action** | | | **System Response** | | |
| 1 | The user navigates to the "Profile" page. | | | 2 | The system displays the user's current profile information. |
| 3 | The user edits the desired fields and clicks "Save." | | | 4 | The system validates and updates the profile information. |
|  |  | | | 5 | The system displays a success message: "Profile updated successfully."  Alternative Flow |
| **Alternative Flow** | | | | | |
| 3.1 | The user enters invalid information (e.g., incorrect email format). | | | **3.2** | The system displays an error message and prompts the user to correct the input. |
|  |  | | |  |  |

### Enter Interest:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | | Enter Interest | | | |
| **Actors** | | Student | | | |
| **Summary** | | Students specify their interests (e.g., genres, subjects, future career paths) to tailor book recommendations. | | | |
| **Pre-Conditions** | | The user must be logged in. | | | |
| **Post-Conditions** | | Interests are saved in the system and used for personalized recommendations. | | | |
| **Special Requirements** | | None | | | |
| **Basic Flow** | | | | | |
| **Actor Action** | | | **System Response** | | |
| 1 | The student navigates to the "Enter Interests" section in their profile. | | | 2 | The system displays a list of interest categories (e.g., genres, subjects). |
| 3 | The student selects or enters their interests and clicks "Save." | | | 4 | The system validates and saves the interests. |
|  |  | | | 5 | The system confirms: "Your interests have been saved successfully." |
|  | **Alternative Flow** | | | | |
| 3.1 | The student submits incomplete or invalid entries | | | 3.2 | The system highlights the errors and requests corrections. |
|  |  | | |  |  |

### Request Books:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | | Request Books | | |
| **Actors** | | Student | | |
| **Summary** | | Students request specific books or genres based on their needs. | | |
| **Pre-Conditions** | | The user must be logged in. | | |
| **Post-Conditions** | | The request is submitted and added to the system for matching with available donations. | | |
| **Special Requirements** | | None | | |
| **Basic Flow** | | | | |
| **Actor Action** | | | **System Response** | |
| 1 | The student navigates to the "Request Books" section. | | 2 | The system displays a form for book requests. |
| 3 | The student fills out the form with book details (e.g., title, genre) and clicks "Submit." | | 4 | The system validates and saves the request. |
|  |  | | 5 | The system displays a confirmation: "Your book request has been submitted." |
|  | **Alternative Flow** | | | |
| 3.1 | The student submits an incomplete or invalid request | | 3.2 | The system highlights missing or incorrect fields and prompts corrections. |

### Search Books:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | | Search Books | | |
| **Actors** | | Student | | |
| **Summary** | | Students search for books available for donation using keywords or filters. | | |
| **Pre-Conditions** | | The user must be logged in. | | |
| **Post-Conditions** | | Relevant search results are displayed to the user. | | |
| **Special Requirements** | | None | | |
| **Basic Flow** | | | | |
| **Actor Action** | | | **System Response** | |
| 1 | The student navigates to the "Search Books" section. | | 2 | The system displays a search bar and optional filters (e.g., genre, author). |
| 3 | The student enters a keyword or applies filters and clicks "Search." | | 4 | The system retrieves and displays matching books. |
|  |  | | 5 | The student selects a book to view its details. |
|  | **Alternative Flow** | | | |
| 3.1 | The student enters a query with no matches. | | 3.2 | The system displays a message: "No books found. Please refine your search." |

### View Recommended Books:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | | View Recommended Books | | |
| **Actors** | | Student | | |
| **Summary** | | The system recommends books to students based on their interests, popularity, and ratings. | | |
| **Pre-Conditions** | | The student must be logged in.  The student must have provided their interests in their profile. | | |
| **Post-Conditions** | | The recommended books are displayed to the student. | | |
| **Special Requirements** | | None | | |
| **Basic Flow** | | | | |
| **Actor Action** | | | **System Response** | |
| 1 | The student navigates to the "Recommended Books" section. | | 2 | The system retrieves books based on the student’s interests, popular genres, and ratings. |
| 3 | The system displays a list of recommended books with metadata (e.g., title, author, rating). | | 4 | The student selects a book to view more details. |
| **Alternative Flow** | | | | |
| The student has not provided any interests. | | | **2.2** | The system displays a message: "No recommendations available. Please update your interests in your profile." |
|  | | |  |  |

### See book status:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | | See Book Status | | |
| **Actors** | | Student | | |
| **Summary** | | Students track the status of their requested books. | | |
| **Pre-Conditions** | | The user must be logged in. | | |
| **Post-Conditions** | | The user is informed about the current status of their requested books. | | |
| **Special Requirements** | | None | | |
| **Basic Flow** | | | | |
| **Actor Action** | | | **System Response** | |
| 1 | The student navigates to the "Recommended Books" section. | | 2 | The system displays a list of the student’s book requests with their statuses (e.g., Pending, Approved, Delivered). |
| 3 | The student selects a specific request to view more details. | | 4 | The system shows the current status and any relevant updates. |
| **Alternative Flow** | | | | |
| 3.1 | The student has no active book requests. | | 3.A | The system displays a message: "You have no active book requests." |

## Database Design *(Optional)*

## Class Diagram (*Optional)*

## Sequence diagram *(Optional)*

## Any Other Artifact…

## GUI Graphical User Interfaces (*Optional)*

This section should give the GUI dumps of each screen, with reference to the user. The navigation flow of each user is also required, and each GUI should mark the functionality/use case that it covers.

# Implementation and Test Cases

For each chapter provide a paragraph of introduction and in the end a paragraph of conclusions.

## Implementation

### Implementation of First Component/Algorithm

**Introduction**

In this chapter, we delve into the implementation of the **Ehsas-Hub** platform. We will cover the core components of the system that have been developed so far, focusing on the major algorithms implemented, such as the **Recommendation System**, and **Volunteer Coordination** functionalities. Additionally, we will describe the platforms, APIs, and libraries that were used in the system. This chapter will also discuss the test cases that validate the system’s functionality, ensuring its performance, security, and reliability.

### Prototype

The initial prototype of **Ehsas-Hub** has been developed to showcase the core functionalities of the platform. This prototype focuses on the primary use cases, such as user registration, book donation management, and personalized recommendations. It provides a basic structure for the system’s user interface, backend logic, and database integration, demonstrating how different user roles (students, donors, volunteers, and admins) interact with the platform.

The prototype is built using the (**MySQL**, **Express.js**, **React.js**, and **Node.js**) and integrates key features like a hybrid recommendation system and volunteer task coordination.

## **Test case Design and description**

**This section will be added in FYP-II.** Summarize the common attributes of test cases. This may include input constraints that must be true for every input in the set of associated test cases, any shared environmental needs, any shared special procedural requirements, and any shared case dependencies. The following scheme is recommended for describing test cases in detail.

### Sample Test case No.1

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| id | TEST CASES | PRECONDITIONS | INPUT DATA | STEPS | EXPECTED RESULT | ACTUAL RESULT | PASS/FAIL |
| 1 | Test Admin Registration Successfully | None | Name, Email, Phone, Password, Gender, Image | Fill registration form fields and press Submit | New admin account created and stored in database | Admin created successfully | Pass |
| 2 | Test Admin Login Successfully | Admin must be registered | Correct Email and Password | Enter Email and Password → Press Login | Admin successfully logged in | Logged in successfully | Pass |
| 3 | Test Admin Login with Incorrect Password | Admin must be registered | Correct Email, Incorrect Password | Enter Email and wrong Password → Press Login | Error message displayed: Incorrect Password | Error displayed | Pass |
| 4 | Test Admin Login with Unregistered Email | None | Incorrect Email, Any Password | Enter unregistered Email → Press Login | Error message displayed: Email not registered | Error displayed | Pass |
| 5 | Test Admin Update Profile | Admin must be logged in | Updated Name, Phone, Image | Update fields and press Save | Admin profile updated successfully | Profile updated | Pass |
| 6 | Test Admin Delete Account | Admin must be logged in | Admin ID | Select admin → Press Delete | Admin record deleted from database | Admin deleted | Pass |

### Sample Test case No.2

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ID | TEST CASES | PRECONDITIONS | INPUT DATA | STEPS | EXPECTED RESULT | ACTUAL RESULTS | PASS/FAIL |
| 1 | Test user registration with valid data | User not registered | Correct name, email, phone, password, address, gender, image | Fill all fields and press Register | User is successfully registered | User registered successfully | Pass |
| 2 | Test user registration with duplicate email | Email already registered | Existing email, new name, phone, password | Enter duplicate email and press Register | "Email already exists" error should appear | Duplicate email error displayed | Pass |
| 3 | Test user login with correct credentials | User already registered | Correct registered email and password | Enter email/password and press Login | User logged in successfully | Login successful | Pass |
| 4 | Test user login with wrong password | User already registered | Correct email, wrong password | Enter email and wrong password, press Login | "Invalid credentials" error should appear | Login failed with error | Pass |
| 5 | Test update user profile information | User logged in | Updated phone or address | Change profile fields and save | Profile updated successfully | Profile updated successfully | Pass |
| 6 | Test delete user | User present in database | Existing user data | Press Delete on the user | User deleted successfully | User deleted successfully | Pass |
| 7 | Test registration with invalid email format | No user registered | Wrong email format (missing '@') | Enter wrong email and press Register | "Invalid Email Format" error should appear | Invalid email error displayed | Pass |
| 8 | Test user status field behavior | New registration | Correct user details | Register and check status field | Status should be Active (1) | Status set correctly | Pass |
| 9 | Test phone number field validation | No user registered | Phone number less than 10 digits | Enter short phone number and press Register | "Invalid Phone Number" error should appear | Phone validation error shown | Pass |
| 10 | Test password encryption | New registration | Correct user data | Register and check database password field | Password should be encrypted (hash) | Password saved encrypted | Pass |

### Sample Test case No.3

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ID | TEST CASES | PRECONDITIONS | INPUT DATA | STEPS | EXPECTED RESULT | ACTUAL RESULTS | PASS/FAIL |
| 1 | Test volunteer registration with valid data | Volunteer not registered | Correct name, email, phone, password, address, profile image | Fill all fields and press Register | Volunteer registered successfully | Volunteer registered successfully | Pass |
| 2 | Test volunteer registration with duplicate email | Email already registered | Existing email, new name, phone, password | Enter duplicate email and press Register | "Email already exists" error should appear | Duplicate email error displayed | Pass |
| 3 | Test volunteer login with correct credentials | Volunteer already registered | Correct email and password | Enter email/password and press Login | Volunteer logged in successfully | Login successful | Pass |
| 4 | Test volunteer login with incorrect password | Volunteer already registered | Correct email, wrong password | Enter correct email and wrong password | "Invalid Credentials" error should appear | Login failed with error | Pass |
| 5 | Test update volunteer profile | Volunteer logged in | Updated phone/address details | Change and save profile | Profile updated successfully | Profile updated successfully | Pass |
| 6 | Test delete volunteer | Volunteer present in database | Existing volunteer data | Press Delete on the volunteer | Volunteer deleted successfully | Volunteer deleted successfully | Pass |
| 7 | Test volunteer phone number validation | No volunteer registered | Phone number with letters or special chars | Enter invalid phone number and press Register | "Invalid Phone Number" error should appear | Validation error displayed | Pass |
| 8 | Test volunteer password strength | No volunteer registered | Weak password without special char, uppercase | Enter weak password and submit | "Weak Password" error should appear | Weak password error displayed | Pass |
| 9 | Test volunteer profile image upload | No volunteer registered | Correct image file (JPG/PNG) | Upload profile image and press Register | Image uploaded and saved | Image saved successfully | Pass |
| 10 | Test volunteer status after registration | New volunteer registration | Correct registration data | Complete registration and check status field | Status should be Active (1) | Status set correctly | Pass |

### Sample Test case No.3

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ID | TEST CASES | PRECONDITIONS | INPUT DATA | STEPS | EXPECTED RESULT | ACTUAL RESULTS | PASS/FAIL |
| 1 | Test if donor can submit with valid inputs | User is logged in and on the Donate Book form | Valid Book Name, Genre, Book Edition, Author Name, Book Image selected | Enter all fields correctly → Click Donate | Book should be donated successfully and saved in database | Book donation successful | Pass |
| 2 | Test form submission without Book Name | User is logged in | Leave Book Name empty, fill other fields correctly | Try submitting the form without entering Book Name | Error should appear (Book Name required) | Error displayed, form not submitted | Pass |
| 3 | Test form submission without Genre | User is logged in | Leave Genre field empty, fill others | Submit the form | Error should appear (Genre required) | Error displayed, form not submitted | Pass |
| 4 | Test form submission without Book Edition | User is logged in | Leave Book Edition empty | Submit the form | Error should appear (Book Edition required) | Error displayed, form not submitted | Pass |
| 5 | Test form submission without Author Name | User is logged in | Leave Author Name empty | Submit the form | Error should appear (Author Name required) | Error displayed, form not submitted | Pass |
| 6 | Test form submission without selecting Book Image | User is logged in | Fill all fields, do not upload image | Submit the form | Form should allow submission OR show warning if image is mandatory | Form submitted successfully if image is optional / Error if mandatory | Pass |
| 7 | Test if file upload accepts only image formats | User is logged in | Upload invalid file type (e.g., .txt, .pdf) in Book Image field | Submit the form | Error should show (Invalid file type) | Proper error displayed | Pass |
| 8 | Test minimum character validation for Book Name | User is logged in | Enter 1-2 characters only in Book Name | Submit the form | Error should show (Minimum 3 characters required) | Error displayed | Pass |
| 9 | Test Book Edition field accepts only numbers | User is logged in | Enter text in Book Edition field (e.g., "First Edition") | Submit the form | Error should show (Only numbers allowed) | Error displayed | Pass |
| 10 | Test successful clearing of form fields after donation | Donation successful | After submitting form | Check if form fields clear/reset automatically | Fields should reset after donation | Form reset successfully | Pass |
| 11 | Test clicking Close button cancels donation | User is on Donate Book form | Click Close without filling anything | Close the modal | Form should close without submitting any data | Modal closed successfully | Pass |

## Test Metrics

Summarize here the common ground of attributes of test case metrics.

### Sample Test case Matric.No.1

|  |  |
| --- | --- |
| Metric: | Purpose |
| Number of Test Cases: | Total number of test cases that you have developed for your system. |
| Number of Test Cases Passed: | The number of test cases that successfully passed |
| Number of Test Cases Failed: | The number of test cases that failed |
| Test Case Defect Density: | (No of test cases failed \* 100)  No of test cases executed |
| Test Case Effectiveness: | No of defects detected using test cases \*100  Total number of defects detected |
| Traceability Matrix: | Traceability is the ability to determine that each feature has a source in requirements and each requirement has a corresponding implemented feature. |

### Sample Test case Metric.No.2

### Sample Test case Metric.No.3

.

.

.

# Experimental Results and Analysis

**This chapter will be added in FYP-II.** Give proper analysis and discussion of experimental results (in plain English text) along with tables of results. **For each chapter provide a paragraph of introduction and in the end a paragraph of conclusions.**

# Conclusion and Future Directions

**This chapter is mandatory.** Give conclusions and summary of the work done. What were your findings and what were the results? Discuss in detail whether the scope of your project was entirely covered or not and whether the objectives of the project were met or not. What challenges did you face and what has been left out and why?

Sum up all the conclusions of all the chapters here to make a conclusion chapter. Do not repeat any text, just summarize it in different words.

Give recommendations for future work also. How your project can be further enhanced or improved? Future recommendations if someone wants to work on it. **For FYP-1 it is mandatory to list down a plan of the work to be done for FYP-2.**

# References

List all important sources of information which have been consulted for this project

# Appendix

## Appendix A: Guidelines

This section should include all supporting information from the project that was not included in the body of the report.  You should include surveys, complex statistical calculations, certain detailed tables and other such information in an appendix.  The information presented in this section is important to support the work presented in the body of the report but would make it more difficult to read and understand if presented within the body of the report.

Cite the appendix items in the report narrative (write "see Appendix A") and organize appendices (e.g., Appendix A, Appendix B,

Any tables, figures, forms, or other materials that are not totally central to the analysis but that need to be included are placed in the Appendix.

## Appendix B: Heading of Sample Appendix B

Following is a sample code with “code” style format.

Void SampleFunction(){

Print “Hello World.”;

}

# Formatting Guidelines

This document also serves as style guide for final year project reports. In order to give a similar high-quality appearance to all final year software project reports this template uses a collection of predefined Microsoft Word formatting styles. **These styles should be used without modification or replacement.** Font in the document is ***“Time New Roman”.*** This template provides following styles:

* **Title** – the main title style
* **Title2** – the subtitle style
* **Body Text** – style for paragraphs
* **Caption** – the style for a figure or table caption
* **Table Description** – the style for description of table, it must be added after caption.
* **Figure Description** - the style for description of figure, it must be added after caption.
* **Code** – the style for program source code

**int x** = 10; // Writing important code

* **Table Header Row** – Style for the header row of table
* **Table Grid** – the style for the data rows in the tables
* **Reference** – The style for references
* **Bullets** – The style for the bullet lists
* **Numbered** **List**– Style for numbered lists

All Heading styles with different level numbers are listed below.

# Heading 1

## Heading 2

### Heading 3

#### Heading 4

##### Heading 5

###### Heading 6

Heading 7

Heading 8

Heading 9

## Tables and Figures

Tables and figures should be centered horizontally. The caption button should be used to insert caption for both the figures and tables. All figures and tables must be numbered properly. Always refer to tables and figures according to their numbers. A table or figure can be cited as follows: ‘see Table1’ or ‘as shown in Table1’. The caption of table should be centered above the table and figure caption should be centered below the figure. Place the tables/figures close to their reference. Use “Table Header Row” and ‘Table Grid’ style for table’s header and data rows respectively. It is compulsory to provide brief description of table/figure after its caption. Styles for table and figure descriptions are “Table Description” and “Figure Description” respectively.

Press Ctrl+Shift+S to see list of styles mentioned above. Figure 1 shows the Apply Style window displaying the list of styles. Select any text then press Ctrl+Shift+S, the Apply Style window will show you the current style applied on that text and if required, you can change the style by selecting any other style from the “Style Name” dropdown.

This is brief description of above figure.

Figure 1: List of Styles

Table 2: This is Sample table caption

This is brief description of following Table.

|  |  |  |  |
| --- | --- | --- | --- |
| Header row | Header row | Header row | Header row |
| Row1 col1 | Row1 col2 | Row1 col3 | Row1 col4 |
| Row2 col1 | Row2 col2 | Row2 col3 | Row2 col4 |

Table 3: This is Sample table caption

This is brief description of following Table.

|  |  |  |  |
| --- | --- | --- | --- |
| Header row | Header row | Header row | Header row |
| Row1 col1 | Row1 col2 | Row1 col3 | Row1 col4 |
| Row2 col1 | Row2 col2 | Row2 col3 | Row2 col4 |

## Equations

Use equation editor to write equations in this report. Use last button of the custom tool bar to invoke equation editor. Similar to tables and figures, equations should also be aligned centered horizontally. Number all equations and insert them in parenthesis. Below is a sample equation and its reference number. An equation can be referenced like this: ‘it is clear from (1)’.

 (1)

## Header/Footer

Notice the headers in this document, before Introduction (i.e. the main content of this document) page numbers are in roman numerals. The page numbers of the actual content start with Arabic numerals i.e. 1, 2, 3 and so on. All of the **odd numbered pages** contain title of your project while the **even numbered pages** contain the section heading (i.e. chapter’s name) in the headers.

## Other Formatting Guidelines

* Keep 2-4 GUIs in one page. Consume as much space as possible. Do not leave most of page blank unnecessarily.
* Do not break tables (or use cases) in multiple pages unless the table is too large to fit in one page.
* Re-arrange the content i.e., text, images, and tables properly to meet above two guidelines.

## References

Always refer to the source of information by inserting the reference number in square brackets like this [5]. The reference numbers can either be added at the end of the sentence or within the sentence without changing the punctuation of sentence. A reference can also be cited as follows: ‘as Ruskey [2] mentioned’. List each source only once on your reference page.



Figure 2: IEEE Reference style

This figure represents the styling information for adding references in IEEE format

**Following is a list of sample reference for various typed of sources in IEEE format.**

1. P.M. Morse and H. Feshback, *Methods* of *Theoretical Physics*. New York: McGraw Hill, 1953. **//Format for Book**
2. S.K. Kenue and J.F. Greenleaf, “Limited angle multifrequency diffiaction tomography,” *IEEE Trans. Sonics Ultrason*., vol. SU-29, no. 6, pp. 213-2 17, July 1982. **//Format for Journal Article**
3. B. Tsikos, “Segmentation of 3-D scenes using multi-modal interaction between machine vision and programmable mechanical scene manipulation,” Ph.D. dissertation, Univ. of Pennsylvania, BCE Dept., Philadelphia, 1987. [Add if applicable: University Microfilms, Inc., University of Michigan, Ann Arbor, Michigan.] **//Format for Dissertation or thesis**
4. R. Finkel, R. Taylor, R. Bolles, R. Paul, and J. Feldman, “An overview of AL, programming system for automation,” in *Proc. Fourth Int. Joint Conf Artif. Intell*., pp. 758-765, Sept. 3-7, 1975. **//Format for Proceedings paper**
5. “Technology threatens to shatter the world of college textbooks, *The Wall Street Journal*, vol 91, pp. Al, A8, June 1, 1993. **//Format for Newspaper article**
6. R. Cox and J. S. Turner, “Project Zeus: design of a broadband network and its application on a university campus,” Washington Univ., Dept. of Comp. Sci., Technical Report WUCS-91-45, July 30, 1991. **//Format for Technical Report**
7. M. Janzen, *Instant Access Accounting*. Computer software. Nexus Software, Inc IBM-PC, 1993. **//Format for** **Software**
8. Fuminao Okumura and Hajime Takagi, “Maglev Guideway On the Yamanashi Test Line,” *http://www.rtri.or.jp/rd/maglev2/okumura.html*, October 24, 1998. **//Format for** **World Wide Web** (give author and title if named)
9. “AT&T Supplies First CDMA Cellular System in Indonesia,” http://www.att.com/press/1095/951011.nsa.html, Feb 5, 1996. **//Format for World Wide Web**