**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: May, 2025

**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 **Tajamul 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, and genres) and provide top rated books. It ensures that all that is done is open and honest so that trust may be built. This does not only get the right resources to the right students but empowers them to reach their full potential.

Table of Contents

[Table of Contents i](#_Toc198214466)

[List of Tables iv](#_Toc198214467)

[List of Figures v](#_Toc198214468)

[Chapter 1: Introduction 7](#_Toc198214469)

[1.1 Introduction: 7](#_Toc198214470)

[1.2 Goals and Objectives 7](#_Toc198214471)

[1.2.1 Goals: 7](#_Toc198214472)

[1.2.2 Objectives: 8](#_Toc198214473)

[1.3 Scope of the Project 8](#_Toc198214474)

[1.3.1 Functional Scope: 8](#_Toc198214475)

[1.3.2 Non-Functional Scope: 10](#_Toc198214476)

[1.4 Conclusion: 11](#_Toc198214477)

[Chapter 2: Literature Review 13](#_Toc198214478)

[2.1 Introduction 13](#_Toc198214479)

[2.2 Background and Problem Elaboration 13](#_Toc198214480)

[2.3 Detailed Literature Review 13](#_Toc198214481)

[2.3.1 Definitions 13](#_Toc198214482)

[2.3.2 Related Research Work 1 14](#_Toc198214483)

[2.3.3 Related Research Work 2 14](#_Toc198214484)

[2.4 Literature Review Summary Table 14](#_Toc198214485)

[2.5 Research Gap 15](#_Toc198214486)

[2.6 Problem Statement 15](#_Toc198214487)

[2.7 Conclusion 16](#_Toc198214488)

[Chapter 3: Requirements and Design 18](#_Toc198214489)

[3.1 Introduction: 18](#_Toc198214490)

[3.2 Requirements 18](#_Toc198214491)

[3.2.1 Functional Requirements 18](#_Toc198214492)

[3.2.2 Non-Functional Requirements 22](#_Toc198214493)

[3.2.3 Hardware and Software Requirements 23](#_Toc198214494)

[3.3 Proposed Methodology 23](#_Toc198214495)

[3.4 System Architecture 24](#_Toc198214496)

[3.5 Use Cases 25](#_Toc198214497)

[3.5.1 Admin Use-Case Diagram: 25](#_Toc198214498)

[3.5.2 Needy Use-case Diagram: 26](#_Toc198214499)

[3.5.3 Donor Use-Case Diagram: 27](#_Toc198214500)

[3.5.4 Volunteer Use-Case Diagram 28](#_Toc198214501)

[3.5.5 Full System Use-Case Diagram: 29](#_Toc198214502)

[3.6 Fully Dressed Use Cases: 30](#_Toc198214503)

[3.6.1 Sign Up: 30](#_Toc198214504)

[3.6.2 Login 30](#_Toc198214505)

[3.6.3 Recover password: 31](#_Toc198214506)

[3.6.4 Edit Profile: 31](#_Toc198214507)

[3.6.5 Donate Books: 32](#_Toc198214508)

[3.6.6 Request Book: 32](#_Toc198214509)

[3.6.7 Manage Donor Request: 33](#_Toc198214510)

[3.6.8 Manage Account: 33](#_Toc198214511)

[3.6.9 Manage Listing: 34](#_Toc198214512)

[3.6.10 Feedback: 34](#_Toc198214513)

[3.6.11 Recommended Books: 35](#_Toc198214514)

[3.6.12 Add to Favorites: 35](#_Toc198214515)

[3.6.13 Requested Book Statistics: 36](#_Toc198214516)

[3.6.14 Search Book: 36](#_Toc198214517)

[3.6.15 Donated Book Statistics: 37](#_Toc198214518)

[3.6.16 Manage Request: 37](#_Toc198214519)

[3.6.17 Logout: 38](#_Toc198214520)

[3.7 Entity Relationship diagram 39](#_Toc198214521)

[3.8 Activity Diagram 40](#_Toc198214522)

[3.9 GUI Graphical User Interfaces 41](#_Toc198214523)

[3.9.1 User Role GUI: 41](#_Toc198214524)

[3.9.2 Admin Role GUI: 46](#_Toc198214525)

[3.9.3 Volunteer Role GUI: 51](#_Toc198214526)

[Chapter 4: Implementation and Test Cases 56](#_Toc198214527)

[4.1 Implementation 56](#_Toc198214528)

[4.1.1 Implementation Overview 56](#_Toc198214529)

[4.1.2 Introduction 56](#_Toc198214530)

[4.1.3 Prototype 56](#_Toc198214531)

[4.1.4 Key Implementation Components 57](#_Toc198214532)

[4.2 Test Cases 61](#_Toc198214533)

[4.2.1 Admin Test Cases 61](#_Toc198214534)

[4.2.2 Needy Test Case 62](#_Toc198214535)

[4.2.3 Volunteer Test Case 63](#_Toc198214536)

[4.2.4 Donor Test Case 64](#_Toc198214537)

[4.3 Test Metrics 65](#_Toc198214538)

[4.3.1 Common Attributes of Test Case Metrics 65](#_Toc198214539)

[4.3.2 Test Summary Table 66](#_Toc198214540)

[4.4 Conclusion 66](#_Toc198214541)

[Chapter 5: Experimental Results and Analysis 68](#_Toc198214542)

[5.1 Introduction 68](#_Toc198214543)

[5.2 Experimental Setup 68](#_Toc198214544)

[5.2.1 Platform Performance Evaluation 68](#_Toc198214545)

[5.2.2 Recommendation System Effectiveness 70](#_Toc198214546)

[5.2.3 Authentication and Security 71](#_Toc198214547)

[5.3 Conclusion 72](#_Toc198214548)

[Chapter 6: Conclusion and Future Directions 74](#_Toc198214549)

[6.1 Introduction 74](#_Toc198214550)

[6.2 Achievements and Improvements 74](#_Toc198214551)

[6.2.1 Front-End Achievements: 74](#_Toc198214552)

[6.2.2 Backend Achievement’s 74](#_Toc198214553)

[6.2.3 Recommendation Engine: 75](#_Toc198214554)

[6.2.4 Database Enhancements: 75](#_Toc198214555)

[6.3 Critical Review 75](#_Toc198214556)

[6.3.1 Strengths 75](#_Toc198214557)

[6.3.2 Weaknesses: 75](#_Toc198214558)

[6.4 Future Recommendations 76](#_Toc198214559)

[6.4.1 Enhancements to Current Modules: 76](#_Toc198214560)

[6.4.2 Additional Features: 76](#_Toc198214561)

[6.4.3 Future Specific Work Plan: 76](#_Toc198214562)

[6.5 Conclusion 76](#_Toc198214563)

[References 77](#_Toc198214564)

List of Tables

Table 2.1: Literature Review Summary Table 14

Table 3.1: Functional Requirements of Needy 18

Table 3.2: Functional Requirements of Donor 19

Table 3.3: Functional Requirements of Volunteer 20

Table 3.4: Functional Requirements of Admin 20

Table 3.5: Fully Dressed Use Case (Sign Up) 30

Table 3.6: Fully Dressed Use Case (Login) 30

Table 3.7: Fully Dressed Use Case (Recover Password) 31

Table 3.8: Fully Dressed Use Case (Edit Profile) 31

Table 3.9: Fully Dressed Use Case (Donate Books) 32

Table 3.10: Fully Dressed Use Case (Request Book) 32

Table 3.11: Fully Dressed Use Case (Manage Donor Request) 33

Table 3.12: Fully Dressed Use Case (Manage Account) 33

Table 3.13: Fully Dressed Use Case (Manage Listing) 34

Table 3.14: Fully Dressed Use Case (Feedback) 34

Table 3.15: Fully Dressed Use Case (Recommended Books) 35

Table 3.16: Fully Dressed Use Case (Add to Favorites) 35

Table 3.17: Fully Dressed Use Case (Requested Book Statistics) 36

Table 3.18: Fully Dressed Use Case (Search Book) 36

Table 3.19: Fully Dressed Use Case (Donated Book Statistics) 37

Table 3.20: Fully Dressed Use Case (Manage Request) 37

Table 3.21: Fully Dressed Use Case (Logout) 38

Table 4.1: Admin Test Case 61

Table 4.2: Needy Test Case 62

Table 4.3: Volunteer Test Case 63

Table 4.4: Donor Test Case 64

Table 4.5: Test Summary Table 66

Table 5.1: Functional Performance Evaluation 69

Table 5.2: Recommendation System Evaluation 70

Table 5.3: Authentication Metrics 71

List of Figures

Figure 3.1: System Architecture 24

Figure 3.2: Admin Use-Case Diagram 25

Figure 3.3: Needy Use-Case Diagram 26

Figure 3.4: Donor Use-Case Diagram 27

Figure 3.5: Volunteer Use-Case Diagram 28

Figure 3.6: Full System Use-Case Diagram 29

Figure 3.7: Entity Relationship Diagram (ERD) 39

Figure 3.8: Whole system Activity Diagram 40

Chapter 01:

Introduction

# Introduction

## 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. Ehsas Hub, a digital platform that makes it easy for donors, students, and volunteers to meet, can fill this gap. The primary 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:

* **To develop a centralized book donation platform:** Goal is creating an online platform where people can contribute books, specifically educational material, to needy students. The idea is to simplify the process of donation and make books easily available to those in need.
* **To develop an AI-based book recommendation system:** Through machine learning algorithms, Ehsas Hub will suggest books to users from their preferred genre, favorite book, and requested book. The aim is to get users to discover appropriate education related content that is related with their interests.
* **To enable coordination of volunteers and donors:** The site will manage volunteers and donors using a simple-to-use interface. Volunteers will be able to monitor their work, while donors will be informed of their donations and the effect they are creating.
* **To Create a Scalable, Secure, and User-Friendly Web Application:** The aim is to have a platform that is scalable, secure, and reliable, enabling growth in the future. The application should be simple to use, supporting various user roles like students, donors, volunteers, and admin.

### Objectives:

* **Develop Core Functionalities:** User Registration & Authentication: Provide users with the ability to register and securely log in with role-based access control (e.g., donors, volunteers, admin, needy).
* **Book Donation Management:** Provide a way for donors to submit books to donate, which will then be listed for needy users to request.
* **AI-Driven Book Recommendations:** Integrate a recommendation engine to recommend books to users based on their preferences and education needs.
* **Volunteer Task Administration:** Enable volunteers to monitor tasks involving book donation collection.
* **Admin Panel for Management:** Offer administrators with management tools for users, books, and donations

## Scope of the Project

The project Ehsas Hub is meant to develop a detailed platform that would link donors, volunteers, and students in need of study materials, mostly in the form of book donation. The scope of the project comprises both functional and non-functional parts that facilitate an effortless user experience. Following are the important areas encompassed within the project's scope:

### Functional Scope:

#### User Registration and Authentication:

* **Roles:** There will be several user roles on the platform, such as Admin, Donor, Volunteer, and Needy (Student).
* **Authentication**: User login and registration will be secured through JWT (JSON Web Tokens) for authenticating users.
* **Role-based Access Control (RBAC):** There will be different permissions for each type of user to access various sections of the platform.

#### Book Donation Management:

* **Donor Interface:** Donors can donate books (with fields like book name, author, genre, and book image).
* **Approval System:** Admin can approve/reject book donations based on specific parameters.
* **Book Availability:** Once approved, the books will be available for needy users to request.

#### Personalized Book Suggestions:

* **AI Integration:** An AI-powered recommendation system will recommend books to underprivileged students according to their interests, learning requirements, and past interactions with the platform.
* **Machine Learning Model:** NCF, Clustering or Hybrid Models, leveraging Tensor Flow and other machine learning algorithms, will drive the recommendation engine.

#### Volunteer Coordination:

* **Task Management:** Volunteers will be able to view and accept tasks associated with book collection from donors, which is near to them.

#### Admin Dashboard:

* **Admin Control:** Admin users will have total control over the platform, including user management (donors, volunteers, and needy), book approval, task allocation, and platform monitoring.
* **Data Overview:** Admins will be able to view all data, such as donation logs, volunteer tasks, user activity, and more.

#### Frontend Interface:

* **User interface:** The frontend is based on React, allowing a very intuitive and user-friendly interface for all kinds of users.
* **Dashboard:** This would comprise a specifically designed dashboard to track donations, book availability, volunteer activities and more.
* **Responsive Design:** The website will ensure mobile-friendliness and responsiveness across different types of devices.

#### Book Search and Request System:

* **Search Functionality:** Needy users are able to search for books by categories, subjects, and other parameters.
* **Request Process:** Needy users are able to request books using a simplified process and will get updates on email.

### Non-Functional Scope:

#### Reliability

* The system should handle errors gracefully and should not lose data during Donations.
* The platform must be able to recover quickly from unexpected failures Like passwords etc.

#### Security

* Strong authentication (JWT) and password encryption should be implemented to ensure that user data is secure.
* Sensitive data, such as user information and donation details, should be encrypted both during transmission and storage.

#### Usability

* The platform should have an intuitive and easy-to-use interface for all user roles (admin, donor, volunteer, needy).
* It should be accessible on both desktop and mobile devices, providing a responsive design for different screen sizes.

#### Maintainability

* The system should be modular, with clearly defined components that can be easily updated or replaced in the future.
* Proper documentation and error logs should be maintained to help with ongoing support and updates.

#### Compatibility

* The platform should work on common web browsers like **Chrome**, **Firefox**, and **Safari, Web Browsers.**
* It should be compatible with both **Windows** and **Ubuntu Linux** operating systems.

#### Resource Efficiency

* The system should operate efficiently without excessive consumption of memory or CPU, ensuring smooth operation on typical hardware.

## Conclusion:

The Ehsas Hub initiative aims to solve the issue of book availability by creating a platform that brings donors, volunteers, and students together. The principal objectives of the project are to make the process of donating simpler, enhance educational content access, and provide users with book recommendations based on AI. The goals include building essential features for user registration, donation management, and integrating AI, ensuring security, scalability, and usability throughout the platform.

**Chapter 02:   
Literature Review**

# 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 with an emphasis on book recommendation platforms; this chapter provides a thorough analysis of recommendation systems. It examines fundamental ideas, current studies, approaches used in comparable systems, and highlights important gaps pertinent to the Ehsas-Hub project.

## 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 the ones 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 2.1: Literature Review Summary Table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 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

The research gap in current book recommendation systems is their narrow attention to user-specific goals and educational aims. The majority of current systems strongly depend on user ratings or common interests, which do not pay attention to the specific requirements of particular user segments, like students within an educational context. These systems usually face the cold-start problem, data sparsity, and computational complexity, in addition, although hybrid recommendation systems have been investigated, they are computationally expensive and need well-curated datasets, which makes them less appropriate for platforms that cater to a broad set of stakeholders. There is limited research that integrates personalized recommendations based on particular academic or educational objectives with scalable and efficient solutions for platforms such as Ehsas Hub. Thus, there is a requirement for a recommendation system that not only takes ratings into account but also encompasses user interests, academic objectives, and computational efficiency.

## Problem Statement

Current book recommendation systems mainly concentrate on general-purpose recommendations, mostly overlooking the specific needs of nonprofit educational platforms such as Ehsas Hub. Issues including plague current systems:

* **Cold-Start Problem:** New members who have little or no historical information are problematic for personalized recommendations.
* **Data Sparsity**: Lack of user or item interaction data can lead to poor recommendations, particularly in collaborative filtering techniques.
* **Limited Personalization:** Most recommendation systems fail to well-personalize their suggestions with respect to precise academic interests, objectives, or user preferences.
* **Scalability Issues:** Current systems, particularly hybrid models, can be computationally intensive to handle large sets of data and thus are not appropriate for nonprofit platforms with limited resources.

## Conclusion

In summary, current book recommendation systems are primarily based on user ratings, mostly ignoring deeper, more personalized ones like users' research interests or special preferences. Though content-based and collaborative filtering methods are generally employed, they have limitations when dealing with the special requirements of academic platforms, especially in suggesting books according to a user's research objectives or interests. Such systems are also faced with problems such as the cold-start issue, sparsity of data, and computational expense, particularly for nonprofit platforms like Ehsas Hub, that demand personalized recommendations for students. Most systems are unable to incorporate such personalized interests and are centered on ratings only. The literature clearly points out an evident void in developing scalable and efficient recommendation systems that trade-off both ratings and user-specific interests.

Ehsas Hub seeks to fill this gap by creating a solution that offers personalized book suggestions according to user interests and learning objectives alongside solving typical issues such as cold-start issues and data sparsity.

**Chapter 03:**

**Requirements and Design**

# Requirements and Design

## Introduction:

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:

Table 3.1: Functional Requirements of Needy

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

#### Donor:

Table 3.2: Functional Requirements of Donor

|  |  |
| --- | --- |
| ID | Requirements |
| FR-2.1 | |  | | --- | | Donor will be able to Sign Up. |  |  | | --- | |  | |
| FR-2.2 | Donor will be able to Login. |
| FR-2.3 | Donor will be able to forget/recover their Password. |
| FR-2.4 | Donor will be to view profile. |
| FR-2.5 | Donor will be able to edit profile. |
| FR-2.6 | Donor will be able to donate books. |
| FR-2.7 | |  | | --- | |  |  |  | | --- | |  |   Donor will be able to view donated book stats. |
| FR-2.8 | |  | | --- | | Donor will be able to view donated book list. |  |  | | --- | |  | |
| FR-2.9 | Donor will be able to give feedback. |
| FR-2.10 | Donor will be able to Logout. |

#### Volunteer:

Table 3.3: Functional Requirements of Volunteer

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

#### Admin:

Table 3.4: Functional Requirements of Admin

|  |  |
| --- | --- |
| ID | Requirements |
| FR-4.1 | |  | | --- | | Admin will be able to sign up. |  |  | | --- | |  | |
| FR-4.2 | |  |  | | --- | --- | | Admin will be able to login |  |  |  | | --- | |  | |
| FR-4.3 | |  | | --- | | Admin will be able to forget/recover password. |  |  | | --- | |  | |
| FR-4.4 | |  | | --- | | Admin will be able to view profile. |  |  | | --- | |  | |
| FR-4.5 | |  | | --- | | Admin will be able to edit profile. |  |  | | --- | |  | |
| FR-4.6 | Admin will be able to manage accounts. |
| FR-4.7 | Admin will be able to approve account. |
| FR-4.8 | Admin will be able to reject account. |
| FR-4.9 | Admin will be able to freeze account. |
| FR-4.10 | Admin will be able to active account. |
| FR-4.11 | Admin will be able to view donor request. |
| FR-4.12 | Admin will be able to accept request. |
| FR-4.13 | Admin will be able to reject request. |
| FR-4.14 | Admin will be able to view approved request. |
| FR-4.14 | Admin will be able to view in process request. |
| FR-4.16 | Admin will be able to view completed request. |
| FR-4.17 | Admin will be able to view needy request. |
| FR-4.18 | Admin will be able to accept needy request. |
| FR-4.19 | Admin will be able to reject needy request. |
| FR-4.20 | Admin will be able to view needy approved request. |
| FR-4.21 | Admin will be able to view needy in process request |
| FR-4.22 | Admin will be able to view needy completed request. |
| FR-4.23 | Admin will be able to view volunteer request. |
| FR-4.24 | Admin will be able to view volunteer approved request. |
| FR-4.25 | Admin will be able to view volunteer completed request. |
| FR-4.26 | Admin will be able to view account statistics. |
| FR-4.27 | Admin will be able to view active list. |
| FR-4.28 | Admin will be able to active book. |
| FR-4.29 | Admin will be able to deactivate book. |
| FR-4.30 | Admin will be able to view feedbacks. |
| FR-4.31 | Admin will be able to log out. |

### Non-Functional Requirements

#### Reliability

* The system should handle errors gracefully and should not lose data during Donations.
* The platform must be able to recover quickly from unexpected failures like passwords etc.

#### Security

* Strong authentication (JWT) and password encryption should be implemented to ensure that user data is secure.
* Sensitive data, such as user information and donation details, should be encrypted both during transmission and storage.

#### Usability

* The platform should have an intuitive and easy-to-use interface for all user roles (admin, donor, volunteer, needy).
* It should be accessible on both desktop and mobile devices, providing a responsive design for different screen sizes.

#### Maintainability

* The system should be modular, with clearly defined components that can be easily updated or replaced in the future.
* Proper documentation and error logs should be maintained to help with ongoing support and updates.

#### Compatibility

* The platform should work on common web browsers like **Chrome**, **Firefox**, and **Safari, Web Browsers.**
* It should be compatible with both **Windows** and **Ubuntu Linux** operating systems.

#### Resource Efficiency

* The system should operate efficiently without excessive consumption of memory or CPU, ensuring smooth operation on typical hardware.

### 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 have an agile approach with 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, Account management, donation management, and recommendation system.
* **Testing**: Validate functionality, performance, and security.

## C:\Users\A\AppData\Local\Microsoft\Windows\INetCache\Content.Word\System Diagram Part 2 (1).jpgSystem Architecture

Figure 3.1: System Architecture

## Use Cases

### Admin Use-Case Diagram:

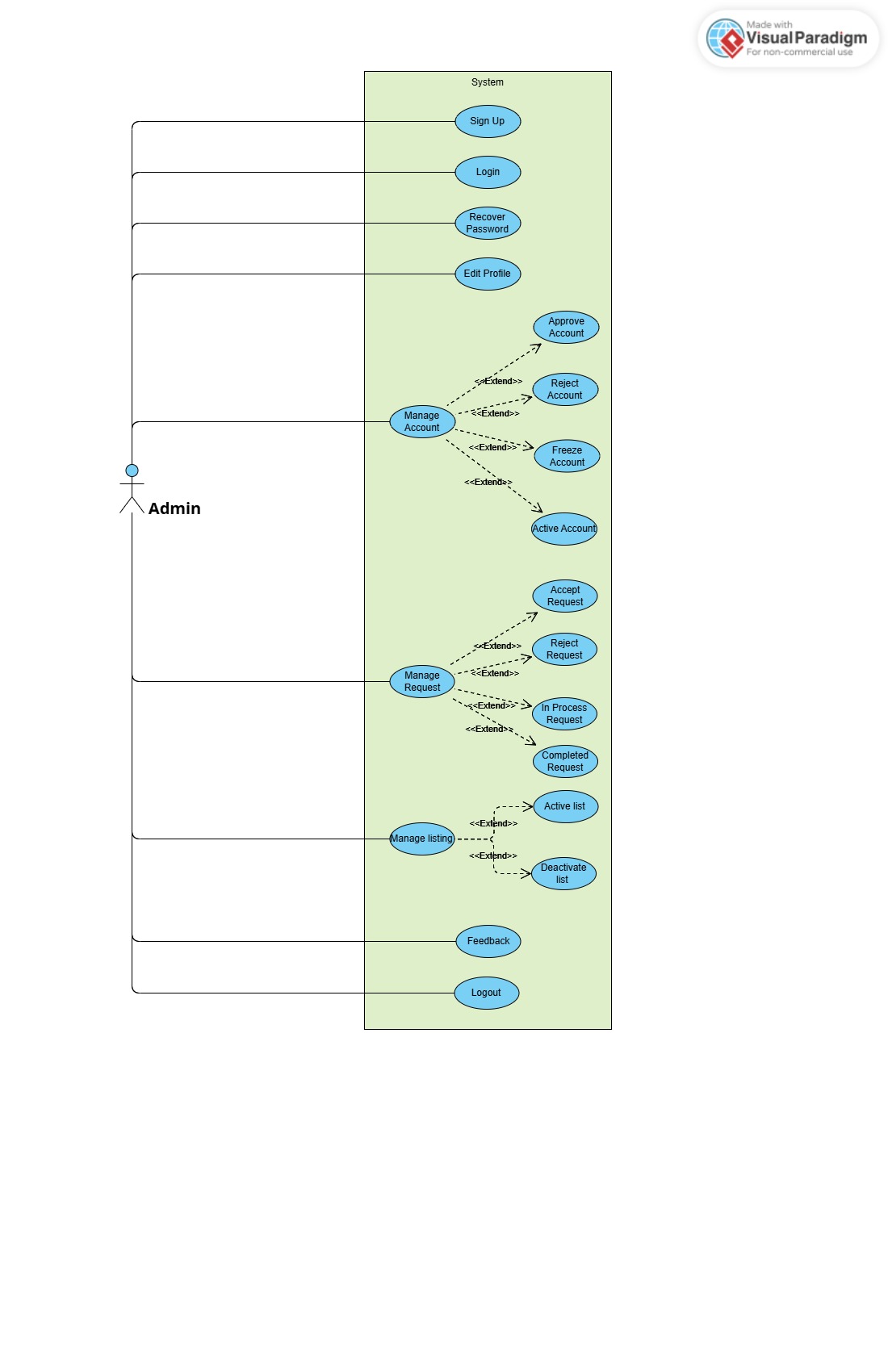


Figure 3.2: Admin Use-Case Diagram

### Needy Use-case Diagram:

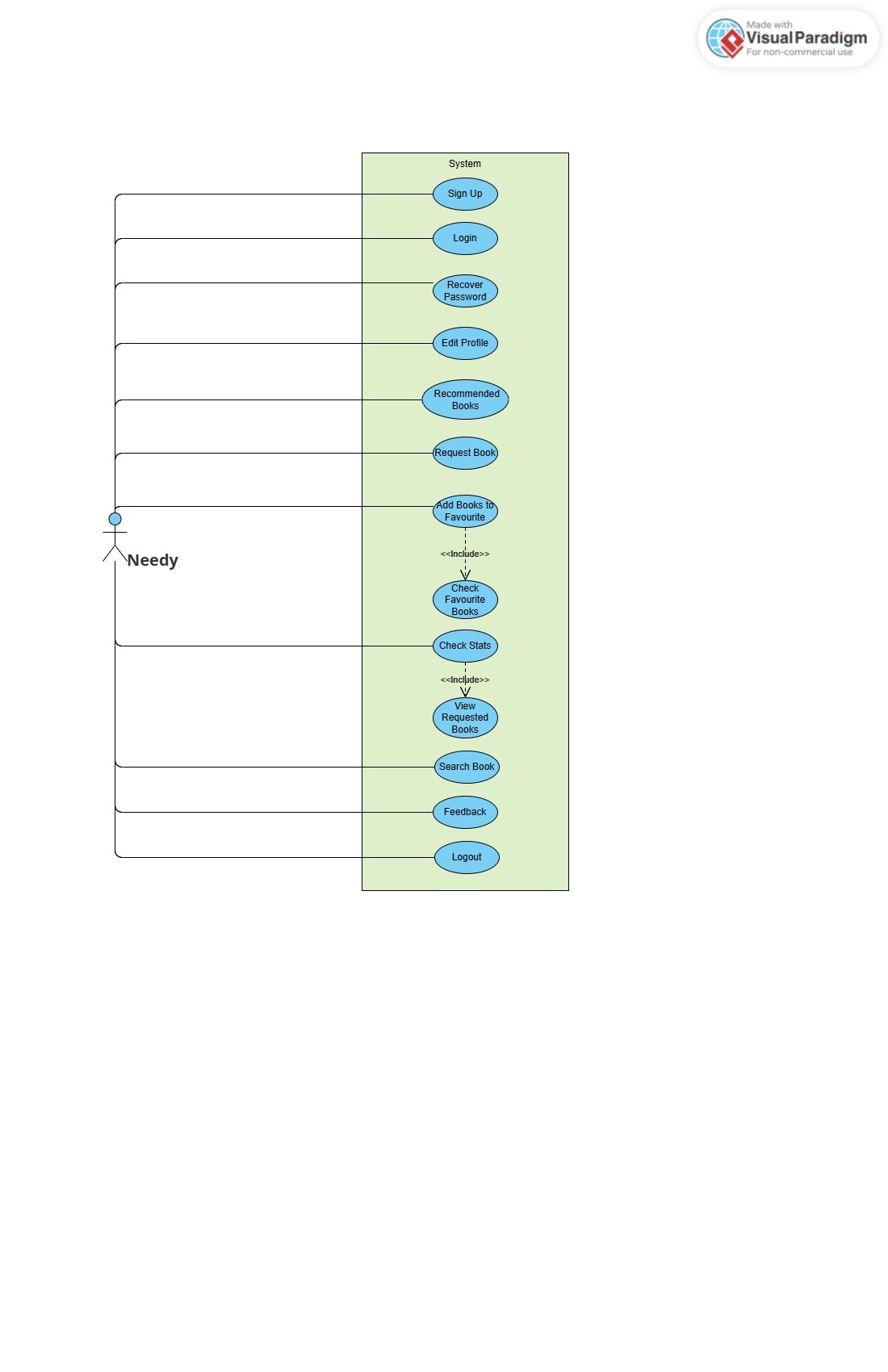


Figure 3.3: Needy Use-Case Diagram

### Donor Use-Case Diagram:

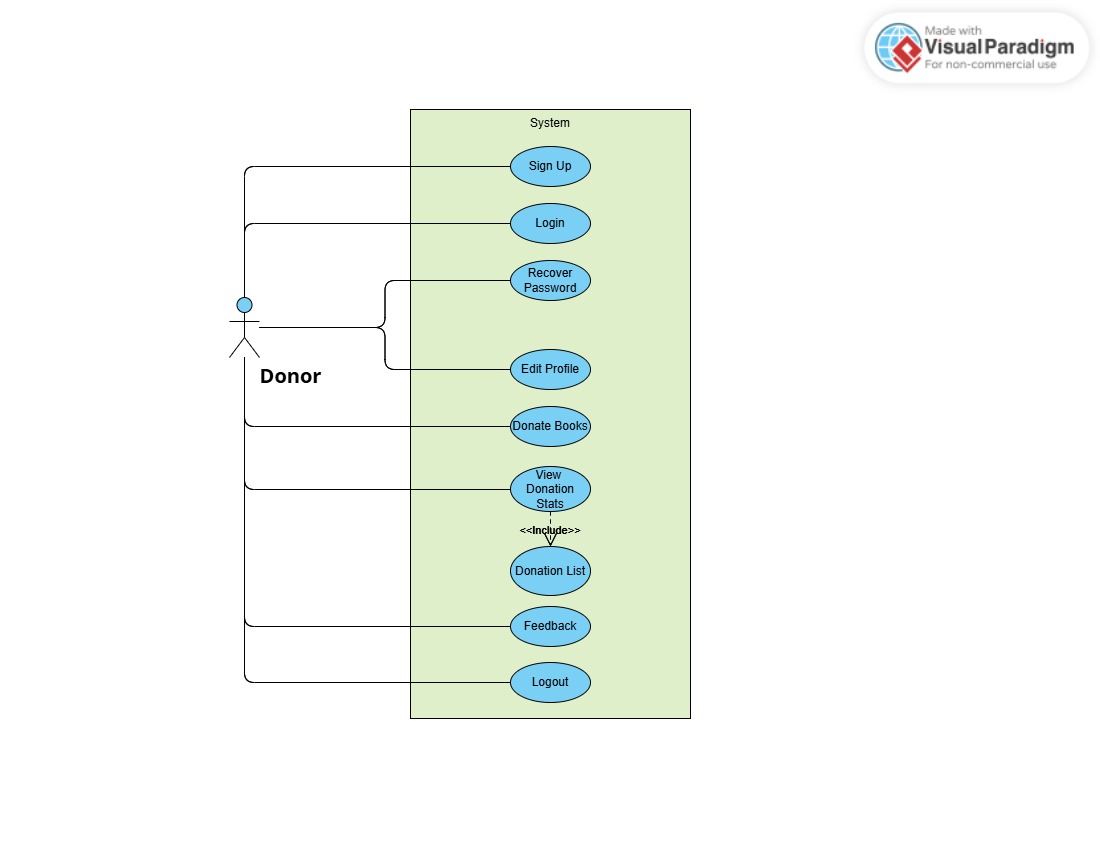
****

Figure 3.4: Donor Use-Case Diagram

### Volunteer Use-Case Diagram

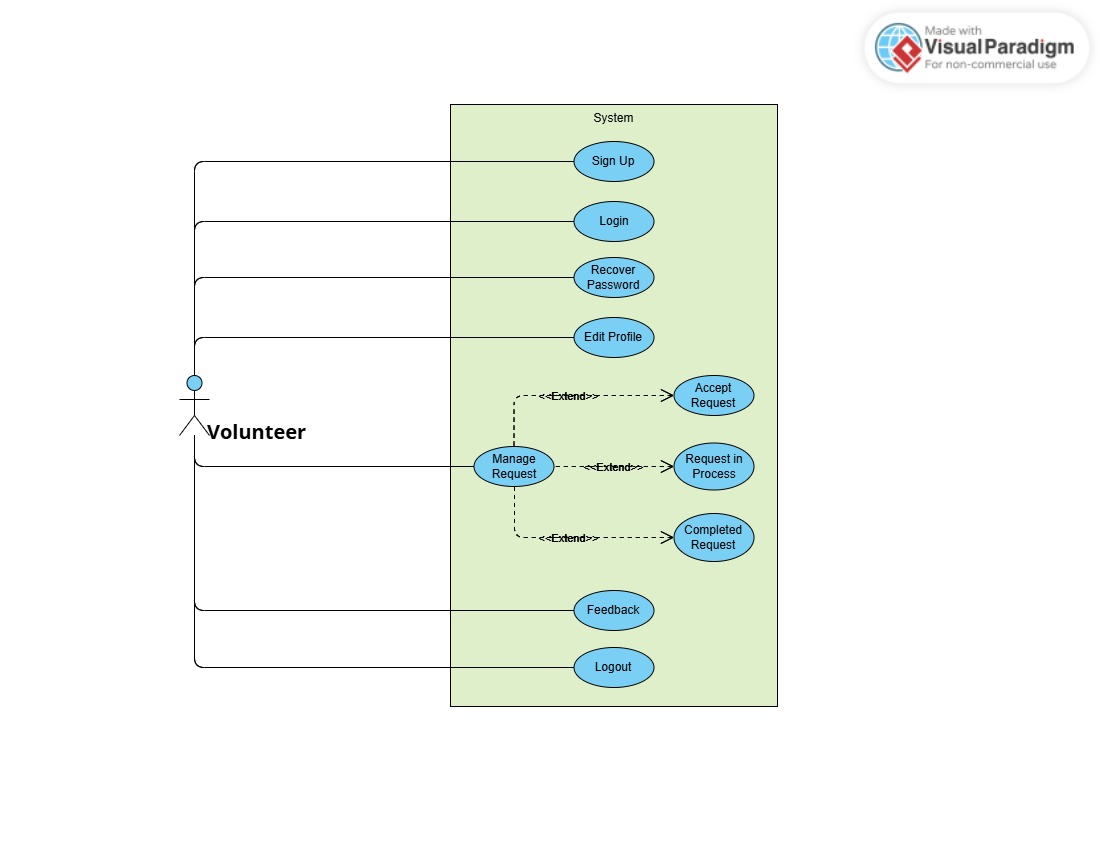


Figure 3.5: Volunteer Use-Case Diagram

### Full System Use-Case Diagram:

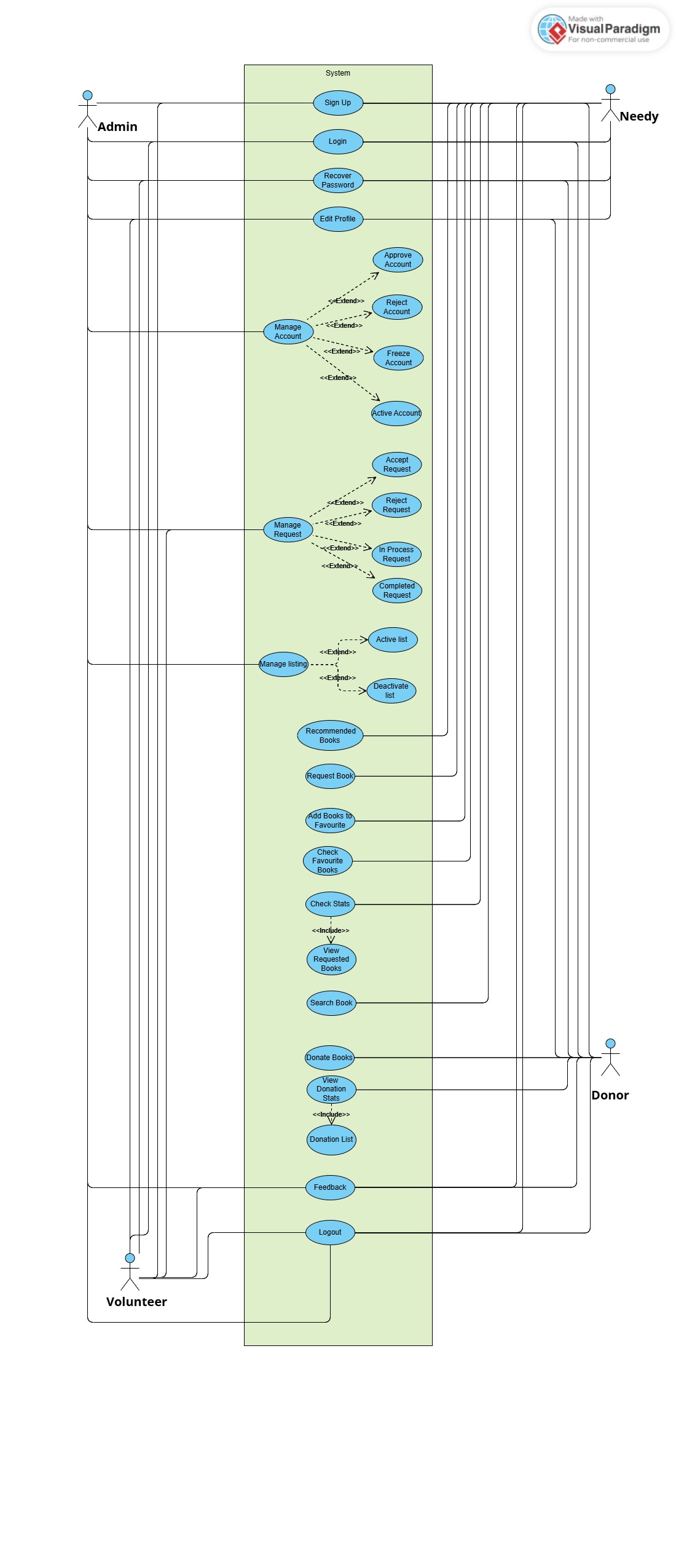


Figure 3.6: Full System Use-Case Diagram

## Fully Dressed Use Cases:

### Sign Up:

Table 3.5: Fully Dressed Use Case (Sign Up)

|  |  |
| --- | --- |
| Use CaSE ID | UC-001 |
| uSE CASE NAME | Sign Up |
| Primary Actor | Donor / Needy / Volunteer / Admin |
| Precondition | User is on the registration page |
| Postcondition | User account is successfully created |
| Main Flow | 1. User accesses the Sign-Up page  2. Enters required details (name, email, password etc.)  3. Clicks Sign Up  4. System validates input  5. System saves data and creates account  6. User is notified of successful registration |
| Alternate Flow | 4a. Missing or invalid data: system prompts correction  4b. Email already exists: system notifies and blocks submission |

### Login

Table 3.6: Fully Dressed Use Case (Login)

|  |  |
| --- | --- |
| Use Case ID | UC-002 |
| Use Case Name | Login |
| Primary Actor | Donor / Needy / Volunteer / Admin |
| Precondition | User is registered and on login screen |
| Postcondition | User is logged into the system |
| Main Flow | 1. User enters email and password  2. Clicks Login  3. System verifies credentials  4. Redirects to user dashboard |
| Alternate Flow | 3a. Invalid credentials: system shows error  3b. Account is frozen: system denies access and displays message |

### Recover password:

Table 3.7: Fully Dressed Use Case (Recover Password)

|  |  |
| --- | --- |
| Use Case ID | UC-003 |
| Use Case Name | Recover Password |
| Primary Actor | Donor / Needy / Volunteer / Admin |
| Precondition | User is on password recovery screen |
| Postcondition | Forgot Password OTP is sent to user’s email |
| Main Flow | 1. User enters registered email  2. Clicks 'Verify Email'  3. System sends OTP via email  4. User resets password via OTP |
| Alternate Flow | 1a. Email not registered: system notifies user |

### Edit Profile:

Table 3.8: Fully Dressed Use Case (Edit Profile)

|  |  |
| --- | --- |
| Use Case ID | UC-004 |
| Use Case Name | Edit Profile |
| Primary Actor | Donor / Needy / Volunteer / Admin |
| Precondition | User is logged in and on profile page |
| Postcondition | Profile is updated |
| Main Flow | 1. User navigates to profile page 2. Edits fields (Email, Profile picture, address etc.) 3. Verify email with OTP 4. Clicks Save 5. System updates database 6. Confirms update to user |
| Alternate Flow | 4a. Invalid inputs: system prompts correction |

### Donate Books:

Table 3.9: Fully Dressed Use Case (Donate Books)

|  |  |
| --- | --- |
| Use Case ID | UC-005 |
| Use Case Name | Donate Books |
| Primary Actor | Donor |
| Precondition | Donor is logged in |
| Postcondition | Donation created successfully. |
| Main Flow | 1. Donor selects 'Switch to donate'  2. Fills form with book details  3. Submits donation  4. System records donation and confirmation by email. |
| Alternate Flow | 3a. Missing book details: system prompts for completion |

### Request Book:

Table 3.10: Fully Dressed Use Case (Request Book)

|  |  |
| --- | --- |
| Use Case ID | UC-006 |
| Use Case Name | Request Book |
| Primary Actor | Needy |
| Precondition | Needy is logged in |
| Postcondition | Book request is submitted |
| Main Flow | 1. See books on Dashboard  2. Selects a book and clicks 'Request'  3. System saves the request  4. Request Confirmation email sent to user |
| Alternate Flow | 2a. your request has not been submitted you have exceed your limit for this month |

### Manage Donor Request:

Table 3.11: Fully Dressed Use Case (Manage Donor Request)

|  |  |
| --- | --- |
| Use Case ID | UC-007 |
| Use Case Name | Manage Request |
| Primary Actor | Volunteer |
| Precondition | Volunteer is logged in |
| Postcondition | Request in process |
| Main Flow | 1. Volunteer views pending requests  2. Accept one to manage  3. Accepted request status will be change to in process |
| Alternate Flow | 1a. System shows message no data available |

### Manage Account:

Table 3.12: Fully Dressed Use Case (Manage Account)

|  |  |
| --- | --- |
| Use Case ID | UC-007 |
| Use Case Name | Manage Account |
| Primary Actor | Admin |
| Precondition | Admin is logged in |
| Postcondition | Account Status is Updated to (approved, Reject, frozen, Active.) |
| Main Flow | 1. Admin views list of pending accounts  2. Approves, rejects, or freezes and active as needed  3. System updates status  4. System send email to user about the account status |
| Alternate Flow | 1a. System shows message no data available |

### Manage Listing:

Table 3.13: Fully Dressed Use Case (Manage Listing)

|  |  |
| --- | --- |
| Use Case ID | UC-009 |
| Use Case Name | Manage Listing |
| Primary Actor | Admin |
| Precondition | Admin is logged in |
| Postcondition | Deactivated Successfully |
| Main Flow | 1. Admin navigates to Active List  2. View active list  3. Chooses Deactivate  4. System update the status of book |
| Alternate Flow | 3a. No data available |

### Feedback:

Table 3.14: Fully Dressed Use Case (Feedback)

|  |  |
| --- | --- |
| Use Case ID | UC-010 |
| Use Case Name | Feedback |
| Primary Actor | All Users |
| Precondition | User on landing page |
| Postcondition | Feedback is Saved Successfully |
| Main Flow | 1. User accesses feedback form  2. Fill feedback form  3. Submit feedback form  3. System saves feedback |
| Alternate Flow | 2a. If any field empty, system prompts appear “fields are required” |

### Recommended Books:

Table 3.15: Fully Dressed Use Case (Recommended Books)

|  |  |
| --- | --- |
| Use Case ID | UC-011 |
| Use Case Name | Recommended Books |
| Primary Actor | Needy |
| Precondition | Needy Must be logged in |
| Postcondition | Needy can request book and add to favorite |
| Main Flow | 1. User will view recommended books on dashboard |
| Alternate Flow |  |

### Add to Favorites:

Table 3.16: Fully Dressed Use Case (Add to Favorites)

|  |  |
| --- | --- |
| Use Case ID | UC-012 |
| Use Case Name | Add to Favorites |
| Primary Actor | Needy |
| Precondition | Needy must be Logged in and on Dashboard |
| Postcondition | Book added to favorites |
| Main Flow | 1. User will view available books on dashboard  2. User will click on like button  3. Book will be added to favorites |
| Alternate Flow | 1a. No Available Books |

### Requested Book Statistics:

Table 3.17: Fully Dressed Use Case (Requested Book Statistics)

|  |  |
| --- | --- |
| Use Case ID | UC-013 |
| Use Case Name | Requested Book Statistics |
| Primary Actor | Needy |
| Precondition | Needy Should be on profile page |
| Postcondition | View request book |
| Main Flow | 1. Needy clicks to profile page  2. Needy view requested books statistics  3. Clicks on requested books  4. View all requested book details |
| Alternate Flow | 4a. No book requested |

### Search Book:

Table 3.18: Fully Dressed Use Case (Search Book)

|  |  |
| --- | --- |
| USE CASE ID | UC-014 |
| USE CASE NAME | Search Book |
| PRIMARY ACTOR | Needy |
| PRECONDITION | Needy must be on the dashboard |
| POSTCONDITION | Searched books will be displayed |
| MAIN FLOW | 1. Needy accesses the dashboard  2. Needy enters book title in the search bar  3. System fetches matching books from the database  4. System displays the list of books that matches the title. |
| ALTERNATE FLOW | 3a. If no match is found, system displays “No books found” |

### Donated Book Statistics:

Table 3.19: Fully Dressed Use Case (Donated Book Statistics)

|  |  |
| --- | --- |
| Use Case ID | UC-015 |
| Use Case Name | Donated Book Statistics |
| Primary Actor | Donor |
| Precondition | Donor Should be on Profile page |
| Postcondition | View donated books |
| Main Flow | 1. Donor clicks on profile page  2. Donor view donated book statistics  3. Clicks on donated books  4. View all donated books |
| Alternate Flow | 4a. No donated books found |

### Manage Request:

Table 3.20: Fully Dressed Use Case (Manage Request)

|  |  |
| --- | --- |
| USE CASE ID | UC-016 |
| USE CASE NAME | Manage Request |
| PRIMARY ACTOR | Admin |
| PRECONDITION | Admin must be logged in |
| POSTCONDITION | Request status will be updated to either **Accepted**, **Rejected**, or **Completed** |
| MAIN FLOW | 1. Admin logs in and navigates to the Dashboard  2. Admin clicks on "Manage Requests" tab  3. System displays all pending book requests  4. Admin selects a request  5. Admin updates the request status (Accepted/Rejected/Completed)  6. System updates the request status in the database |
| ALTERNATE FLOW | 3a. If no requests are available, system shows message: “No data available” |

### Logout:

Table 3.21: Fully Dressed Use Case (Logout)

|  |  |
| --- | --- |
| Use Case ID | UC-017 |
| Use Case Name | Logout |
| Primary Actor | Admin / Donor / Volunteer / Needy |
| Precondition | All user must be Logged in |
| Postcondition | Logout Successfully |
| Main Flow | 1. User is on dashboard  2. Clicks on profile picture or dropdown  3. Selects "Logout" from the menu  4. System terminates session and redirects to login screen |
| Alternate Flow |  |

## Entity Relationship diagram

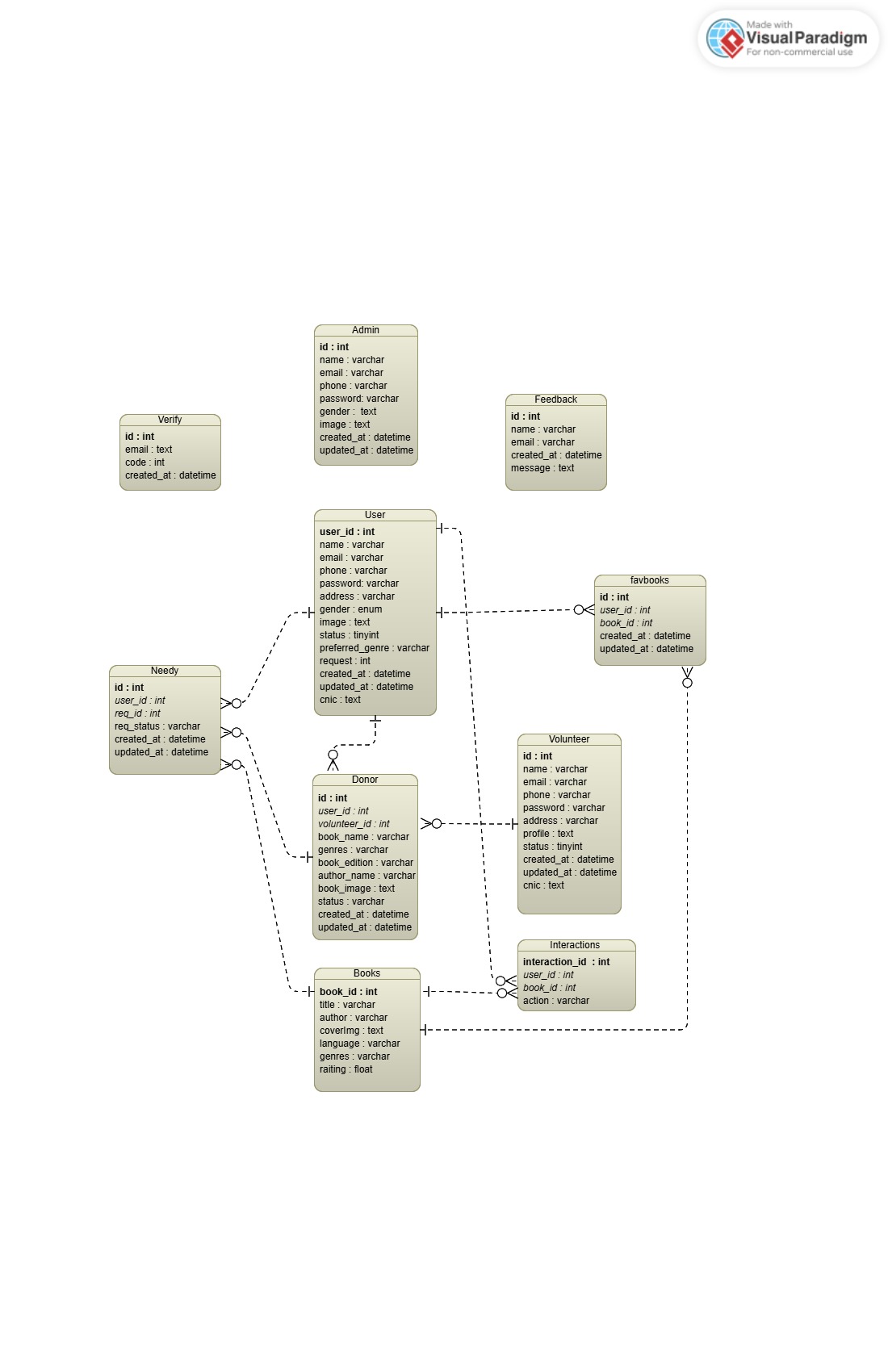


Figure 3.7: Entity Relationship Diagram (ERD)

## Activity Diagram

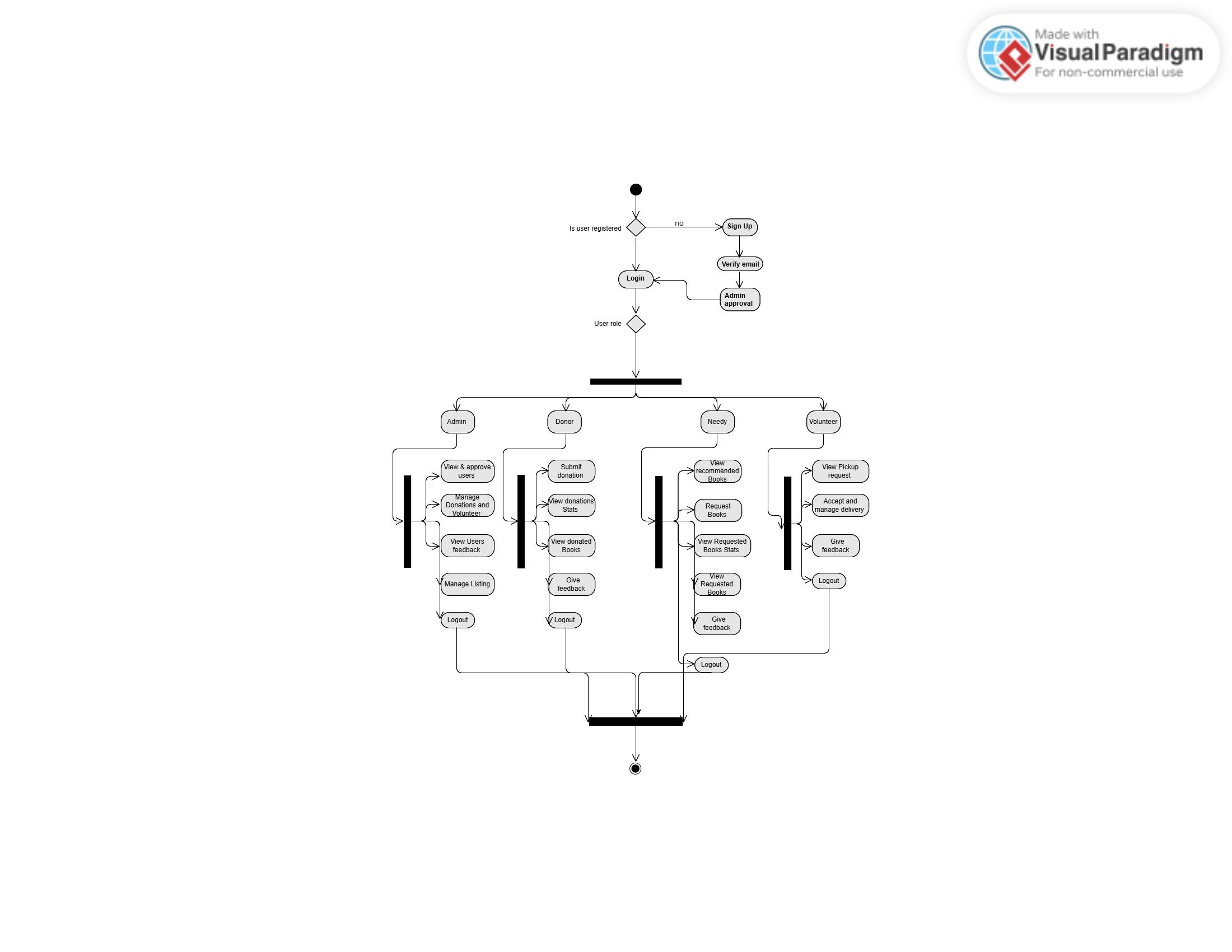
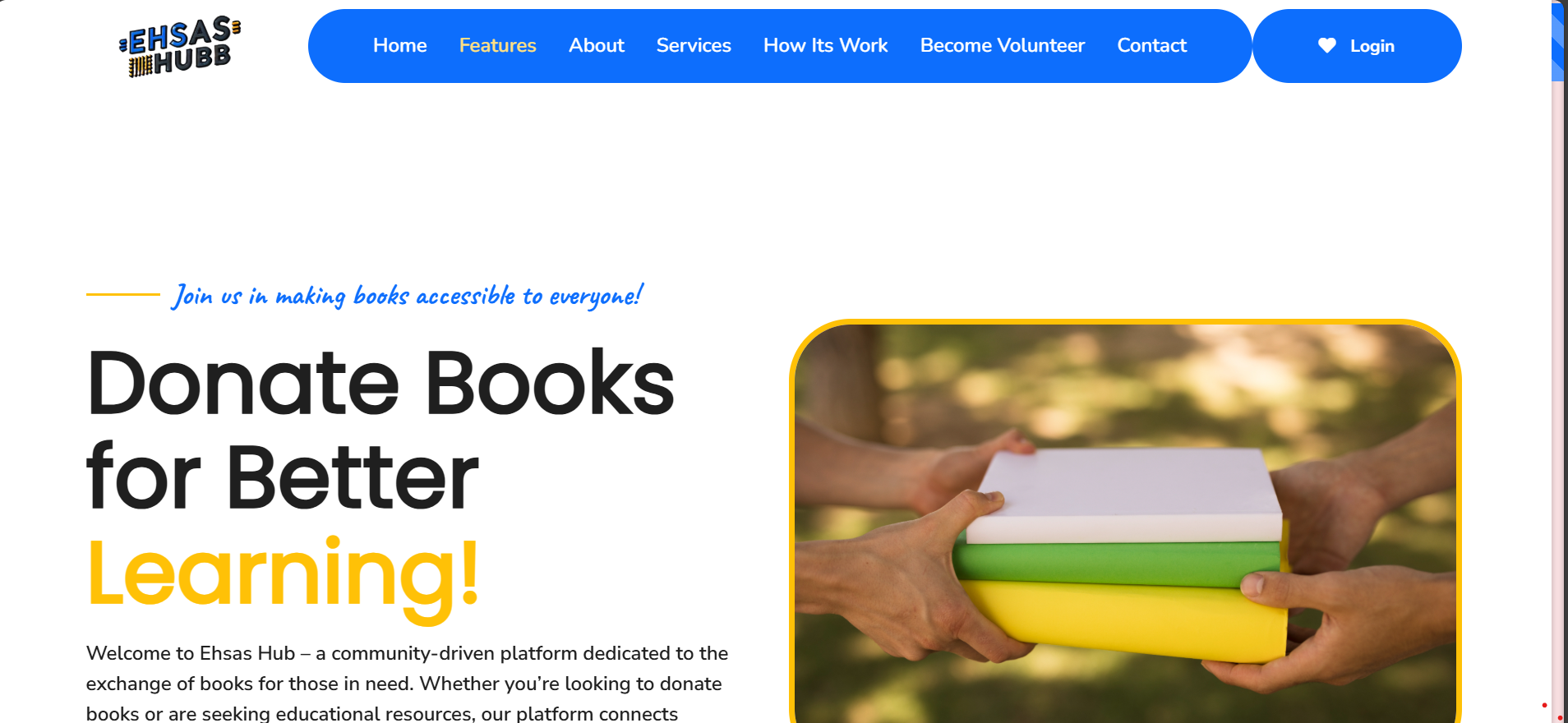


Figure 3.8: Whole system Activity Diagram

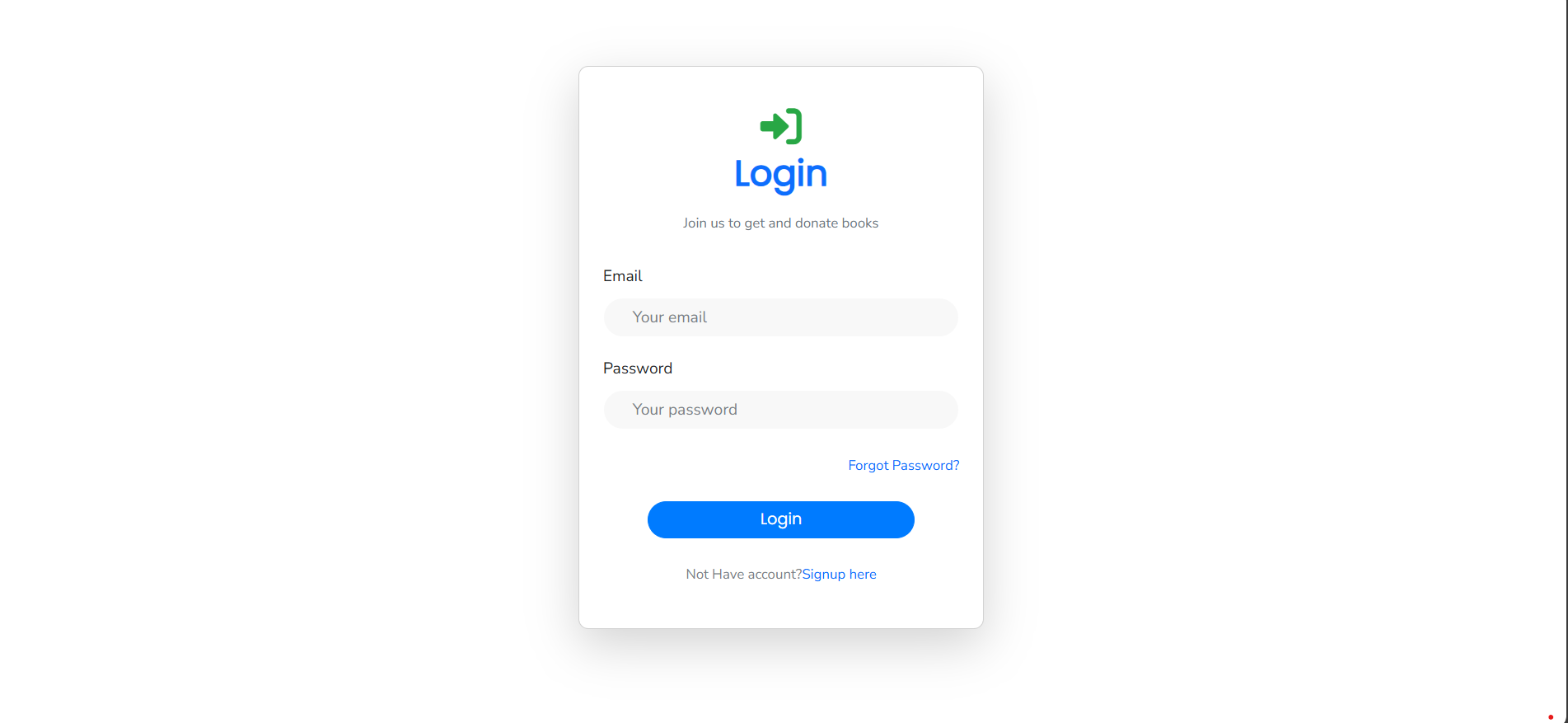
## GUI Graphical User Interfaces

### User Role GUI:

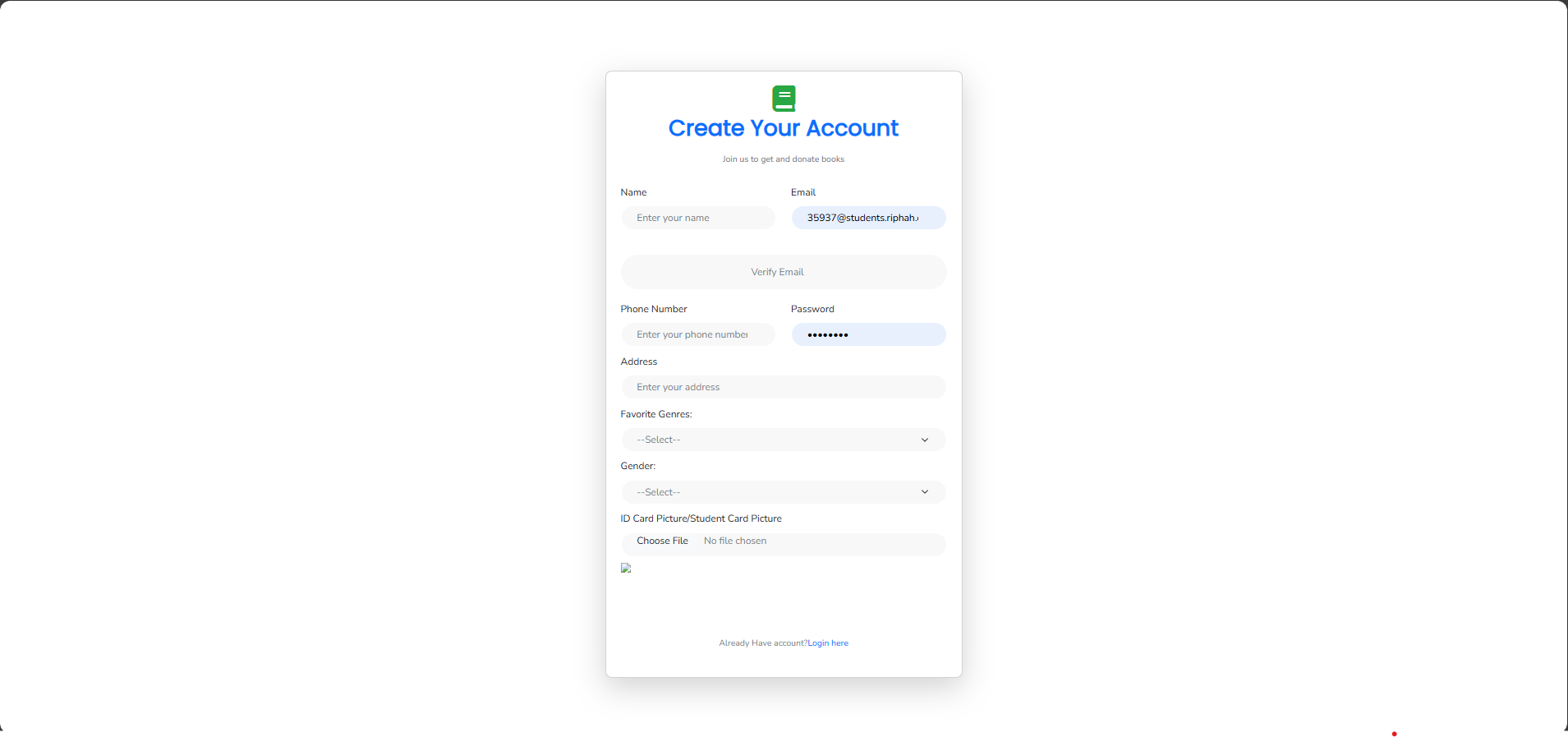
* This is the landing page of Ehsas-hub website.



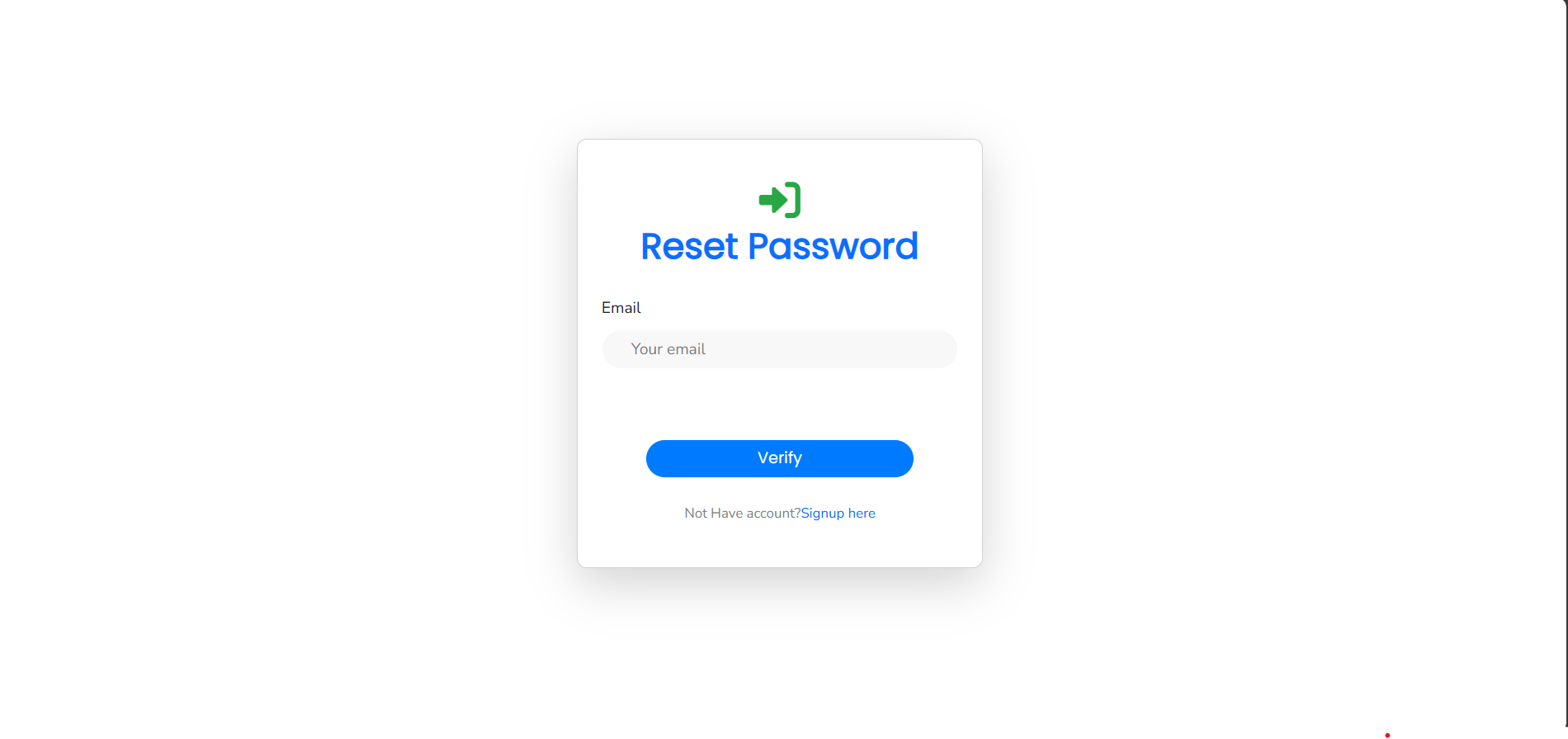
* Login page for user (Needy/Donor).

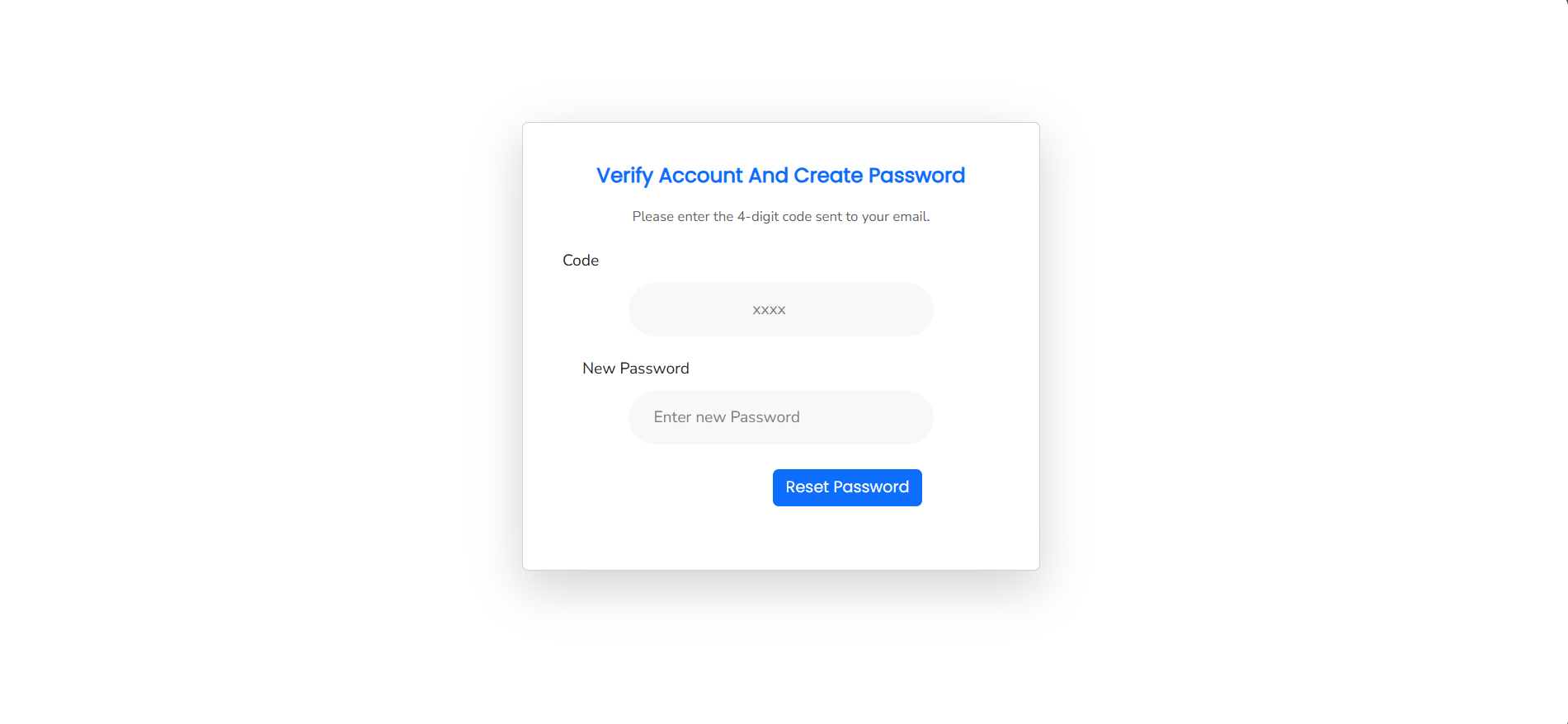


* Sign up for user (Needy/Donor). User will fill this form and verify its provided email after that he can submit this signup request for approval.

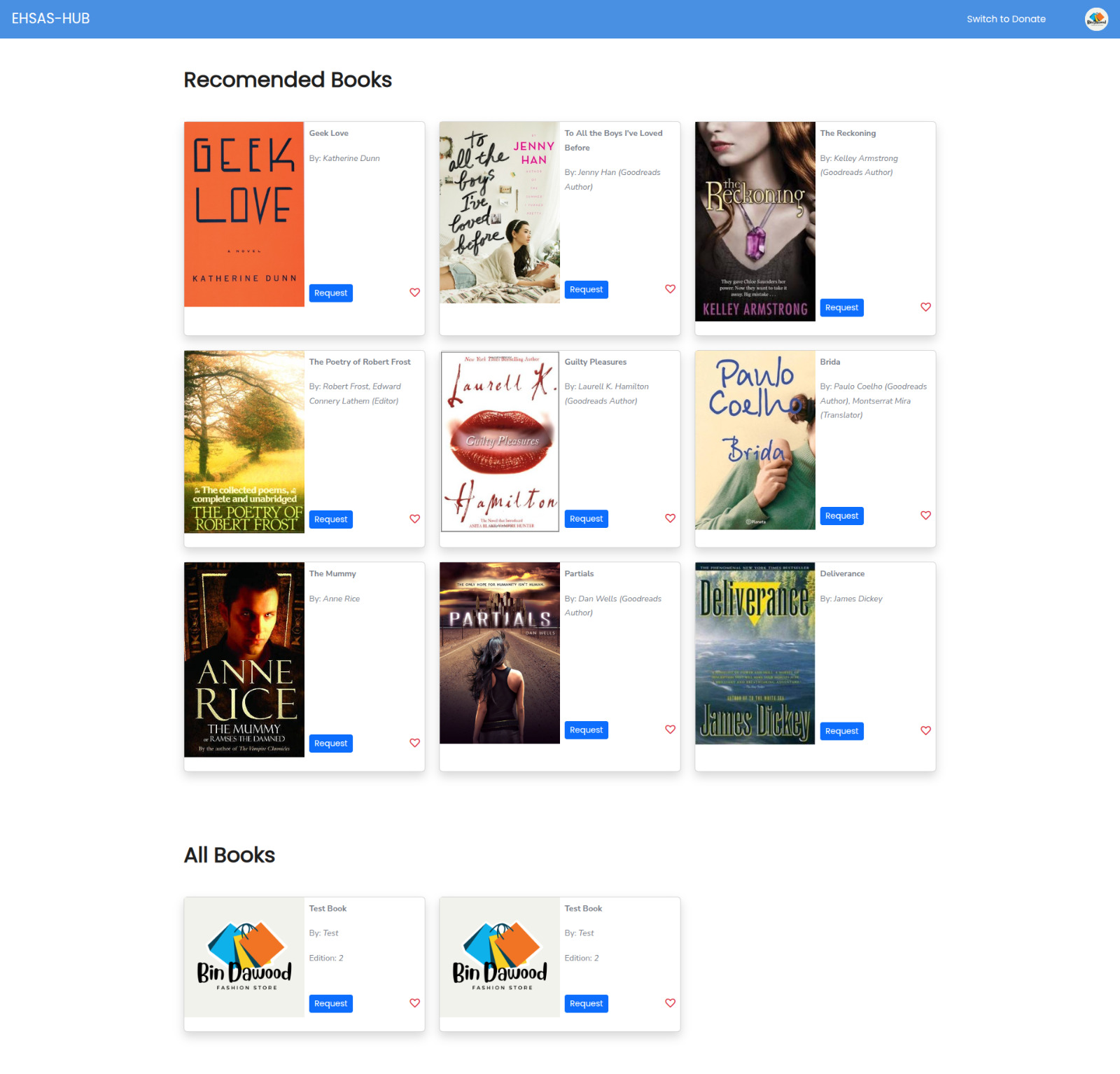


* Password reset page for all user (Needy/Donor/Volunteer/Admin).
* Email will be required and a code will appear in email.
* After entering the correct code, set new password and click the reset button and new password will be saved.

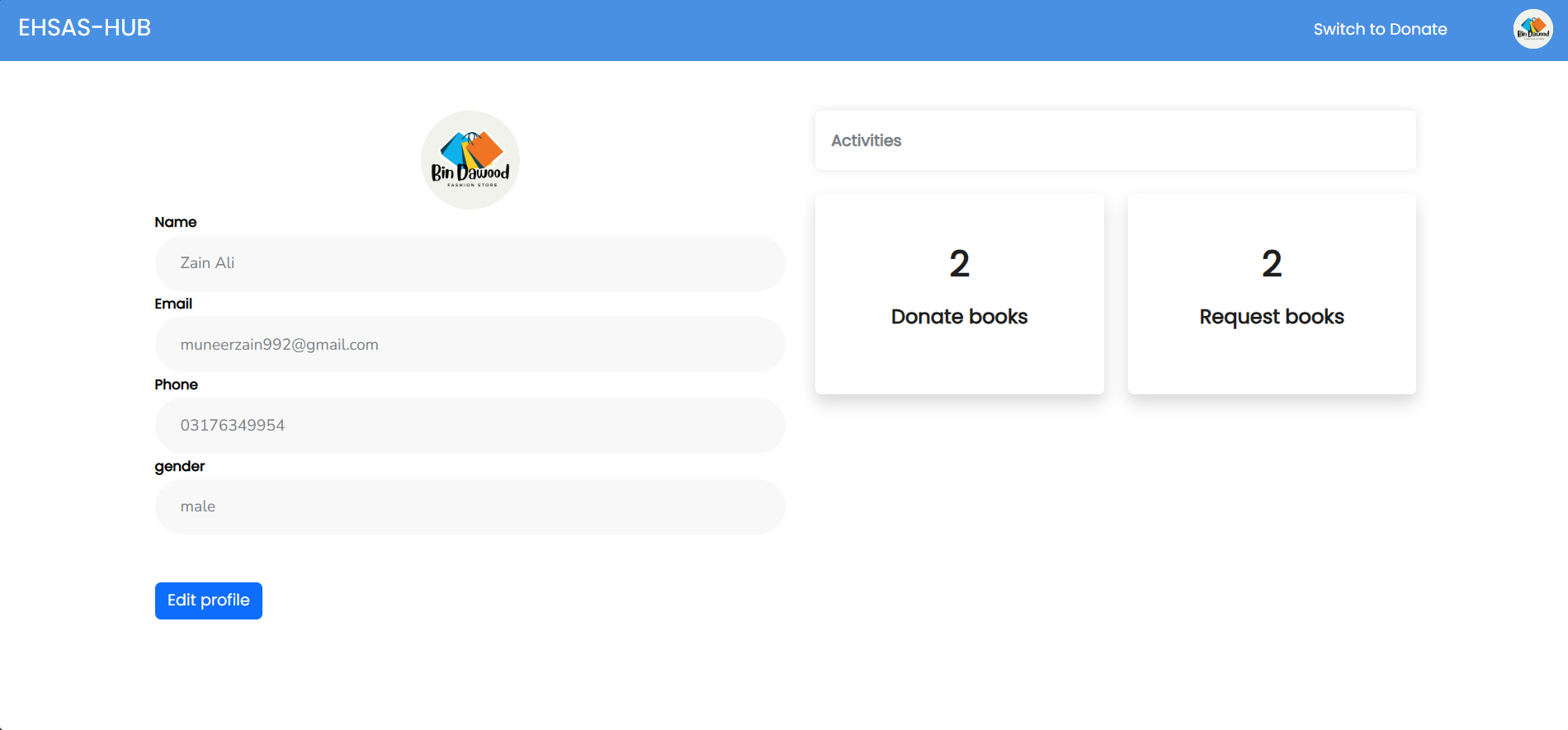


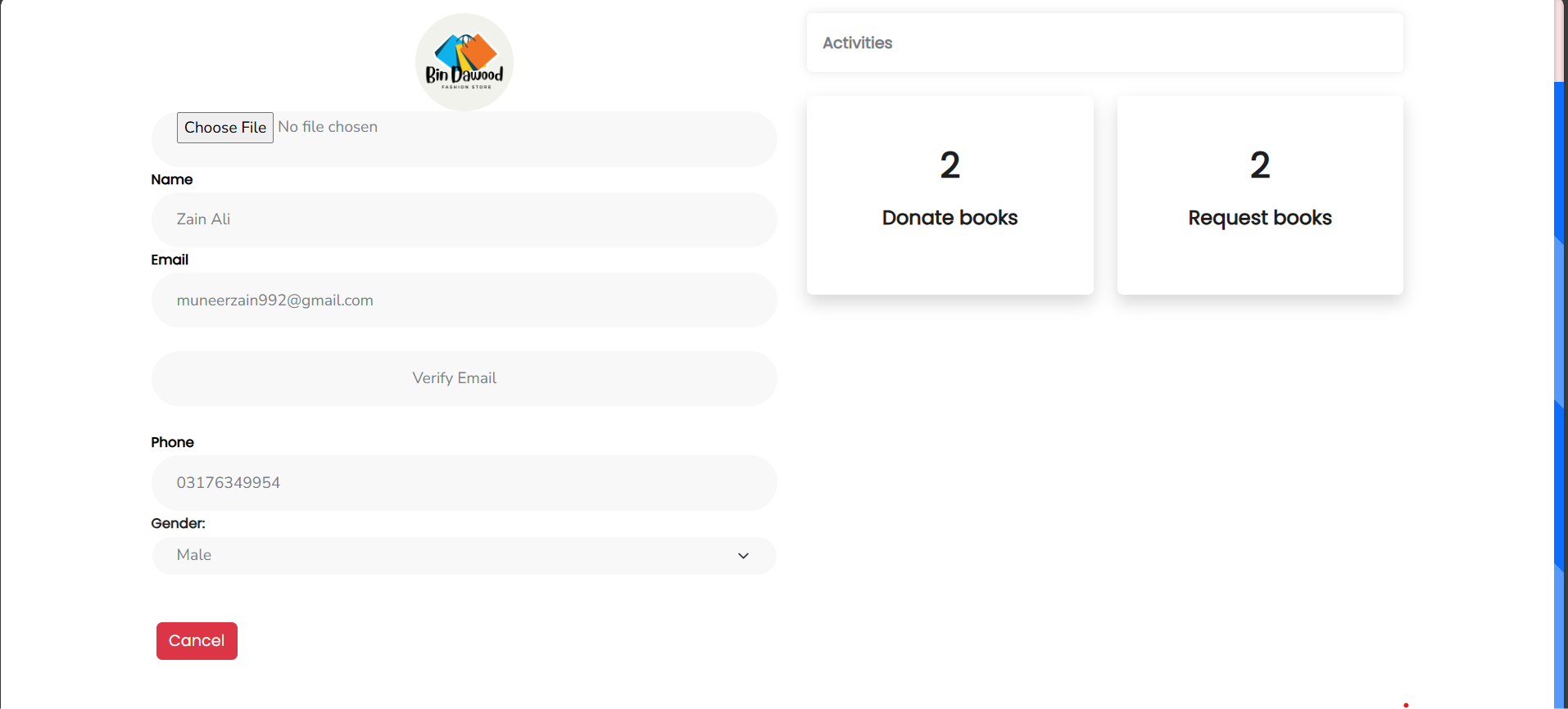


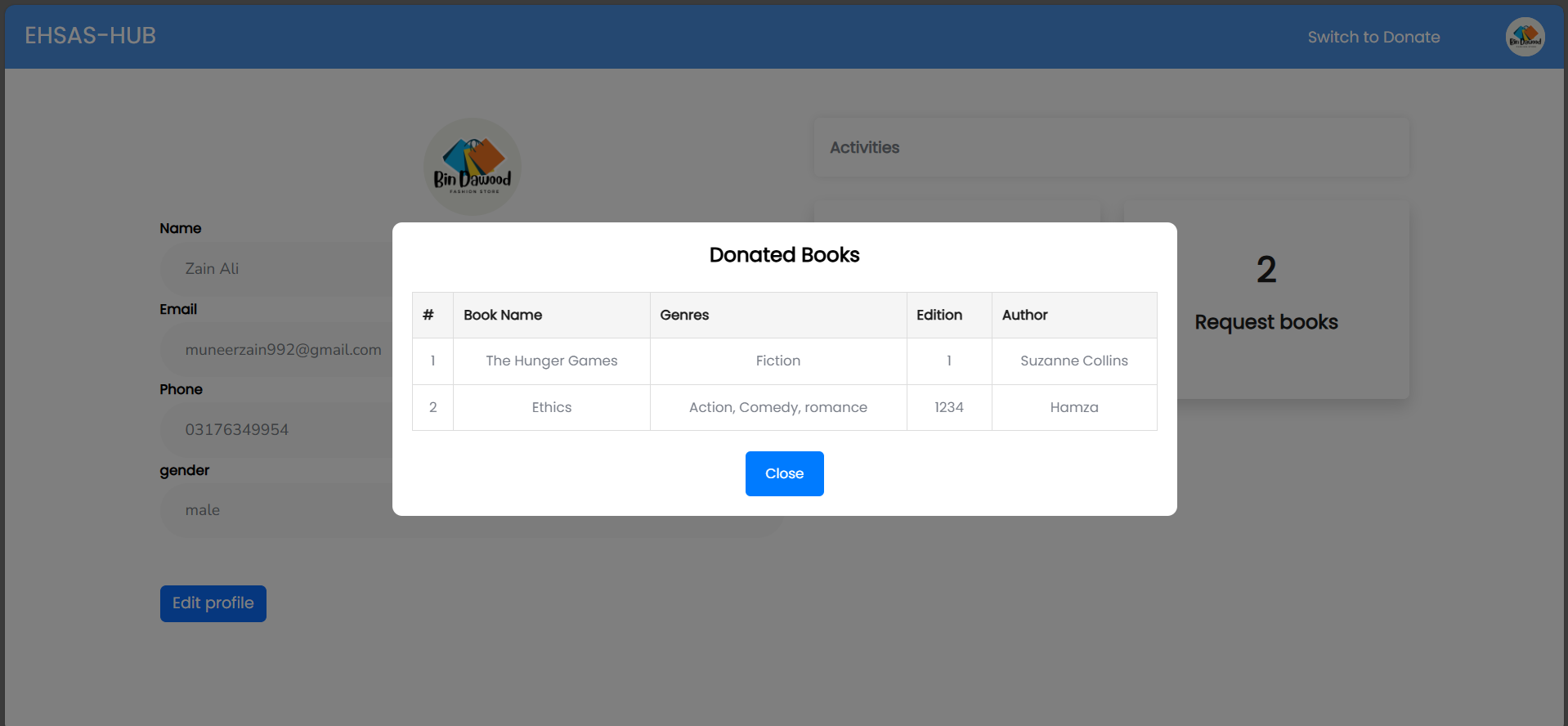
* This is user (Needy/Donor) dashboard.
* Books recommended by model according to preferred genre and all books available on Ehsas-hub will be shown to user.
* If user want to donate any book, he will select switch to donate and a form will appear for donation.

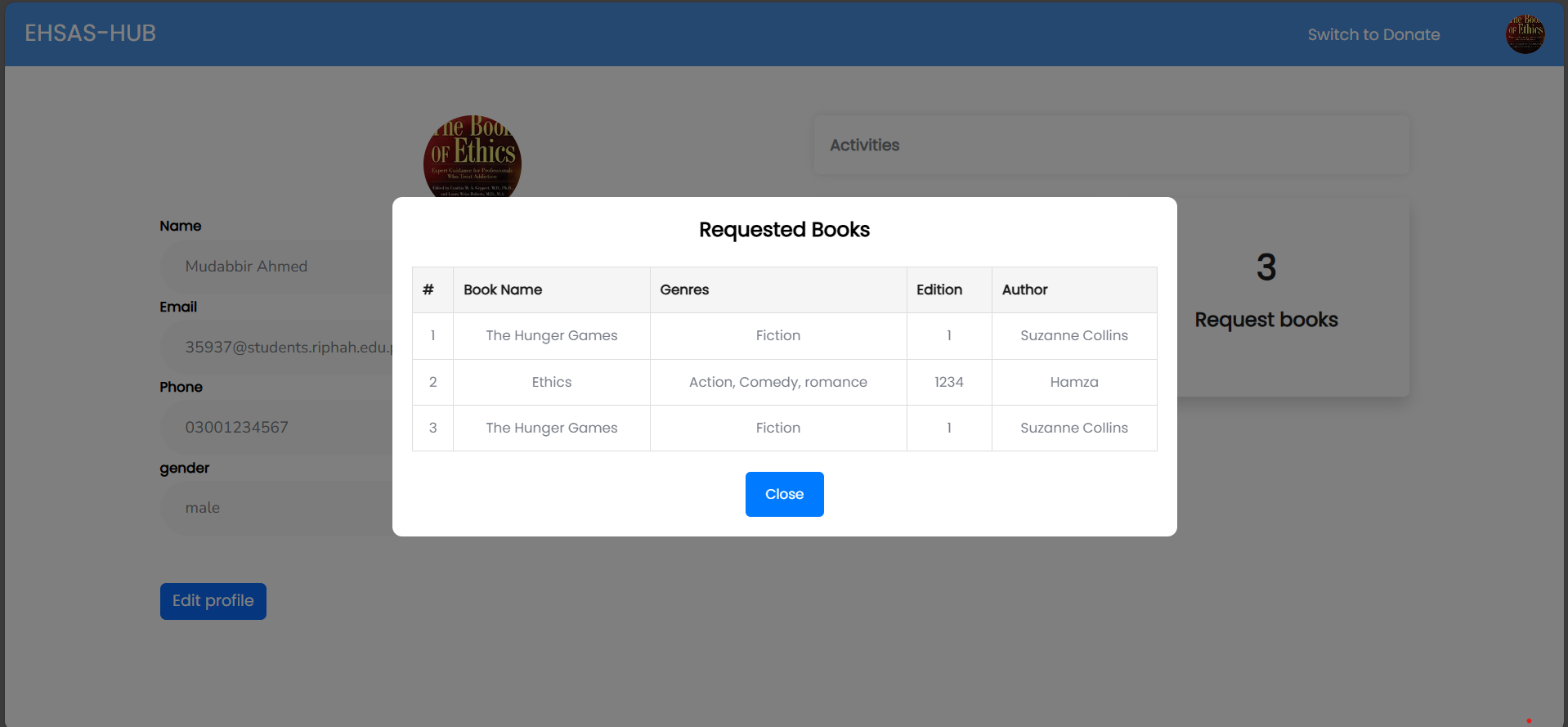


* This is user (Needy/Donor) profile view.
* User can edit profile but every time he will verify register email first then can update its profile.
* View total books donated and requested with details.



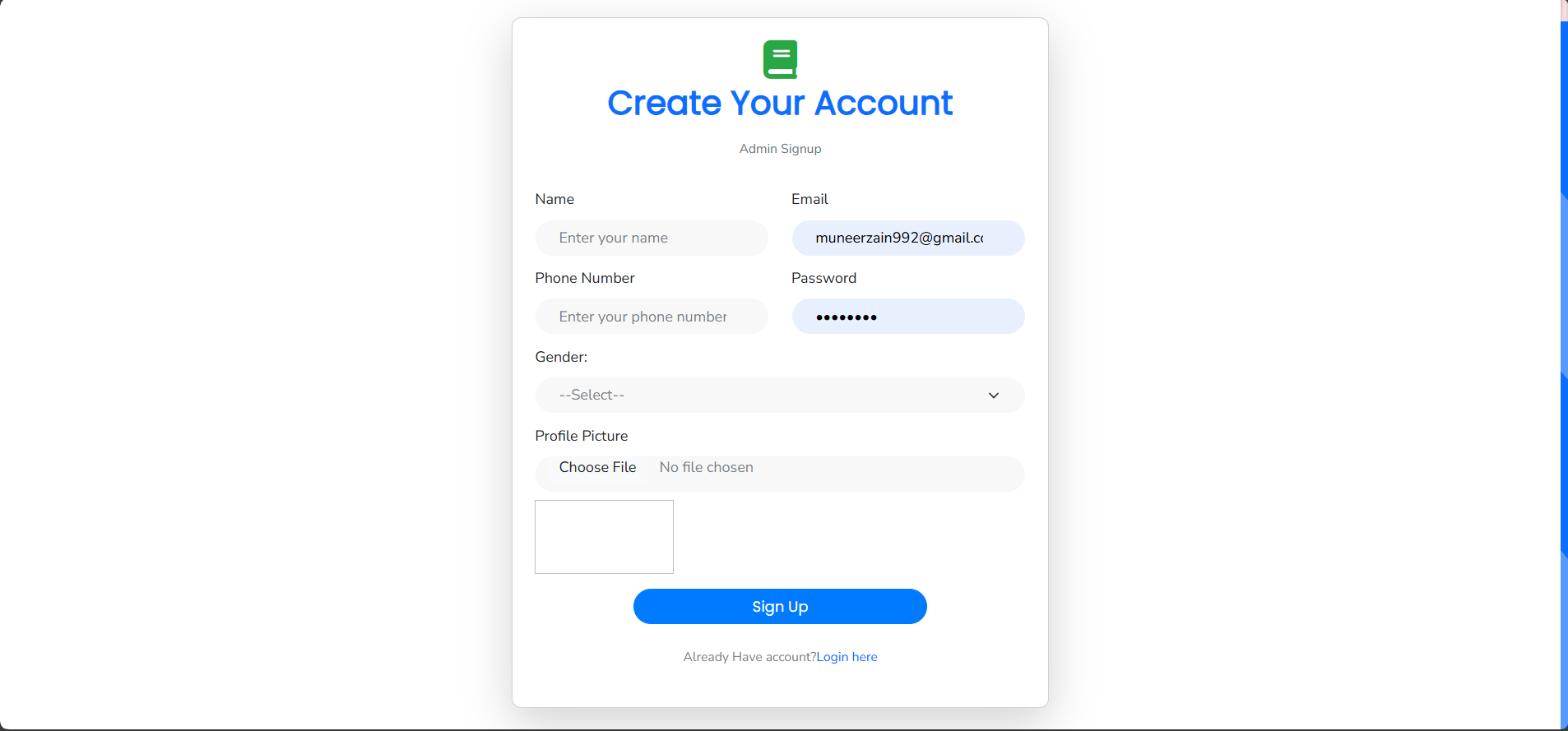


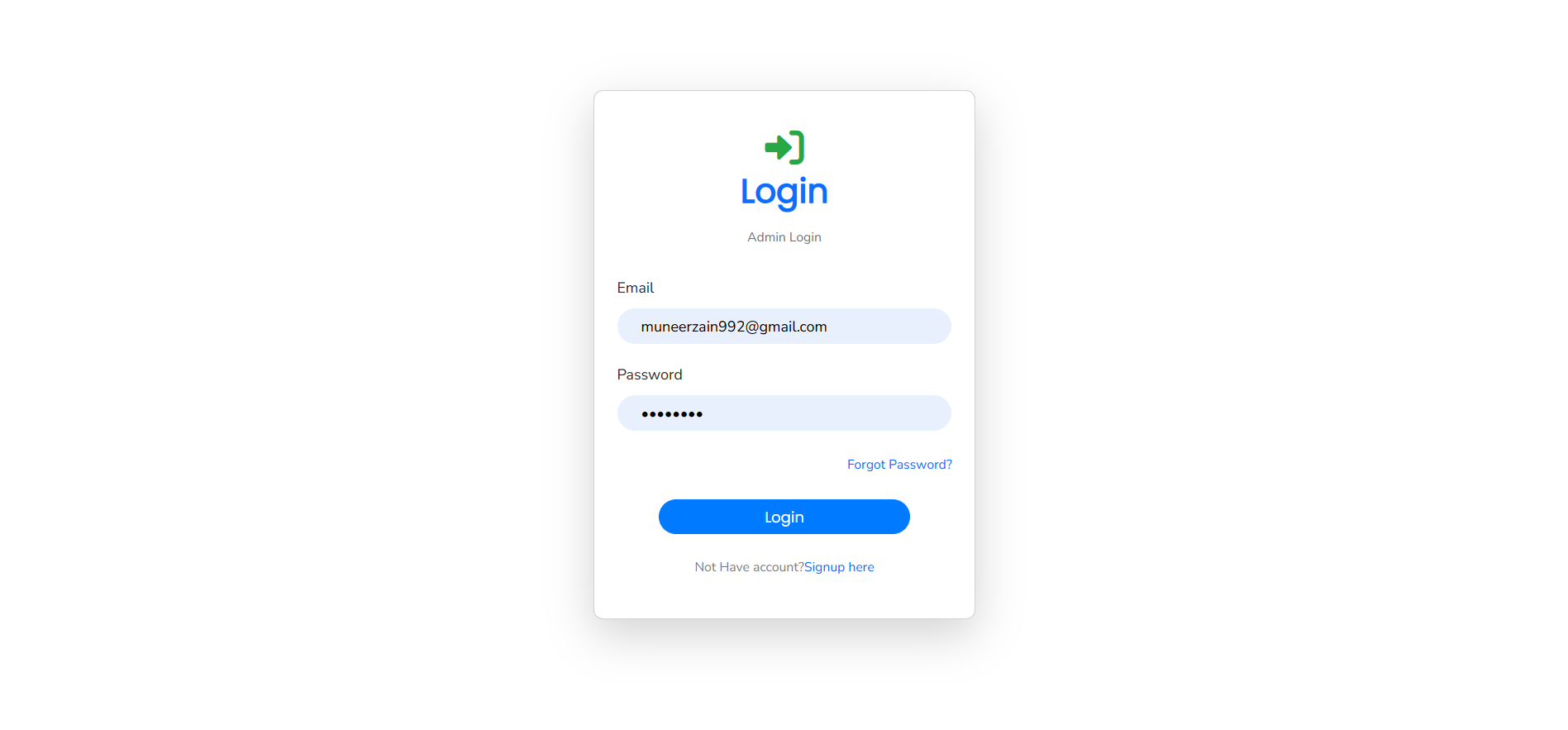




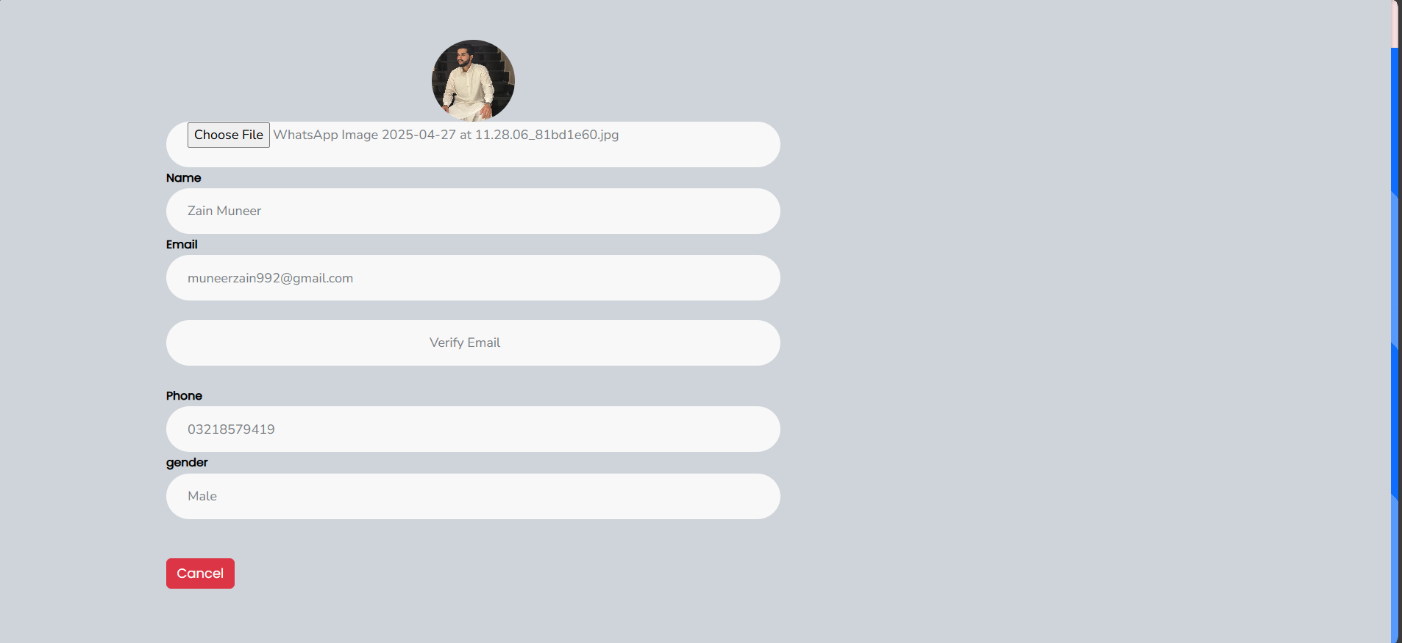
### Admin Role GUI:

* Admin Sign up/login page

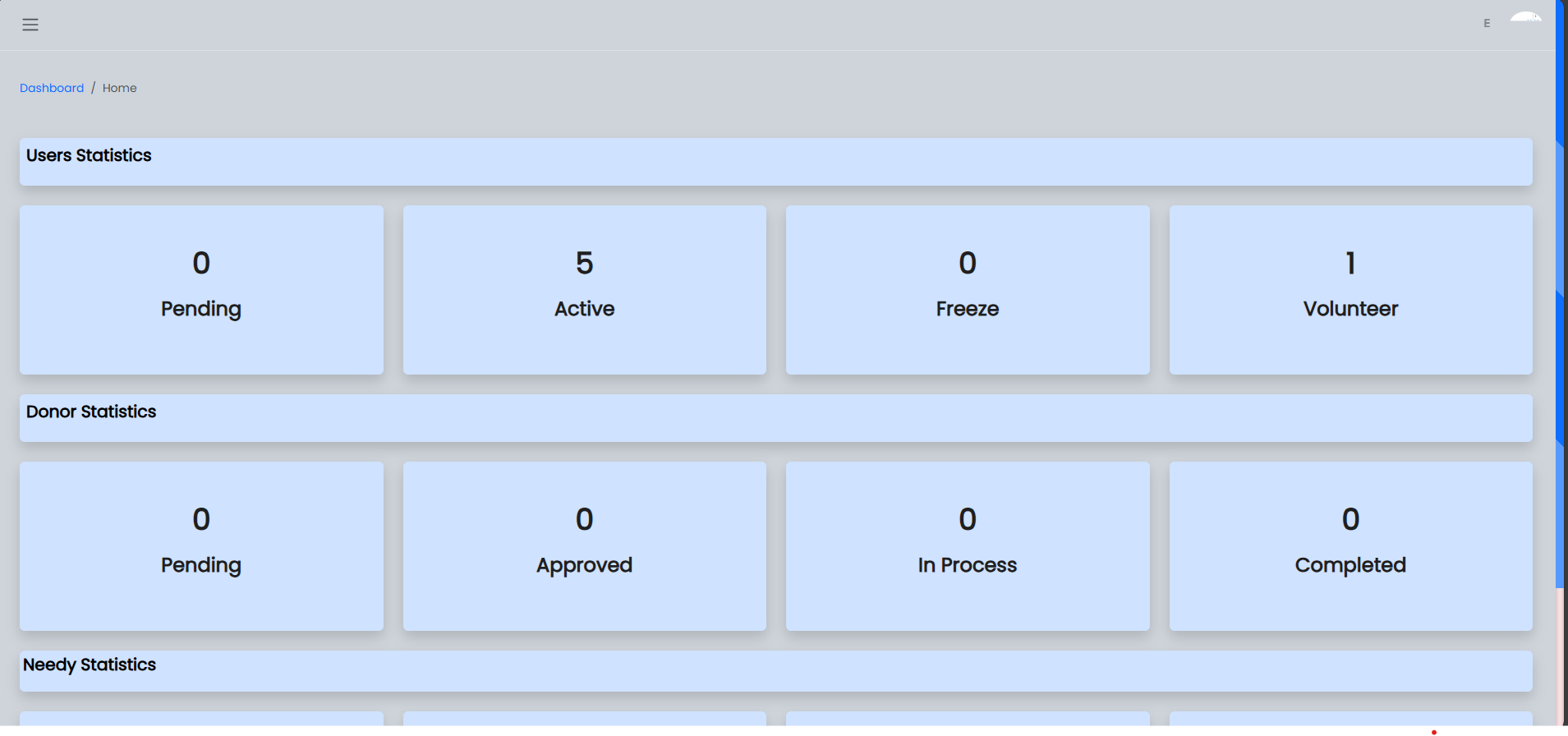




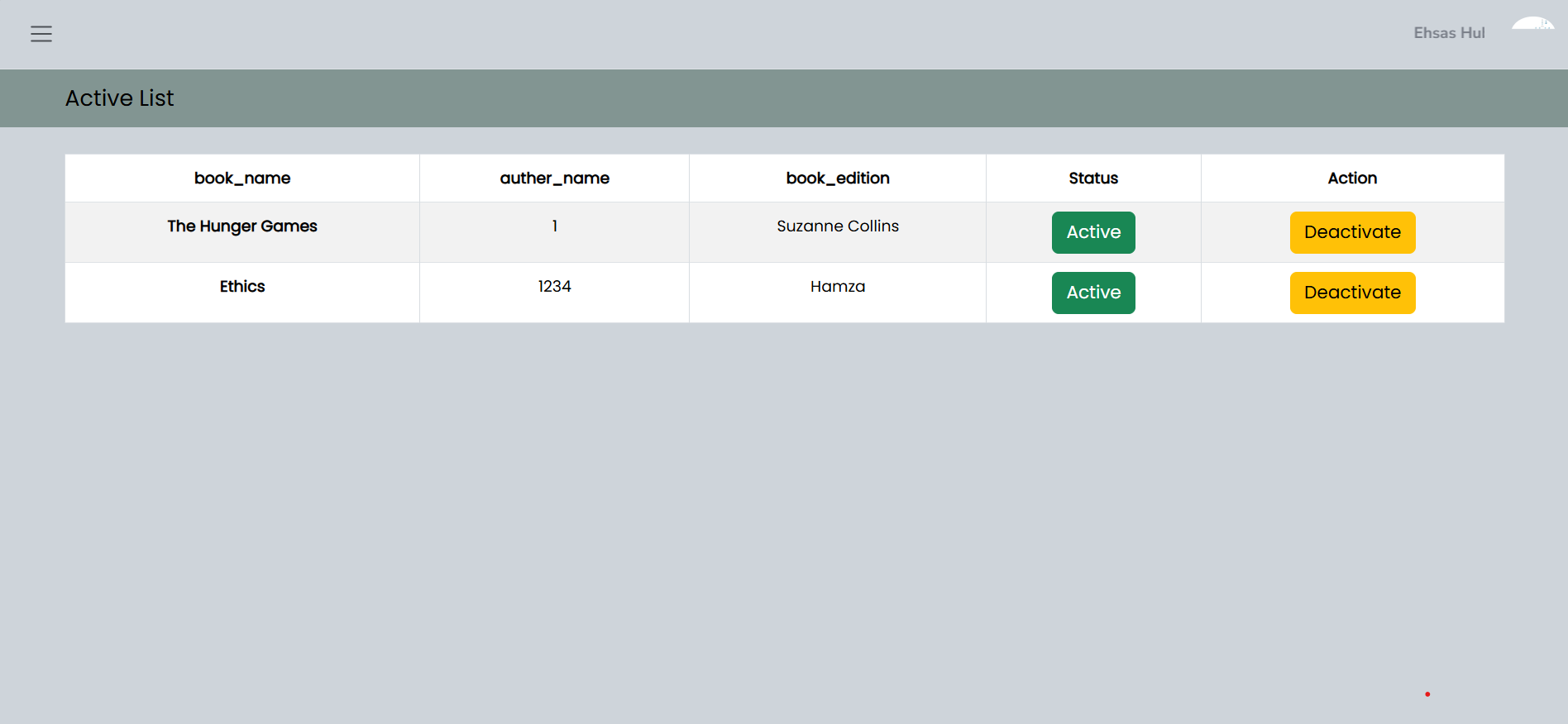
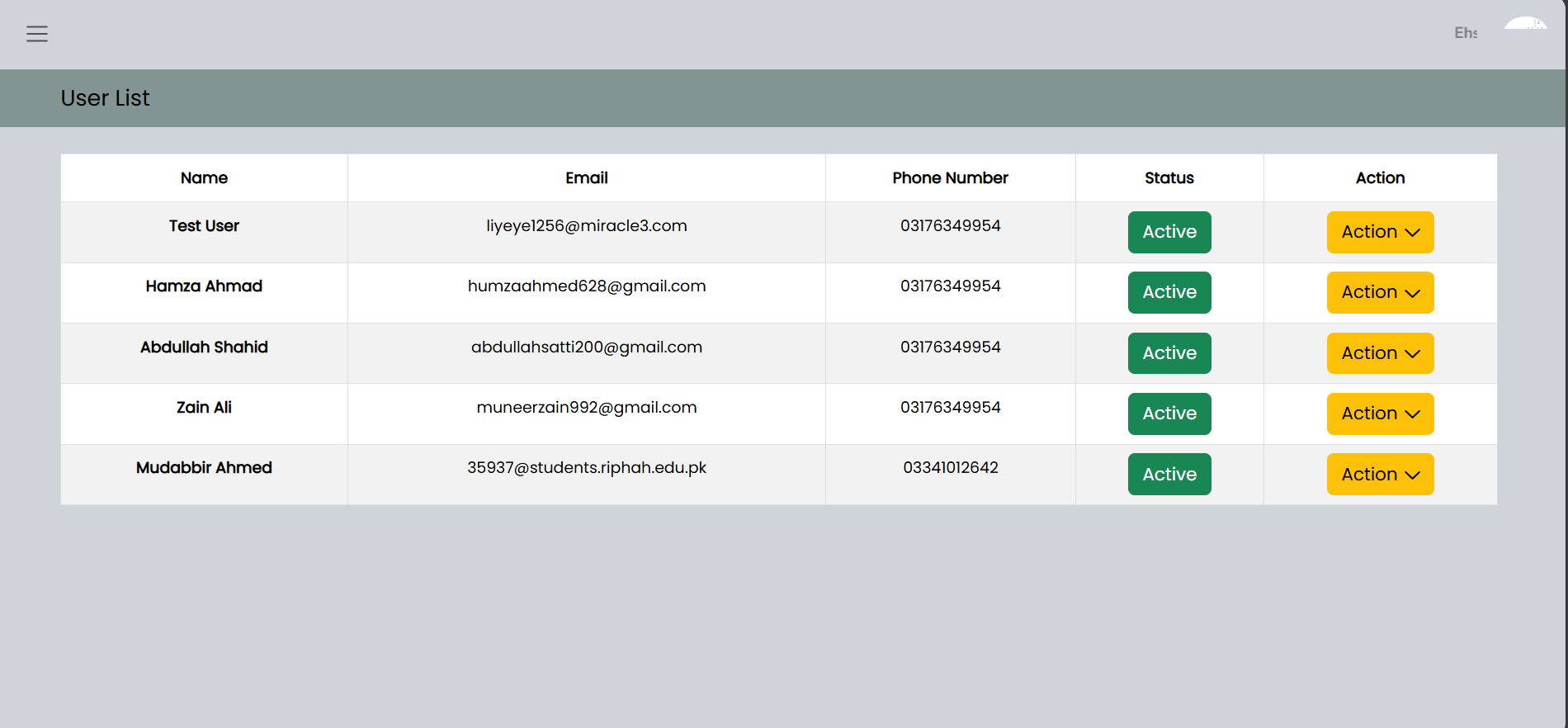
* Admin profile view Page

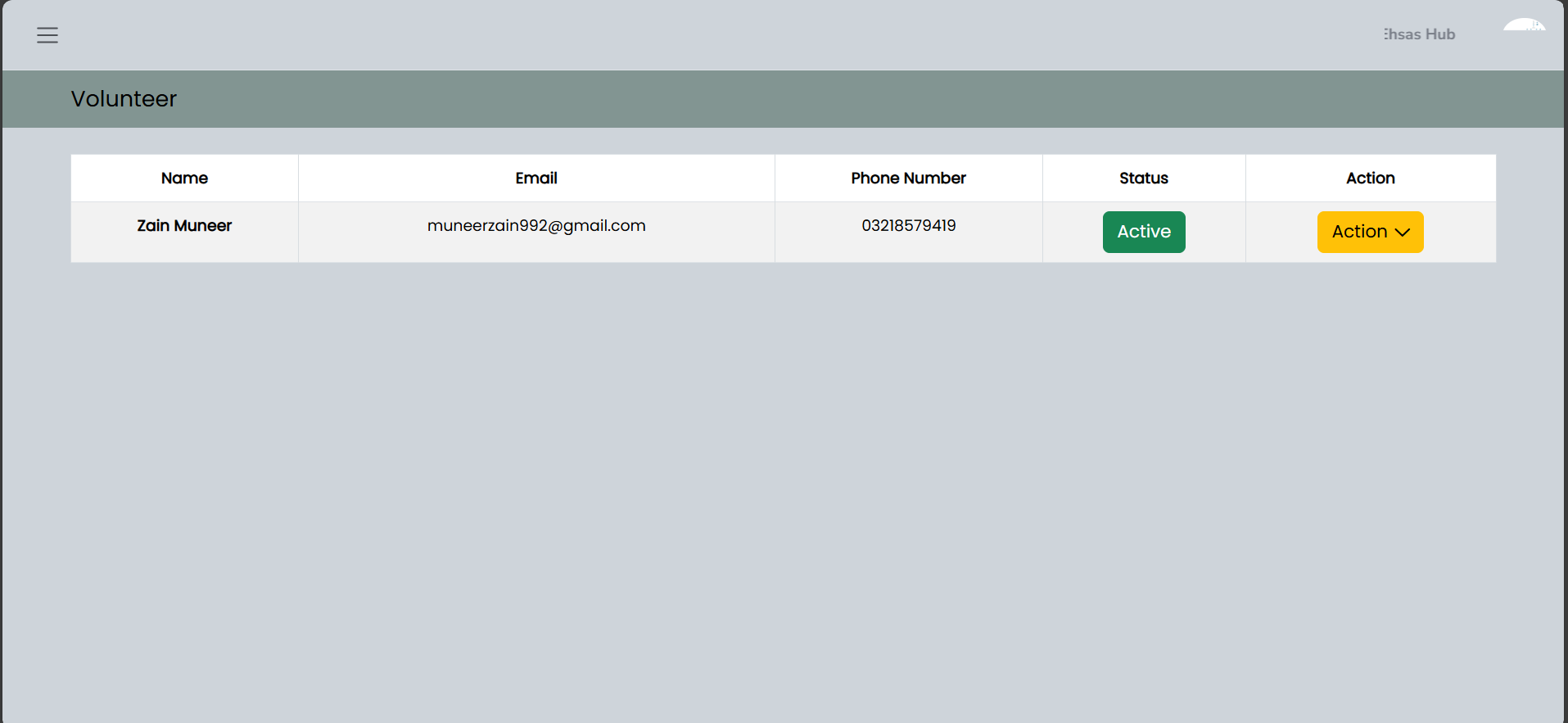


* Admin Statistic page for all users activities

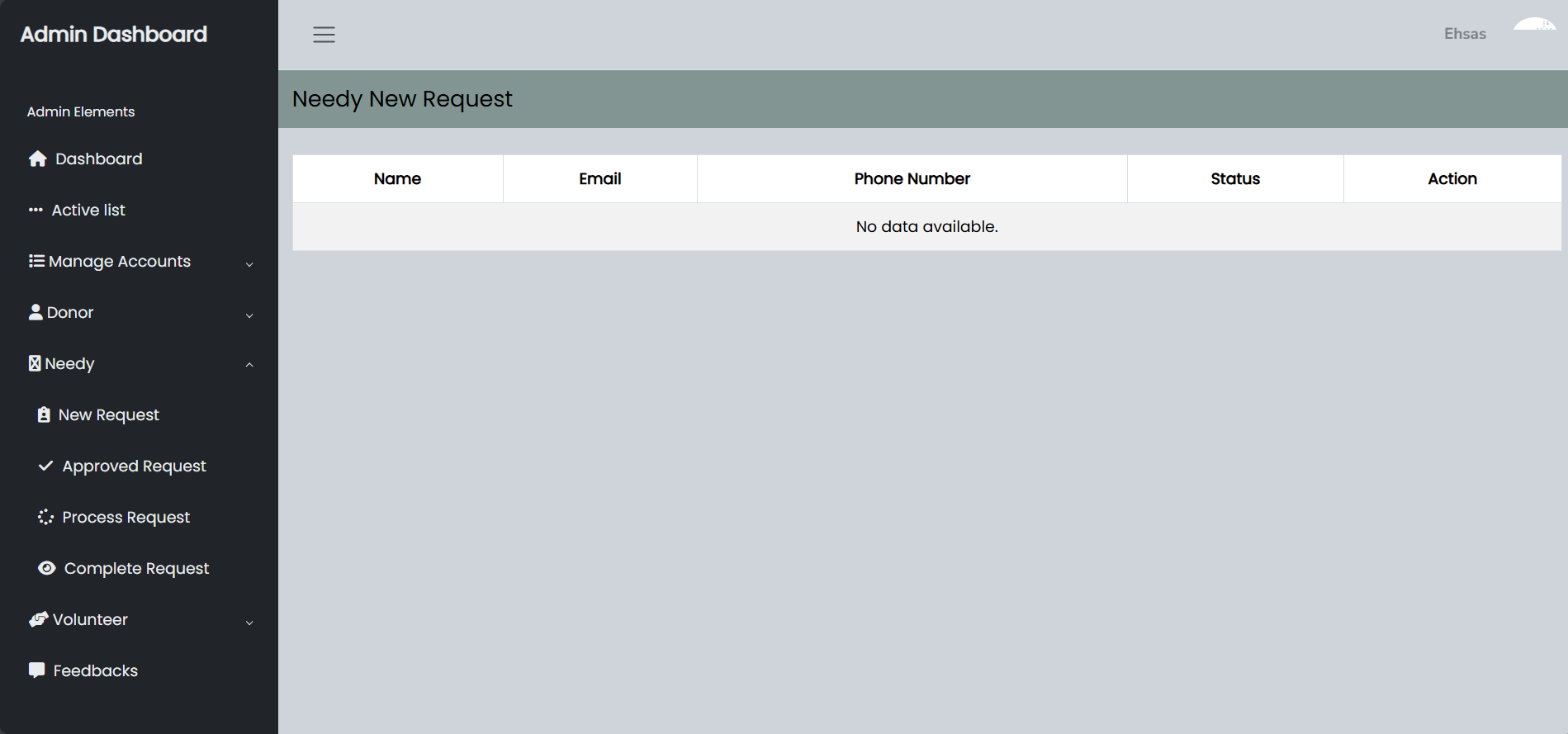


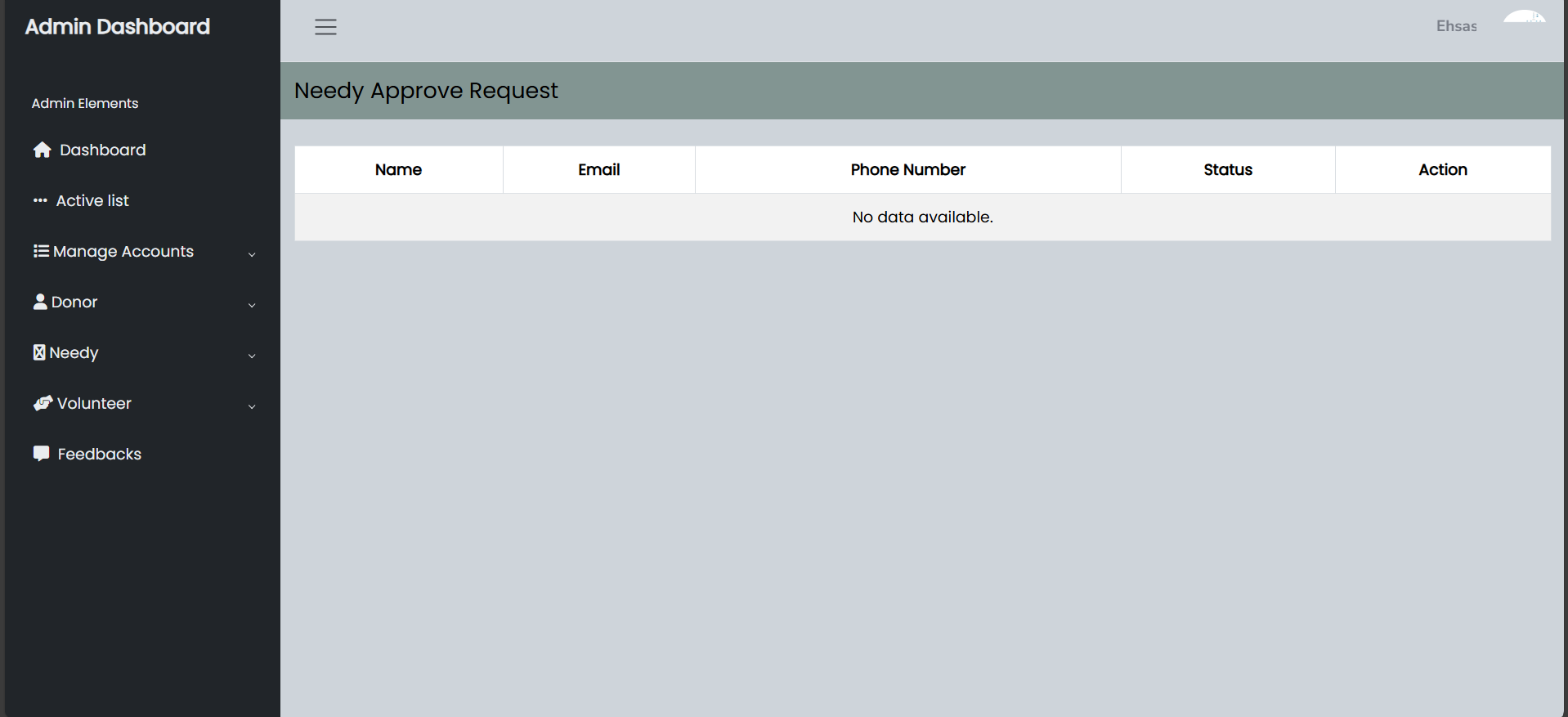
* Active user/book/volunteer list page

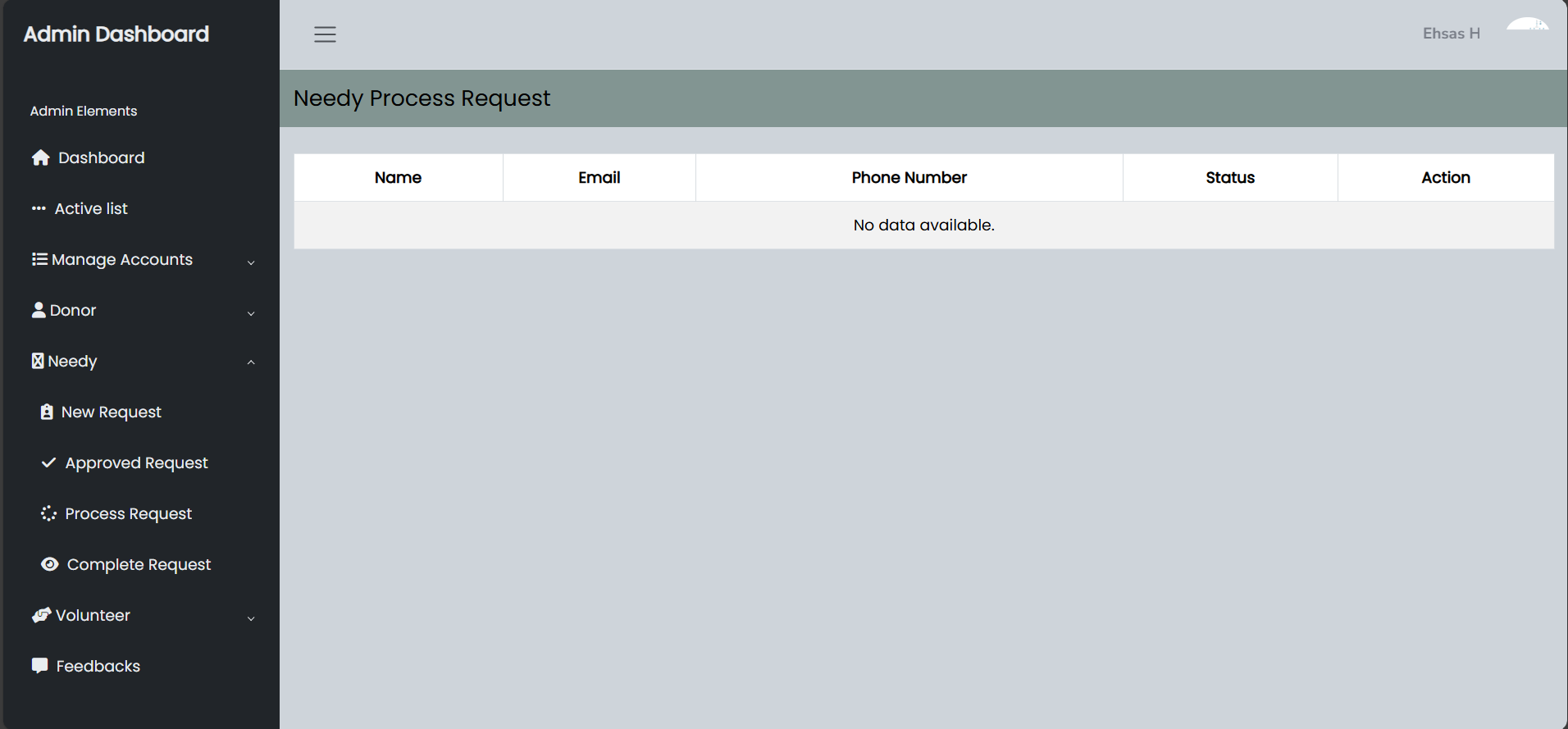
 

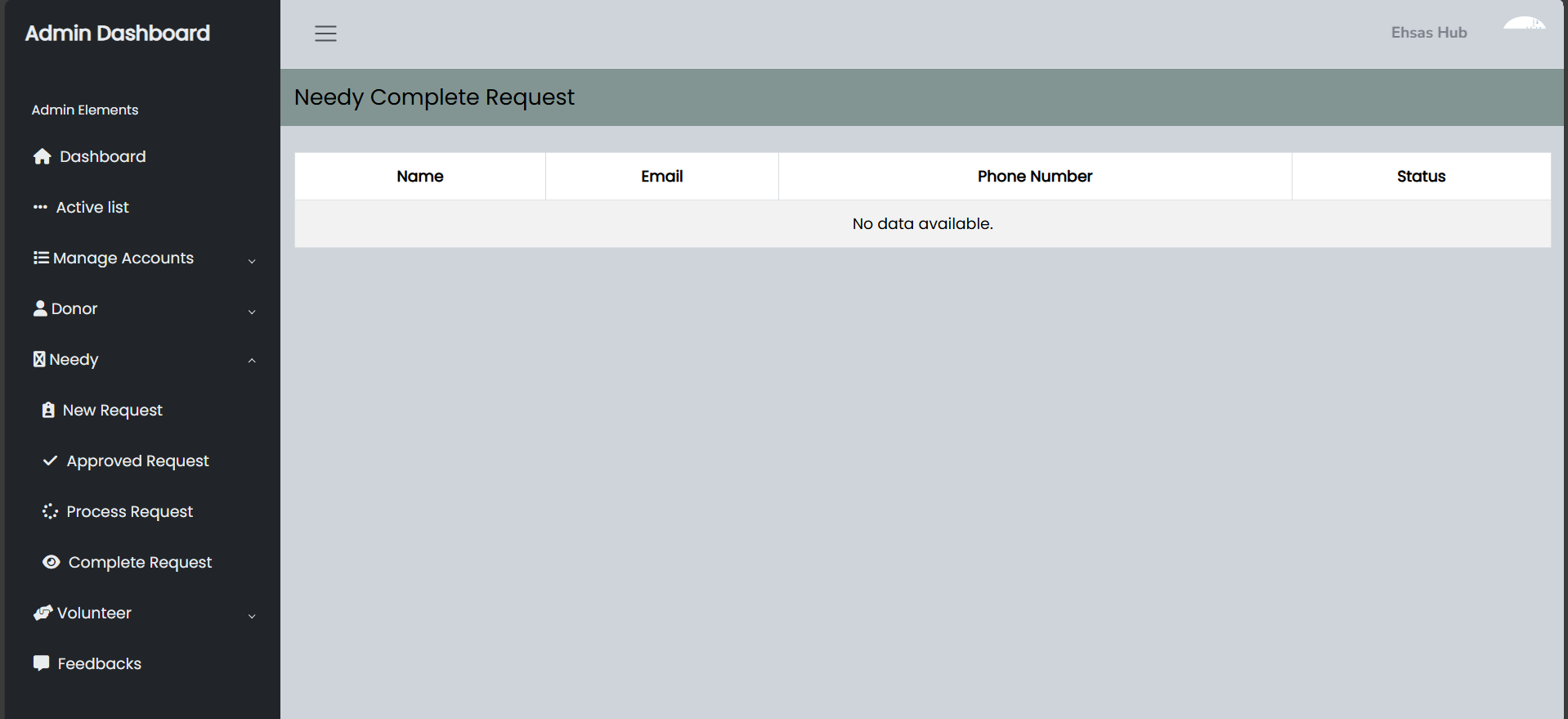


* Needy new/inprocess/approved/completed request

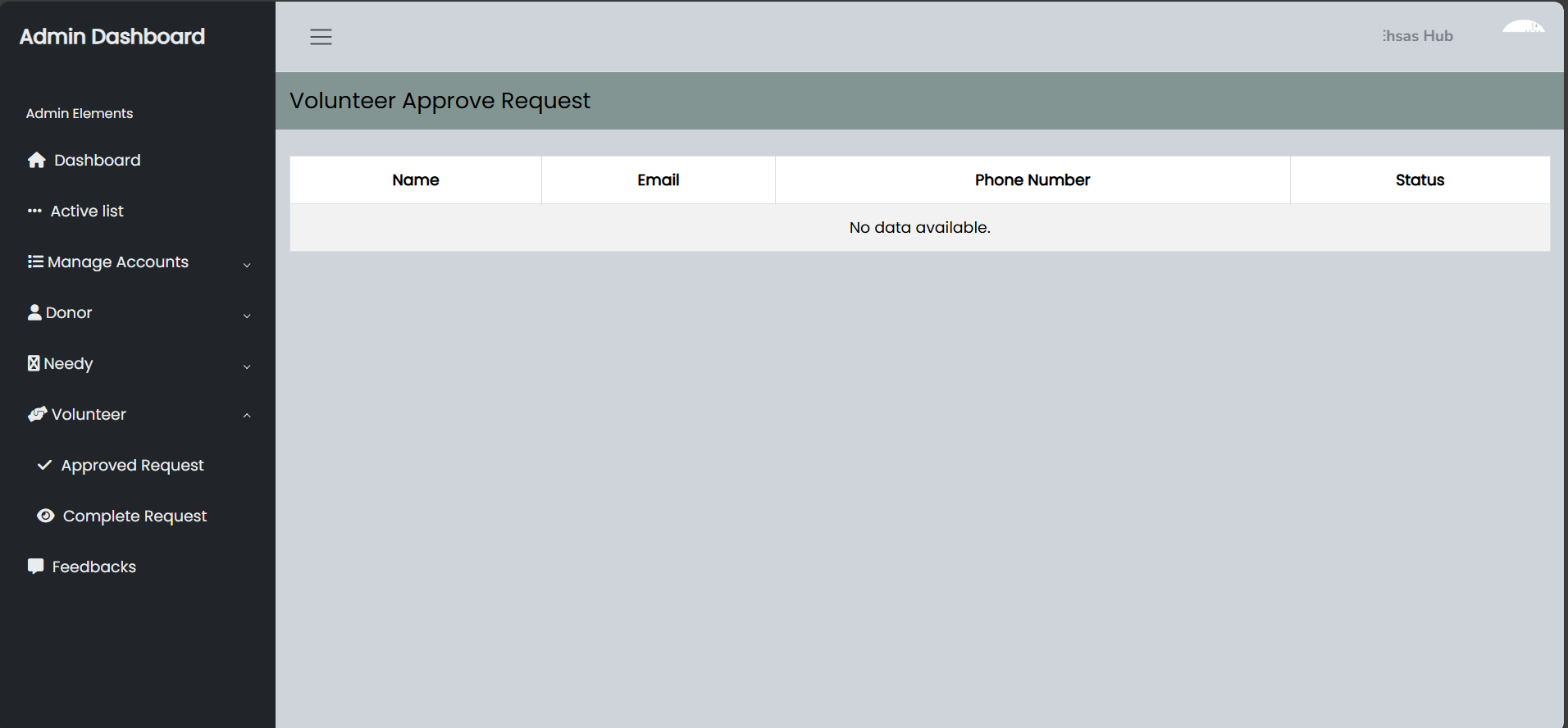


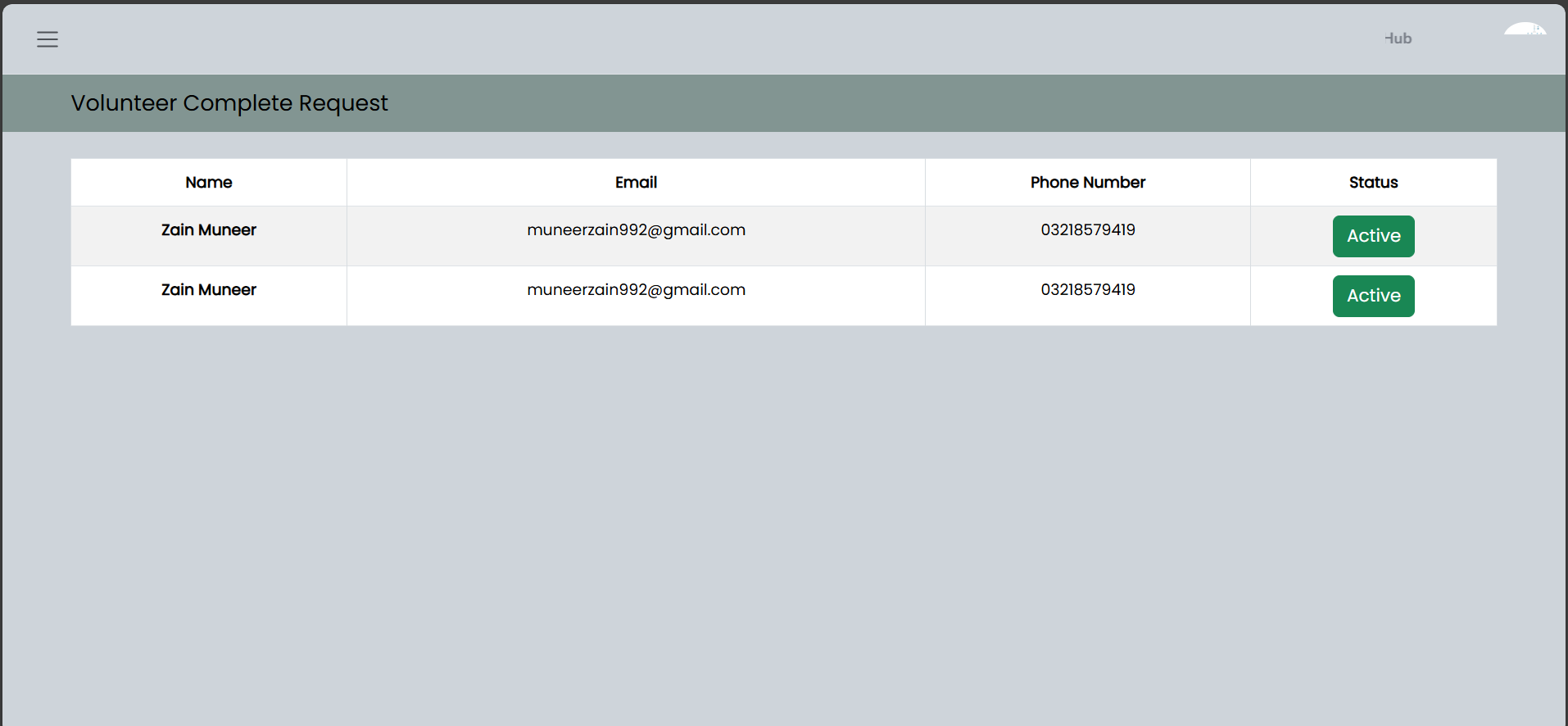




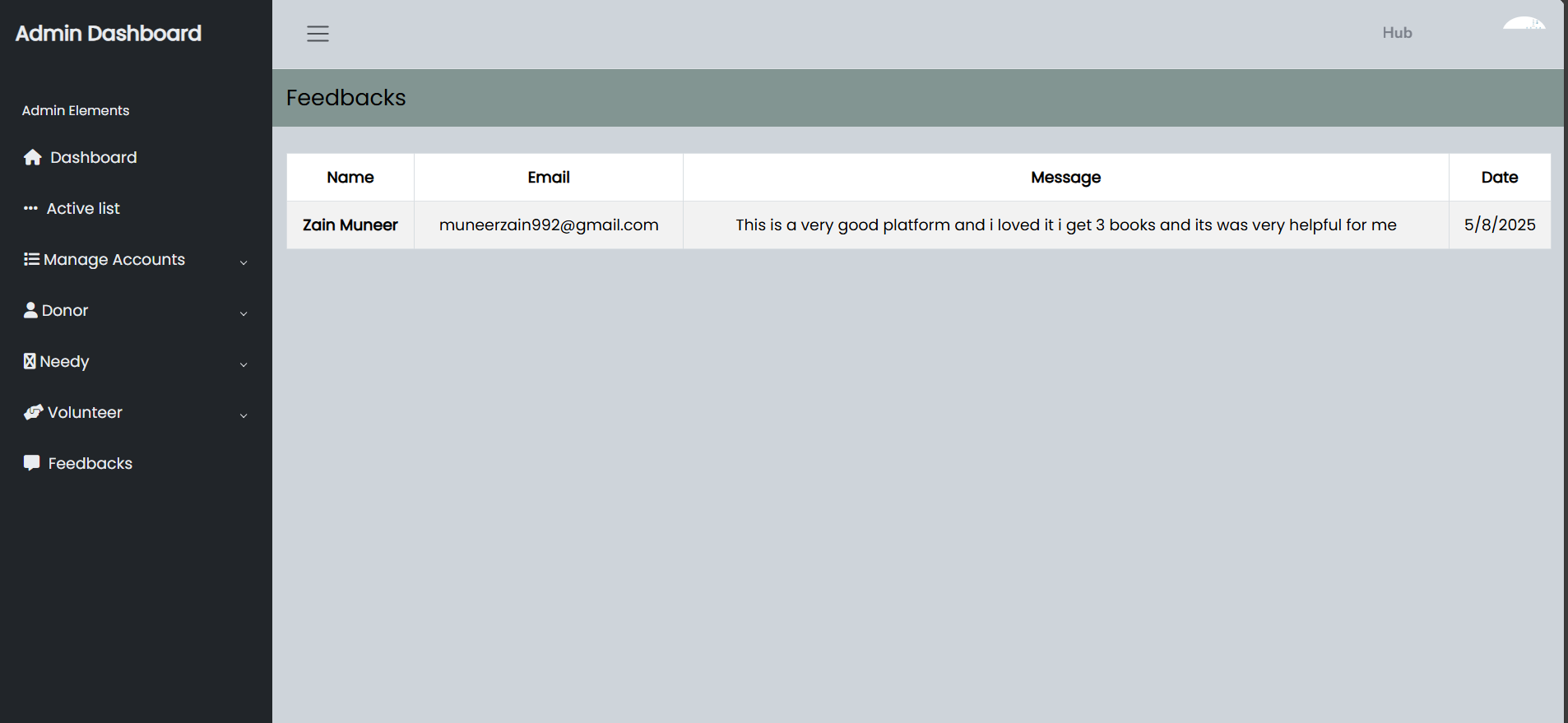


* Volunteer Approved/completed request page.



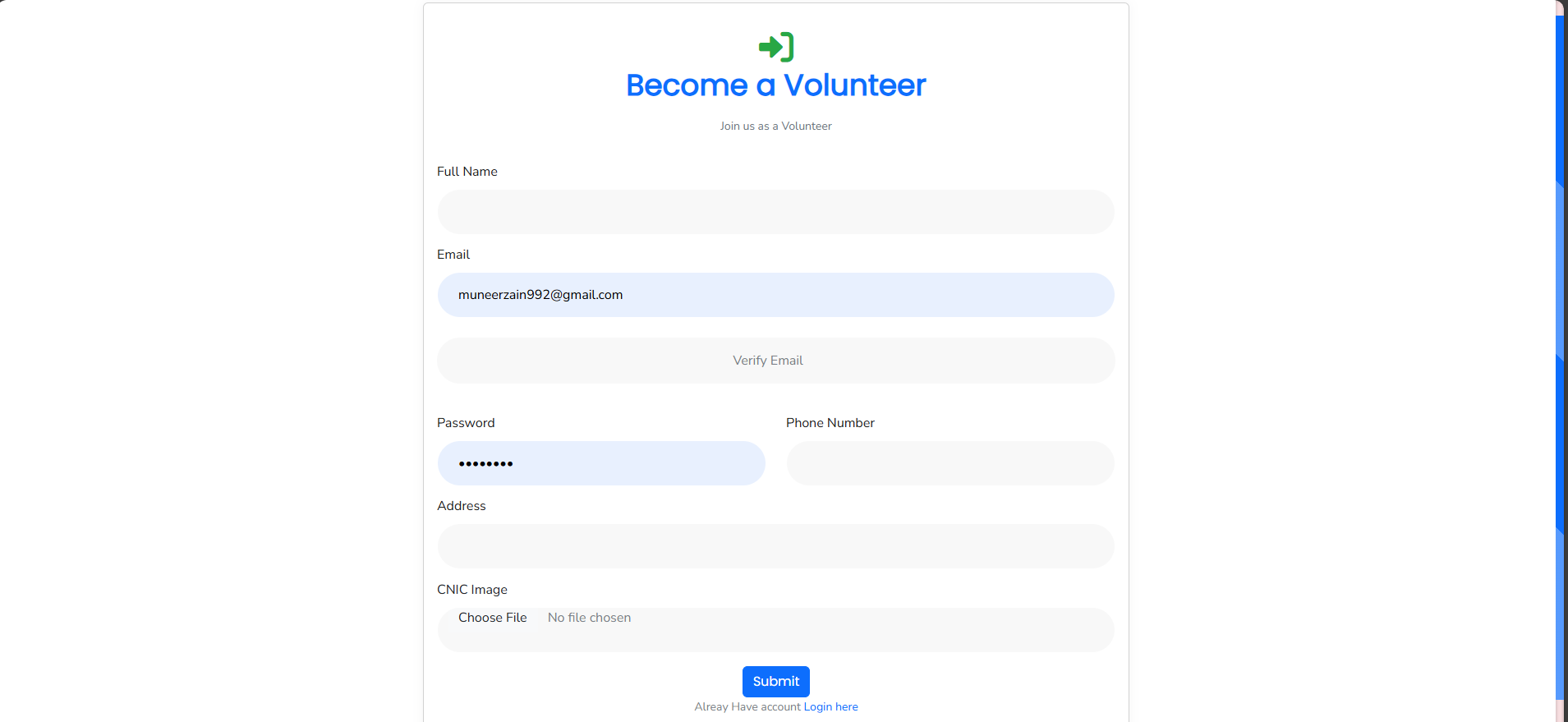


* Feedbacks from user

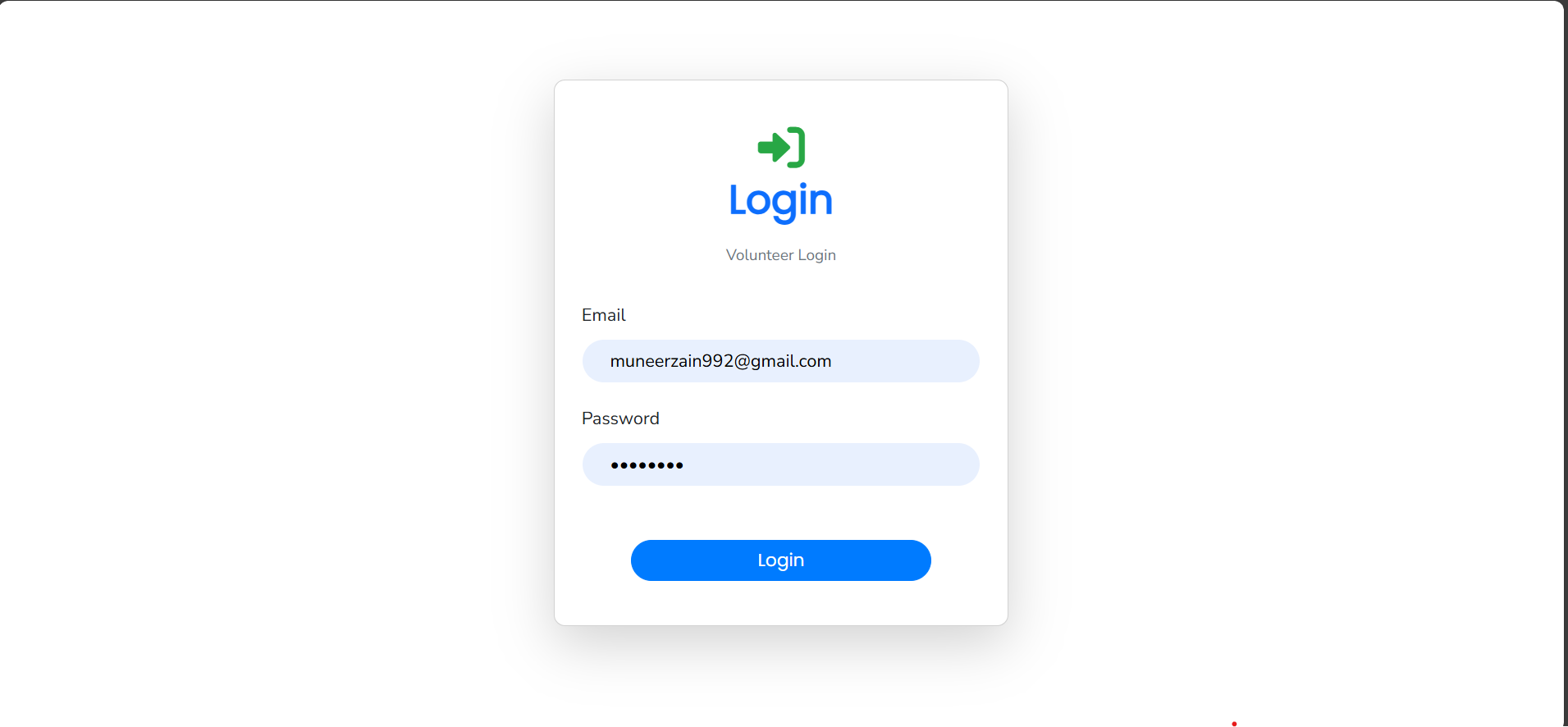


### Volunteer Role GUI:

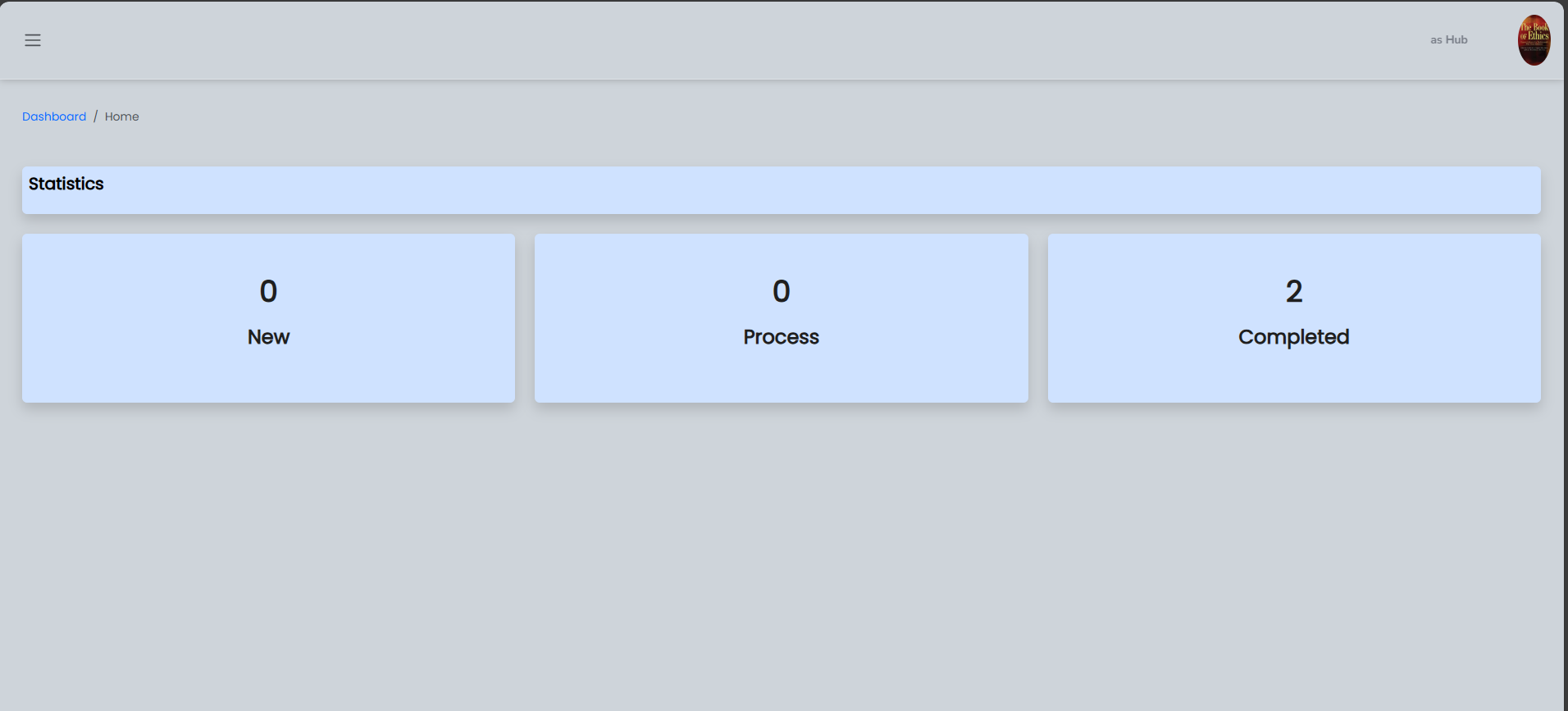
* Volunteer Sign up page.



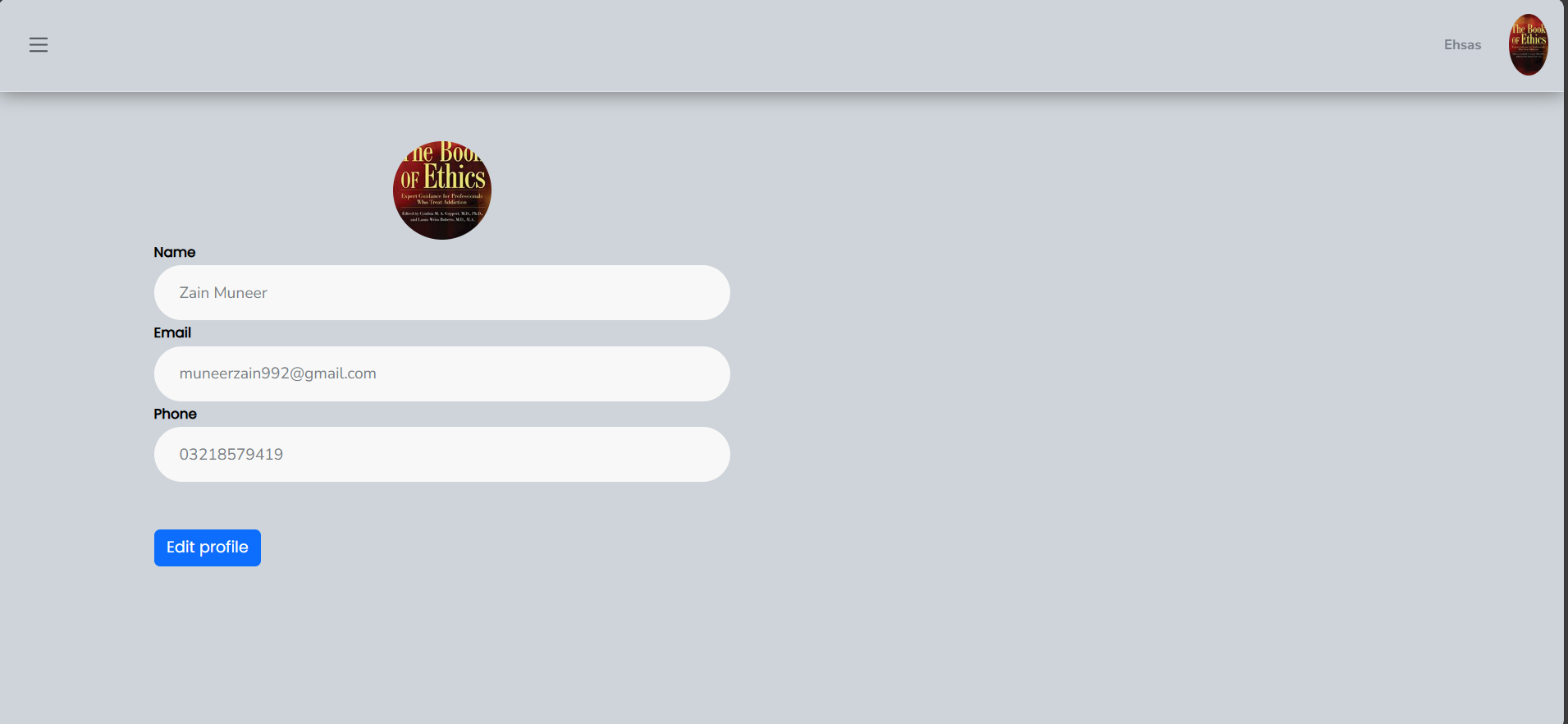
* Volunteer Login Page

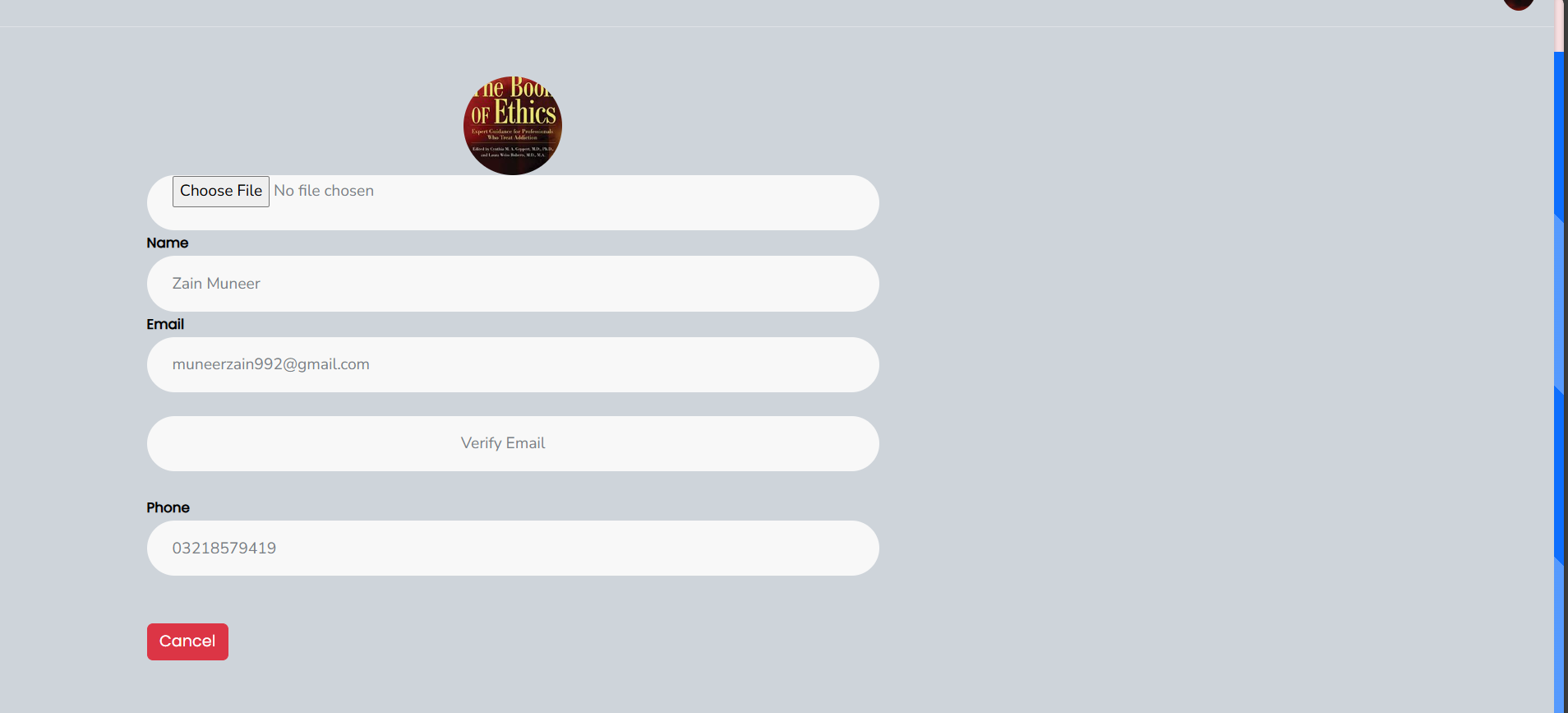


* Volunteer Statistics Page.



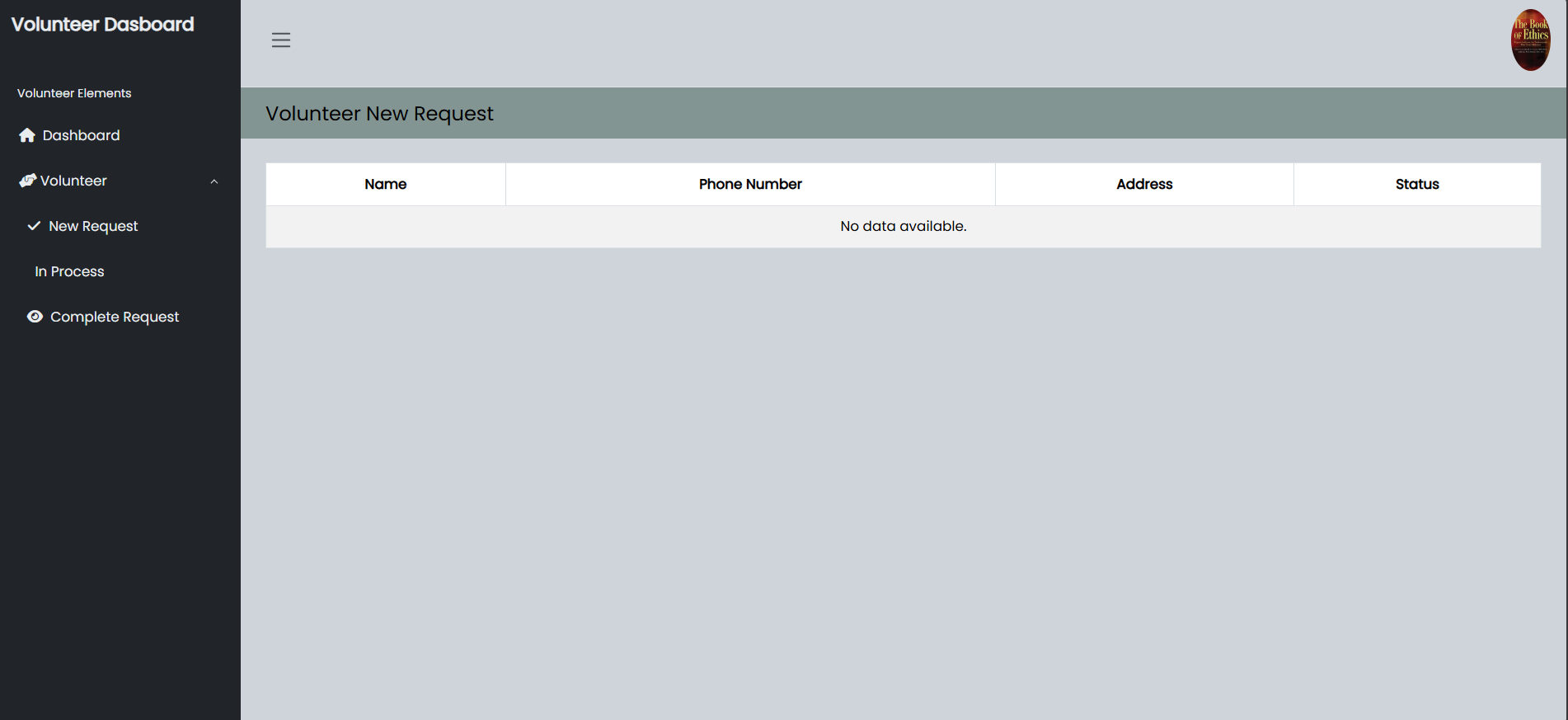
* Volunteer Profile view/edit page.





* Volunteer new/in-process/Completed Request page.







**Chapter 04:**

**Implementation and Test Cases**

# Implementation and Test Cases

## Implementation

### Implementation Overview

Ehsas Hub implemented using the MySQL database through XAMPP server for local development, along with Express.js, React.js, and Node.js for backend and frontend development. The AI recommendation system built in Python using Flask. The system integrates multiple roles—donors, (needy), volunteers, and admins—with functionalities like book donation/request, OTP-based authentication, recommendation engine, and profile verification. This project implements a hybrid recommendation system that leverages both collaborative filtering using deep learning (Neural Collaborative Filtering - NCF) and content-based filtering (genre clustering with BERT embeddings). The system recommends books to users based on their interactions with books and their preferred genres. It is exposed through a Flask API and interacts with data stored in a MySQL database.

### 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 display 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.

### Key Implementation Components

#### Frontend (React.js):

* **User Interfaces:** Dynamic views for different roles.
* **Routing:** Implemented using React Router DOM.
* **HTTP Requests:** Axios used for connecting frontend to backend APIs.

#### Form Validation:

* Passwords must include at least 8 characters, one uppercase, one lowercase, and one numeric digit
* All fields must be filled. Any missing field will trigger an error message.
* Book donation/request forms validate genre, edition (numeric only), image format (JPG/PNG only), and title length (minimum 3 characters).

#### Backend (Node.js & Express.js):

* RESTful APIs for login, registration, book management, and volunteering.
* Authentication: JWT and bcrypt for token-based secure login.
* Email Services: Node mailer for OTP email verifications.
* Profile Control: OTP verification before allowing any profile updates.
* Validations: Strong validation for registration (email, password format), login, donations, book requests, and mandatory field checks.

#### Recommendation System:

This Recommendation System implements a hybrid recommendation system that leverages both collaborative filtering using deep learning (Neural Collaborative Filtering - NCF) and content-based filtering (genre clustering with BERT embeddings). The system recommends books to users based on their interactions with books and their preferred genres. It is exposed through a Flask API and interacts with data stored in a MySQL database.

##### 4.1.4.4.1 Technologies Used

* Python
* TensorFlow / Keras
* Scikit-learn
* SentenceTransformers (BERT Embeddings)
* Flask (REST API)
* SQLAlchemy (ORM for MySQL)
* MySQL (Relational Database)
* Pandas / NumPy (Data manipulation)

##### 4.1.4.4.2 Dataset Description & Preprocessing

* Books Dataset: Contains book details such as ID, title, genre, and author.
* Users Dataset: Contains user profiles including user ID, name, age, and preferred genre.
* Interactions Dataset: Contains logs of user actions (view, like, click, etc.) with books.
* Duplicates in the books dataset were removed.
* Fake user profiles were generated using the Faker library.
* Interactions were synthetically created to simulate user behavior.

##### 4.1.4.4.3 Clustering Using BERT Embedding’s

* BERT model (`all-MiniLM-L6-v2`) was used to generate embeddings for book genres.
* KMeans clustering was applied to group books into similar thematic clusters.
* This clustering helps enhance recommendations for new users based on their preferred genres.

##### 4.1.4.4.4 Neural Collaborative Filtering (NCF) Model

* A binary classifier model was built using the Keras Functional API.
* The model takes a user ID and book ID, embeds both, and processes them through a multi-layer perceptron (MLP).
* Positive samples were taken directly from the interactions dataset.
* Negative samples were generated by randomly selecting books not interacted with by a user.
* The model was trained to distinguish between interacted and non-interacted items.

##### 4.1.4.4.5 Model Architecture

* Input layers for user ID and book ID
* Embedding layers (size: 64)
* MLP with Dense → Dropout → Dense → Dropout → Dense layers
* Final Dense layer with sigmoid activation

##### 4.1.4.4.6 Flask API Implementation

* A Flask API was developed to expose the recommendation engine.
* When a GET request is made to `/recommend/<user\_id>`, the system returns a list of recommended book titles.
* The system supports both cold-start (new user) and warm-start (existing user) recommendations.

##### 4.1.4.4.7 Recommendation Logic

* New User: Recommends books based on the user's preferred genre.
* Existing User: Predicts book preference scores using the trained NCF model. Filters top N recommendations and further refines using genre clustering to improve personalization.

#### Database (MySQL)

##### 4.1.4.5.1 Tables

* **users**: Stores general users including students (needy). Fields: user\_id (PK), name, email, phone, password, gender, address, image, status, preferred\_genre, request\_id (FK to needy), created\_at, updated\_at.
* **admin:** Stores administrators. Fields: id (PK), name, email, phone, password, gender, image, created\_at, updated\_at.
* volunteer: Stores volunteer data. Fields: id (PK), name, email, phone, password, address, image, gender, status, created\_at, updated\_at.
* **donor:** Book donors. Fields: id (PK), user\_id (FK), volunteer\_id (FK), book\_name, genres, edition, author\_name, image, status, created\_at, updated\_at.
* **books:** All books in the system. Fields: bookid (PK), title, author, covering, language, genres, rating.
* **needy:** Links students to their book requests. Fields: id (PK), user\_id (FK), req\_id, request\_status, created\_at, updated\_at.
* **interactions:** Tracks recommendations and user-book interactions. Fields: interaction\_id (PK), user\_id (FK), book\_id (FK), action.
* **feedback:** Collects user feedback. Fields: id (PK), name, email, message, created\_at.
* **verify:** OTP verification store. Fields: id (PK), email, code, created\_at.

##### 4.1.4.5.2 Validation enforced through:

* NOT NULL constraints for mandatory fields
* Strong password policy
* ENUM values for controlled status and gender fields
* Foreign key constraints for relational integrity

## Test Cases

### Admin Test Cases

Table 4.1: Admin Test Case

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 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 and press Submit | Admin account created and stored in DB | Admin created successfully | Pass |
| 2 | Test Admin Login Successfully | Admin must be registered | Correct Email and Password | Enter Email and press Login | Admin logged in successfully | Logged in successfully | Pass |
| 3 | Test Admin Login with Incorrect Password | Admin must be registered | Correct Email, Wrong Password | Enter Email and wrong Password → Press Login | Error message:  Invalid credentials | Error displayed | Pass |
| 4 | Test Admin Login with Unregistered Email | None | Unregistered Email, Any Password | Enter Email → Press Login | Error: Invalid credentials | Error displayed | Pass |
| 5 | Test Admin Edit Profile | Admin must be logged in | Updated Name, Phone, Image and verify otp | Update fields → Press Save | Admin profile updated | Profile updated | Pass |

### Needy Test Case

Table 4.2: Needy Test Case

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 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 Edit/Update user profile information | User logged in | Updated Email, phone or address | Change profile fields and verify email with OTP and save | Profile updated successfully | Profile updated successfully | Pass |
| 6 | Test registration with invalid email format | No user registered | Wrong email format | Enter wrong email and press Register | "Invalid Email Format" error should appear | Invalid email error displayed | Pass |
| 7 | Test user status field behavior | New registration | Correct user details And approved | Register and check status field | Status should be Active (1) | Status set correctly | Pass |
| 8 | Test phone number field validation | No user registered | Phone number less than 11 digits | Enter short phone number and press Register | "Invalid Phone Number" error should appear | Phone validation error shown | Pass |
| 9 | Test password encryption | New registration | Correct user data | Register and check database password field | Password should be encrypted (hash) | Password saved encrypted | Pass |

### Volunteer Test Case

Table 4.3: Volunteer Test Case

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 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 Edit/update volunteer profile | Volunteer logged in | Updated Email, phone/address details | Change and save profile | Profile updated successfully | Profile updated successfully | Pass |
| 6 | 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 |
| 7 | Test volunteer password strength | No volunteer registered | Weak password without special char, uppercase | Enter weak password and submit | error should appear | error displayed | Pass |
| 8 | 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 |
| 9 | Test volunteer status after registration | New volunteer registration | Correct registration data And approved | Complete registration and check status field | Status should be Active (1) | Status set correctly | Pass |

### Donor Test Case

Table 4.4: Donor Test Case

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ID | TEST CASES | PRECONDITIONS | INPUT DATA | STEPS | EXPECTED RESULT | ACTUAL RESULT | PASS/FAIL |
| 1 | Test book donation with valid inputs | User logged in | Valid book name, genre, edition, author, image | Fill donation form → Submit | Book added to DB | Book saved | Pass |
| 2 | Test missing book name | Logged in | Leave book name empty | Submit form | Error: Fiels are required | Error shown | Pass |
| 3 | Test invalid genre | Logged in | Leave genre empty | Submit form | Error shown | Error displayed | Pass |
| 4 | Test edition field with text | Logged in | Enter "First" in edition | Submit | Error Displayed | Error shown | Pass |
| 6 | Test minimum book title length | Logged in | Enter 2 characters | Submit | Error: Title too short | Error shown | Pass |
| 7 | Test cancel donation form | Logged in | Click Close on modal | Close the modal | Modal closes, no action | Closed | Pass |

## Test Metrics

### Common Attributes of Test Case Metrics

Test case metrics provide a structured approach to evaluate the quality and performance of software testing. In Ehsas Hub, the following common attributes were used across all modules:

* **Total Number of Test Cases:** Indicates the overall coverage of testing across all modules and functionalities.
* **Test Case Pass Rate:** The ratio of test cases that passed successfully against the total executed.
* **Test Case Failures:** Number of tests that did not meet expected outcomes, helping identify bugs or logic flaws.
* **Defect Density:** Represents the percentage of test cases that failed out of the total executed, calculated as

Defect Density = (Failed Test Cases / Total Test Cases) \* 100.

* **Test Case Effectiveness:** Measures the proportion of test cases that successfully detected defects, calculated as

Effectiveness = (Defects Found by Tests / Total Defects) \* 100.

* **Traceability Matrix:** Ensures that each requirement is linked to corresponding test cases to verify that all features are tested and validated.
* **Validation Checks:** Common validation logic (e.g., non-empty fields, password complexity, file format, numeric inputs) was standardized and reused across different test forms.

To provide comprehensive and consistent testing coverage, these metrics were used uniformly throughout the admin, user, volunteer, and book donation/request modules.

### Test Summary Table

Table 4.5: Test Summary Table

|  |  |  |
| --- | --- | --- |
| Metric | Description | Value |
| Total Test Cases | Combined across all modules | 37 |
| Passed | All test cases executed successfully | 37 |
| Failed | - | 0 |
| Test Case Effectiveness | (37/37) \*100 | 100% |
| Defect Density | (0/37) \*100 | 0% |

## Conclusion

This chapter discussed the Ehsas Hub implementation process, including the technical architecture, component breakdown, and project-wide validation techniques. Our backend makes use of MySQL, which has stringent database and code validation guidelines. Complexity criteria are enforced by password validation. Security for profile updates is ensured by OTP verification. To better match recommendations with user preferences, the recommendation system employs a hybrid approach. Every aspect of the system passed the first functional testing with 100% efficacy, demonstrating that Ehsas Hub is safe, scalable, and designed with the goal of enabling students to access educational materials with ease.

**Chapter 05:**

**Experimental Results and Analysis**

# Experimental Results and Analysis

## Introduction

This chapter presents the experimental setup, performance evaluation, and result analysis of our application "Ehsas Hub". Ehsas Hub is a platform that links administrators, volunteers, needy users, and contributors to donate and suggest books. Validating the efficacy of key features such user interaction flow, account approval procedures, book donation/request processing, and the hybrid recommendation system is the goal of these trials. In order to guarantee correctness, usability, and dependability, we additionally assess platform performance in a variety of user roles and scenarios.

## Experimental Setup

### Platform Performance Evaluation

#### Objective

To evaluate each role's essential characteristics and user experience from start to finish, admin, volunteer, needy, and donor.

#### Environmental tools

* **Device Used:** Dell Latitude Laptop
* **Specifications:** 16gb Ram 512 SSD Core i5 8th gen
* **Network:** 4G , Nayatel Wi-Fi
* **Internet Speed:** 3-5 Mbps
* **Software:** Ehsas-Hub Web Application (Mobile responsive)

Table 5.1: Functional Performance Evaluation

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Procedure | Action | Expected Time | Actual Time | Result | Notes |
| Account Registration | Sign up with details, genre, and email OTP | ≤ 5 sec | 6 sec | 90% | Network dependent |
| Admin Account Approval | Admin dashboard accepts new users | Instant | Instant | 100% | Works as intended |
| Login | Enter email/password | ≤ 5 sec | 3 sec | 100% | Secure and smooth |
| Donate Book | Fill form and submit book | ≤ 5 sec | 5-6 sec | 90% | Image upload takes time |
| Request Book | Choose and request a book | ≤ 5 sec | 4 sec | 100% | Success confirmation email |
| Volunteer Accept Request | Volunteer accepts pickup nearby | ≤ 3 sec | 3-4 sec | 95% | Needs location optimization |
| View Recommended Book | Browse all books | ≤ 4 sec | 3 sec | 100% | Smooth rendering |
| Edit Profile / Logout | Update info / logout | ≤ 3 sec | 2 sec | 100% | No issues found |

### Recommendation System Effectiveness

#### Objective

To test the hybrid recommendation system, which uses Neural Collaborative Filtering (NCF) and content-based filtering using BERT embedding is for suggesting books, based on preferred genres and interaction history.

#### Environment & Tools

* **Libraries Used:** pandas, numpy ,flask, tensorflow, sentence-transformers, scikit-learn, faker, tf-keras, dotenv, openpyxl, sqlalchemy, pymysql, hf\_xet
* **Backend:** Flask REST API
* **Database:** MySQL with SQLAlchemy
* **Synthetic Data:** User interactions generated via Faker

Table 5.2: Recommendation System Evaluation

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Component | Action | Expected Outcome | Actual Outcome | Accuracy | Notes |
| Genre-based Suggestions | Display relevant books after login | Relevant book list shown | 90% match | 90% | Based on initial signup genre |
| Interaction Learning | Recommend based on Search, likes, requests | Personalized suggestions | 85% accuracy | 85% | Improves over time |
| Response Time | Load recommendations | ≤ 5 sec | 4-5 sec | 100% | Acceptable speed under load |
| Cold Start Test | New user with no interactions | Genre-only based suggestions | 80% match | 80% | Initial fallback to genre model |

### Authentication and Security

#### Objective

To confirm the safe and effective operation of the password reset and edit profile of any user, OTP verification, and login functions.

#### Environment & Tools

* **Device Used:** Dell Latitude Laptop
* **Specifications:** 16gb Ram 512 SSD Core i5 8th gen
* **Software:** Web Frontend with email services (Mailer linked to Ehsas-Hub domain)

Table 5.3: Authentication Metrics

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Feature | Action | Expected Time | Actual Time | Success Rate | Notes |
| Email OTP Verification | Register + receive code | ≤ 2 min | 1.5 min | 100% | Code received on Gmail |
| Login Authentication | Email/password login | ≤ 5 sec | 2-3 sec | 100% | Token stored securely |
| Forgot Password Flow | Request Forgot password | ≤ 3 min | 2.5 min | 100% | Secure via email confirmation |
| Verify Email on Edit Profile | Verify email OTP before profile is updated | ≤ 2 min | 1.8 min | 100% | OTP ensures security for user changes |

## Conclusion

All of the Ehsas Hub platform's modules—registration, book donation/request, volunteer coordination, and personalized recommendations—show excellent functioning and user satisfaction, according to the experimental research. When it came to genre-based and interaction-based recommendations, the hybrid recommendation algorithm achieved up to 90% accuracy. With secure email-based verification, user approval and authentication processes operated effectively. While there is need for improvement in terms of response times and volunteer location optimization, the platform is reliable and prepared for practical use. These findings support Ehsas Hub's usefulness in expediting book contributions via an ecosystem powered by technology.

**Chapter 06:**

**Conclusion and Future Directions**

# Conclusion and Future Directions

## Introduction

The main objective of Ehsas Hub's conception and development was to provide a centralized, AI-assisted platform that would accelerate volunteer and admin coordination, allow book contribution, and enhance needy kids' ability to access learning resources. This chapter offers a thorough summary of the findings from the implementation process, evaluations of every element, and suggestions for further improvement. We review our achievements and consider areas that might use improvement.

## Achievements and Improvements

Throughout the development of Ehsas Hub, several technical and operational milestones were achieved that validate the robustness and feasibility of the platform:

### Front-End Achievements:

* **User Experience Optimization:** React.js was used to design an intuitive and responsive user interface for multiple roles (admin, donor, volunteer, and Needy).
* **Validation Enhancements:** All forms enforce strong password rules, mandatory field checks, and file type validations, improving data consistency and security.
* **Modular Navigation:** Seamless routing between modules such as donation, registration, login, and feedback has been established using React Router.

### Backend Achievement’s

* **Secure Authentication:** JWT-based login with password encryption (bcrypt) and OTP verification using Node mailer ensures secure user operations.
* **Role-Based Functionality:** Each role accesses specific APIs designed to maintain operational clarity and data segregation.
* **API Validation:** Express-validator ensures structured input validation across all endpoints.

### Recommendation Engine:

* **Hybrid Model Integration:** We developed a Neural Collaborative Filtering (NCF) model for personalized book recommendations.
* **Successful Training and Evaluation:** The system uses actual interaction data to fine-tune suggestions, boosting usability.

### Database Enhancements:

* **MySQL with XAMPP:** Relational schema designed with foreign keys, NOT NULL constraints, and ENUM types to maintain integrity.
* **Modules Covered:** Admin, Users, Donors, Volunteers, Feedback, Book Interactions, and OTP Verification modules were fully developed and interconnected.
* **Detailed ERD Mapped:** Relationships and constraints were implemented exactly as mapped in the ER diagram.

## Critical Review

The Ehsas Hub platform tackles the issues of needy empowerment, donation transparency, and book accessibility. The creation of a full-stack platform with multi-role support and integrated AI recommendation was part of the scope.

### Strengths

* **Social Impact Focused:** Aimed at educational upliftment using technology
* **Machine Learning Integration:** Used modern algorithms for personalized learning support.
* **Secure and Scalable:** Clean modular codebase and robust authentication mechanisms.

### Weaknesses:

* **UI Aesthetics:** Visual design could benefit from improved styling and user interaction cues.
* **Performance Optimization:** Database queries can be further optimized for high concurrency.
* **Limited Real-Time Updates:** Chat or live support functionalities were not included but could enhance coordination.
* **Google Map:** Use Map location will be Helpful to enhance the Donor location for pick up.

## Future Recommendations

For Future, the following future improvements and scope extensions are proposed:

### Enhancements to Current Modules:

* Improve UI with animated transitions and better visual hierarchies.
* Add file format previews for book cover uploads.
* Provide status tracking for donation and request submissions.

### Additional Features:

* **Chat System:** Real-time communication between users and volunteers.
* **Mobile Application:** Flutter-based cross-platform app for accessibility.
* **Gamified Volunteering:** Add badges and leaderboards to motivate volunteers.
* **Feedback Analytics:** Automatically categorize user feedback using NLP.

### Future Specific Work Plan:

* Implement real-time notification system (Node.js + Socket.io).
* Deploy the application using cloud services (e.g., Vercel/Heroku for frontend, Render for backend).
* Conduct user testing in real environments (e.g., colleges, libraries).

## Conclusion

In conclusion, Ehsas Hub successfully met its objectives, including secure user registration, book donation workflows, multi-role access, and personalized book recommendations. All core functionalities were implemented and tested with 100% success rate in unit testing. The project offers a scalable base for further work and meaningful social contribution. Future directions include the expansion of features, UI improvement, integration of real-time components, and deployment for public use. With a clear roadmap for Future, Ehsas Hub stands ready for refinement and broader impact in the educational tech domain.

# References

[1] F. Ricci, L. Rokach, and B. Shapira, Recommender Systems Handbook. Boston, MA: Springer, 2011.

[2] Y. Koren, R. Bell, and C. Volinsky, “Matrix factorization techniques for recommender systems,” IEEE Computer, vol. 42, no. 8, pp. 30–37, Aug. 2009.

[3] R. Burke, “Hybrid recommender systems: Survey and experiments,” User Modeling and User-Adapted Interaction, vol. 12, no. 4, pp. 331–370, Nov. 2002.

[4] X. Su and T. M. Khoshgoftaar, “A survey of collaborative filtering techniques,” Advances in Artificial Intelligence, vol. 2009, Article ID 421425, 2009.

[5] S. Bhattacharyya, S. Jha, K. Tharakunnel, and J. C. Westland, “Data mining for credit card fraud: A comparative study,” Decision Support Systems, vol. 50, no. 3, pp. 602–613, Feb. 2011.

[6] K. R. Seeja and M. Zareapoor, “FraudMiner: A novel credit card fraud detection model based on frequent itemset mining,” The Scientific World Journal, vol. 2014, Article ID 252797, 2014.

[7] M. Omokaro, P. C. Zhang, and E. H. Chi, “Volunteerism through self-tracking and personal informatics: How quantified self data can support civic engagement,” in Proc. ACM Conf. Human Factors in Computing Systems (CHI), 2015, pp. 4873–4882.

[8] S. Wu, J. M. DiMicco, and D. R. Millen, “Detecting professional versus personal closeness using an enterprise social network site,” in Proc. 32nd ACM Conf. Human Factors in Computing Systems, 2010, pp. 1955–1964.

[9] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T. S. Chua, “Neural collaborative filtering,” in Proc. 26th Int. Conf. World Wide Web (WWW), 2017, pp. 173–182.

[10] P. Covington, J. Adams, and E. Sargin, “Deep neural networks for YouTube recommendations,” in Proc. 10th ACM Conf. Recommender Systems (RecSys), 2016, pp. 191–19