Student Care

F I N A L Y E A R P R O J E C T

CONNECTING STUDENTS IN NEED WITH DONORS WHO CARE

Submitted By

Numan Ali	**20021519-137**
Abdul Rehman	**20021519-049**

BS (Computer Science)
Session 2020 – 2024

Supervised By

Mr. Najeeb Ur Rehman, Assistant Professor

FACULTY OF COMPUTING AND INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF GUJRAT

STATEMENT OF SUBMISSION

This is certified that **Mr. Numan Ali** (20021519-137) and **Mr. Abdul Rehman** (20021519-049) have successfully completed the final year project titled as **Student Care** in the Department of Computer Science, University of Gujrat, to fulfill the partial requirement of the degree of Bachelor of Science in Computer Science.

_____
Project Supervisor
Najeeb-Ur-Rehman
Assistant Professor
Department of Computer Science

_____
Project Management Office
Department of Computer Science

_____
Chairperson/Head of Department
Department of Computer Science

Acknowledgement

We truly acknowledge the cooperation and help made by **Mr. Najeeb-Ur-Rehman**, Assistant Professor, Department of Computer Science, University of Gujrat. He has been a constant source of guidance throughout the course of this project. We would also like to thank him for his help and guidance throughout this project. We are also thankful to our friends and families whose silent support led us to complete our project.

Date: 25 June 2024

Abstract

Access to education is a fundamental right, yet many university students face financial constraints that hinder their ability to pay tuition fees. Student Care Application aims to address this issue by creating a platform that connects university students in need with compassionate donors. Through a rigorous verification process and an intuitive interface, the application will facilitate seamless transactions, ensuring that deserving students receive the support they need.

Table Of Contents

Table Of Figures

Chapter 1: Project Feasibility Report

1.1. Introduction

Student Care seeks to bridge the financial gap in higher education by providing a user-friendly platform that allows university students to seek financial assistance for their tuition fees. This innovative application will revolutionize the way charitable donations are made, ensuring transparency, security, and impact while offering a lifeline for students facing unexpected medical crises.

1.2. Project/Product Feasibility Report

The Student Care web application is designed to be technically straightforward, functional, and effective in its purpose. The project is feasible in all aspects, including time and the availability of appropriate software and applications. By utilizing these resources, we can successfully complete the project. The Student Care web application aims to streamline the process of connecting students in need with donors willing to provide essential items, thereby promoting community engagement and support. The feasibility report includes the following types:

- Technical
- Operational
- Economic
- Schedule
- Specification
- Information
- Motivational
- Legal and Ethical

1.2.1. Technical Feasibility

The Student Care web application is developed using widely supported and reliable technologies such as Angular for the frontend, Node.js with Express for the backend, and MongoDB for the database. These technologies ensure robust performance and scalability, making the project technically feasible.

1.2.2. Operational Feasibility

The operational aspect of Student Care involves seamless integration and usability for both students and donors. The web application is designed to be user-friendly, ensuring that users can easily navigate through the platform to request or offer donations. This ease of use guarantees operational feasibility.

1.2.3. Economic Feasibility

The economic feasibility of Student Care is evaluated based on the cost-effectiveness of development and maintenance. Open-source technologies and cloud hosting solutions are employed to minimize expenses. Additionally, the potential social benefits and the positive impact on the student community justify the investment.

1.2.4. Schedule Feasibility

The project timeline is well-structured, with clearly defined milestones and deadlines. The development team is committed to adhering to the schedule, ensuring that the project is completed within the allocated timeframe. This makes the project schedule feasible.

1.2.5. Specification Feasibility

The specifications of the Student Care web application are clear and achievable. The functional and non-functional requirements are well-documented, and the development team has the expertise to meet these specifications effectively.

1.2.6. Information Feasibility

The information required for the successful implementation of the Student Care project, including user requirements, donor profiles, and transaction data, is accessible and manageable. The use of MongoDB ensures efficient handling of large volumes of data.

1.2.7. Motivational Feasibility

The project team is highly motivated and dedicated to the cause of helping students in need. The social impact of the project serves as a significant motivational factor, driving the team to deliver a high-quality solution.

1.2.8. Legal & Ethical Feasibility

The Student Care project adheres to all relevant legal and ethical standards. User data privacy and security are prioritized, and measures are taken to ensure compliance with data protection regulations. The platform promotes ethical practices by encouraging community support and responsible donations.

1.3. Project/Product Scope

Scope of the Project

The project aims to create a comprehensive Charity Application facilitating direct, transparent financial assistance for university students. It covers user registration, verification, donation management, and real-time notifications.

Development Schedule

The development timeline spans twelve months, including initial research and planning, six months for development, and six months for testing and refinement.

Development Process

The project will follow the Agile development methodology, with iterative development and regular feedback loops to ensure the application meets user needs and quality standards.

Techniques, Tools, and Platform

- **Frontend:** Angular
- **Backend:** Node.js with Express
- **Database:** MongoDB
- **Tools:** Visual Studio Code, Git for version control.

Reasoning

- **Angular for Frontend Development:** Chosen for its robust framework, flexibility, and strong community support, making it ideal for developing a dynamic and responsive web application.
- **Node.js with Express for Backend:** Selected for its efficiency in handling asynchronous operations and scalability, which is essential for handling numerous user requests and data processing.
- **MongoDB for Database:** Opted for its flexibility in handling unstructured data and scalability, which is crucial for the dynamic nature of donation listings and user profiles.

Features of the App

- User Registration and Authentication
- Student & Donor Profiles
- Payment Processing
- Admin Dashboard
- Request for Donation
- Donation Management

Justification for the Project

The Student Care web application aims to create a supportive community by connecting students in need with donors willing to help. By facilitating the donation process, this application will help reduce financial burdens on students, promote educational equity, and encourage a culture of giving**.**

Constraints

The application is developed for web users and can be accessed through any device with internet connectivity.

1.4. Project/Product Costing

This project focuses on developing a web-based platform, Student Care, aimed at facilitating donations for students in need. Most of the system involves code development, ensuring efficient resource planning and usage. We will estimate the project's cost using function point analysis based on the system's functionalities provided to end users.

1.4.1. Project Cost Estimation by Function Point Analysis

We will use function point analysis to estimate the cost of the project.

Measurement Parameter	**Count**
1. Number of external inputs (EI)	6
2. Number of external outputs (EO)	6
3. Number of external inquiries (EQ)	2
4. Number of internal files (ILF)	2
5. Number of external interfaces (EIF)	0

Table 1: Domain Characteristics Table

Weighting factor					
Measurement Parameter	Count	Simple	Average	Complex	
Number of user inputs	6	3	4	6	=18
Number of user outputs	6	4	5	7	=24
Number of user inquiries	2	3	4	6	=6
Number of files	2	7	10	15	=14
Number of external Interfaces	0	5	7	10	=0
					Total=62

Total Complexity Adjustment Factor (Fi) = (1+2+2+5+3+3+1+3+5+4+5+4+4+4) =46

FP est. = GFP * VAF

FP est. = Count Total * [0.65 + 0.01 * (Fi)]

= 62*[0.65+0.01*(46)]

= 68.82

Finally, Total Project Cost and Total Project Effort are calculated given the average productivity parameter for the system. The formulae are given as follows:

Productivity parameter = 68.82/3 = 22.94

Cost / FP = labor rate / productivity parameter

=10000 / 22.94 = 435.92

Total Project Cost = FP est. * (cost / FP)

= 68.82*435.92
= 30,000
Total Estimated Effort = FP est. / productivity parameter
= 68.82/22.94
= 3

1.4.2. Project Cost Estimation by using COCOMO

Project Size (in KLOC) = 100 KLOC
Product attributes = 1.10
Hardware attributes = 0.1
Personnel attributes = 0.5
Software tools attributes = 1.15
Project attributes = 1.05

COCOMO II Constants

a = 2.4
b = 1.05
∏(EMi) = 1.10 * 0.95 * 0.85 * 1.15 * 1.05 = 1.07

Calculation for Each Phase

Name	Size in KLOC	Calculation	Estimated Cost (Person-Months)
Proposal Submission	0.2	$2.4 * (0.2)^{1.05} * 1.07$	0.47
SRS	1	$2.4 * (1)^{1.05} * 1.07$	2.56
Design UI/UX	0.5	$2.4 * (0.5)^{1.05} * 1.07$	1.24
Frontend Development	15	$2.4 * (15)^{1.05} * 1.07$	44
Backend Development	10	$2.4 * (10)^{1.05} * 1.07$	28
Testing &Deployment	0.5	$2.4 * (0.5)^{1.05} * 1.07$	1.24

Table 2: Project Cost Estimation Table

Estimating Schedule and Cost

Schedule Calculation

Project Schedule = Total Effort / Productivity Rate

Project Schedule = 77/ 2.5 ≈ **30 months**

Estimated Semesters = Project Schedule / Semester Length

Estimated Semesters = 30 / 8 ≈ **4 months**

Cost Calculation

Total Estimated Effort for all phases: 77 Person-Months

Developer Salary: Rs. 500 per person-month

Total Cost = Total Effort * Developer Salary

Total Cost = 77 * 500 ≈ **Rs. 38500**

1.4.3. Activity Based Costing

Sr.	Activities	Resources	Cost Rate	Duration
1	Web Application Design, Layout, Structure	Angular, Visual Studio Code, HTML/CSS, TypeScript	Free	6 Weeks
2	Front End	Angular, Visual Studio Code, TypeScript	Free	2 Weeks
3	Database Design, Development and Connectivity	Visual Studio Code, MongoDB	Free	2 Weeks
4	Back End	Node.js, Express.js	Free	6 Weeks
5	Testing	Visual Studio Code, Postman, Browser Dev Tools	Free	2 Weeks
6	Documentation	MS Word	Free	2 Weeks

Table 3: Activity Base Costing Table

1.5. Task Dependency Table

Task	Task Name	Duration(days)	Dependencies
T1	Idea finalizing	7	
T2	Proposal document	8	T1
T3	Requirement gathering	30	T2
T4	Prototyping Design, Layout, Structure	36	T3
T5	Development, Design, Layout, Structure	26	T3, T4
T6	Backend Coding	19	T4, T5
T7	Database Connectivity	40	T6
T8	Software testing with dummy data	14	T7

Table 4: Task Dependency Table

1.6 CPM - Critical Path Method

Activity	Immediate Predecessor	Duration (Days)
T1	None	7
T2	T1	8
T3	T2	30

T4	T3	36
T5	T3, T4	26
T6	T4, T5	19
T7	T6	40
T8	T7	14

Table 4: CPM Table

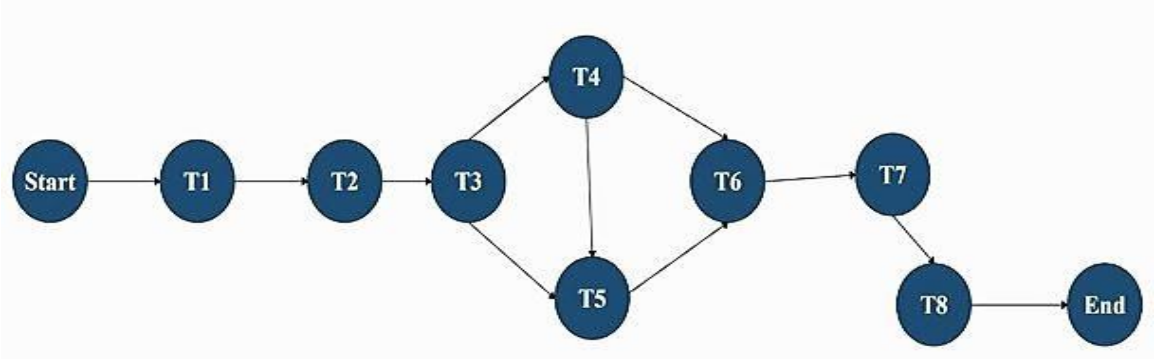


Figure 1:Network Diagram

The critical path is the longest path amongst all, so critical path of our project is:
T1, T2, T3, T4, T5, T6, T7, T8

Activity	Duration	ES		EF	LS	LF
T1	7	0	7		0	7
T2	8	7	15		7	15
T3	30	15	45		15	45
T4	36	45	81		45	81
T5	26	81	107		81	107
T6	19	107	126		107	126
T7	40	126	166		126	166
T8	14	166	180		166	180

Table 5: Identify the Critical Path

Possible paths are:
T1-T2-T3-T4- T6-T7-T8= 7+8+30+36+19+40+14=154
T1-T2-T3- T5-T6-T7-T8=7+8+30+26+19+40+14=144
T1-T2-T3- T4-T5-T6-T7-T8=7+8+30+36+26+19+40+14=180

The critical path is the longest path amongst all, so critical path of our project is:

T1, T2, T3, T4, T5, T6, T7, T8

1.7. Project Milestones and Deliverables

- **Project Initiation (1-2 weeks)**
  Define the project vision, goals, and initial user stories for Student Care. Establish the project infrastructure and prioritize the initial backlog for development.
- **Prototype Development (2-4 weeks)**
  Develop a basic prototype of the application with essential features. Conduct user testing to gather feedback for refining subsequent sprints.
- **User Interface Enhancement (4-6 weeks)**
  Our focus will be on improving the user interface design based on user feedback and ensuring the application is user-friendly and responsive on various devices.
- **Core Functionality Implementation (6-15 weeks)**
  Develop key features, including student profiles, donation management, and transparent impact tracking. Rigorously test to ensure the accuracy and reliability of these features.
- **Community Building (15-17 weeks)**
  Introduce features for peer interaction, alumni engagement, and consultant support. Incorporate user-generated content and usability testing.
- **Trust and Verification Mechanisms (17-19 weeks)**
  Build mechanisms for data verification and source validation, ensuring the accuracy and reliability of student profiles and donation records.
- **Real-time Updates and Notifications (19-22 weeks)**
  Enable real-time updates for student profiles, academic performance, and donation tracking. Provide timely notifications to users about changes and contributions.
- **Performance Optimization and Scaling (22-26 weeks)**
  Optimize application performance, conduct load testing, and enhance scalability to accommodate a growing user base and increasing support.
- **Final Testing and Issue Resolution (26-32 weeks)**
  Comprehensive testing will be carried out, and any reported issues or bugs will be addressed to ensure a polished and stable final release.
- **Release and Continuous Improvement (Ongoing)**
  Upon launch, gather user feedback and insights for continuous iteration and enhancement of Student Care based on evolving user needs and emerging challenges.

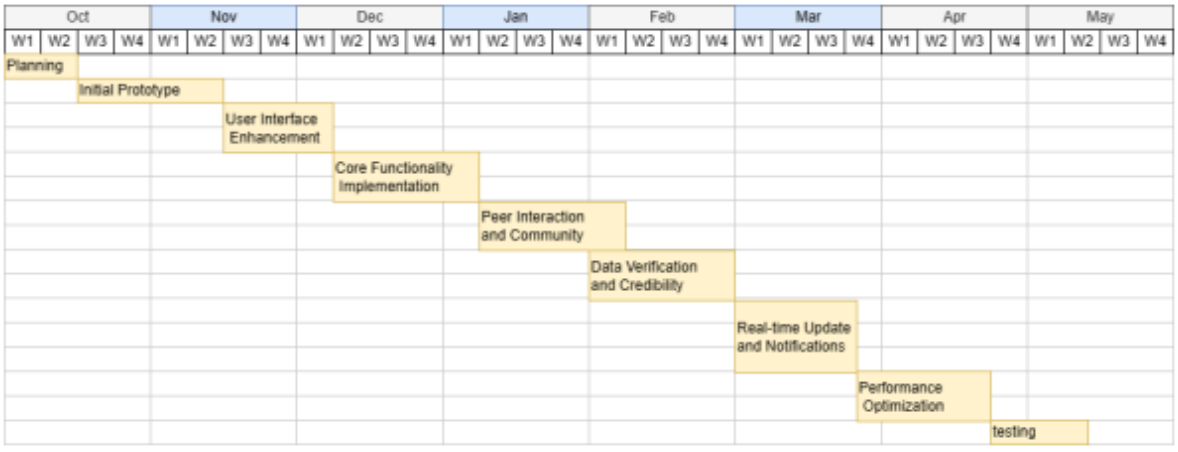


Figure 2:Gantt Chart

1.8. Work Division among Group Members

Team Members	Roll No.	Skill Set	Task
Abdul Rehman	20021519-049	Frontend + Backend Development	Frontend Development, Backend Development
Numan Ali	20021519-137	Frontend + Backend Development	Backend Development, Database Design, API Development

Table 6: Work Division Table

1.9 Tools and Technology with reasoning

- **Visual Studio Code**: Primary IDE for frontend development in Angular.
- **Angular**: Frontend framework for building responsive and scalable web applications.
- **Node.js and Express**: Backend development for server-side logic and API implementation.
- **MongoDB**: NoSQL database for storing student and donor information securely.
- **MS Office**: For documentation purposes, including requirements gathering, user stories, and project reports.

1.10. Vision Document

In today's educational landscape, accessing resources and donations can be challenging for students in need. Student Care aims to bridge this gap by providing a centralized platform where students can easily find donations and donors can contribute directly to those in need. Our vision is to create a supportive community where education resources are shared and accessible to all, fostering a more equitable and inclusive learning environment.

Unique Selling Proposition:

- **Streamlined Access:** Student Care offers a straightforward interface for students to request and receive donations effortlessly.
- **Community Engagement:** Facilitates connections between donors and students, promoting a sense of community and shared support.
- **Transparency and Trust:** Implementing a rating and feedback system ensures trust and accountability within the platform.

1.11. Risk List

Development level risks

- Technical complexities in integrating backend APIs with frontend functionalities.
- Potential delays due to unexpected bugs or software conflicts.

Technology level risks

- The system can be shut down due to power resources, failed web browsers due to internet problems.

Software level risks

- The software level risks may be occurred due to no internet availability in some areas.
- It can occur if there is sudden growth in technology requirements and standards.

Maintenance and design level risks

Estimation: The uncertainty about the parameters of work time, cash flow process and the rate of defects, repair, and duration can be occurred. Therefore, the development team can go wrong in calculating estimation about these factors.

Reusability

The functionality of our project can be affected due to the reusability of the modules.

Scope

The total features requested may be beyond what the development team can deliver in the time available.

Other factors:

- Changes in the requirements of the project can affect.
- Health factors of the team members

1.12. Product Features/Product Decomposition

The app will include the following key features:

- **Student Dashboard**: Where students can create profiles and list their specific needs for donations.
- **Donor Portal**: Enables donors to browse student requests, donate items or funds, and track their contributions.
- **Messaging System**: In-app messaging for direct communication between students and donors to facilitate donation arrangements.

Chapter 2: System Requirements Specifications

2.1. Introduction

"Student Care" aims to facilitate seamless connections between compassionate donors and deserving students in need. This online application provides a convenient platform for donors to contribute funds, emergency support, and educational resources to support students through their academic journey.

Through this system, donors can easily make direct donations to verified students, ensuring a transparent and impactful contribution process. The application simplifies the process of connecting donors with students in need, making it effortless for both parties.

2.2. System Specifications

In the current landscape, various charitable organizations and institutions operate with a primary focus on providing financial aid and support to students in need. These organizations typically engage in traditional methods of fundraising, including events, campaigns, and offline donations. The process of connecting donors with students often involves intermediaries, and the transparency of fund utilization may not be as direct or immediate. The existing system relies on conventional charity models, where students in need often must navigate through complex application procedures to access financial assistance. Donors, on the other hand, may not have a direct and transparent view of how their contributions are utilized or the specific impact on individual students. Communication between donors and students might be limited, and the overall process may lack the efficiency and immediacy required to address urgent needs. The existing system may not fully leverage modern technology to streamline the donation process and enhance the connection between donors and beneficiaries. With **Student Care**, we aim to revolutionize this existing system by introducing a user-centric, transparent, and efficient platform that directly connects donors with students in need. This application will address the limitations of the current system, providing a more immediate, accessible, and impactful way for donors to support students through their academic journeys.

2.3. Organizational Chart

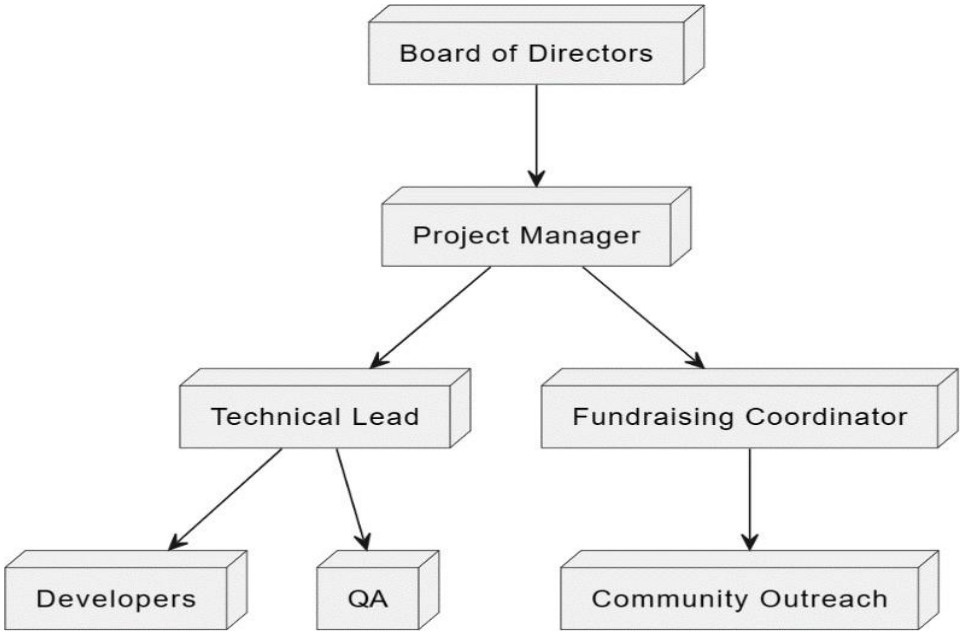


Figure 3:Organizational Chart

2.4. Roles and Responsibilities

Board of Directors

- Provides overall strategic guidance and decision-making for the project.

Project Manager

- Oversees the entire project, ensuring its successful execution.
- Acts as a liaison between the Board of Directors and the project team.

Technical Lead

- Manages the technical aspects of the project, including app development and integration.
- Coordinates with developers and quality assurance (QA) team.

Developers

- Responsible for the development and maintenance of the Student Care Charity Application.

QA (Quality Assurance)

- Ensures the quality and functionality of the application through testing and feedback.

Fundraising Coordinator

- Manages fundraising activities and donor relations.
- Works closely with the Community Outreach team.

Community Outreach

- Engages with the community to promote the Student Care Charity Application.
- Facilitates communication between donors and students in need.

2.5. Scope of System

The Student Care Charity Application is designed to operate within the domain of student financial support, emphasizing transparent and direct connections between donors and students in need. The scope of the system includes:

- **Financial Assistance**
  The system focuses on providing a platform for donors to contribute funds directly to students for various needs, including tuition fees, emergency health situations, and essential educational resources.
- **Direct Donor-Student Interaction**
  The system facilitates direct communication and interaction between donors and students. Donors can view student profiles, academic achievements, and financial needs, fostering a personalized connection.
- **Payment Integration**
  The system integrates secure payment methods to facilitate direct transactions between donors and students, ensuring a seamless and trustworthy process.

2.6. Summary of Requirements

The initial requirements for the Student Care Charity Application are outlined to establish a foundational understanding of the system's high-level functionalities. Key aspects of the system include:

- **User Registration and Verification**
  Donors and students can register on the platform, with a robust verification process ensuring the authenticity of user profiles.
- **Direct Fund Donations**
  The system enables donors to make direct financial contributions to verified students, covering various needs such as tuition fees and educational resources.
- **Direct Donor-Student Interaction**
  The platform facilitates direct communication between donors and students, creating a personalized and supportive connection.
- **Payment Integration**
  Secure payment methods are integrated to enable direct transactions between donors and students, streamlining the donation process.
- **Transaction History and Reporting**
  Detailed transaction history and reporting features allow users and administrators to monitor the flow of funds and assess the impact of donations.

2.7. Identifying External Entities

The process of identifying external entities for the Student Care Charity Application involves two phases

Over Specify Entities from Abstract

Based on the initial abstract, the following entities are identified:

- **Donor**
  Individuals or organizations contribute funds to support students.
- **Student**
  Individuals seeking financial assistance for education.
- **Administrator**
  Oversees and manages the overall functionality of the Student Care Charity Application.

Perform Refinement

Verified Donor

- Donors who have completed a verification process to ensure their legitimacy and commitment.
- Verified Student
- Students who have undergone a verification process to confirm their eligibility and financial needs.

Admin/Coordinator

- Administrators or coordinators responsible for managing and overseeing the platform's operations, including user verification and dispute resolution.

Financial Institution

- External entity representing the secure payment gateway integrated into the application.

Emergency Service Provider

- External entity associated with the emergency fund module, ensuring immediate support during health crises.

2.8. Context Level Data Flow Diagram

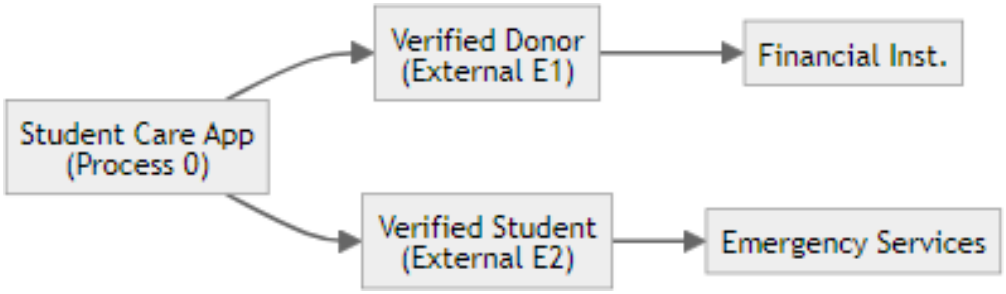


Figure 4:Context Level Data Flow

2.9. Capture "Shall" Requirements

User Registration and Verification	• The system shall allow users to register as either Verified Donors or Verified Students.
Donor Functionalities	• The system shall provide Verified Donors with the ability to make direct financial contributions to Verified Students.
Student Functionalities	• Verified Students shall be able to create profiles and list their financial needs on the platform. • The system shall allow students to request emergency funds in case of health crises.
Direct Donor-Student Interaction	• The system shall facilitate direct communication and interaction between Verified Donors and Verified Students.
	• Donors shall receive notifications and updates on the impact of their contributions.
Payment Integration	• The system shall integrate secure payment methods to enable direct transactions between Verified Donors and Verified Students.
Emergency Fund Module	• The system shall have a dedicated module for emergency fund disbursement. • Emergency funds shall be disbursed promptly upon verification of the student's health crisis.

Transaction History and Reporting	• The system shall maintain a comprehensive transaction history for both donors and students. • Reporting features shall provide insights into the utilization of funds and the impact of donations.

Table 7: Shall Requirements Table

2.10. Allocate Requirements

Para #	Initial Requirements	Use_Case_Name
1	Users "shall" register to access the platform.	User_Registration
2	Users "shall" be able to create profiles and list their financial needs on the platform.	Student_Profile_and_Listing
3	The system "shall" allow students to request emergency funds in case of health crises.	Emergency_Fund_Request
4.1	Users "shall" have the ability to search for students based on academic performance, financial needs, or keywords.	Donor_Search_and_Discovery
4.2	The system "shall" implement filters for refining donor search results.	Donor_Search_and_Discovery
5.1	Users "shall" be able to send messages to each other regarding financial assistance.	Messaging
5.2	The system "shall" include real-time messaging functionality.	Messaging
6.1	Users "shall" be able to make direct financial contributions to verified students.	Direct_Fund_Contribution
7.1	Users "shall" be able to provide feedback and ratings after financial assistance.	Donor_Feedback
7.2	The system "shall" display average donor ratings on profiles.	Donor_Feedback
8.1	The system "shall" send notifications for new messages, fund contributions, and feedback.	Notification_Management
9.1	Users "shall" have profiles displaying their academic performance, financial needs, and fund utilization history.	Student_Profile

9.2	Users "shall" be able to edit and update their profiles.	Student_Profile
10.1	The system "shall" implement privacy settings for student profiles and contact information.	Security_Settings
10.2	Users "shall" be able to set visibility preferences for their profiles.	Security_Settings

Table 8: Allocate Requirements Table

2.11. Prioritize Requirements

Para#	Priority	Initial Requirements	Use_Case_Name
1	Highest	Users "shall" register to access the platform.	User_Registration
2	High	Users "shall" be able to create profiles and list their financial needs on the platform.	Student_Profile_and_Listing
3	Medium	The system "shall" allow students to request emergency funds in case of health crises.	Emergency_Fund_Request
4.1	High	Users "shall" have the ability to search for students based on academic performance, financial needs, or keywords.	Donor_Search_and_Discovery
4.2	High	The system "shall" implement filters for refining donor search results.	Donor_Search_and_Discovery
5.1	High	Users "shall" be able to send messages to each other regarding financial assistance.	Messaging
5.2	High	The system "shall" include real-time messaging functionality.	Messaging
6.1	High	Users "shall" be able to make direct financial contributions to verified students.	Direct_Fund_Contributi on
7.1	High	Users "shall" be able to provide feedback and ratings after financial assistance.	Donor_Feedback
7.2	High	The system "shall" display average donor ratings on profiles.	Donor_Feedback
8.1	High	The system "shall" send notifications for new messages, fund contributions, and feedback.	Notification_Manageme nt
9.1	Medium	Users "shall" have profiles displaying their academic performance, financial	Student_Profile

		needs, and fund utilization history.	
9.2	Medium	Users "shall" be able to edit and update their profiles.	Student_Profile
10.1	Highest	The system "shall" implement privacy settings for student profiles and contact information.	Security_Settings
10.2	Highest	Users "shall" be able to set visibility preferences for their profiles.	Security_Settings

Table 9: Prioritize Requirements Table

2.12. Requirements Traceability

Req ID	Para#	Priority	Initial Requirements	Buid	Use_Case_Name
1	1.1	Highest	Users "shall" register to access the platform.	B1	User_Registration
2	2.2	High	Users "shall" be able to create profiles and list their financial needs on the platform.	B1	Student_Profile_and_Listing
3	3.1	Medium	The system "shall" allow students to request emergency funds in case of health crises.	B1	Emergency_Fund_Request

4	4.1	High	Users "shall" have the ability to search for students based on academic performance, financial needs, or keywords.	B1	Donor_Search_and_Discovery
5	4.2	High	The system "shall" implement filters for refining donor search results.	B1	Donor_Search_and_Discovery
6	5.1	High	Users "shall" be able to send messages to each other regarding financial assistance.	B1	Messaging
7	5.2	High	The system "shall" include real-time messaging functionality.	B1	Messaging
8	6.1	High	Users "shall" be able to make direct financial contributions to verified students.	B1	Direct_Fund_Contribution
10	7.1	High	Users "shall" be able to provide feedback and ratings after financial assistance.	B1	Donor_Feedback
11	7.2	High	The system "shall" display average donor ratings on profiles.	B1	Donor_Feedback
12	8.1	High	The system "shall" send notifications for new messages, fund contributions, and feedback.	B1	Notification_Management
13	9.1	Medium	Users "shall" have profiles displaying their academic performance, financial needs, and fund utilization history.	B1	Student_Profile

14	9.2	Medium	Users "shall" be able to edit and update their profiles.	B1	Student_Profile
15	10.1	Highest	The system "shall" implement privacy settings for student profiles and contact information.	B1	Security_Settings
16	10.2	Highest	Users "shall" be able to set visibility preferences for their profiles.	B1	Security_Settings

Table 10: Requirements Traceability Table

2.13. High Level Use Case Diagram

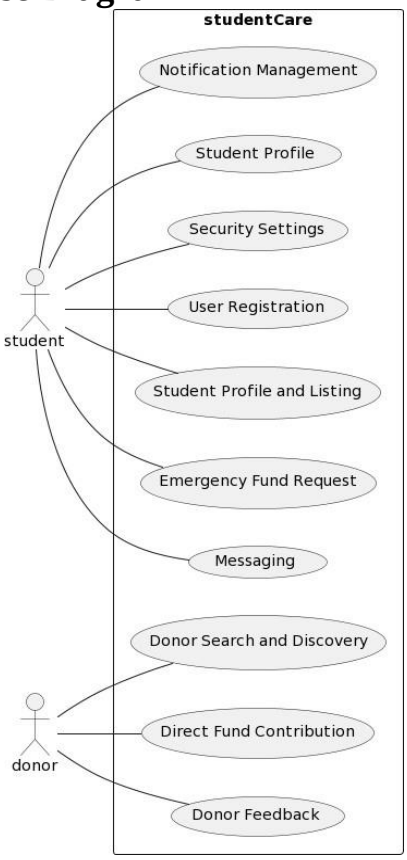


Figure 5:Use Case

2.14. Analysis Level Use Case Diagram

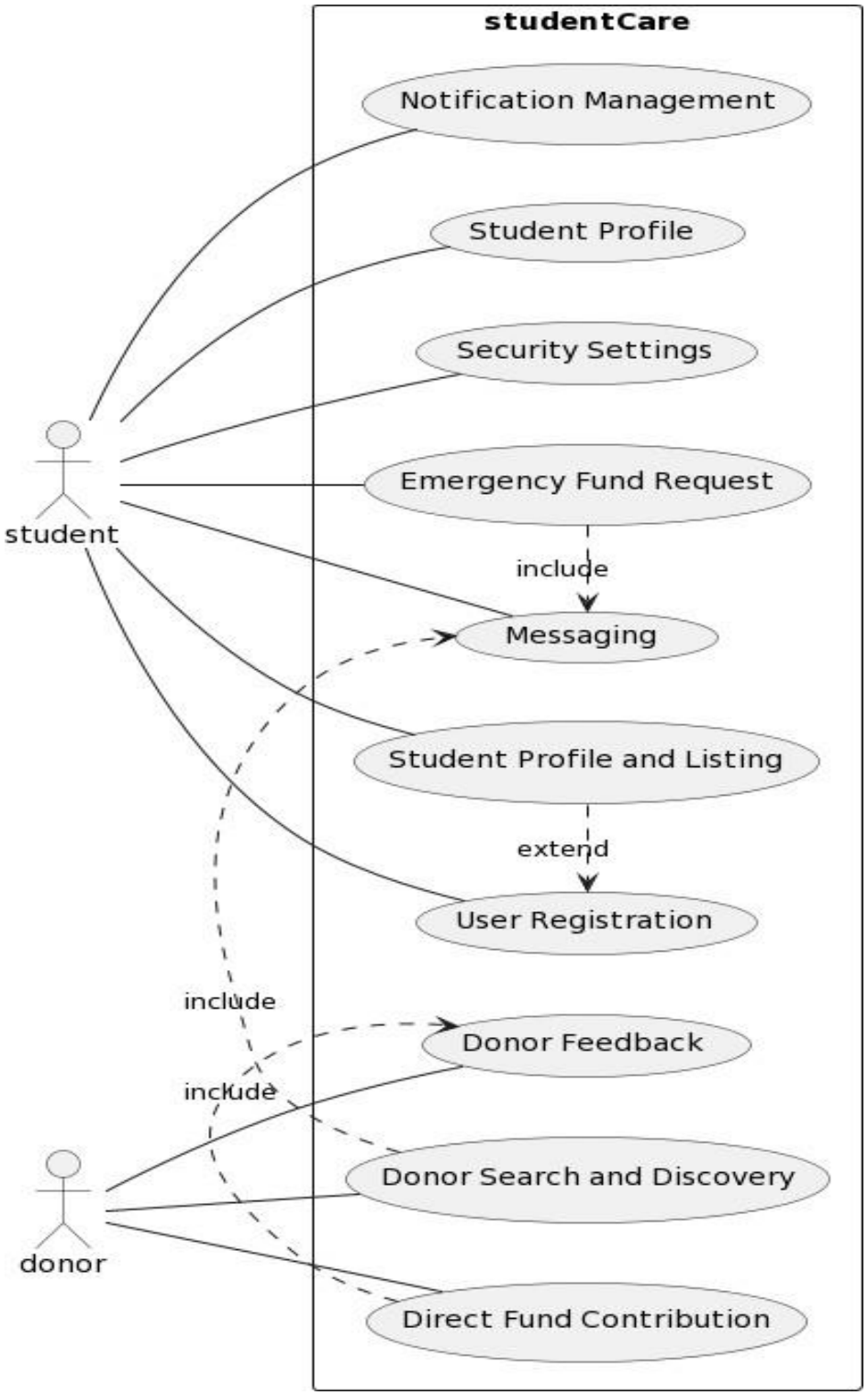


Figure 6:Analysis Level Use Case

2.15. Use Case Description

2.15.1 Student Registration

Brief Description

This use case describes the process of a student registering on the student Care platform. The primary goal is to enable students to create profiles and list their financial needs.

Preconditions

- The student has access to the student Care platform.
- The student has a valid email address for account verification.

Basic Flow

- Students navigate to the registration page.
- Student provides necessary information: name, email, password.
- Student clicks on the "Register" button.
- System verifies the email address format.
- System sends a verification email to the student.
- Student clicks on the verification link in the email.
- System verifies the email link and activates the student's account.
- Student is redirected to the student profile creation page.
- Student adds details such as academic information and financial needs.
- Student clicks on the "Submit" button.

Alternate Flows

- Invalid Email: If the email format is invalid, the system prompts the student to enter a valid email.
- Existing Account: If the email is already registered, the system prompts the student to log in.
- Incomplete Registration: If the student leaves mandatory fields blank, the system notifies them to complete the required information.

Post Conditions

- The student's profile is created successfully.
- The student can log in and access the full features of the student Care platform.
- Feel free to tailor this template to fit specific use cases or add more details as needed.

2.15.2 Requesting Emergency Funds

Brief Description

This use case outlines the process of a student requesting emergency funds through the student Care platform in case of health crises.

Preconditions

- The student is registered and logged into the student Care platform.
- The student has encountered a health crisis requiring financial assistance.

Basic Flow

- Student navigates to the "Emergency Fund Request" section.
- Student provides details about the health crisis, including required funds and supporting documents.

- Student clicks on the "Submit Request" button.
- System validates the request and requires documentation.
- If valid, the system sends a notification to potential donors.
- Donors review the request and decide whether to contribute.
- If donors contribute, the system processes the funds and notifies the student.

Alternate Flows

- Invalid Request: If the request is incomplete or lacks necessary documentation, the system prompts the student to provide the required information.
- No Donor Contributions: If no donors contribute within a specified time, the system notifies the student.

Post Conditions

- The student receives financial assistance for the health crisis.
- Donors receive feedback on the impact of their contributions.
- Feel free to adapt this template for other use cases or provide specific details you'd like to include.

Chapter 3: System Design

3.1 Data Model

In the previous deliverable, analysis of the system is completed. So, we understand the current situation of the problem domain. Now we are ready to strive for a solution for the problem domain by using an object-oriented approach. Now we discuss these artifacts one by one as follows.

3.2 Data Model

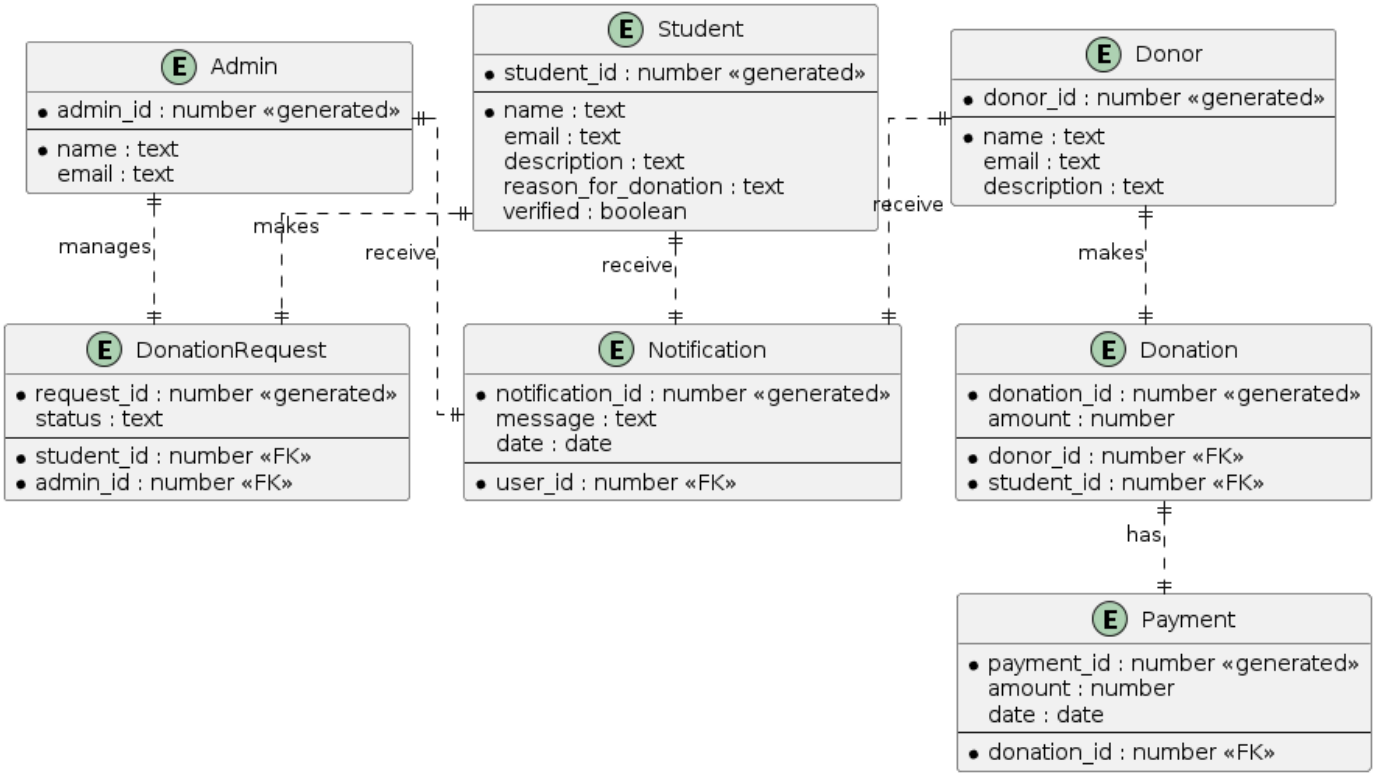


Figure 7:Data Model

3.3 Class Diagram

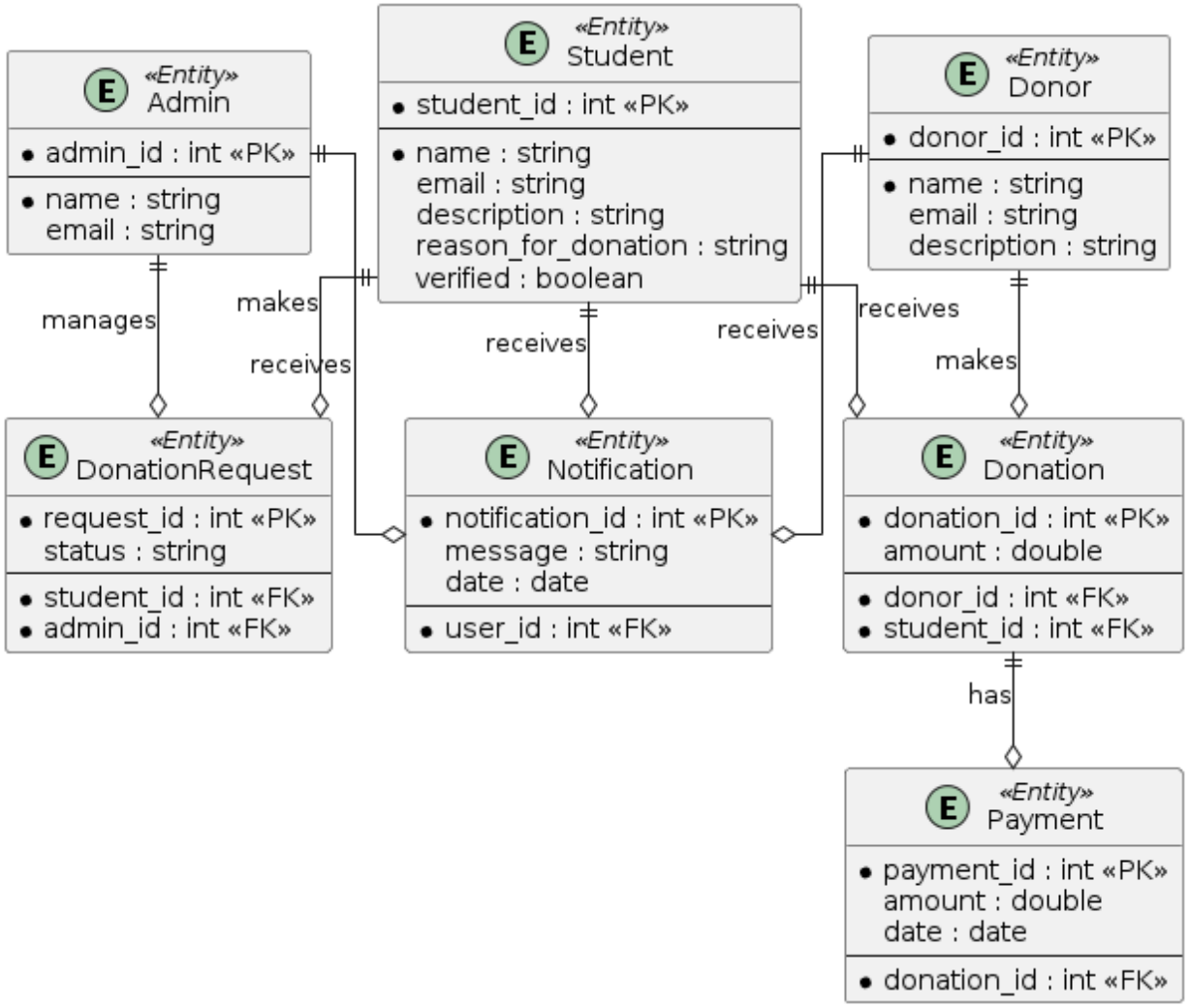


Figure 8:Class Diagram

3.4 Sequence Diagram

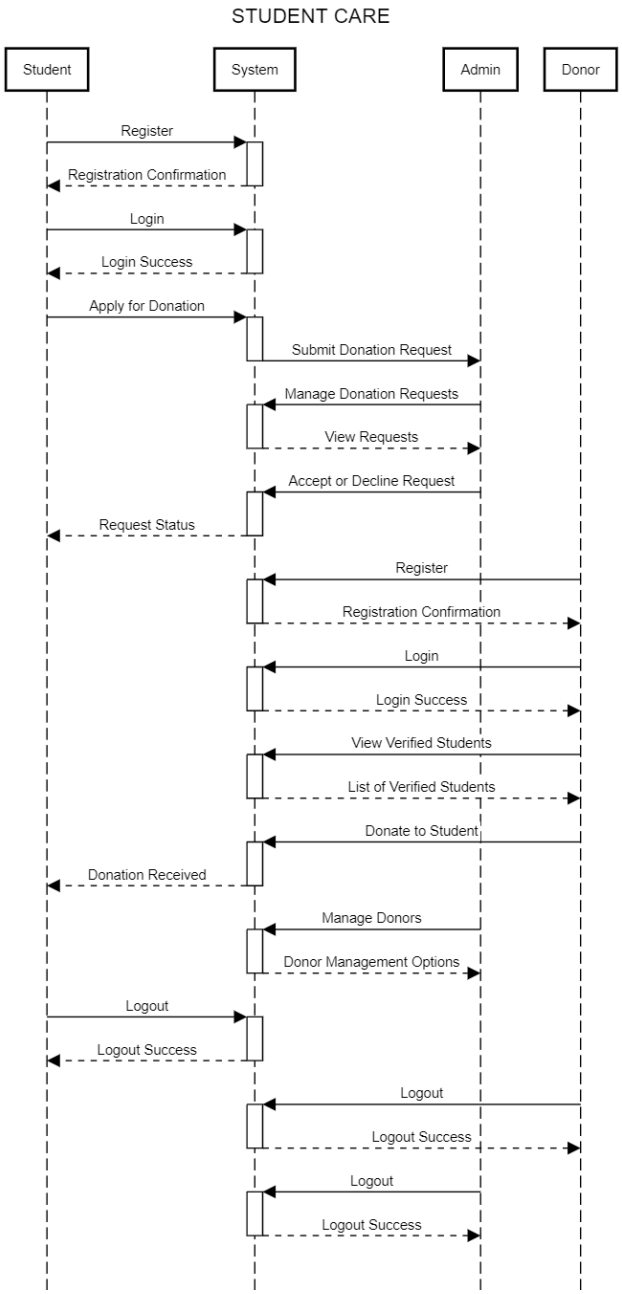


Figure 9:Sequence Diagram

3.5 State Chart Diagram

3.5.1 Student State Chart Diagram

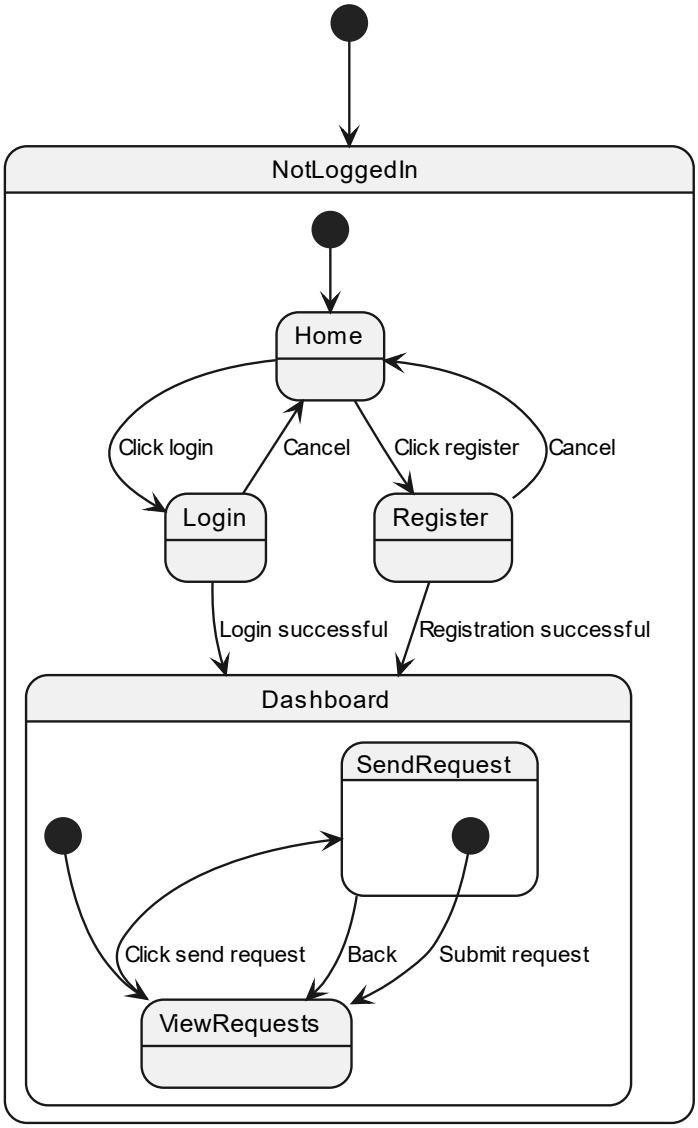


Figure 10:Student State Diagram

3.5.2 Admin State Chart Diagram

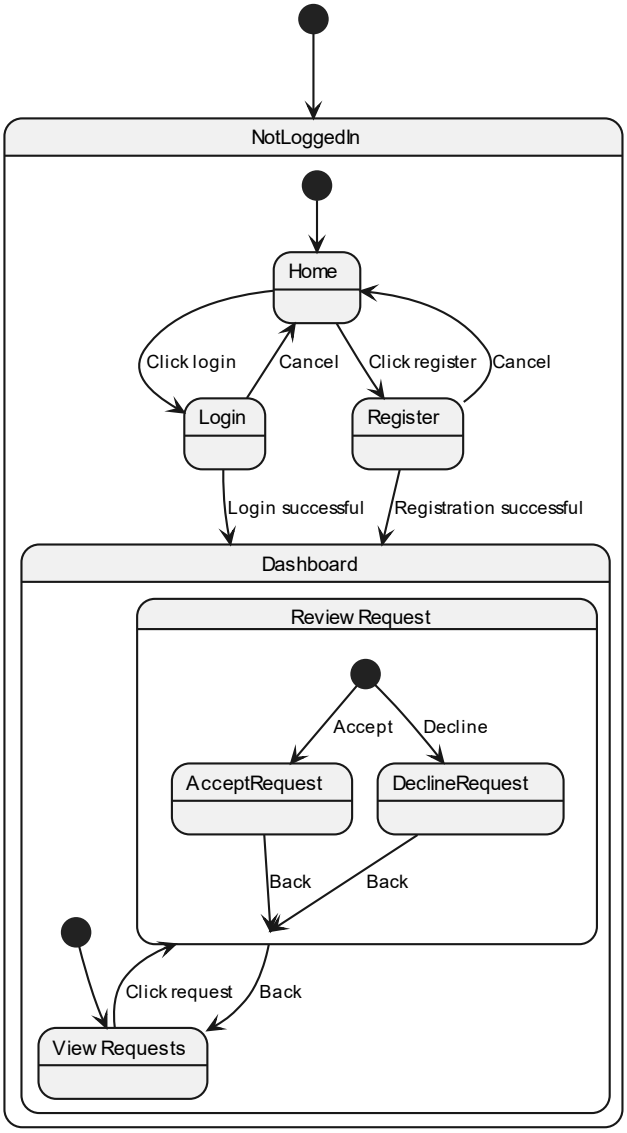


Figure 11:Admin State Chart Diagram

3.5.3 Donor State Chart Diagram

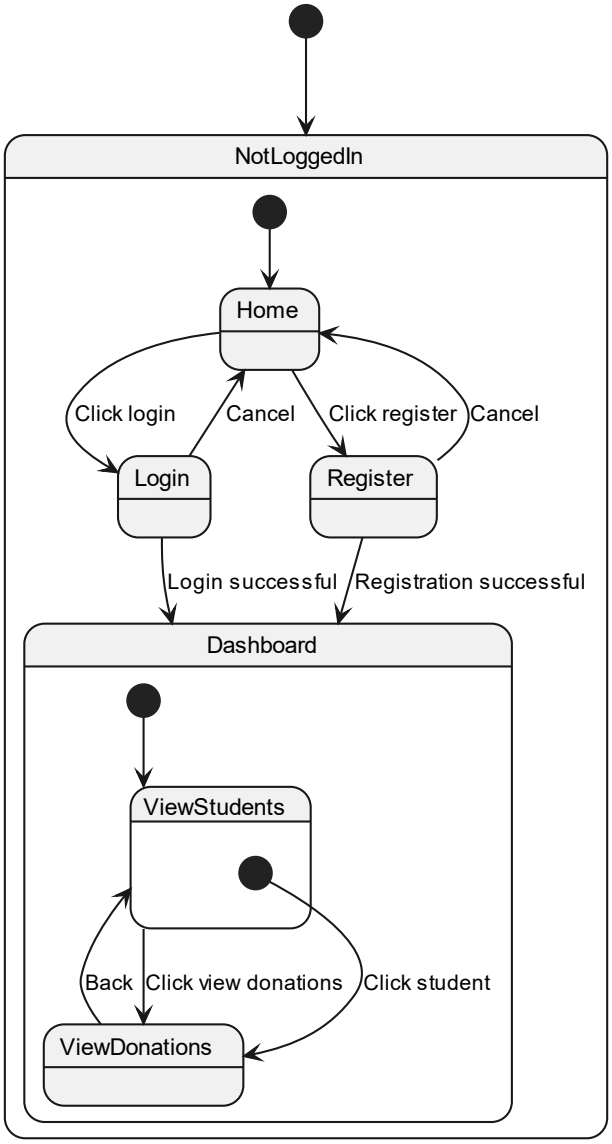


Figure 12:Donor State Chart

Chapter 4: User Interface Design

4.1. Introduction

A User Interface Design mainly consists of 3 parts:

4.2. Site Map

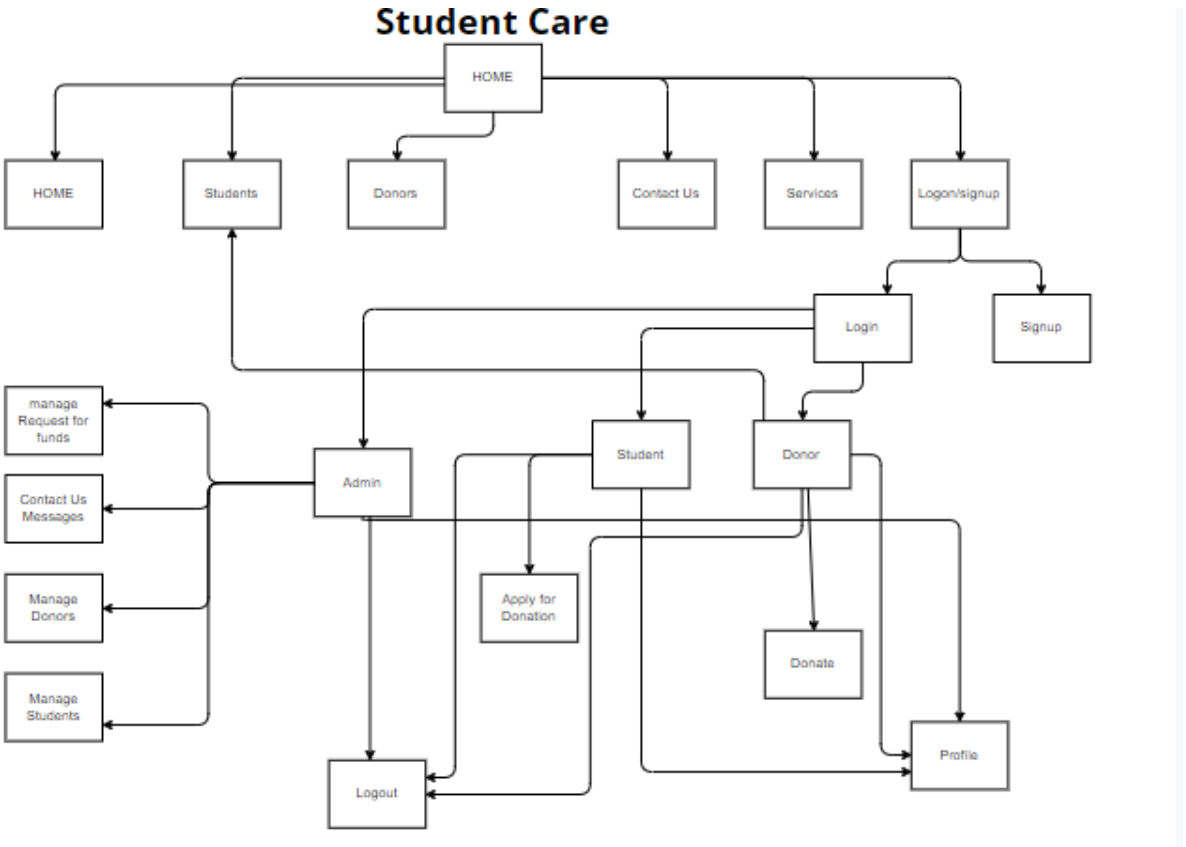


Figure 13:Site Map

4.3. Story Boards

There are different attributes involved in storyboards to represent our system described below.

4.3.1. Environment

The web application "Student Care" facilitates donation exchanges within a community, emphasizing sustainable practices and fostering connections among users. It enables donors to contribute to students in need by donating educational resources and financial support.

4.3.2. Visual Cues

In Student Care, admins access a web interface for managing donation requests from students, including approval and platform management. Students log in to request donations for educational needs and track request statuses. Donors use the platform to browse and donate to verified student requests, track contributions, and access donation records. The interface is designed to be intuitive and user-friendly, ensuring efficient interaction tailored to each user's role and responsibilities.

4.3.3. User Input

The system ensures accessibility via web browsers, providing a seamless experience for users to navigate and interact with donation requests. Donors can easily submit their donations and track their contributions through the platform.

4.3.4. Machine Output

During the login process, Student Care verifies donor credentials to ensure secure access to their dashboard and functionalities for managing donations effectively.

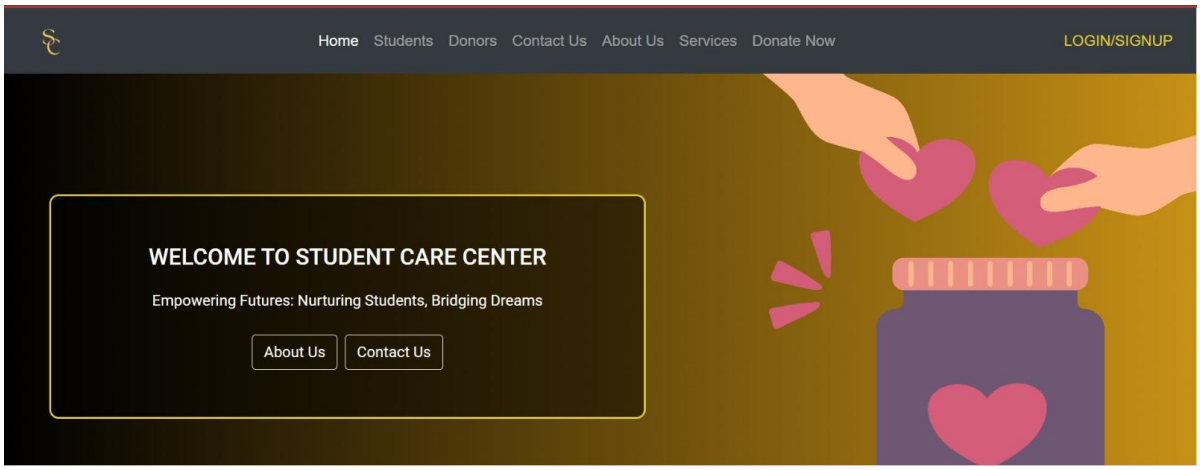
4.3.5. Technology

Student Care is developed using Angular for frontend development, Node.js and Express for backend services, and MongoDB for database management. These technologies enable scalability, real-time data management, and efficient user interactions.

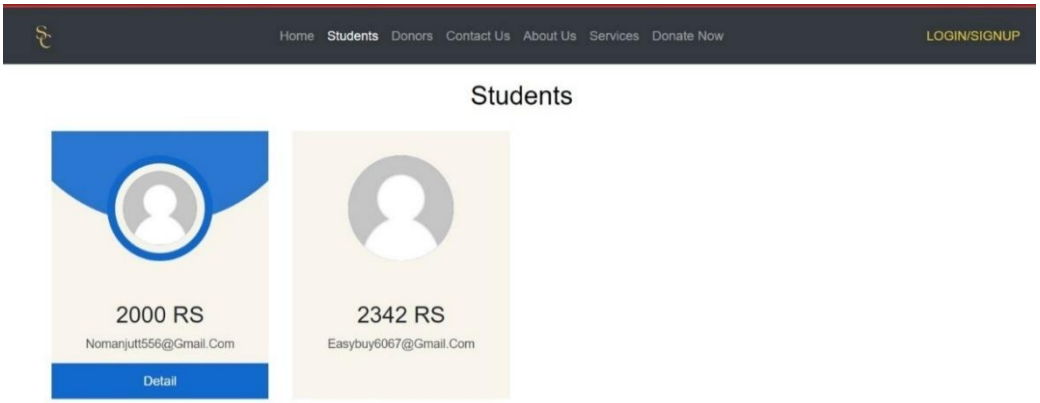
4.3.6. Quality of Experience

During the development of Student Care, we have employed cutting-edge technologies to craft a streamlined and intuitive platform. Student Care offers users a seamless experience for facilitating donations, fostering community support, and ensuring efficient management.

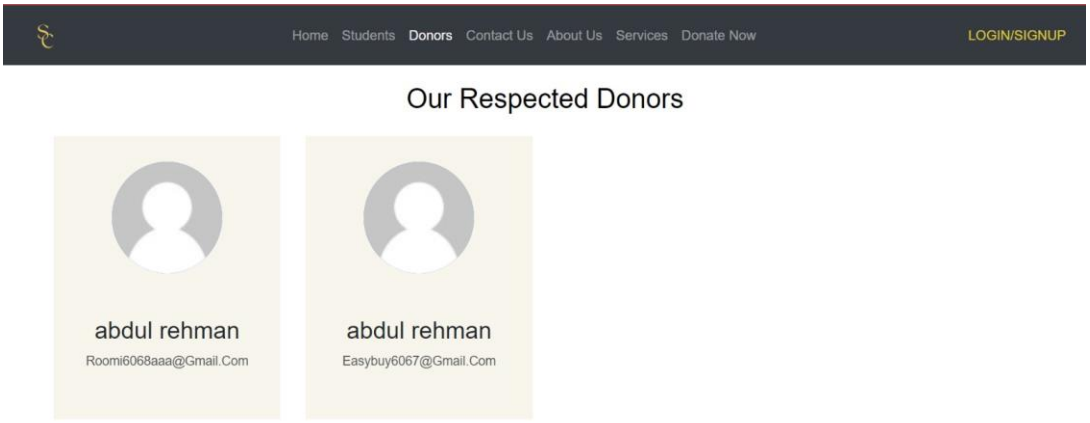
4.3.7.Storyboards/Wireframes



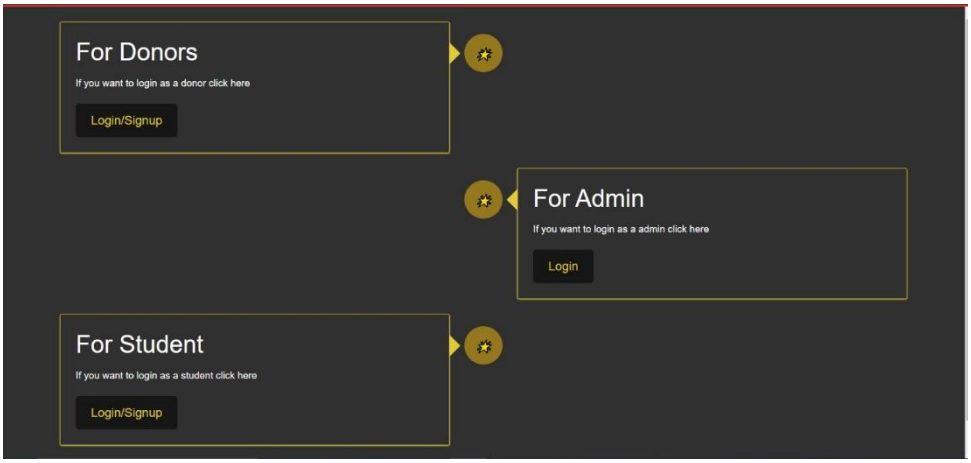
Home Screen



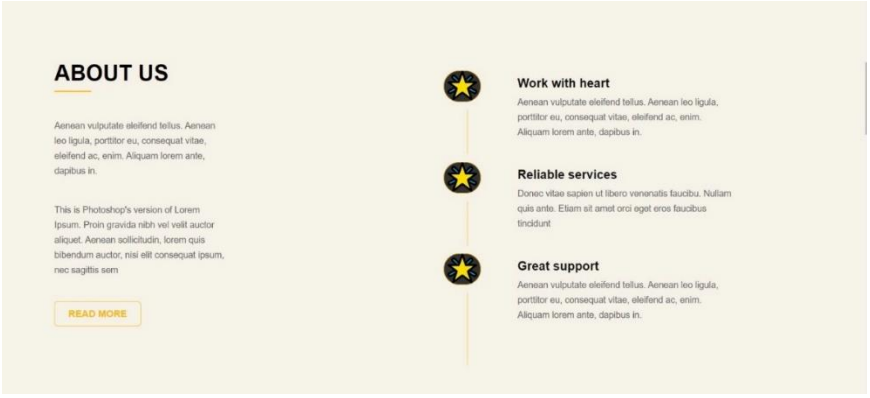
Students Page



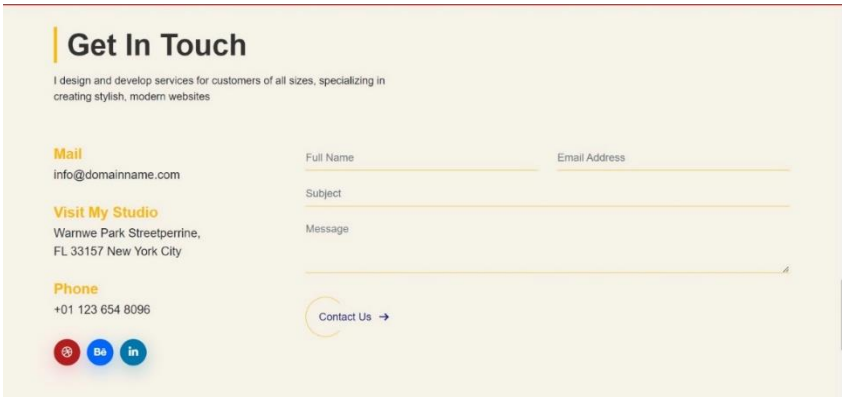
Donors Page



Login/signup Page



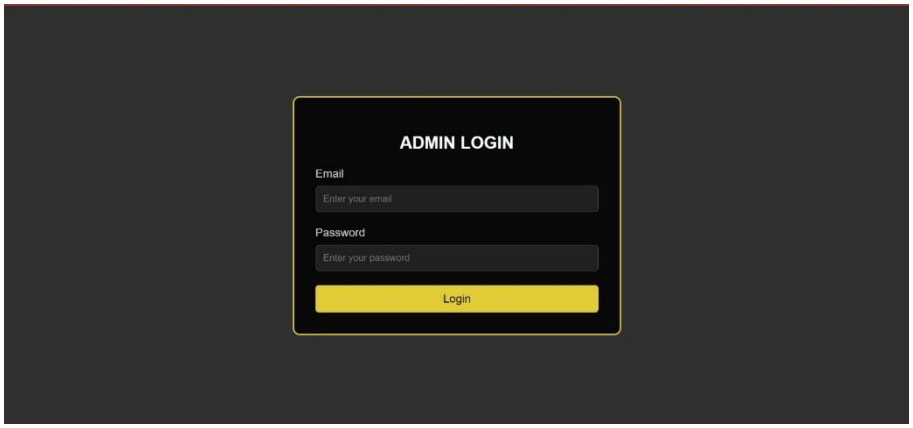
About Us Page



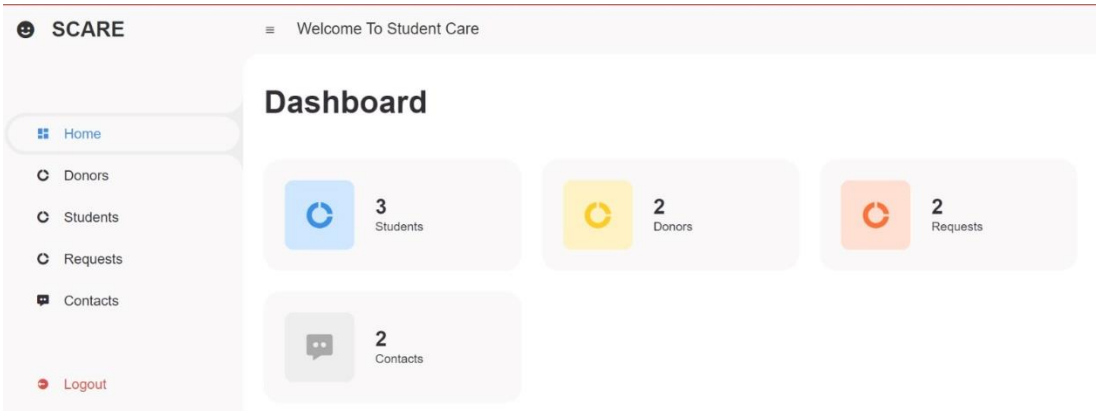
Contact Us Page

4.4. Navigation Maps

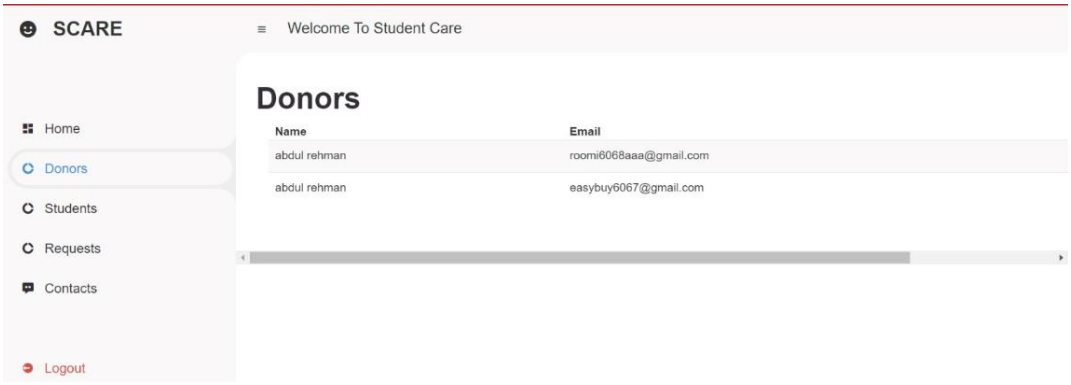
4.4.1 Login as Admin



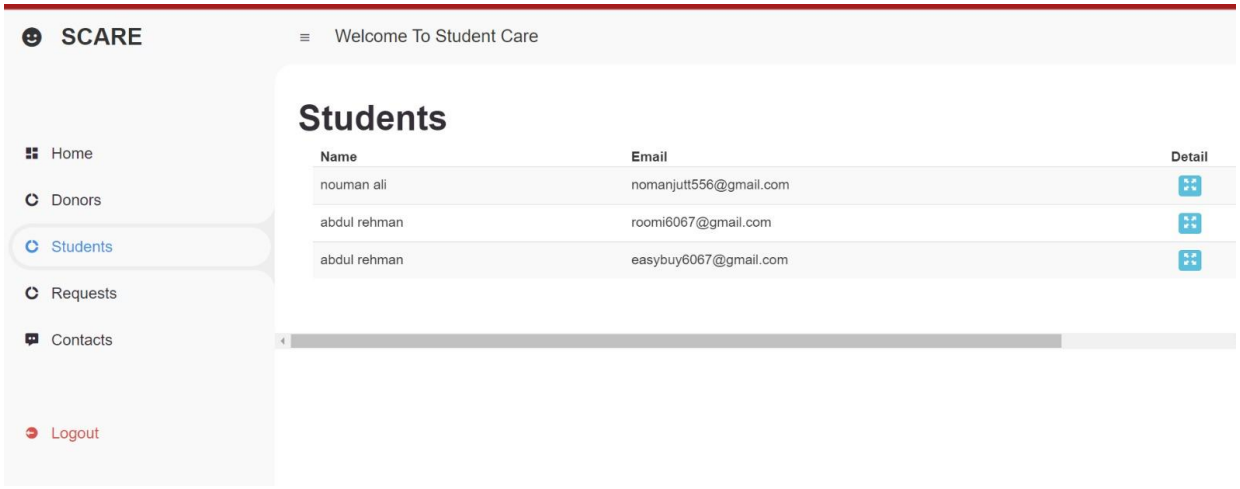
Admin’s login Page



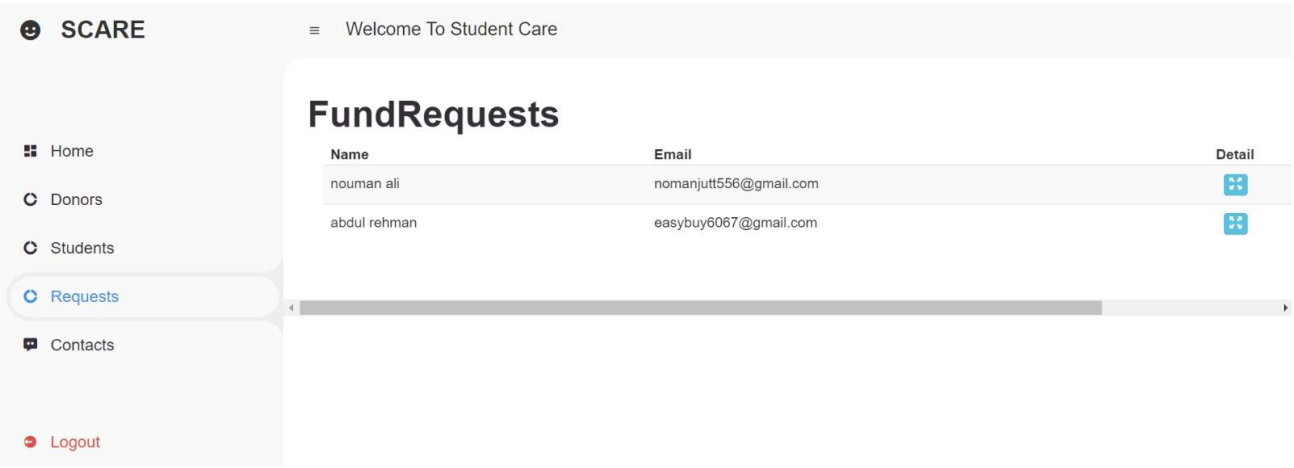
Admin Dashboard Page



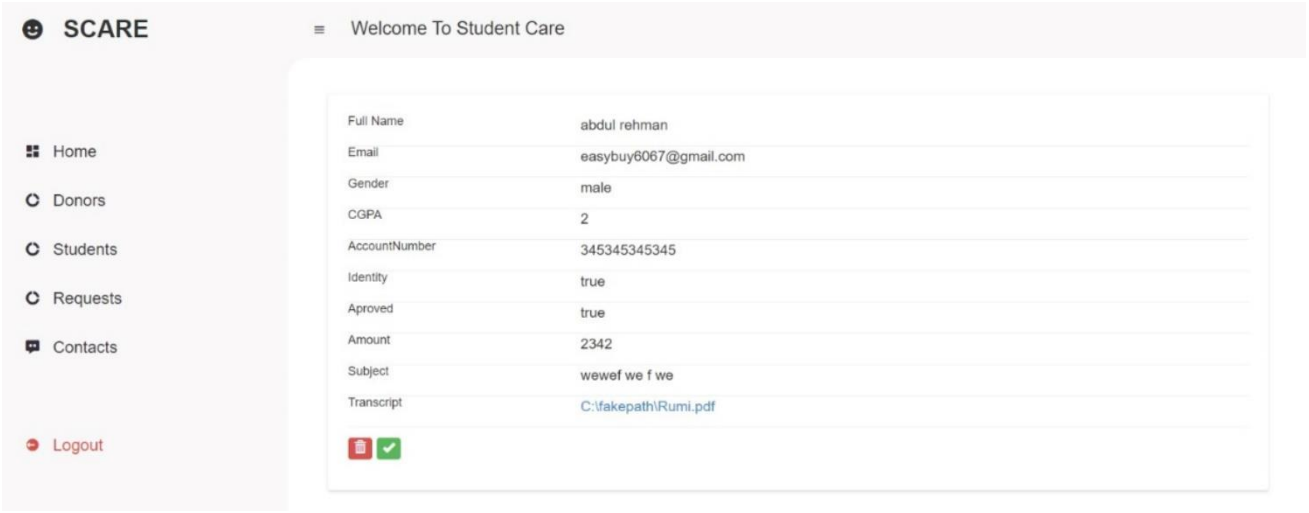
Donors view Page



Students view Page

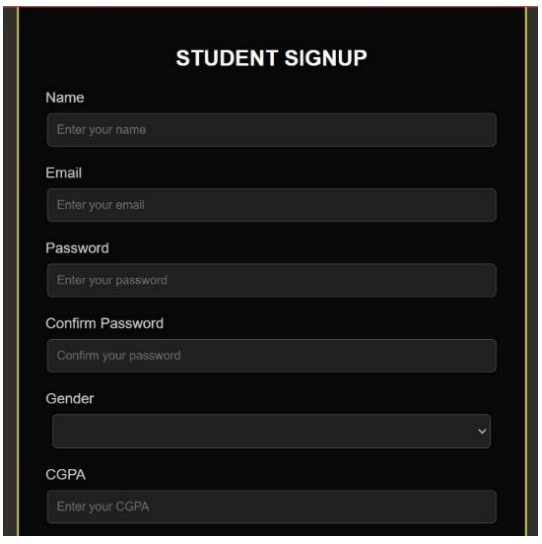


Fund Request view Page

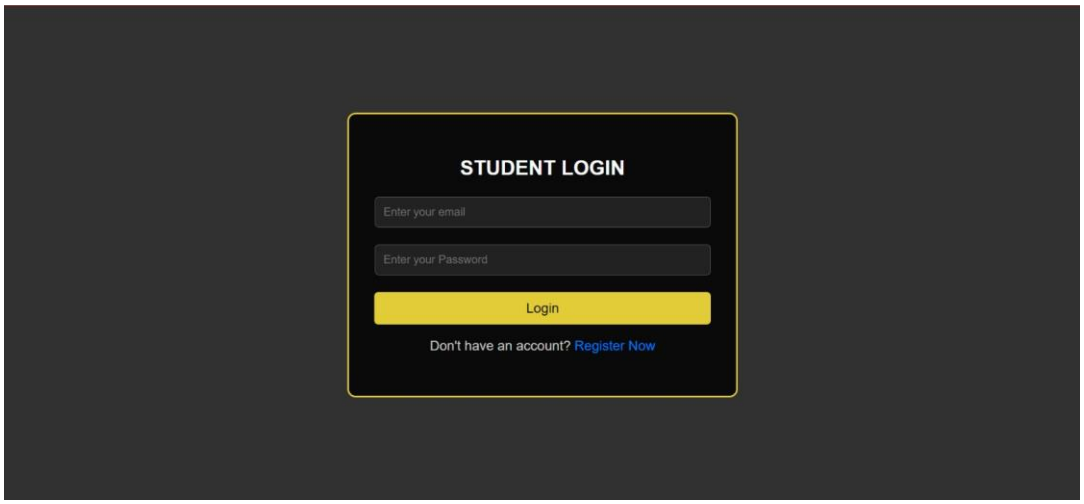


Detailed Request view Page

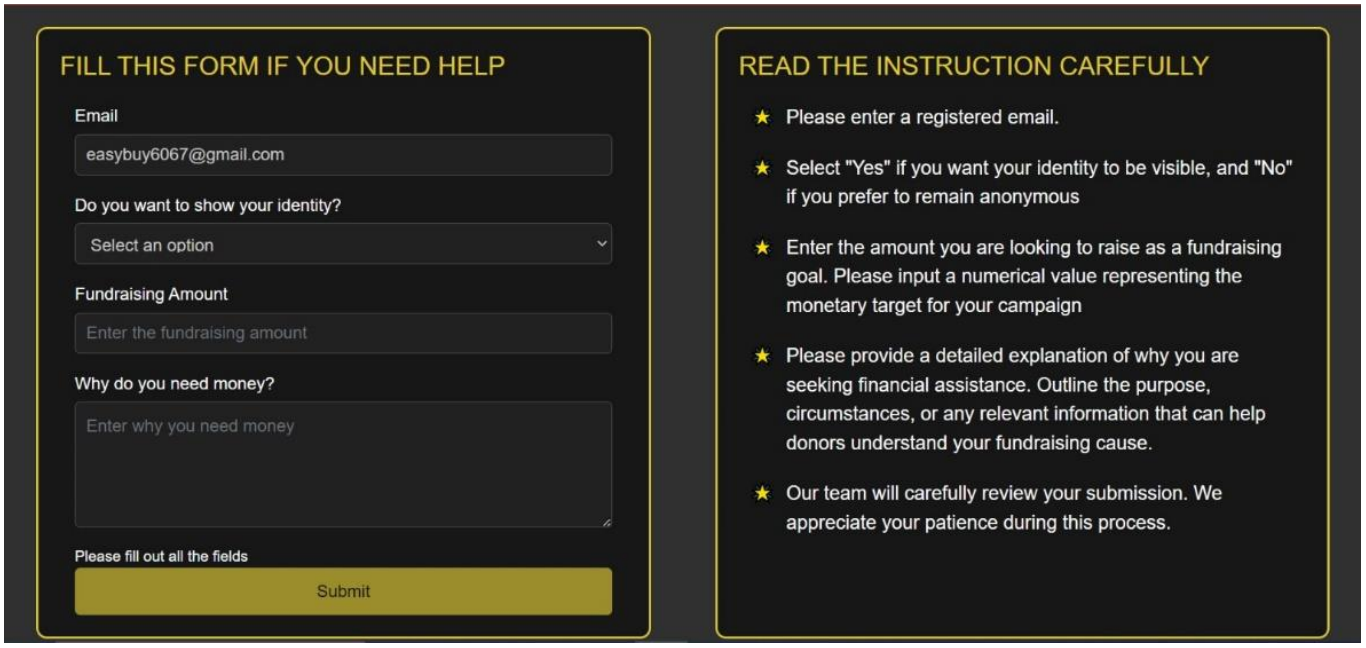
4.4.2 Login & Signup as Student



Signup Page

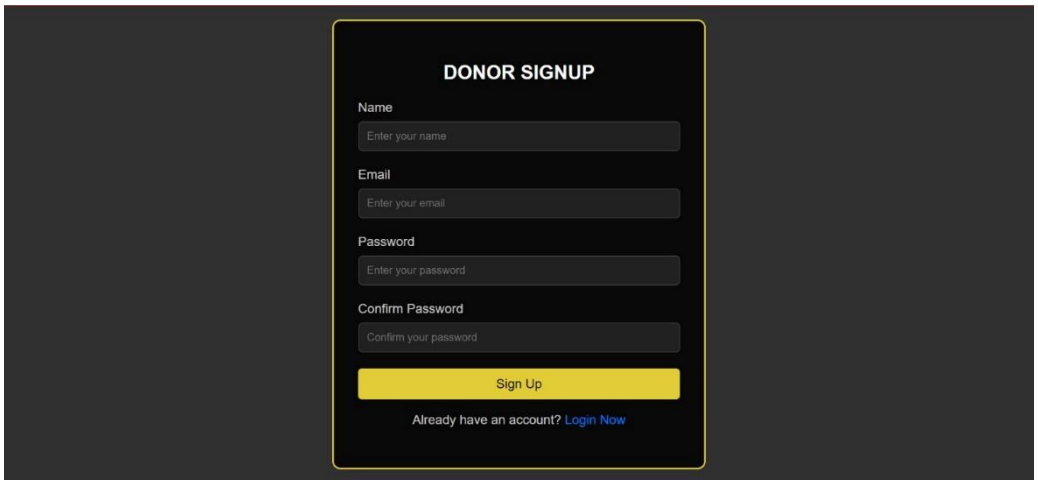


Login Page

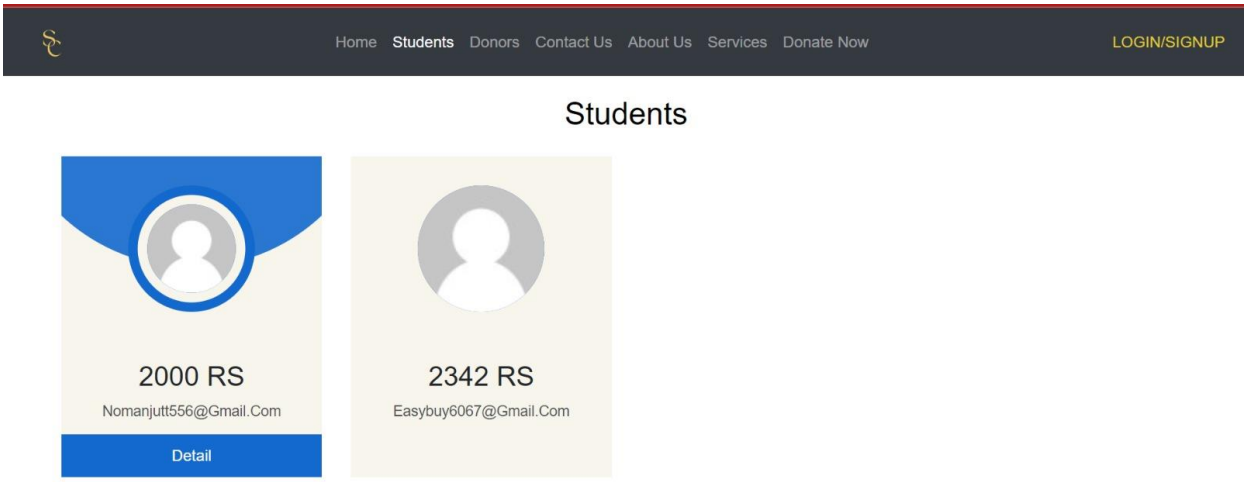


Application for Donation Page

4.4.2 Login & Signup as Donate



Signup Page



View Students Page

Chapter 5: Software Testing

5.1 Introduction

This chapter is centered around the IEEE standard for software testing. This particular standard outlines a series of fundamental test documents that pertain to the dynamic facets of software testing, which involve the actual execution of procedures and code. The standard provides clear definitions for the purpose, structure, and content of each of these core documents. Although the documents detailed within the standard primarily pertain to dynamic testing, several of them can also be relevant to other testing activities. For example, the test plan and test incident report could find utility in activities such as design and code reviews. This standard comprehensively addresses the documentation needs for both the initial testing during software development and the subsequent testing phases for software releases.
The subsequent list represents the standardized artifacts that are required to be included within this chapter.

- Test Plan
- Test Design Specification
- Test Case Specification
- Test Procedure Specification
- Test Item Transmittal Report
- Test Log
- Test Incident Report
- Test Summary Report

5.2. Test Plan

5.2.1. Purpose

The purpose of this plan is to test the real-time developed system. To prescribe the scope, approach, resources, and schedule of the testing activities. To identify the items being tested, the features to be tested, the testing tasks to be performed, the personnel responsible for each task, and the risks associated with this plan.

5.2.2. Outline

A test plan shall have the following structure:

- Test plan identifier
- Introduction
- Test items
- Features to be tested.
- Features not to be tested.
- Approach
- Item pass/fail criteria.
- Suspension criteria and resumption requirements
- Test deliverables
- Testing tasks
- Environmental needs
- Responsibilities
- Staffing and training needs
- Schedule
- Risks and contingencies
- Approvals

5.2.2.1 Test Plan Identifier

Testing Level TP-001

5.2.2.2 Introduction

This test plan encompasses the comprehensive testing of the entire system, comprising modules and features collectively. The assessment of developed system features will be executed manually over a period of 6-9 days. Additionally, the testing process will take place within an environment that is populated with relevant test data.

5.2.2.3 Test Items

- User Registration Module
- Login and Authentication Module
- Student Verification Module
- Fund Request and Approval Module
- Payment Processing Module
- Donor Interaction and Messaging Module
- Notifications and Alerts Module
- Integration with Database and Backend APIs
- Compatibility Testing on Various Devices
- Performance and Load Testing
- Cross-Browser and Cross-Device Testing

5.2.2.4 Features to be Tested

All features will be tested to check whether they are functioning properly. All modules will be tested.

- User Registration and Sign-Up
- User Login and Authentication
- Student Verification Process
- Fund Request Submission and Tracking
- Payment Processing and Confirmation
- Donor Interaction
- Notification System
- Account Profile Management
- View Donation History
- Validation in Forms
- Cross-Browser and Cross-Device Functionality
- Logout Functionality
- Integration with Database and Backend APIs
- Performance and Load Testing
- Compatibility Testing on Various Devices
- Admin Dashboard and Reports
- Manage Student Accounts
- Manage Donor Accounts
- Monitor Donations and Resolve Issues

5.2.2.5 Features not to be Tested

- Integration with External Third-Party Service: The testing of integration with external services like payment gateways or messaging services is beyond the control of the Student Care application and falls under the responsibility of those service providers.
- Third-Party Hardware Compatibility: Ensuring compatibility with specific third-party hardware devices or peripherals, such as printers or scanners, is outside the scope of Student Care's testing plan.
- Non-Functional Testing of Third-Party Service: Comprehensive performance, security, or compatibility testing of third-party services (e.g., external payment processors) will not be performed as these are managed by their respective providers.

- Network Infrastructure Testing: Testing the underlying network infrastructure or its components, such as routers, switches, or firewalls, is not part of Student Care's testing plan and is assumed to be the responsibility of the network administrators.
- Hardware Reliability: The reliability of hardware devices used to access the application, such as computers or mobile devices, is not within the testing scope of Student Care and is expected to be handled by the users or their IT departments.
- Operating System Updates: Compatibility testing with new or upcoming operating system updates beyond the current versions supported is not covered in this plan as these updates are frequent and outside the control of Student Care.
- External Data Sources: The accuracy or reliability of data obtained from external sources (e.g., APIs for location services) is not included in the testing plan, as these are managed by the external data providers.
- Third-Party Software Integration: In-depth testing of third-party software integrated with Student Care, unless explicitly specified, is not covered, as these integrations are managed by the third-party providers.
- Business Logic of External Services: Testing the business logic of external services or APIs integrated with Student Care is not within the scope, as the functionality and reliability of these services are under the control of their respective providers.

5.2.2.6. Approach

Individual members will conduct independent testing of the system services. Each member will manually perform the testing of specific features and designate the outcome as either Pass or Fail for each test case. The testers will document both the observed result and any pertinent information. Once all tests have been concluded, the test manager will review the test reports and provide appropriate feedback to the team.

5.2.2.7 Item Pass/Fail Criteria

Criteria is set with the help of software requirements specification version 1.0. All testing tasks will be performed within the ratio of 100% in which the passing criteria is 75%. If the test results are less than 75%, it will be considered a failure.

- If the system doesn't work properly, it will be considered as fail case.
- If expected page of application won't appear then it will be considered as fail case

5.2.2.8 Suspension criteria and resumption requirements

Testing work is performed by the individual to ensure that all the functional activities are working well in the system. If the system experiences any error during the middle of a test case, avoid the system to be shutdown. Instead of it, pause the testing procedure and perform updating to deal with error. So here are some criteria for which we will pause the test work for the application e.g.

- If any test cases failed.
- If modules are not working well. Resumed the test procedure and checked the results. In case of failure, repeat the cycle until the required results are obtained.

5.2.2.9 Test Deliverables

After completion, the test result will be saved, and the test manager should circulate the complete test report to the team members and other concerned authorities.

5.2.2.10 Testing Tasks

The following activities must be completed:

- Test plan should be prepared.
- Functional specifications written and delivered to the testing team.
- Perform the tests.
- Prepare test summary report.

Task ID	Testing Task	Inter Task Dependencies
TT_001	Application connectivity with internet	None
TT_002	Admin Registration	TT_001
TT_003	Donor Registration	TT_001
TT_004	Student Registration	TT_001
TT_005	Admin Login Page	TT_001
TT_006	Donor Login Page	TT_001
TT_007	Student Login Page	TT_001
TT_008	Authentication of Users (Admin, Donor, Student)	TT_002, TT_003, TT_004, TT_005, TT_006, TT_007
TT_009	Admin Verification of Students	TT_008
TT_010	Notification System	TT_008, TT_009
TT_011	Payment Method Integration	TT_008
TT_013	User CRUD operations (Admin, Donor, Student)	TT_002, TT_003, TT_004, TT_005, TT_006, TT_007
TT_014	Application UI testing	TT_001 to TT_0013
TT_015	Server availability	TT_002, TT_003, TT_004, TT_006, TT_007

5.2.2.11 Environmental Needs

The following outlines the fundamental hardware and software prerequisites for the environment required to run our application:

- Internet Access: Stable and reliable internet connection required.
- Access: Users need authorized login credentials.
- Browser Compatibility: Supports modern browsers like Chrome, Firefox, Safari, Edge.
- Hardware Requirements: Basic requirements for running modern web applications.

5.2.2.12 Responsibilities

Since each team member has been assigned specific tasks, they are individually responsible for testing their respective assignments. A designated team member has been chosen to compile all the test results, store them, and distribute them to the relevant parties.

5.2.2.13 Staffing and training needs

There is no need for external staff members to be involved in the testing of this project. All the necessary testing will be conducted by the team members themselves. Each member possesses domain knowledge relevant to their respective fields.

5.2.2.14 Schedule

The testing is expected to span from 6 to 9 days.

Testing Task	Time
Unit Testing	2 Days
Integration Testing	1 Days
User Registration Testing	1 Days
Login and Authentication	1 Day
Student Management	1 Day
Donor Management	1 Day
Payment Integration	1 Day

Table 1: Testing Schedule

5.2.2.15 Risks and Contingencies

The absence of a fundamental understanding of application testing by the tester could potentially lead to delays or inadequate testing execution.

- Testing Expertise: Limited understanding of testing methodologies may delay or compromise testing quality. Regular training sessions will enhance testers' skills.
- Time Management: Effective project scheduling is critical due to extensive testing requirements. Strict timelines and regular reviews will ensure timely completion.
- Internet Connectivity: Unreliable internet access can disrupt testing and affect results. Backup plans and stable connections are essential to mitigate this risk.
- Risk Identification: Failure to identify hidden risks may impact testing and project success. Ongoing risk assessments and proactive measures will address potential issues early.
- Data Availability: Insufficient test data can hinder thorough testing and accurate evaluation. Robust data management strategies will ensure adequate and reliable data for testing purposes.

5.2.2.16 Approvals

Name	Title	Signature
Mr. Najeeb-Ur-Rehman	Project Manager	

5.3 Test Design Specification

5.3.1 Purpose

It records which design features of a test item are to test, and how successful the test of these features would be recognized. Test design specification ensures all functional and design requirements are implemented as specified in the documentation.

5.3.2 Outline

A test plan shall have the following structure.

a. Test plan identifier
b. Introduction
c. Test items
d. Features to be tested.
e. Features not to be tested.
f. Approach.
g. Item pass/fail criteria.
h. Suspension criteria and resumption requirements
i. Test deliverables
j. Testing tasks
k. Environmental needs
l. Responsibilities
m. Staffing and training needs
n. Schedule
o. Risks and contingencies
p. Approvals.

5.3.2.1 Test Plan Identifier

Test design specification identifier is **TP_002**

5.3.2.2. Introduction

A Test Plan Identifier includes:

- Project Authorization: Student Care was authorized and approved for development as part of an educational initiative aimed at enhancing student support services.
- Project Plan: The project goal is to create a comprehensive student care platform that integrates various functionalities to support students' well-being and academic success.
- Quality Assurance Plan: By following industry best practices and continuous performance monitoring, our objective is to deliver a robust and user-friendly application.
- Configuration Management Plan: We ensure consistency, control, and thorough documentation of development and testing environments throughout the project lifecycle.
- Relevant Policies: Adherence to relevant policies, best practices, and ethical considerations ensures alignment with industry norms and regulatory requirements.
- Relevant Standards: Application of established standards ensures that Student Care meets quality expectations and delivers a high-standard user experience

5.3.2.3. Test Items

By identifying the test items, we ensure a systematic and thorough assessment of the app's features and capabilities.

- User Frontend
- Backend
- Rest API’s
- Database
- User Profile

5.3.2.4. Test Items

- Access the Application
- Login as a Registered User
- View Student Profiles
- Verify Student

- Send Donation Requests
- View Donation Requests Received
- View Donation Requests Sent
- Accept/Decline Incoming Donation Requests
- Cancel Outgoing Donation Requests
- Logout

5.3.2.5. Features not to be tested

- Third Party Services
- Backward Compatibility
- Load Testing for Extreme Conditions
- Connectivity Failures

5.3.2.6. Approach

The test approach is divided into three main phases: Unit testing, integration testing and system testing. Unit testing focuses on the smallest unit of software design. In this test an individual unit or group of interrelated units are tested. The objective of integration testing is to take unit tested components and build a program structure component that are combined to produce output. Every time a new module is added leads that have been dictated by design. Integration testing is testing in which a group of to changes in program. This type of testing makes sure that the whole component works properly even after adding components to the complete program.

5.3.2.7 Item Pass/Fail Criteria

Criteria is set with the help of software requirements specification version 1.0. All testing tasks will be performed within the ratio of 100% in which the passing criteria is 75%. If the test results are less than 75%, it will be considered a failure.

- If the system doesn’t work properly, it will be considered as fail case.
- If expected page of application won't appear then it will be considered as fail case

5.3.2.8 Suspension criteria and resumption requirements

Suspension Criteria:

We suspended Testing activities under the following circumstances:

- If critical defects are identified that significantly impact the functionality or usability of the app.
- If external factors, such as network outages or system downtime, hinder the ability to conduct testing.
- If there are significant changes to the application's scope, architecture, or requirements.

Resumption Requirements:

We resumed Testing activities under the following conditions:

- Once critical defects are addressed and verified as resolved, testing will be resumed.
- When essential testing resources become available, activities will be resumed.

5.3.2.9 Test Deliverables

Identify the deliverable documents. The following documents should be included:

a. Test plan
b. Test design specifications
c. Test case specifications
d. Test procedure specifications
e. Test item transmittal reports
f. Test logs
g. Test incident reports
h. Test summary reports

5.3.2.10 Testing Tasks

- Test Planning
- Requirements Analysis
- Test Case Design
- Test Data Preparation
- Test Environment Setup
- Test Execution

5.3.2.11 Environmental Needs

- A Windows, Mac, or Linux device with a minimum of 4GB RAM.
- Node.js and npm (Node Package Manager) installed.
- Angular CLI.
- MongoDB server.
- Visual Studio Code or any other preferred code editor.
- Git for version control.
- Internet connection for accessing online resources and APIs.
- Web browser (Chrome, Firefox, etc.) for testing and debugging the application.

5.3.2.12 Responsibilities

Each member is responsible for requirements, designing, preparing documentation, developing and executing the system properly.

5.3.2.13 Staffing and training needs

Each member was able to provide adequate staffing and fulfil training needs.

5.3.2.14 Schedule

Additional test milestones will take more than 14 days.

5.3.2.15 Risks and Contingencies

- Testing Expertise: Limited understanding of testing methodologies may delay or compromise testing quality. Regular training sessions will enhance testers' skills.
- Time Management: Effective project scheduling is critical due to extensive testing requirements. Strict timelines and regular reviews will ensure timely completion.
- Internet Connectivity: Unreliable internet access can disrupt testing and affect results. Backup plans and stable connections are essential to mitigate this risk.
- Risk Identification: Failure to identify hidden risks may impact testing and project success. Ongoing risk assessments and proactive measures will address potential issues early.
- Data Availability: Insufficient test data can hinder thorough testing and accurate evaluation. Robust data management strategies will ensure adequate and reliable data for testing purposes.

5.3.2.16 Approvals

Name	Title	Signature
Mr. Najeeb-Ur-Rehman	Project Manager	

5.4 Test Case Specification

5.4.1 Purpose

The purpose of test case specification is to provide clear guidance to the end user regarding the utilization of the system and its associated application. It identifies necessary inputs and anticipates the anticipated outcomes. This document offers comprehensive, step-by-step procedures for executing the tests effectively.

5.4.2 Outline

A test case specification shall have the following structure:

- A test case specification shall have the following structure:
- Test case specification identifier
- Test items
- Input specifications
- Output specifications
- Environmental needs
- Special procedural requirements

5.4.2.1 Test Case Specification Identifier

Identifier for this module is TPS 001

5.4.2.2. Test Items

Identify and briefly describe the items and features to be exercised by this test case.

For each item, consider supplying references to the following test item documentation:

a. Requirements specification

b. Design specification
c. Users guide
d. Operations guide
e. Installation guide

5.4.2.3. Test Items

By identifying the test items, we ensure a systematic and thorough assessment of the app's features and capabilities.

- User Frontend
- Backend
- APIs
- Database
- User Registration and Authentication
- Fund Request Management
- Payment Processing
- Notification System
- Profile Management
- Admin Controls
- Data Security and Privacy
- System Performance
- User Interface Testing

5.4.2.3. Input Specifications

User Registration and Authentication

- Username, Email, Password, User Type
- User Database
- Confirmation Email

Fund Request Management

- Student ID, Fund Amount, Justification
- Fund Requests Database
- Confirmation Message

Payment Processing

- Payment Method, Amount, Confirmation
- Payment Transactions Database
- Confirmation Email

Notification System

- Notification Type, User ID, Message Content
- Notifications Database
- Notification Message

Profile Management

- User ID, Profile Information
- User Profile Database
- Update Confirmation

Admin Controls

- Admin ID, Action Type, Relevant Data
- Admin Actions Database
- Action Confirmation

5.4.2.4. Output Specifications

User Registration and Authentication

- Successful Registration Confirmation
- Error Messages for Invalid Inputs
- Account Activation Email

Fund Request Management

- Fund Request Submission Confirmation
- Request Status Updates (Approved, Declined, Pending)
- Error Messages for Invalid Requests

Payment Processing

- Payment Successful Confirmation
- Payment Failed/Error Messages
- Transaction Receipts

Notification System

- Successful Notification Sent Confirmation
- Error Messages for Failed Notification

Profile Management

- Profile Update Successful Confirmation
- Error Messages for Invalid Inputs

Admin Controls

- Action Successful Confirmation
- Error Messages for Invalid Actions
- Notifications Sent to Relevant Users

5.4.2.5. Environmental Needs

5.4.2.5.1. Hardware

- Device with at least 4GB RAM
- 2GHz dual-core processor
- 20GB available storage

5.4.2.5.2. Software

- Operating System: Windows 7 or higher, macOS 10.13 or higher, Linux distributions (e.g., Ubuntu 18.04).
- Development Tools: Node.js, Angular CLI, MongoDB
- Browsers: Latest versions of Chrome, Firefox, Safari, and Edge

5.4.2.6. Special Procedural Requirements

- **Setup**: Ensure all test data is preloaded into the database.
- **Operator Intervention**: Manual verification of specific scenarios such as fund approval and notifications.
- **Wrap Up**: Ensure all test data is cleared and the system is reset to initial state

5.4.2.7. Inter Case Dependencies

- TC_002 (Login to the application) must be executed before TC_004 (Student Registration)

- TC_004 (Student Registration) must be executed before TC_005 (Fund Request Management)
- TC_005 (Fund Request Management) must be executed before TC_010 (Notification System).

5.5 Test Procedure Specification

5.5.1. Purpose

It will specify how the tester will physically run the test, the physical set-up required, and the procedure steps that need to be followed. It contains a sequence of actions required for the execution of a test.

5.5.2 Outline

A test procedure specification shall have the following structure:

- Test procedure specification identifier
- Purpose
- Special requirements
- Procedure steps

5.5.2.1. Test Procedure Specification identifier

Identifier for this module is TPS _001

5.5.2.2. Purpose

This module describes the testing procedure for the entire Student Care system. All tests will be performed manually within the specified time constraints. The procedure executes the following test cases: TC_001, TC_002, TC_003, TC_004, TC_005, TC_006, TC_007, TC_008, TC_009, TC_010, TC_011, TC_012, TC_013. The purpose is to specify the steps for executing these test cases.

5.5.2.3. Special Requirements

No special requirements are needed for testing this web application manually.

5.5.2.4. Procedure Steps

5.5.2.4.1. Set Up

- Ensure the application is functional.
- Ensure a stable internet connection.
- Prepare necessary test data.
- Log in with appropriate credentials.

5.5.2.4.2. Start

- Connect the application to the internet.
- Log in as an admin, donor, or student.

5.5.2.4.3. Proceed

Execute test cases in the order of their dependencies:

- Start with test cases that have no dependencies or where dependencies have already been resolved.

5.5.2.4.4. Shut down

- Log out from the application.
- Disconnect the server if using a local environment.

5.5.2.4.5. Restart

- Reconnect to the internet.
- Restart the server if using a local environment.
- Log back into the application.

5.5.2.4.6. Stop

- Log out of the application.

5.5.2.4.7. Wrap up!

- Collect and document all test results.

5.5.2.4.8. Contingencies

We need to submit bug reports. The test manager will use that information to create incident reports.

5.6 Test Item Transmittal Report

5.6.1. Purpose

The purpose of Test Items Transmittal Report is to identify the test items being transmitted for testing. This identifies a software item being given to the software group for testing.

5.6.2 Outline

A test item transmittal report shall have the following structure:

a. Transmittal report identifier
b. Transmitted items
c. Location
d. Status
e. Approvals

5.5.2.1. Transmittal Report Identifier

Identifier for this module is TR _001

5.6.2.2. Transmitted Items

The Student Care application, including the admin, donor, and student modules, is transmitted to the testing team. All team members are responsible for its transmittal and use during testing.

5.6.2.3. Location

The Student Care system is a web application hosted on a central server. Users can access the application via the internet.

5.6.2.4. Status

- The status of all test items being transmitted is passed.
- All the test items are in working state and functioning properly.
- The actual result is the same as expected in the test plan.
- The required system is achieved.

5.6.2.5. Approvals

Name	Title	Signature
Mr. Najeeb-Ur-Rehman	Project Manager	

5.7. Test log

5.7.1. Purpose

The purpose of the test log is to provide a sequential record of relevant details about the execution of tests. This log represents the results of testing of the whole system.

5.7.2. Outline

A test log shall have the following structure:

a. Test log identifier.
b. Description.
c. Activity and event entries.

5.7.2.1. Test Log Identifier

The test log identifier is TL_001

5.7.2.2. Description

All the items were tested. The testing was conducted manually, and a few errors occurred but were fixed. Later, the website and functionalities were properly working.

5.7.2.3. Activity and Event Entries

Activity	Authorization	Date
Application connectivity with internet	Numan Ali	20-05-2024
Admin Registration	Numan Ali	21-05-2024
Donor Registration	Numan Ali	21-05-2024
Student Registration	Numan Ali	22-05-2024
Admin Login Page	Abdul Rehman	22-05-2024
Donor Login Page	Abdul Rehman	21-05-2024
Student Login Page	Abdul Rehman	21-05-2024
Authentication of Users (Admin, Donor, Student)	Abdul Rehman	22-05-2024
Admin Verification of Students	Abdul Rehman	23-05-2024
Notification System	Numan Ali	25-05-2024
Payment Method Integration	Numan Ali	25-05-2024
User CRUD operations (Admin, Donor, Student)	Numan Ali	26-05-2024
Application UI testing	Abdul Rehman	25-05-2024
Server availability	Abdul Rehman	28-05-2024

5.7.2.3.1. Execution Description

All the tests are performed by the team members as listed above. This testing will be completed within 4 weeks. The whole testing process was supervised by Mr. Najeeb-Ur-Rehman, the project supervisor.

5.7.2.3.2. Procedure Results

Task ID	Testing Task	Result	Status
TT_001	Application connectivity with internet	Application is connected	Pass
TT_002	Admin Registration	Admin Registered Successfully	Pass
TT_003	Donor Registration	Donor Registered Successfully	Pass
TT_004	Student Registration	Student Registered Successfully	Pass

TT_005	Admin Login Page	Admin Login Successful	Pass
TT_006	Donor Login Page	Donor Login Successful	Pass
TT_007	Student Login Page	Student Login Successful	Pass
TT_008	Authentication of Users (Admin, Donor, Student)	Authenticated Successfully	Pass
TT_009	Admin Verification of Students	Verified Successfully	Pass
TT_010	Notification System	Notifications Working	Pass
TT_011	Payment Method Integration	Payment Integration Successful	Pass
TT_013	User CRUD operations (Admin, Donor, Student)	CRUD Operations Successful	Pass
TT_014	Application UI testing	UI Working Properly	Pass
TT_015	Server availability	Server Working Properly	Pass

5.7.2.3.3. Environmental Information

- **Hardware**: Windows or Mac device with at least 4GB RAM.
- **Software**: Angular, Node.js, Express.js, MongoDB.
- **Other Requirements**: Stable internet connection.

5.7.2.3.4. Anomalous Events

Some anomalous events occurred with server availability but were resolved with modifications in earlier tests. A success status was achieved upon completion of testing.

5.8. Test Incident Report

5.8.1. Purpose

Before performing the actual test, all the test procedures have been designed and it is assured that all the features to be tested well but unexpectedly some sudden incidents happen due to which testing process needs to be stop for the time being taken to recover from such incidents.

5.8.2. Outline

A test incident report shall have the following structure:

a. Test incident report identifier
b. Summary
c. Incident description
d. Impact

5.8.2.1. Test Incident Report Identifier

Test incident report identifier is TIR-001

5.8.2.2. Summary

Summarize the incident. Identify the test items involved.

- Test Item: Registration
- ID: TIR-001

5.8.2.3. Incident Description

This description includes the following items:

- **Inputs**: User registration data (name, email, password, role)

- **Expected results**: User should be registered successfully and receive a confirmation email.
- **Actual results**: Registration process failed; no confirmation email sent.
- **Anomalies**: Database connection error during the registration process.
- **Date and time**: 24-05-2024, 10:30 AM
- **Procedure step**: During the submission of the registration form.
- **Environment**: Local development environment, MongoDB server, Node.js backend.
- **Attempts to repeat**: Issue repeated three times; failed consistently.
- **Testers**: Numan Ali
- **Observers**: Abdul Rehman

5.8.2.4. Impact

The incident had a major impact on the registration function of the Student Care application. Since registration is a critical feature, the failure of this function prevented users from creating new accounts, thereby halting further testing that depends on registered users. This issue required immediate attention and resolution before continuing with other test cases.

5.9. Test Summary Report

5.9.1. Purpose

The test summary report is a formal document summarizing the results of all testing performed on the features of the Student Care application. This document details the various activities conducted as part of the testing for the web application and its connectivity.

5.9.2. Outline

A test summary report for Student Care shall have the following structure:

- Test summary report identifier
- Summary
- Variances
- Comprehensiveness assessment
- Summary of results
- Evaluation
- Summary of activities
- Approvals

5.9.2.1. Test Summary Report Identifier

Test summary report identifier is **TSR _001**

5.9.2.2. Summary

All functionalities of the Student Care application were tested as per the test plan. Each test was performed according to its specified plan, and a test log was prepared accordingly. All components were tested; there were some bugs as expected, but they were subsequently fixed. Currently, each component is functioning properly.

5.9.2.3. Variances

Some errors arose during the testing of the application due to the underlying environment. Specifically, server connectivity was unreliable and took an excessive amount of time to establish.

5.9.2.4. Comprehensiveness Assessment

The testing process was carried out exactly as described in the test plan. All features outlined in the test plan were tested.

5.9.2.4. Summary of Results

Activity	Date
Application connectivity with internet	20-05-2024
Admin Registration	21-05-2024
Donor Registration	21-05-2024
Student Registration	22-05-2024
Admin Login Page	22-05-2024
Donor Login Page	21-05-2024
Student Login Page	21-05-2024
Authentication of Users (Admin, Donor, Student)	22-05-2024
Admin Verification of Students	23-05-2024
Notification System	25-05-2024
Payment Method Integration	25-05-2024
User CRUD operations (Admin, Donor, Student)	26-05-2024
Application UI testing	25-05-2024
Server availability	28-05-2024

5.9.2.6. Evaluation

The testing process was straightforward and sufficient for this phase. Active and comprehensive evaluation of problems found in the final phase led to the conclusion that users can easily perform the necessary functions of the Student Care application.

5.9.2.7. Approvals

Name	Title	Signature
Mr. Najeeb-Ur-Rehman	Project Manager	

Chapter 6: User Manual

6.1 Student Care User Manual

6.1.1 Home Page

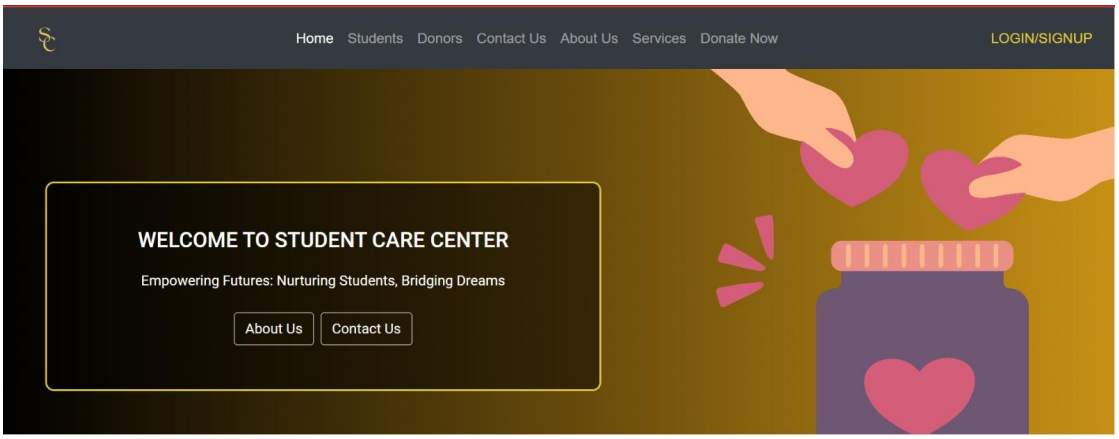


Figure 14:Home Page

navigate to different sections of the platform. It provides options for students to apply for donations, donors to view and contribute to verified student requests, and administrators to manage donation requests and donor activities."

6.1.2 About Us Page

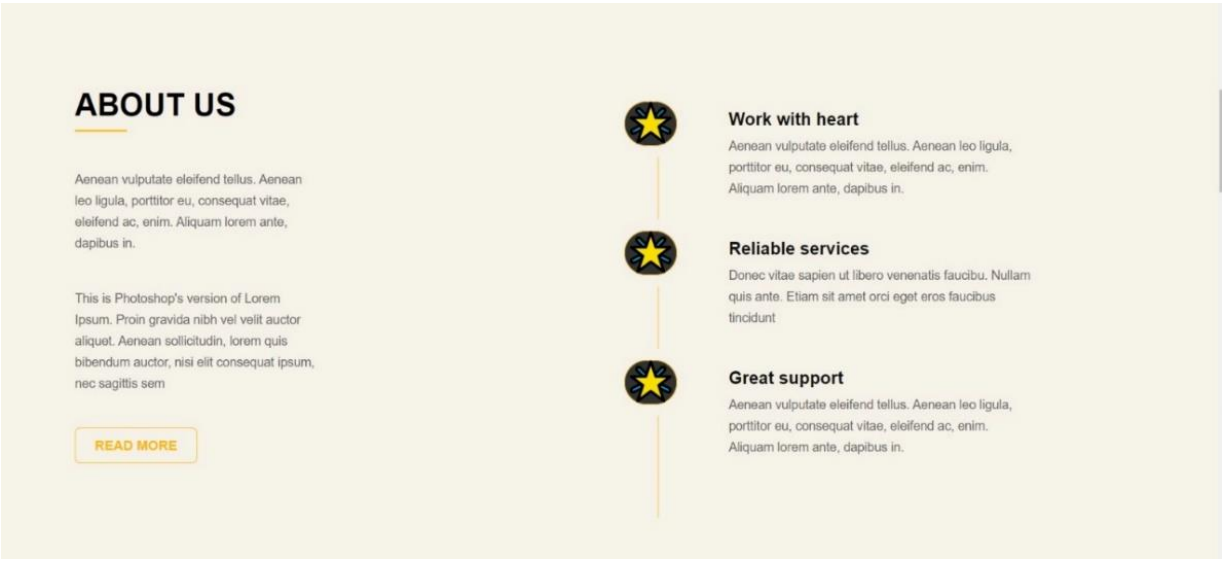


Figure 15:About Us

"The 'About' page on Student Care provides information about the platform's mission, goals, and how it facilitates donations and support among students, donors, and administrators."

6.1.3 Contact Us Screen

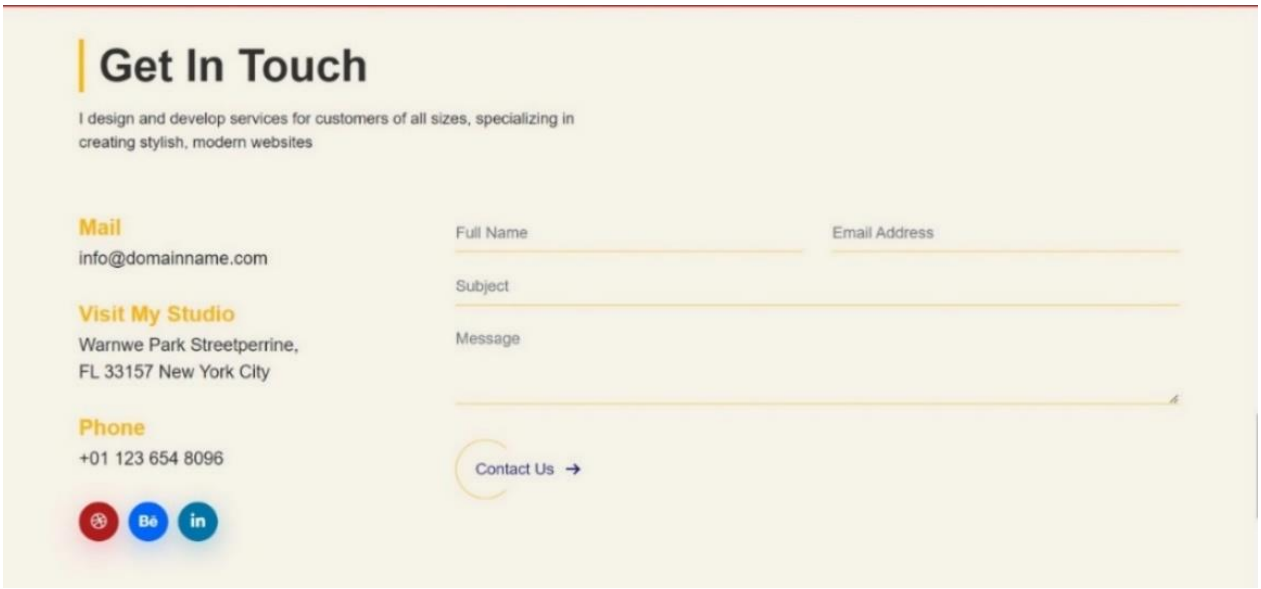


Figure 16:Contact Us

this is contact page where user can contact us about feedback and any bug issue.

6.1.4 Services Page

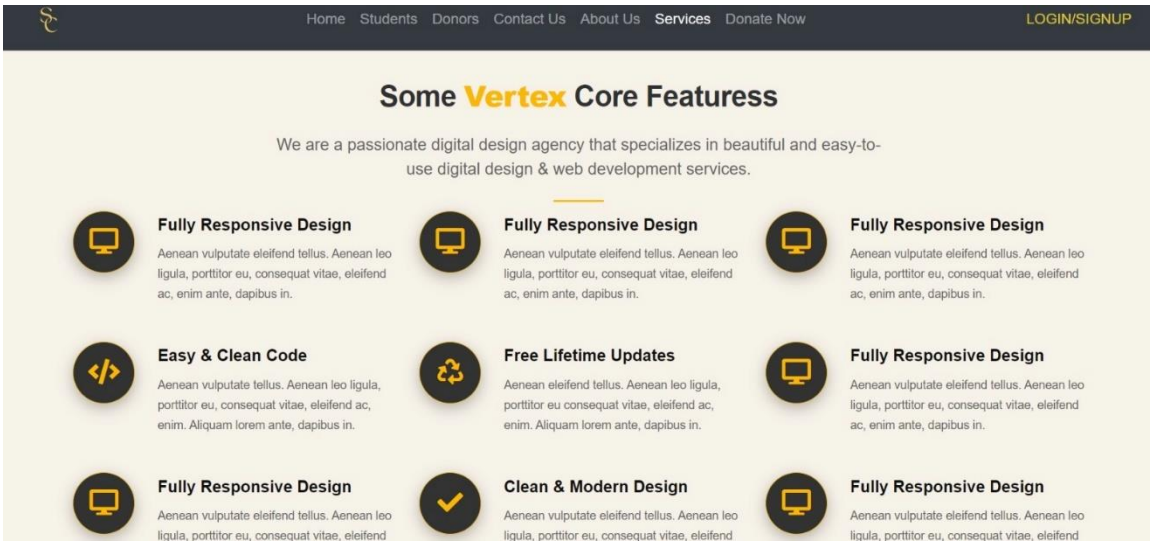


Figure 17:Services page

We are providing these services.

6.1.5 Students

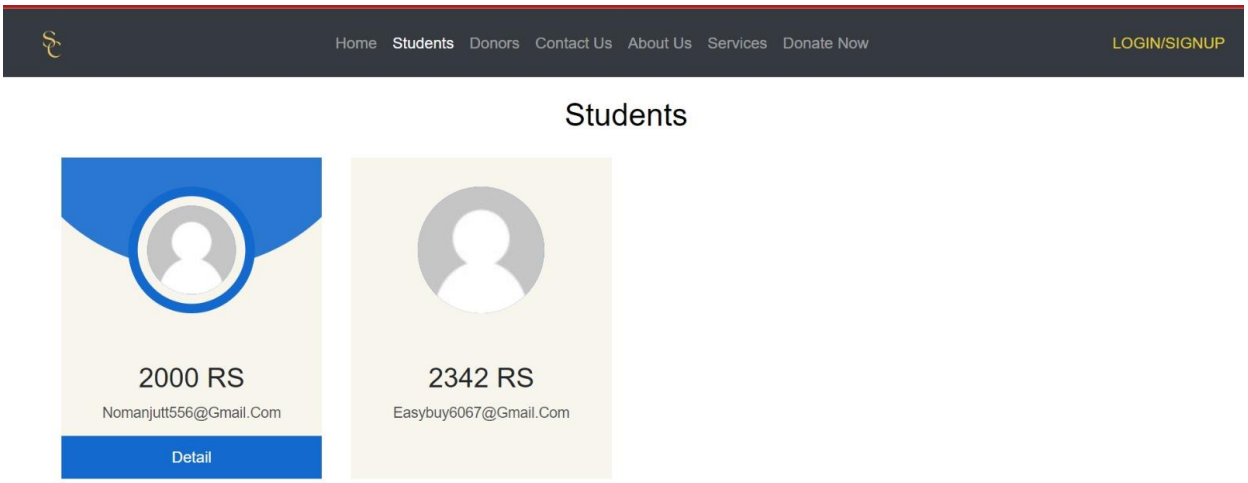


Figure 18:Students

6.1.6 Donor

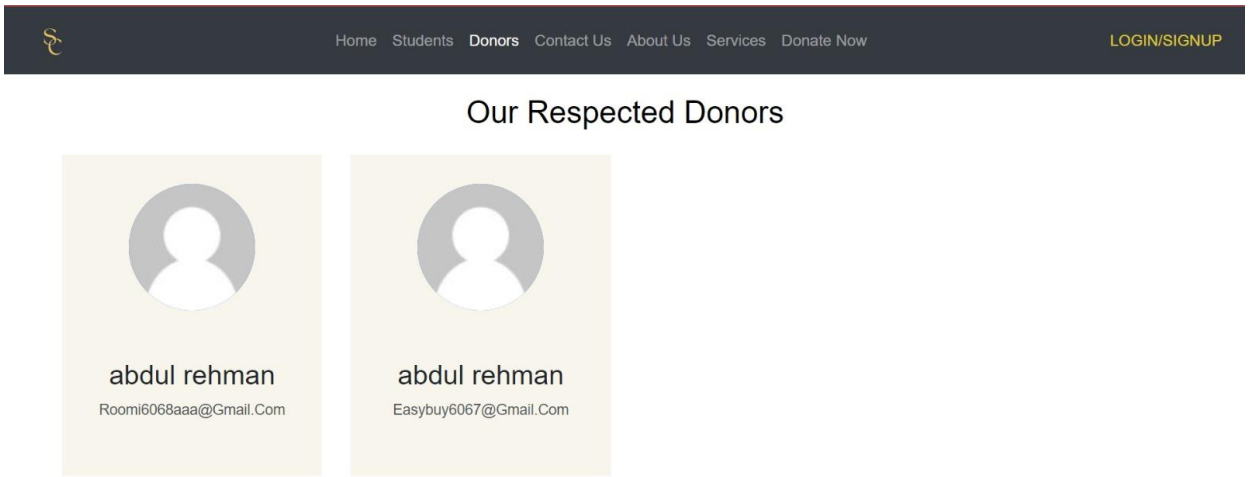


Figure 19:Donors

6.1.7 Users Login & Registration

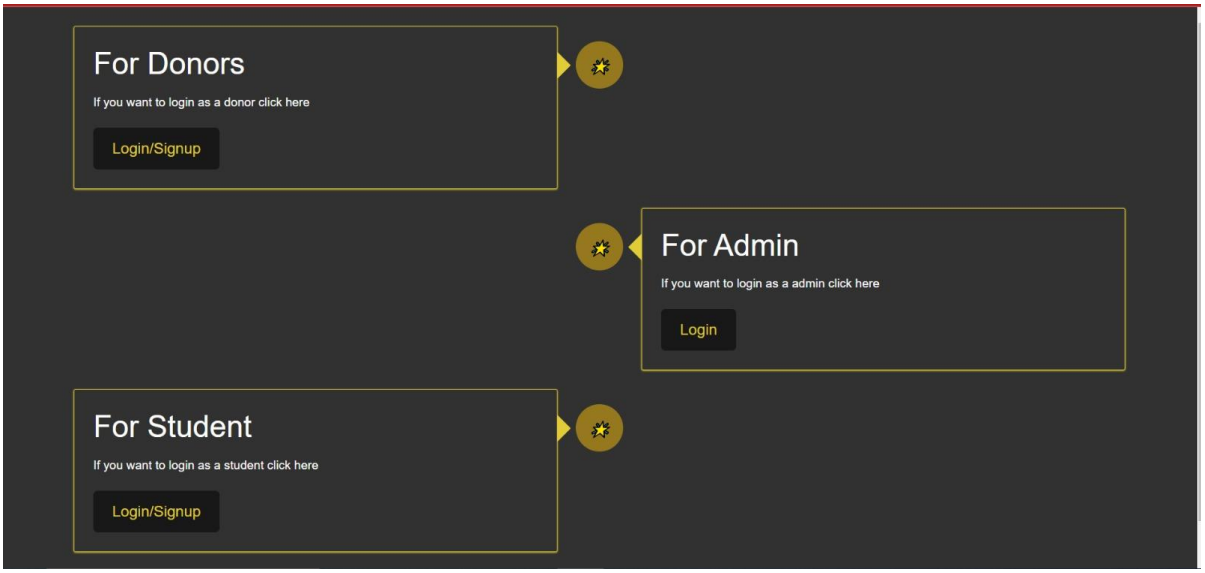


Figure 20:User Login & Registration

6.1.7.1 Donor Signup

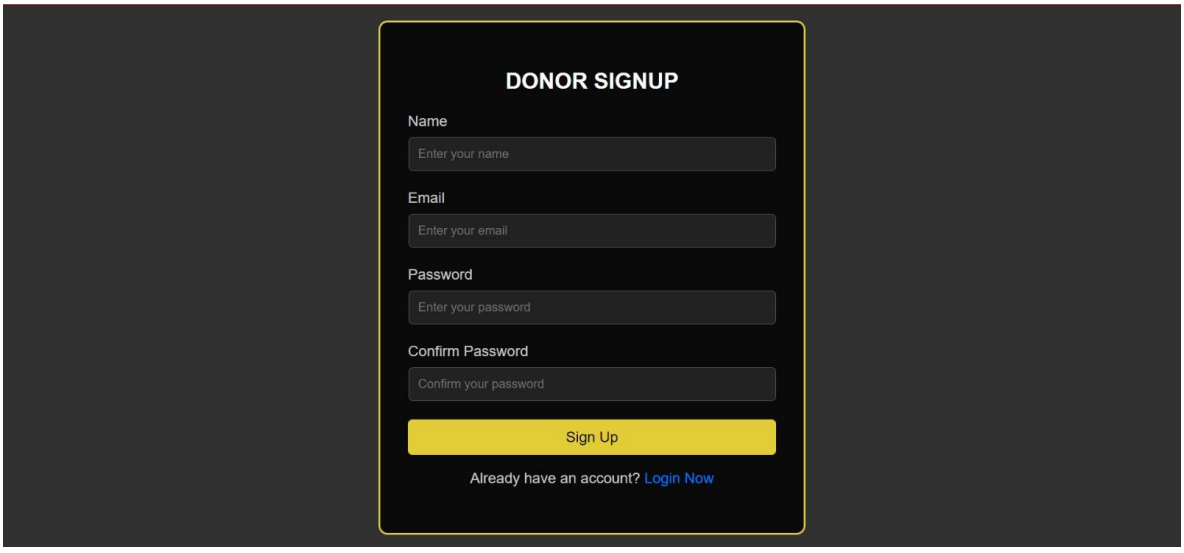


Figure 21:Donor Signup

6.1.7.2 Student Signup

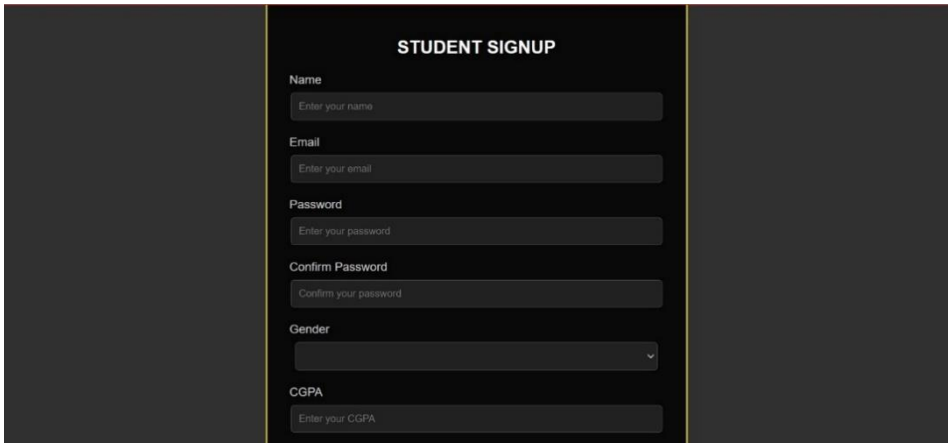


Figure 22:Student Signup

6.1.7.3 Admin Login

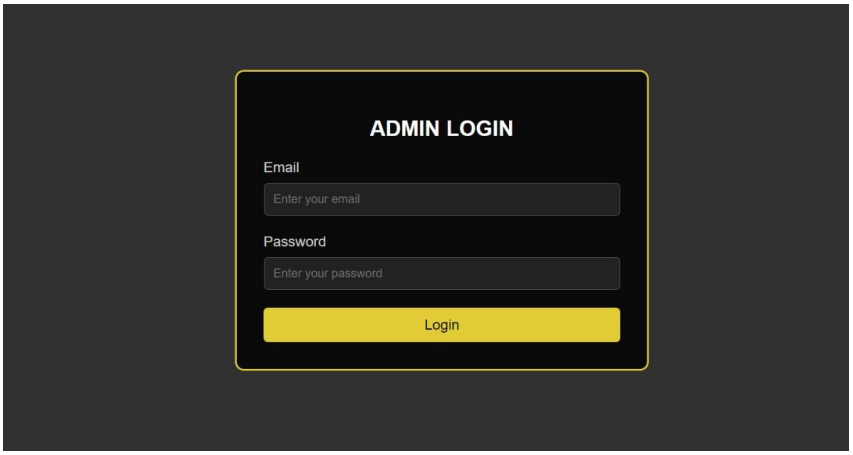


Figure 23:Admin Login

6.1.7.4 Student Login

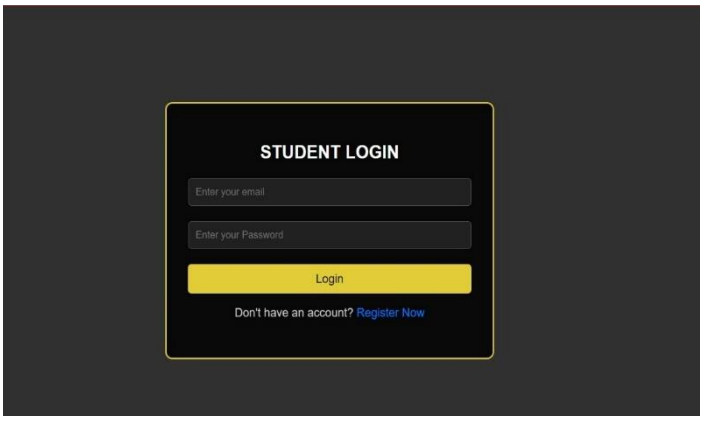


Figure 24:Student Login

6.1.8 Apply Donation Page

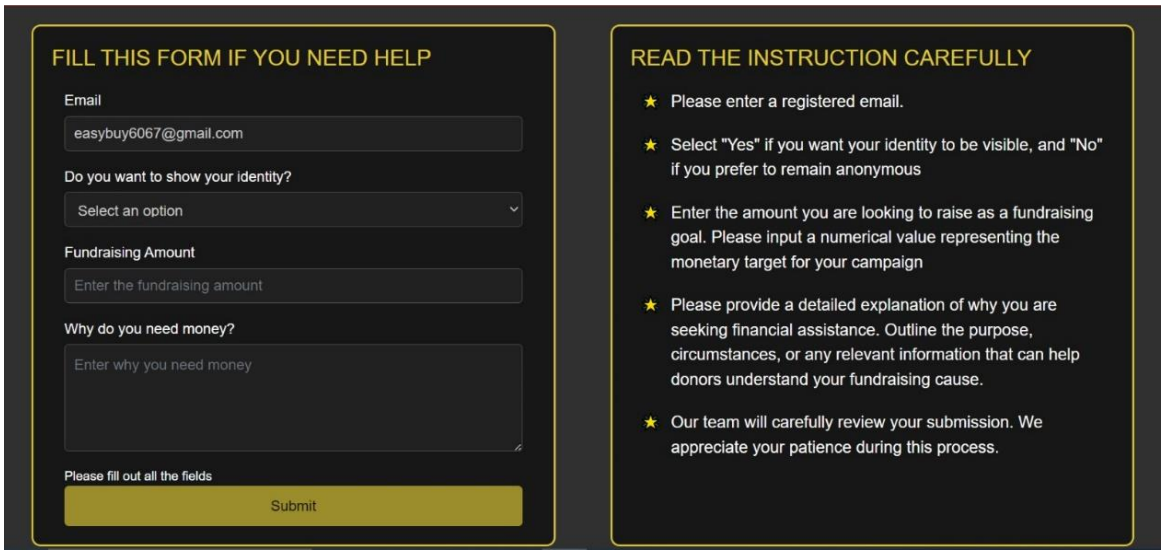


Figure 25:Apply for Need

6.1.9 Admin Dashboard

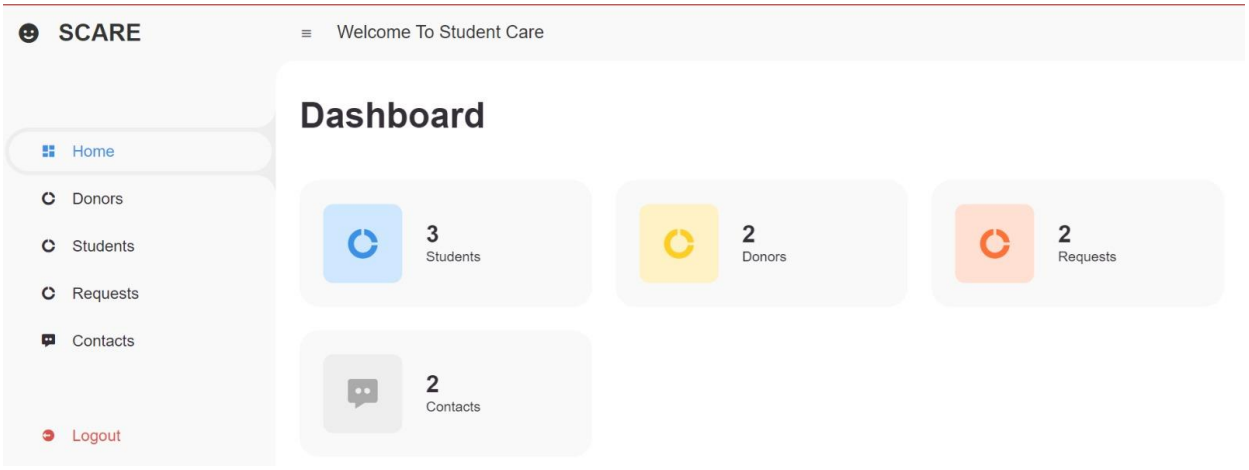


Figure 26:Admin Dashboard

6.1.9.1 Donor & Students Page

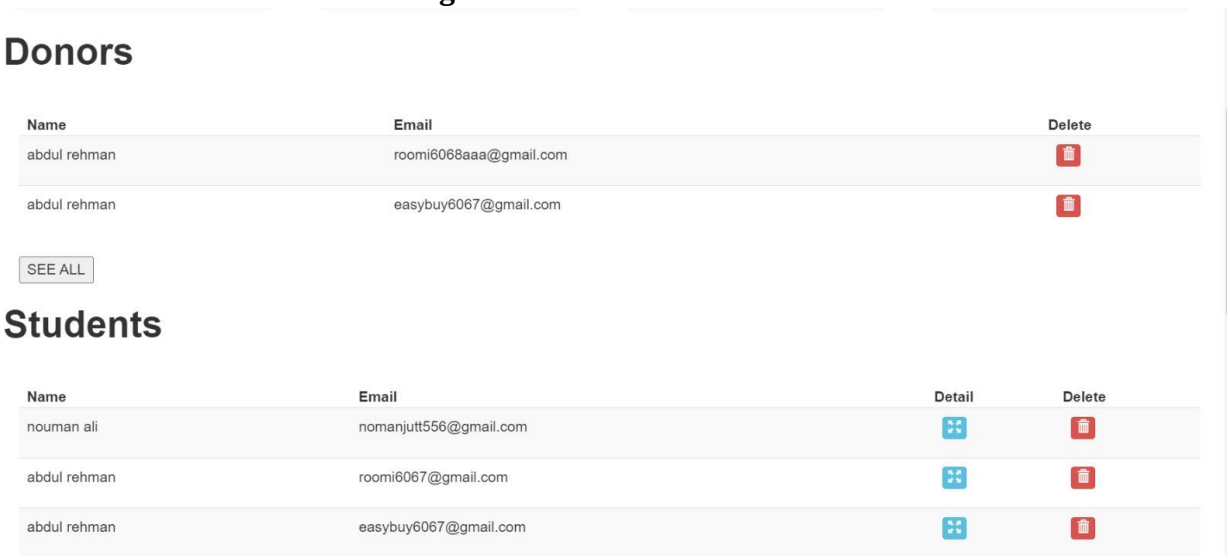


Figure 27: Donor & Student

6.1.9.2 Contact & Requests Page

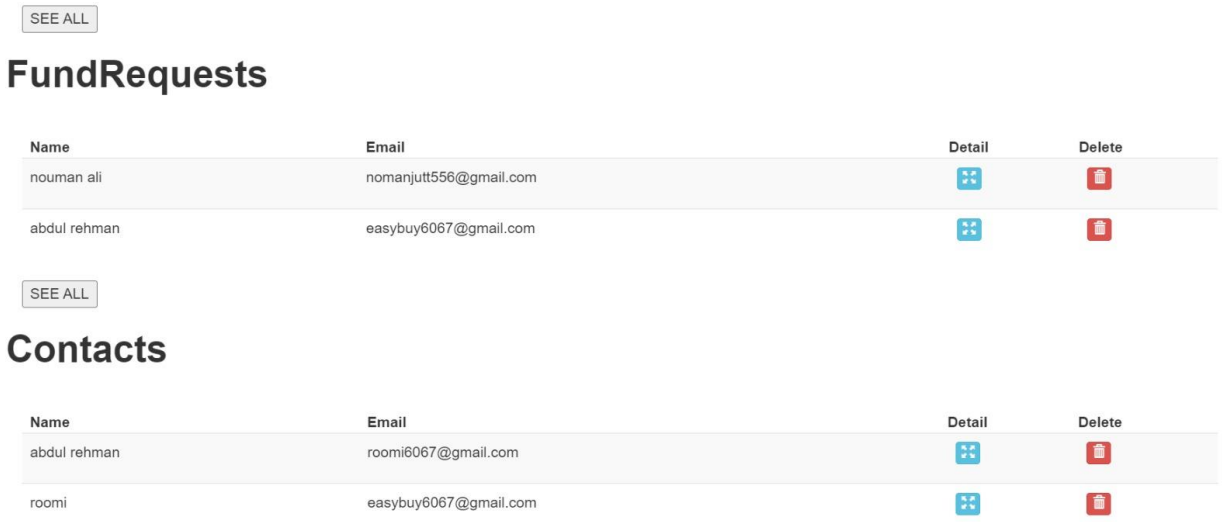


Figure 28:Fund & Request

6.1.9.3 Detail Request Page

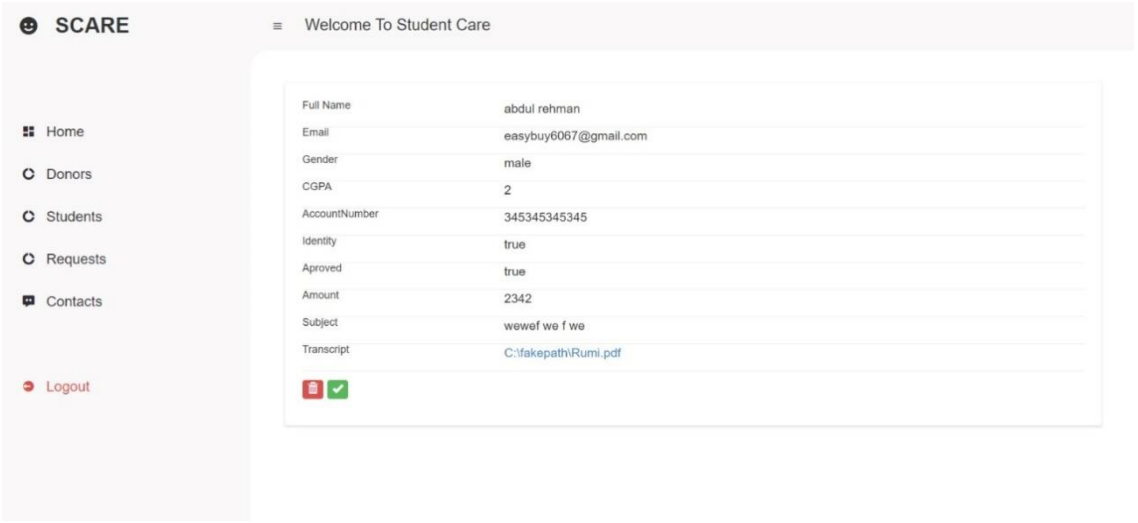


Figure 29:Detailed Request

6.1.10 User Profile Page

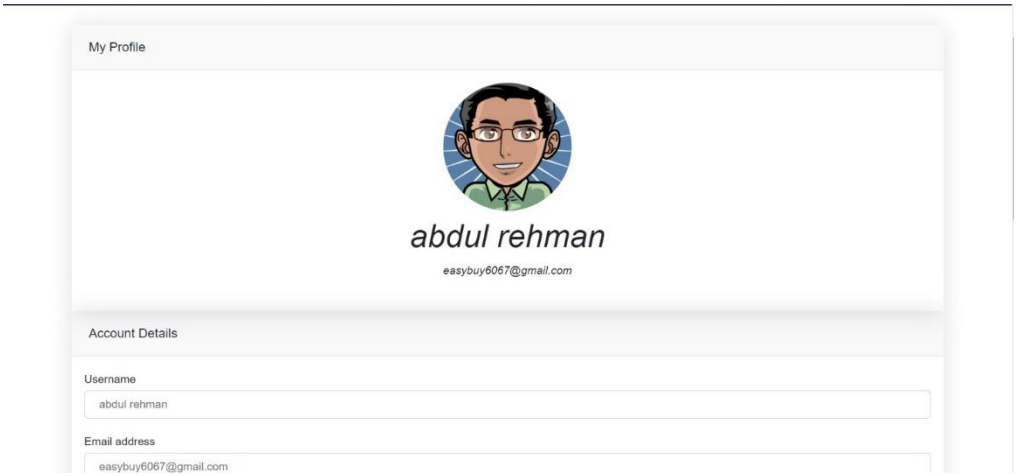


Figure 30:User Profile

6.1.11 Not Found Page