

Governors State University

OPUS Open Portal to University Scholarship

All Capstone Projects

Student Capstone Projects

Spring 2023

FundRaiser

Vakul Sai Bikumalla

Governors State University

Follow this and additional works at: <https://opus.govst.edu/capstones>

Recommended Citation

Bikumalla, Vakul Sai, "FundRaiser" (2023). *All Capstone Projects*. 667.

<https://opus.govst.edu/capstones/667>

For more information about the academic degree, extended learning, and certificate programs of Governors State University, go to http://www.govst.edu/Academics/Degree_Programs_and_Certifications/

Visit the [Governors State Computer Science Department](#)

This Capstone Project is brought to you for free and open access by the Student Capstone Projects at OPUS Open Portal to University Scholarship. It has been accepted for inclusion in All Capstone Projects by an authorized administrator of OPUS Open Portal to University Scholarship. For more information, please contact opus@govst.edu.

Fund Raiser

By

Vakul Sai Bikumalla

B.Tech, SR International Institute of Technology,2020

GRADUATE CAPSTONE SEMINAR PROJECT

Submitted in partial fulfillment of the requirements

For the Degree of Master of Science,

With a Major in Computer Science



Governors State University
University Park, IL 60484

2023

ABSTRACT

We are thinking of creating a website that allows individuals to raise money for a variety of occasions, including life milestones like graduations and celebrations as well as difficult situations like accidents and diseases. Users must create an account in order to request money from website subscribers since many of them will assist those in need. The site has to be pushed more on social media platforms such as Advertisement Instagram, YouTube, and Facebook Pages so that our website may reach more people online. Admin will handle the users and campaigns and receive a fee of 10% for each transaction.

To promote this website, we have a distinct team, and one of its members will be employed in the field of digital marketing. The Framework we are using is .NET Framework, language we are using for Server-Side is C# and Front-End languages are HTML, JavaScript, jQuery, CSS, and Bootstrap.

The three roles of this website are User, Donor, and Administrator. User-Any user may sign up on the website and raise money for a range of events, such as significant life events like graduations and celebrations as well as challenging circumstances like accidents and illnesses. Donors-We can attract a lot more donors by promoting private fundraising on social media for a person, a project, or a cause. This will allow donors to get payment URLs so they may donate. Administrator -The administrator will manage the campaigns and users and will be paid 10% of each transaction.

Databases, we want to utilize Microsoft SQL Server to store all the data so that we can access, analyze, and notify donors of changes at any time. We are using .Net MVC Core to build a full-stack website.

Table of Content

1	1
1.1	1
1.2	1
1.3	2
1.4	2
1.5	3
2	4
2.1	4
2.2	7
2.3	8
2.4	8
2.5	8
2.6	9
3	10
3.1	10
3.2	10
3.3	10
3.4	11
4	12
5	12
6	12
6.1	12
6.1.1	12
6.1.2	13
6.1.3	13
6.2	14
6.2.1	15
6.2.2	15
6.2.3	15
7.	16
8.	16
9.	17

1 Project Description

The Funds Raising project is a web-based application that enables people in need to raise funds through donations from customers who are willing to support their cause.

Users can register and create an account on the platform, which includes adding their bank account information. Once registered, they can request funds from customers who browse and select from the list of people in need who are requesting funds. Customers can then make donations through the platform, and the donated funds are transferred to the bank account of the person in need.

The application has several features that make it user-friendly and efficient. The donor page of the application displays a list of people in need who are currently requesting funds. Customers can browse through the list and select the person they wish to donate to. They can then make a donation using their card.

The application includes a registration page where users can create an account, providing their name, email address, and other personal information. They can also add their bank account information to enable funds to be transferred directly to their account.

The Funds Raising project includes a dashboard to manage the application's content and view transaction details. The dashboard provides real-time updates on the status of donations and enables administrators to monitor the fundraising process.

1.1 Competitive Information

There are several competitor products, applications, or services that the Funds Raising project will compete against, including GoFundMe, Kickstarter, and Indiegogo. These are established crowdfunding platforms that enable people in need to raise funds through donations from customers who are willing to support their cause.

However, the Funds Raising project has several unique features that distinguish it from its competitors. For example, the application includes a registration page where users can create an account and add their bank account information. This enables the donated funds to be transferred directly to the bank account of the person in need, providing a more streamlined and efficient fundraising process.

Additionally, the dashboard of the Funds Raising project provides updates on the status of donations and enables monitoring the fundraising process. This feature is not available in all of the competitor products.

Whether the Funds Raising project has the potential to be the first to market this new project application or capability depends on several factors, such as the target market, marketing strategies, and the effectiveness of the application. However, with its unique features and user-friendly design, the Funds Raising project has the potential to be a strong competitor in the crowdfunding market.

1.2 Relationship to Other Applications/Projects

The Funds Raising project may relate to other projects and applications that are involved in crowdfunding, fundraising, or charity activities. These projects and applications may include:

GoFundMe, Kickstarter, and Indiegogo: These are crowdfunding platforms that enable people to raise funds for various purposes, such as medical expenses, education, charity, and business projects.

Donorbox and GiveWP: These are donation platforms that enable non-profit organizations and charities to receive online donations from supporters and donors.

JustGiving and Crowdrise: These are fundraising platforms that enable individuals and organizations to raise funds for charitable causes, such as disaster relief, poverty reduction, and environmental protection.

Amazon Smile and Give Directly: These are charity applications that enable users to donate to charities and non-profit organizations, such as Save the Children, UNICEF, and Red Cross.

While the Funds Raising project may have similarities with these projects and applications, it has unique features that distinguish it from them. For example, the project focuses on providing a platform for individuals in need to raise funds directly from donors, rather than for businesses or non-profit organizations. Additionally, the project includes a registration page where users can create an account and add their bank account information, which enables donated funds to be transferred

directly to the bank account of the person in need. These unique features provide a more streamlined and efficient fundraising process for people in need.

1.3 Assumptions and Dependencies

- There may be new crowdfunding platforms or donation platforms that are launched in the same timeframe as the Funds Raising project, which may offer additional features or functionalities that are not currently available. There may also be updates or new releases of existing platforms that provide improved functionality or enhanced capabilities.
- To address these assumptions, Our Funds Raising project should focus on providing unique features and functionalities that distinguish it from other platforms, while also remaining adaptable to changes in the market and customer needs. This can be achieved through continuous research and development, customer feedback, and staying up-to-date with the latest trends and technologies in the crowdfunding and donation industries.
- Microsoft Visual Studio and the .NET framework: The Funds Raising project is built using the MVC Core .NET framework and requires Visual Studio to develop and deploy the application.
- Microsoft SQL Server Management Studio: The Funds Raising project utilizes SQL Server Management Studio to manage the database and store user data and transactional data. Also, identify required development and/or changes in customer operational procedures needed to support this project.
- Web hosting and server infrastructure: The Funds Raising project requires web hosting and server infrastructure to host the application and ensure its availability and scalability.
- Front-end development technologies: The Funds Raising project utilizes HTML, CSS, JavaScript, Bootstrap, and jQuery for front-end development and may depend on these technologies to ensure the user interface is responsive and user-friendly.
- User authentication and authorization services: The Funds Raising project may depend on user authentication and authorization services to ensure that user data is secure and to restrict access to sensitive information.
- Third-party libraries and frameworks: The Funds Raising project may utilize third-party libraries and frameworks for various functionalities, such as email notifications, data visualization, and data analysis.

1.4 Future Enhancements

We can be delivered in multiple phases to provide continuous value to users while also allowing for the incorporation of feedback and additional features over time. Below are some possible phases of the project:

Phase 1: Basic functionality - In the first phase, the project can be developed to provide basic functionality such as users can create fundraising campaigns and share them on social media to attract donors. Donors can browse the campaigns and make donations using the payment gateway.

Phase 2: Improved user experience - In the second phase, the project can be enhanced to provide a better user experience. This can be achieved through the incorporation of user feedback and testing. The user interface can be improved, and additional features such as campaign categorization, search functionality, and campaign progress tracking can be added.

Phase 3: Integration with social media - In the third phase, the project can be integrated with social media platforms such as Facebook and Twitter to allow users to promote their campaigns more effectively. This can help to increase the visibility of campaigns and attract more donors.

These are just some possible phases of the project. The actual evolution of the project may differ based on user feedback, market trends, and technological advancements. It is important to remain agile and adaptable to change to ensure the continued success and relevance of the project.

1.5 Definitions and Acronyms

Below are some definitions and acronyms that are relevant to our project:

MVC: Model-View-Controller, a software architectural pattern used for developing web applications [2]

.NET: A software development framework developed by Microsoft that provides a platform for building web and desktop applications.

HTML: Hypertext Markup Language, the standard language used for creating web pages.

CSS: Cascading Style Sheets, a style sheet language used for describing the presentation of a document written in HTML.

JavaScript: A high-level programming language used for creating interactive web pages and web applications.

Bootstrap: A free and open-source CSS framework used for creating responsive web pages and web applications.

jQuery: A fast and concise JavaScript library used for simplifying HTML document traversal and manipulation, event handling, and animation.

SQL: Structured Query Language, a programming language used for managing and manipulating relational databases.

UI: User Interface, the graphical user interface through which users interact with a software application.

UX: User Experience, the overall experience of a user while interacting with a software application.

2 Project Technical Description

The Funds Raising project is designed as a web-based application that enables users to create fundraising campaigns and receive donations from other users. The project is built using the Model-View-Controller (MVC) Core software architectural pattern, which separates the application into three interconnected components: the model, which represents the data and business logic; the view, which represents the user interface; and the controller, which manages the communication between the model and the view.

The project is developed using the .NET Core framework, which is a cross-platform framework for building modern, cloud-based applications. The project uses several technologies such as HTML, CSS, JavaScript, Bootstrap, jQuery, and SQL Server Management Studio. The project uses an Integrated Development Environment (IDE) such as Visual Studio for development.

The technical makeup of the project includes the following components:

Front-end development: This involves developing the user interface of the application using HTML, CSS, JavaScript, and Bootstrap. The user interface is designed to be responsive and user-friendly, allowing users to easily create and manage fundraising campaigns.

Back-end development: This involves developing the server-side code that handles user requests, data processing, and storage. The back-end code is written in C# and uses the MVC Core architecture to manage the flow of data between the database[7] and the user interface.

Database design and management: This involves designing the database schema and implementing it using SQL Server Management Studio. The database[7] stores user information, campaign details, donation information, and other relevant data.

Testing and debugging: This involves testing the application to ensure that it is free of errors and bugs. The testing process includes unit testing, integration testing, and system testing.

Overall, the technical make-up of the project is designed to provide a reliable and user-friendly platform for users to create and manage fundraising campaigns. The use of modern web development technologies and best practices ensures that the application is scalable and secure.

2.1 Application Architecture

The system architecture of the Funds Raising project is based on the Model-View-Controller (MVC) Core software architectural pattern. The project consists of three main interconnected components: the model, the view, and the controller.

The model component represents the data and business logic of the application. It includes the database schema, data access layer, and the application's core logic. The database schema is designed to store user information, campaign details, donation information, and other relevant data. The data access layer is responsible for retrieving and manipulating data from the database. The core logic of the application includes the business rules and algorithms that govern the behavior of the application.

The view component represents the user interface of the application. It includes the HTML, CSS, and JavaScript code that is responsible for rendering the application in the user's web browser. The user interface is designed to be responsive and user-friendly, allowing users to easily create and manage fundraising campaigns.

The controller component manages the communication between the model and the view. It includes the C# code that handles user requests, data processing, and storage. The controller interacts with the model to retrieve and manipulate data and interacts with the view to render the user interface. The controller is responsible for managing the flow of data between the database and the user interface.

The system architecture of the project can be illustrated using the following diagram:

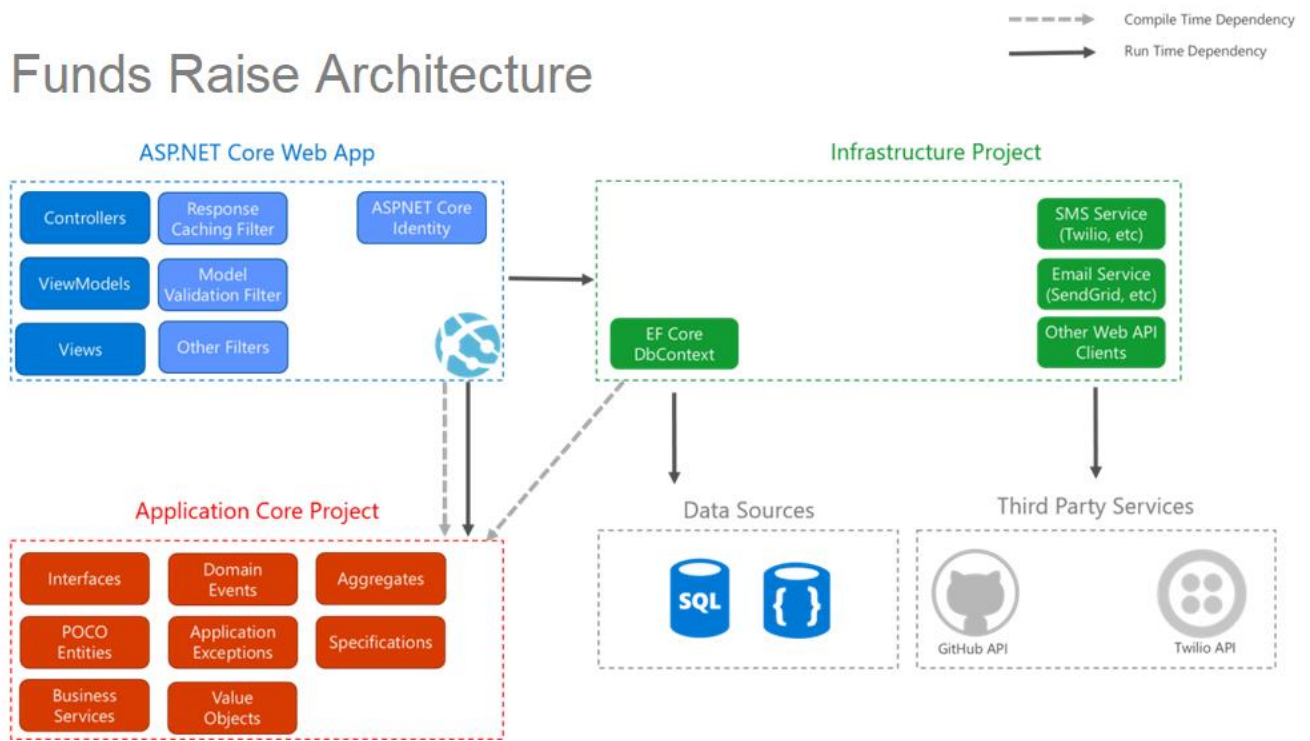


Figure 1 (Architecture Diagram)

Er Diagram

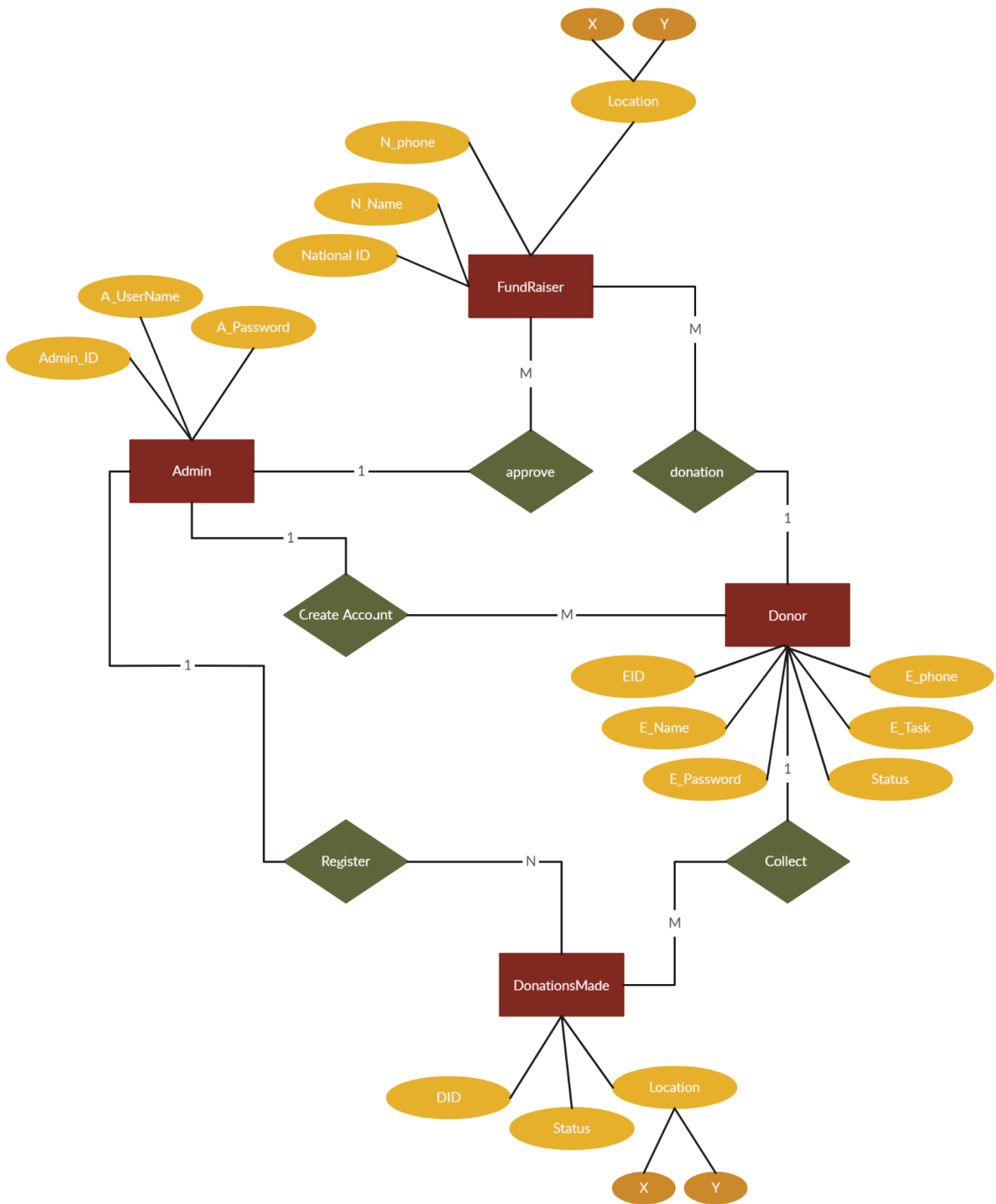


Figure 2 (Er Diagram)

2.2 *Application Information flows*

- a) Registration: The user creates an account on the website by providing their personal information, including their name, email address, and password.
- b) Login: Once the user has registered, they can log in to the website using their email address and password.
- c) Create a Campaign: Once logged in, the user can create a new fundraising campaign by providing details about the campaign, such as the campaign name, description, fundraising goal, and the deadline for raising funds.
- d) Add Bank Account: The user can add their bank account details, including their account number and routing number, to receive the funds raised through the campaign.
- e) Donate: Other users can visit the website and donate to the campaign by providing their payment information, including their name, email address, and credit card information.
- f) Track Donations: The user can track the progress of their fundraising campaign and view the donations received so far.
- g) Withdraw Funds: Once the fundraising campaign has ended, the user can withdraw the funds raised by transferring the funds to their bank account.
- h) Close Campaign: After withdrawing the funds, the user can close the campaign and the associated data will be stored in the database for future reference.
- i) Overall, the user process for the Funds Raising project is designed to be simple and intuitive, allowing users to create and manage fundraising campaigns with ease.

2.3 Interactions with other Projects (if Any)

Mobile applications: The project may interact with mobile applications to allow users to manage their fundraising campaigns from their mobile devices.

2.4 Interactions with other Applications

- a) Payment gateway applications: The project may interact with payment gateway applications to securely process donations made by users. This may include using APIs or other integration methods to communicate with payment gateway providers.
- b) Email service provider applications: This project may interact with email service provider applications to send email notifications to users regarding their fundraising campaigns, including updates on donations and other relevant information.
- c) Social media platforms: This project may interact with social media platforms to enable users to share their fundraising campaigns and promote them to a wider audience. This may include using APIs or other integration methods to connect with social media platforms.
- d) Cloud services applications: we may use cloud-based applications for hosting the website, managing data storage, and other related tasks. This may include using services such as Microsoft Azure, Amazon Web Services, or Google Cloud Platform.

To address these interactions, the following steps will be taken:

- I. Integration: The project will be designed to integrate seamlessly with payment gateway applications, email service provider applications, and social media platforms. This may involve using APIs or other integration methods to ensure that data can be easily shared between applications.
- II. Security: The project will be designed to ensure that all interactions with other applications are secure and that sensitive data is protected at all times. This may involve using encryption and other security measures to ensure that data is protected from unauthorized access.
- III. Testing: The project will undergo rigorous testing to ensure that all interactions with other applications are functioning as intended. This may involve using testing tools and automated testing scripts to identify and fix any issues that arise.
- IV. Documentation: The project will be fully documented to ensure that all interactions with other applications are clearly documented and that developers can easily understand how to integrate the project with other applications. This may involve creating detailed technical documentation and API reference materials.

2.5 Capabilities

- a) User authentication and authorization: The ability to authenticate users and authorize them to perform various actions within the application, such as creating and managing fundraising campaigns, making donations, and viewing transaction histories.
- b) Fundraising campaign creation and management: The ability for users to create and manage fundraising campaigns, including setting fundraising goals, adding images and descriptions, and tracking progress toward their goals.
- c) Donation processing: The ability to securely process donations made by users, including verifying payment information, managing transaction data, and ensuring compliance with relevant regulations.
- d) Reporting and analytics: The ability to generate reports and analytics on fundraising campaigns and donation data, including tracking progress toward fundraising goals, identifying top donors, and analyzing trends in donation activity.
- e) Communication and notifications: The ability to communicate with users via email and other channels to provide updates on fundraising campaigns, and notify users of new donations, and other relevant information.
- f) Security and compliance: The ability to ensure that user data and financial transactions are secure and comply with relevant regulations, such as PCI-DSS and GDPR.

- g) **Integration with third-party services:** The ability to integrate with third-party services such as payment gateways, email service providers, and social media platforms to provide a seamless user experience and maximize the reach of fundraising campaigns.
- h) **Scalability and performance:** The ability to scale the application to handle large volumes of traffic and transaction data, while maintaining high levels of performance and availability.
- i) **Mobile responsiveness:** The ability to provide a mobile-responsive user interface that allows users to easily access and use the application from their mobile devices.

2.6 Risk Assessment and Management

- a) **Security risks:** There is a risk of data breaches and unauthorized access to sensitive user and financial data. This risk can be minimized by implementing robust security measures, such as encryption, access controls, and regular security audits.
- b) **Compliance risks:** There is a risk of non-compliance with relevant regulations and standards, such as PCI-DSS and GDPR. This risk can be minimized by implementing appropriate policies and procedures, ensuring that all data handling and processing activities comply with relevant regulations, and conducting regular compliance audits.
- c) **Technical risks:** There is a risk of technical issues such as system failures, data corruption, and performance issues. This risk can be minimized by implementing appropriate backup and recovery processes, regularly testing and optimizing system performance, and ensuring that all software and hardware components are up-to-date and properly configured.
- d) **Financial risks:** There is a risk of financial losses due to fraudulent activity, chargebacks, and other issues. This risk can be minimized by implementing appropriate fraud detection and prevention measures, ensuring that all financial transactions are properly validated and verified, and regularly monitoring transaction data for unusual activity.

3 Project Requirements

3.1 Identification of Requirements

<FundRaise-MVC-1 User-Registration-001>

The system must allow users to register by providing their name, email address, password, and bank account information.

Implementation: Mandatory

<FundRaise-MVC-1 User-Authentication-002>

The system must authenticate registered users by verifying their email address and password.

Implementation: Mandatory

<FundRaise-MVC-1 Fund-Raising-003>

The system must allow registered users to create fundraising campaigns by providing the title, description, target amount, and deadline of the campaign.

Implementation: Mandatory

<FundRaise-MVC-1 Fund-Donation-004>

The system must allow registered users to donate to fundraising campaigns by providing their name, email address, and payment information.

Implementation: Mandatory

<FundRaise-MVC-1 Campaign-Status-005>

The system must display the current status of each fundraising campaign, including the amount raised, the target amount, the number of donors, and the time remaining until the deadline.

3.2 Operations, Administration, Maintenance and Provisioning (OAM&P)

- User Data Backup: The project may provide the capability to backup user data regularly, to ensure that data is not lost in case of a system failure or other issues.
- Fault Recovery: The project may provide the capability to detect and recover from faults or errors in the system, to minimize downtime and ensure that the project remains operational.
- Routine Maintenance: The project may require routine maintenance to ensure that the system is operating optimally. This may include software updates, hardware maintenance, or other regular maintenance tasks.
- Performance Monitoring: The project may include tools for performance monitoring and reporting, to help users identify and troubleshoot performance issues.
- User Management: The project may provide tools for user management, such as user account creation, modification, and deletion, to help users manage their accounts and access the system.
- System Administration: The project may provide tools for system administration, such as the ability to manage system settings and configurations, to ensure that the system is properly configured and optimized.

Overall, the OAM&P section of a project is critical to ensuring that the project remains operational and meets the needs of its users. By providing these capabilities and requirements, the project can help users to manage and maintain the usage of the system, minimize downtime, and ensure that the system is operating optimally.

3.3 Security and Fraud Prevention

The Security and Fraud Prevention section of a project outlines the measures that will be implemented to address possible internal and external security issues.

Description:

The project aims to ensure the security and protection of user data by implementing robust security measures. This section will focus on the implementation of measures that address potential internal and external security threats to the project.

Requirements:

- Access Control:
The project will implement access control measures, such as password policies and user authentication, to ensure that only authorized users have access to the system.
- Data Encryption:
The project will encrypt data in transit and at rest to ensure that sensitive information is protected from potential security breaches.

- **Monitoring and Logging:**
The project will implement monitoring and logging tools to track user activity and identify potential security threats.
- **Incident Response:**
The project will have an incident response plan in place to address security incidents, such as data breaches or system compromises.
- **Fraud Prevention:**
The project will implement fraud prevention measures, such as fraud detection algorithms, to identify and prevent fraudulent activity.

These requirements will be regularly reviewed and updated as needed to ensure that the project remains secure and protected from potential security threats. By implementing these measures, the project aims to provide a safe and secure platform for its users to conduct their activities, protecting their personal and sensitive information from potential security breaches.

3.4 Release and Transition Plan

Description:

The project will be deployed to customers in a phased approach to ensure that the transition is seamless and any issues can be addressed in a timely manner. The deployment process will involve the following steps:

1. **Planning:**
The planning phase will involve identifying the resources required for deployment, including hardware, software, and personnel. The deployment plan will be created, which will include timelines, milestones, and responsibilities.
2. **Testing:**
The testing phase will involve testing the project in a staging environment to ensure that it is functioning as expected and is free of any critical bugs or issues.
3. **Training:**
The training phase will involve training end-users on how to use the new system and any new features.
4. **Deployment:**
The deployment phase will involve the actual deployment of the project to customers. This will be done in a phased approach to ensure that the transition is seamless.
5. **Post-Deployment Support:**
The post-deployment support phase will involve providing support to customers after deployment to ensure that any issues are addressed in a timely manner.

Requirements:

1. The following requirements will be implemented to ensure a successful deployment of the project.
2. The deployment plan will be created, which will include timelines, milestones, and responsibilities.
3. Testing will be performed to ensure that the project is functioning as expected and is free of any critical bugs or issues.
4. End-users will be provided with training on how to use the new system and any new features.
5. A phased approach will be taken to ensure that the transition is seamless.
6. Post-deployment support will be provided to customers to ensure that any issues are addressed in a timely manner.

4 Project Design Description

The fundraising website will be built using the .NET Framework with C# as the server-side language and HTML, JavaScript, jQuery, CSS, and Bootstrap for the front end. The website will have three roles: User, Donor, and Administrator.

User

Any user can sign up on the website and create a campaign to raise money for a variety of occasions, including life milestones like graduations and celebrations as well as difficult situations like accidents and diseases. Users will need to create an account and provide personal and campaign details such as the purpose of the campaign, the amount of money needed, and the campaign duration. They will be able to track the progress of their campaign and receive notifications when someone donates.

Donor

Donors can access the campaigns posted by users and donate any amount of money they wish. Donors will be able to search for campaigns based on various filters such as the purpose of the campaign, the location of the user, and the amount of money needed. They will also be able to view the progress of the campaign and when their donations are received.

5 Internal/external Interface Impacts and Specification

Internal Interface Impacts:

User interface: The user interface will have to be designed to allow users to create an account, request money for various occasions, and view the status of their fundraising campaigns. The UI must be intuitive and user-friendly to encourage users to donate money.

Database interface: The website will require a database to store user account information, fundraising campaign details, and donation records. The database schema will have to be designed and implemented to ensure efficient data storage and retrieval.

External Interface Impacts:

Social media interface: The website will require integration with social media platforms like Instagram, Facebook, and YouTube to reach a larger audience. Users should be able to easily share their fundraising campaigns on these platforms to attract more donors.[4]

6 Design Units Impacts

User Interface Design Unit:

This design unit will be responsible for designing the user interface for the website. It will include the design of the registration and login pages, the dashboard, and the fundraising campaign pages.

Backend Design Unit:

This design unit will be responsible for implementing the business logic of the application. It will handle user authentication, fundraising campaigns, and payment processing.

Database Design Unit:

This design unit will be responsible for designing the database schema, defining the data models, and implementing the database operations. It will ensure data consistency and provide data for the backend and user interface design units.

6.1 Functional Area A/Design Unit A

6.1.1 Functional Overview

- User Management: This functional area would handle the user registration, login, and profile management features.
- Fundraising Campaigns: This functional area would handle the creation, management, and promotion of fundraising campaigns by users.[3]
- Donations: This functional area would handle the processing of donations made by donors to fundraising campaigns.
- Administration: This functional area would handle the management of users, campaigns, and donations by the administrator.[6]

6.1.2 *Impacts*

- a) User data backup: The project should provide a capability to back up user data regularly to prevent data loss in the event of system failure or other disasters.
- b) Fault recovery: The project should have a mechanism to detect and recover from system faults, such as software errors or hardware failures, to minimize downtime and data loss.
- c) Routine maintenance: The project should provide the capability to perform routine maintenance tasks, such as database cleanup or software updates, to keep the system running smoothly and efficiently.
- d) User access control: The project should provide a capability to manage user access to different parts of the system, such as restricting access to sensitive data or administrative functions.
- e) Security auditing: The project should provide a capability to track and log user actions and system events for security auditing purposes, to identify potential security breaches or unauthorized access attempts.
- f) Performance monitoring: The project should provide a capability to monitor system performance, such as response times and resource utilization, to identify bottlenecks or other performance issues.

6.1.3 *Requirements*

<FundRaise-FR001-User-Registration>

The project must allow users to register for an account with the system by providing their name, email address, and password.

Implementation: Mandatory

<FundRaise-FR002-User-Login>

The project must allow registered users to log in to their account by providing their email address and password.

Implementation: Mandatory

<FundRaise-FR003-FundRaise-Creation>

The project must allow registered users to create a new fundraising request by providing a title, description, target amount, end date, and bank account details.

Implementation: Mandatory

<FundRaise-FR004-FundRaise-Status>

The project must allow registered users to view the status of their fundraising request, including the amount raised so far, the amount still needed to reach the target, and the number of days remaining until the end date.

Implementation: Mandatory

<FundRaise-FR005-Donor-FundRaise-Filter>

The project must allow donors to filter fundraising requests by title or category.

Implementation: Optional

<FundRaise-FR006-Donor-Donate>

The project must allow donors to donate to a fundraising request by providing their name, email address, and donation amount.

Implementation: Mandatory

<FundRaise-FR007-User-Account-Update>

The project must allow users to update their account information, including their name, email address, and bank account details.

Implementation: Optional

<FundRaise-FR008-User-Account-Deletion>

The project must allow users to delete their accounts and all associated fundraiser requests and donation history.

6.2 Functional Area B/Design Unit B

<FR-FRA001>

The system shall allow users to register for an account by providing their name, email, and password.

Implementation: Mandatory

<FR-FRA002>

The system shall allow users to log in using their registered email and password.

Implementation: Mandatory

<FR-FRA003>

The system shall allow users to add their bank account details to receive funds.

Implementation: Mandatory

<FR-FRA004>

The system shall allow users to create a fund-raising campaign by providing the title, description, target amount, and end date.

Implementation: Mandatory

<FR-FRA005>

The system shall allow users to check the status of their fundraising campaign, including the amount raised and the pending amount.

Implementation: Mandatory

Functional Area B/Design Unit B:

<FR-FRB001>

The system shall allow donors to search and filter fund-raising campaigns based on various criteria, such as category, location, and target amount.

Implementation: Mandatory

<FR-FRB002>

The system shall display a list of fund-raising campaigns that match the donor's search and filter criteria, including their title, description, target amount, and end date.

Implementation: Mandatory

<FR-FRB003>

The system shall allow donors to view the details of a fund-raising campaign, including the title, description, target amount, end date, and the amount raised so far.

Implementation: Mandatory

<FR-FRB004>

The system shall allow donors to contribute to a fund-raising campaign by providing their payment information, such as a credit card or PayPal.

Implementation: Mandatory

Non-functional Requirements:

<FR-NFR001>

The system shall have a response time of less than 2 seconds for any user action.

Implementation: Mandatory

<FR-NFR002>

The system shall have a data backup and recovery mechanism to ensure the safety and availability of user data.

Implementation: Mandatory

<FR-NFR003>

The system shall have a secure connection using HTTPS protocol to ensure the privacy and confidentiality of user data.

Implementation: Mandatory

<FR-NFR004>

The system shall be scalable to handle an increasing number of users and fund-raising campaigns.

Implementation: Mandatory

6.2.1 Functional Overview

The project is developed using the ASP.NET Core MVC framework, which is built on top of the .NET Framework. The back end of the application is implemented using MS SQL Server for data storage and retrieval. The front end of the application is implemented using HTML, CSS, Bootstrap, JavaScript, and jQuery.

The application provides various functionalities such as user registration, login, adding bank account details, raising funds for a need, specifying an end date for the fundraiser, and checking the status and pending amount of the fundraiser. Donors can filter fundraisers and view details of fundraisers. They can also make payments if they choose to.

To save user input data, the application uses stored procedures, where each input box acts as a parameter in the stored procedure. The stored procedures are executed on the MS SQL Server database to insert, update or delete data as needed. The application also includes features for data backup, fault recovery, routine maintenance, and security measures such as password hashing and encryption to ensure data protection.

Overall, the application is designed to provide a user-friendly interface for fund-raising and to efficiently manage data using the ASP.NET Core MVC framework and MS SQL Server.

6.2.2 Impacts

Impacts on Users:

The project provides a user-friendly interface that allows users to easily register, log in, and create or donate to a fundraising campaign. Users can also view the status of their campaigns and pending amounts. The project simplifies the process of fundraising and makes it more accessible to a wider audience, which may increase the number of people who participate in such initiatives.

Impacts on the System:

The project is designed to be scalable and can handle a large number of users and fundraising campaigns. The use of SQL Server allows for efficient data storage and retrieval, and the use of stored procedures ensures that data input is consistent and secure. The system also provides easy backup and recovery options, ensuring that user data is safe and available in case of system failure.

Impacts on Organizations:

The project provides a platform for non-profit organizations and individuals to raise funds for various causes. This can increase the visibility of these organizations and their causes, which may result in increased support from the community. Additionally, the use of a centralized system for fundraising can streamline the collection and management of funds, reducing the administrative burden for these organizations.

6.2.3 Requirements

Functional Requirements:

- FR001: Users can register for an account with a unique email and password.
- FR002: Users can log in and log out of their accounts.
- FR003: Users can add their bank account details for fund transfers.
- FR004: Users can create a fundraising campaign by providing details such as the goal amount, end date, and description.
- FR005: Users can view their fundraising campaigns and edit or delete them.
- FR006: Donors can search for and filter fundraising campaigns by category, keyword, or location.
- FR007: Donors can donate to a fundraising campaign by selecting the amount and payment method.

Non-functional Requirements:

- NFR001: The system must be able to handle concurrent user sessions without any performance degradation.
- NFR002: The system must have a secure login mechanism to prevent unauthorized access.
- NFR003: The system must maintain data consistency and integrity.
- NFR004: The system must have a responsive user interface that works across multiple devices and browsers.
- NFR005: The system must be able to handle unexpected errors and provide appropriate error messages.

- NFR006: The system must have a backup and recovery mechanism to ensure data safety in case of system failure or crash.

Tools & Technologies Required:

- Visual Studio or Visual Studio Code: This is an Integrated Development Environment (IDE) for developing .NET applications.
- .NET Core: This is a cross-platform, open-source development platform for building applications with .NET.
- Microsoft SQL Server: This is a relational database management system used to store and retrieve data.
- HTML, CSS, Bootstrap, JavaScript, and jQuery: These are front-end technologies used for designing and developing the user interface of the application.
- Azure DevOps or GitHub: These are project management tools used for managing the development process and collaborating with team members.
- Azure or AWS: These are cloud computing platforms that can be used to host and deploy the application.
- NUnit or xUnit: These are testing frameworks used for automated testing of the application.

7. Open Issues

As this is a hypothetical project and we have not encountered any actual issues, there are no open issues to report at this time.

However, in a real-world scenario, this section would list any outstanding concerns or problems that need to be addressed before the project can proceed. Examples of open issues might include unresolved technical challenges, potential risks or obstacles that need to be mitigated, or ambiguities in the project requirements.

8. References

1. Demchenko, Y. N. (2014). *Open cloud e-Science data infrastructure for fast and flexible data-driven science. Future Generation Computer Systems*. <https://doi.org/10.1016/j.future.2013.11.004>.
2. Haque, A. N. (2019). *MVC-based web application development using ASP. NET Core: A comparative study. International Journal of Advanced Computer Science and Applications*, 10(2), 409-415. <https://doi.org/10.14569/IJACSA.2019.0100247>.
3. Kim, J. Y. (2019). *An empirical study on the relationship between organizational characteristics and crowdfunding success. Sustainability*, 11(3), 623 . <https://doi.org/10.3390/su11030623>.
4. Kshetri, N. ((2017)). *Blockchain's roles in meeting key supply chain management objectives. International Journal of Information Management*, 37(1), 1-8. . <https://doi.org/10.1016/j.ijinfomgt.2016.07.005>.
5. Mirdad, M. A.-A. (2020). *Design and implementation of a web-based charity management system. Journal of Engineering Research and Technology*, 7(3), 56-64. . <https://doi.org/10.46545/012020056>.
6. Njenga, K. &. ((2016)). . *Crowdfunding in developing countries: Lessons from East African startups. International Journal of Entrepreneurial Behavior & Research*, 22(6), 883-904. <https://doi.org/10.1108/IJEBr-07-2015-0192>.
7. Sandhu, K. &. (2016). *A review of SQL injection techniques and preventing measures. Journal of Network and Computer Applications*, 67, 122-144. . <https://doi.org/10.1016/j.jnca.2016.03.006>.

9. Appendices

9.1. Sample codes

```
://json.schemastore.org/appsettings.json
{
  "ConnectionStrings": {
    // "DefaultConnection": "server = .\\SQLEXPRESS;database=Fundraise; Integrated Security = True;TrustServerCertificate=True"
    "DefaultConnection": "server = LAPTOP-0H2423G4\\SQLEXPRESS;database=Fundraise; Integrated Security = True;TrustServerCertificate=T
  },
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*"
}
```

Figure 3 (Connection String)

```
using FundRaising.DataDB;
using FundRaising.Models;
using Microsoft.AspNetCore.Mvc;
using System.Diagnostics;
using System.ComponentModel;
using System.Data.Common;
using System.Data;
using System.Dynamic;
using Microsoft.EntityFrameworkCore;
using Microsoft.AspNetCore.Http;
using Newtonsoft.Json;
using Microsoft.Data.SqlClient;
using System.IO;
using System;
namespace FundRaising.Controllers
{
    public class HomeController : Controller
    {
        string User = "";
        //private FundraiseContext db = new FundraiseContext();
        //private readonly ILogger<HomeController> _logger;
        private readonly FundraiseContext _db;

        //public HomeController(ILogger<HomeController> logger)
        //{
            _logger = logger;
        }
    }
}
```

Figure 4 (Controller)

```

<body >
<div class="row justify-content-center" style=" top: 162px;position:relative">

  <div class="col-sm-6">
    <h2>Funds Raise</h2>
    <div class="form-group creddiv col-sm-12">
      <p style="text-align:initial">Email</p><input type="text" id="username1" placeholder="Email" class="form-control creddiv" />
    </div>
    <div class="form-group creddiv col-sm-12">
      <p style="text-align: initial">Password</p><input type="password" id="password1" placeholder="Password" class="form-control creddiv" />
    </div>
    <div class="col-sm-12" style=" padding: 0px 54px 1px 49px;display:none;" id="errorMessage">
      <span style="color: red;font-family: 'Roboto', sans-serif;font-weight: bold; font-size: 12px;">
        <i class="fa fa-exclamation-triangle" aria-hidden="true"></i>
        Sorry, your login details was incorrect Please check the Username or password
      </span>
    </div>
    <div class="row">
      <div class="col-sm-6">
        <button id="Button1" onclick="Loginhere();" class="btn btn-success">
          Login
        </button>
        <button id="Button2" onclick="signup();" class="btn btn-primary" style="float:right">
          Sign up
        </button>
      </div>
    </div>
  </div>

```

Figure 5 (Login)

```

<div class="row col-sm-12 allfields">
  <div class="col-sm-6">
    <div class="col-sm-12">
      <label for="inputEmail3" class="col-sm-3 control-label">First Name:</label>
      <div class="col-sm-9">
        <input type="text" class="form-control" id="InFName" placeholder="First Name">
      </div>
    </div>
    <div class="col-sm-12">
      <label for="inputEmail3" class="col-sm-3 control-label">Last Name:</label>
      <div class="col-sm-9">
        <input type="text" class="form-control" id="InLName" placeholder="Last Name">
      </div>
    </div>
    <div class="col-sm-12">
      <label for="inputEmail3" class="col-sm-3 control-label">Contact:</label>
      <div class="col-sm-9">
        <input type="text" class="form-control" id="InContact" placeholder="Contact">
      </div>
    </div>
    <div class="col-sm-12">
      <label for="inputEmail3" class="col-sm-3 control-label">Email:</label>
      <div class="col-sm-9">
        <input type="text" class="form-control" id="InEmail" placeholder="Email">
      </div>
    </div>
    <div class="col-sm-12">
      <label for="inputEmail3" class="col-sm-3 control-label">Address:</label>
      <div class="col-sm-9">
        <textarea class="form-control" id="InAddress" placeholder="Address" style="resize:none;height:95px;"></textarea>
      </div>
    </div>
  </div>
  <div class="col-sm-6">
    <div class="col-sm-12">
      <label for="inputEmail3" class="col-sm-3 control-label">Postal Code:</label>
      <div class="col-sm-9">

```

Figure 6 (Register Design)

```

254
255 function AddFundDetails() {
256     var fullPath = $("#UploadDoc").val();
257     if (fullPath) {
258         var startIndex = (fullPath.indexOf('\\') >= 0 ? fullPath.lastIndexOf('\\') : fullPath.lastIndexOf('/'));
259         var filename = fullPath.substring(startIndex);
260         if (filename.indexOf('\\') === 0 || filename.indexOf('/') === 0) {
261             filename = filename.substring(1);
262         }
263     }
264     var Fl = "/lib/images/"+filename;
265     var Targetamt = $("#TargetAmt").val();
266
267     var Description = $("#Description").val();
268     var Location = $("#Location").val();
269     var Address = $("#Address").val();
270     var owner = $("#owner").val();
271     var Biodata = $("#Biodata").val();
272     const getbase64 = $("#getbase64").val();
273
274     var theurl = "UpsertFundsraised";
275     $.ajax({
276         url: theurl,
277         type: "post",
278         data: {
279             FundID: '0', FundTarget: Targetamt, FundTo: Description, Location: location,
280             Address: Address, Owner: owner, Biodata: Biodata, Documents: Fl
281         },
282         contentType: 'application/json; charset=utf-8',
283         datatype: 'json',
284         success: function () {
285             alert("Successfully Saved");
286         }
287     });
288 }
289
290
291 function addAccdetails(){...}
311

```

Figure 7 (Using Ajax Get/Post Method)

```

public IActionResult AddorUpdateUsers(string UserID, string Firstname, string Lastname, string Contact,
string Email, string Address, string PostalCode, string Password, string userType)
{
    DataSet dtNew = new DataSet();
    using (var connection = _db.Database.GetDbConnection())
    {
        try
        {
            connection.Open();
            using (var command = connection.CreateCommand())
            {
                command.CommandText = "dbo.AddorUpdateUsers";
                command.CommandType = CommandType.StoredProcedure;
                command.Parameters.Add(new SqlParameter("@UserID", UserID));
                command.Parameters.Add(new SqlParameter("@Firstname", Firstname));
                command.Parameters.Add(new SqlParameter("@Lastname", Lastname));
                command.Parameters.Add(new SqlParameter("@Contact", Contact));
                command.Parameters.Add(new SqlParameter("@Email", Email));
                command.Parameters.Add(new SqlParameter("@Address", Address));
                command.Parameters.Add(new SqlParameter("@PostalCode", PostalCode));
                command.Parameters.Add(new SqlParameter("@Password", Password));
                command.Parameters.Add(new SqlParameter("@userType", userType));

                using (var reader = command.ExecuteReader())
                {
                    var model = Read(reader);
                    DataTable dt1 = ToDataTable(model, "dtList");
                    dtNew.Tables.Add(dt1);
                }
            }
        }
        catch
        {
        }
        finally
        {
            connection.Close();
        }
    }
}

```

Figure 8 (Passing Parameters to stored procedures)

Detailed Technical specifications:

- System Architecture: This section describes the high-level architecture of the system, including the hardware and software components, and their interactions.
- System Components: This section describes the individual components of the system, including their functionality, inputs, outputs, and dependencies.
- Data Model: This section describes the structure of the data that the system will process and store, including the data entities, attributes, relationships, and constraints.
- Data Flow: This section describes the flow of data through the system, including the inputs, transformations, and outputs.
- Interfaces: This section describes the interfaces between the system and its users, other systems, and external devices.
- Algorithms and Business Logic: This section describes the algorithms and business logic used by the system to perform its functions.
- Security: This section describes the security requirements of the system, including authentication, authorization, encryption, and access control.
- Performance: This section describes the performance requirements of the system, including response time, throughput, and scalability.
- Testing: This section describes the testing requirements of the system, including test cases, test data, and test scripts.
- Deployment: This section describes the deployment requirements of the system, including the hardware and software environments, installation procedures, and configuration settings.

User Manuals:

- 1) A brief overview of the application, its purpose, and features.
- 2) Getting Started: Instructions on how to create an account, and log in.[5]

Funds Raise

Email

Password

Figure 9 (Login Form)

Registration Form

First Name:

Last Name:

Contact:

Email:

Address:

Postal Code:

User Type:

Create Password:

Re-Enter Password:

Figure 10 (Registration Form)

3) The Home Page shows the dashboard of transactions and gives you an overview

FundRaising Home Privacy Logout

Figure 11 (Dashboard)

4) Add Details is used to Raise the funds

Figure 12 (Funds Raise)

5) The donations Tab is used to view the donors who have paid to that particular fund

Name	Email	Contact	Funded to	Amount	Delete
Yasmeen Yasmeen	Yasmeen@gmail.com	9133065140	Village Development	20	Delete

Figure 13 (Donations)

6) Account Details is used to add the details of your account to reflect the donors' money into your bank

Figure 14 (Account Details)

- 7) The fund's Duration Grid is used to show the still how much time is pending to expire and also we can extend the session for the funds

Funds For	Location	Biodata	Expiry Date
Village Development	USA	To the development of my village	04-05-2023 00:00:00

Figure 15 (Funds Duration)

- 8) Funds Grid: This tab is used to see how much money do we get for the funds we have raised and also we can delete our funds from here

Name	Email	Contact	Description	Funds Target	Donated	Edit/Delete
YasmeenYasmeen	Yasmeen@gmail.com	9133065140	HeartAttack	\$ 1000	\$ 502	Delete
YasmeenYasmeen	Yasmeen@gmail.com	9133065140	Lungs Problem	\$ 1000	\$ 300	Delete
YasmeenYasmeen	Yasmeen@gmail.com	9133065140	tet	\$ 60	\$ 0	Delete
YasmeenYasmeen	Yasmeen@gmail.com	9133065140	Village Development	\$ 50	\$ 20	Delete


Figure 16 (Funds Grid)

- 9) Donor page: This page is used to show funds using the dropdown we can select the funds

Add Donation's

Filter

Village Development



Village Development

Raised	Reached	Left
50	20	30

Donate

Figure 17 (Donor Page)

If clicked on the Donate Button

Description

To the development of my village

Donate now

Enter your Card Details

Card Number*

Expiry Month*

Expiry Year*

Name of Card Holder*

Security Code*

Figure 18 (Donation Details)

There are the details will be asked can donate by using our card Details.