Computer Science and Engineering

The Givers

Software Design Description (SDD)

Version 1.0

Document Number: SDD-001

Team Number: B01, Section A

Team Members (Name and NET-ID): David Bravo (db3454), Qilei Cai (qc542), Tasnim Nehal (tn1078), and Yikai Wang (yw3193)

REVIEW AND APPROVALS

Printed Name and Title	Function (Author, Reviewer, Approval)	Date	Signature
David Bravo	Author	October 6, 2020	David Bravo
Qilei Cai	Author	October 6, 2020	Qilei Cai
Tasnim Nehal	Author	October 6, 2020	Tasnim Nehal
Yikai Wang	Author	October 6, 2020	Yikai Wang
Professor Strauss	Reviewer	October 6, 2020	Professor Strauss

REVISION LEVEL

Date	Revision Number	Purpose
October 6, 2020	Version 1.0	Initial Release

Table of Contents

1. INTRODUCTION

1.1 Purpose

The purpose of this document is to document the System Architecture and the Detailed Design, including operations management, technical support, training, and operators. The intended audience for this document is the development, quality assurance, maintenance, and business analyst teams.

1.2 Scope

The mission of The Givers is to provide a non-profit fundraising platform for charities to appeal to donors and for donors to find and support the charitable efforts they are passionate about. The major features of the platform include an all-inclusive donation platform, custom charity ranking, charity categorization, marketing and analytics and client incentives.

The platform will provide a one-stop solution for donations, where signing up for a single account on the platform enables the user to donate to any charity registered on the platform. Users will be able to filter a large number of charities by the users' own preferences and zero in on their favorites. Location will be given a higher priority in recommending charities to users so that local charities can be highlighted. Users will also be able to sort charities by the number of employees and area of charitable interests.

To help charities advocate for their causes, the platform will feature a news feed to deliver the latest operational updates of charities to users as well as providing visualization for each charity's fundraising statistics. To provide incentives to the charity clients, they will receive bonuses from the platform for meeting fundraising targets or accomplishing noteworthy feats of charity.

1.3 Identification

This paragraph contains the system name, subsystem (if applicable), and release number to which this operations document pertains.

The Givers Software Design Description (SDD) Version 1.0 (Document number: SDD-001), released on October 6, 2020

1.4 Document Summary

The SDD document is meant to give a comprehensive overview of The Givers application including details on system design specifications, software structure, and the deployment architecture with dictionaries and diagrams to further support these details.

Format and content

- Title Page
- Review/Approval Signatures
- Table of Revisions (revision number, date, purpose)
- Approval page
- Table of Contents
- Introduction
- Reference Documents
- System-Wide Design Decisions
- Software Item Detailed Design
- Implementation Architecture (Not Required)
- Deployment Architecture
- Dictionaries
- Software Item Computer Resource Utilization
- Requirements Traceability
- System Design Testing
- Rationale
- Notes
- Appendices

1.5 System Overview

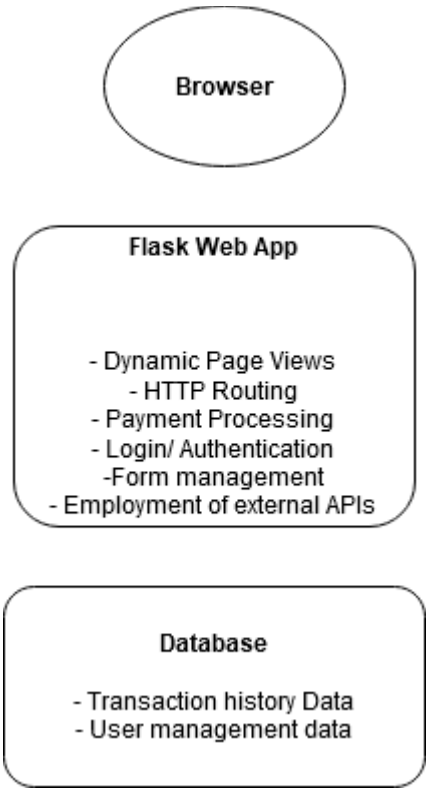
The Givers is built to connect people with their favorite charities. Users can create an account and indicate whether they manage any charities. Users who manage a charity can create and edit their charity page, as well as post any donation goals and news updates. Users who want to donate to charity can search the list of charities, look at recent news, subscribe to charities, and select payment options for donation. These interactions provide a convenient way for users to engage with charities.

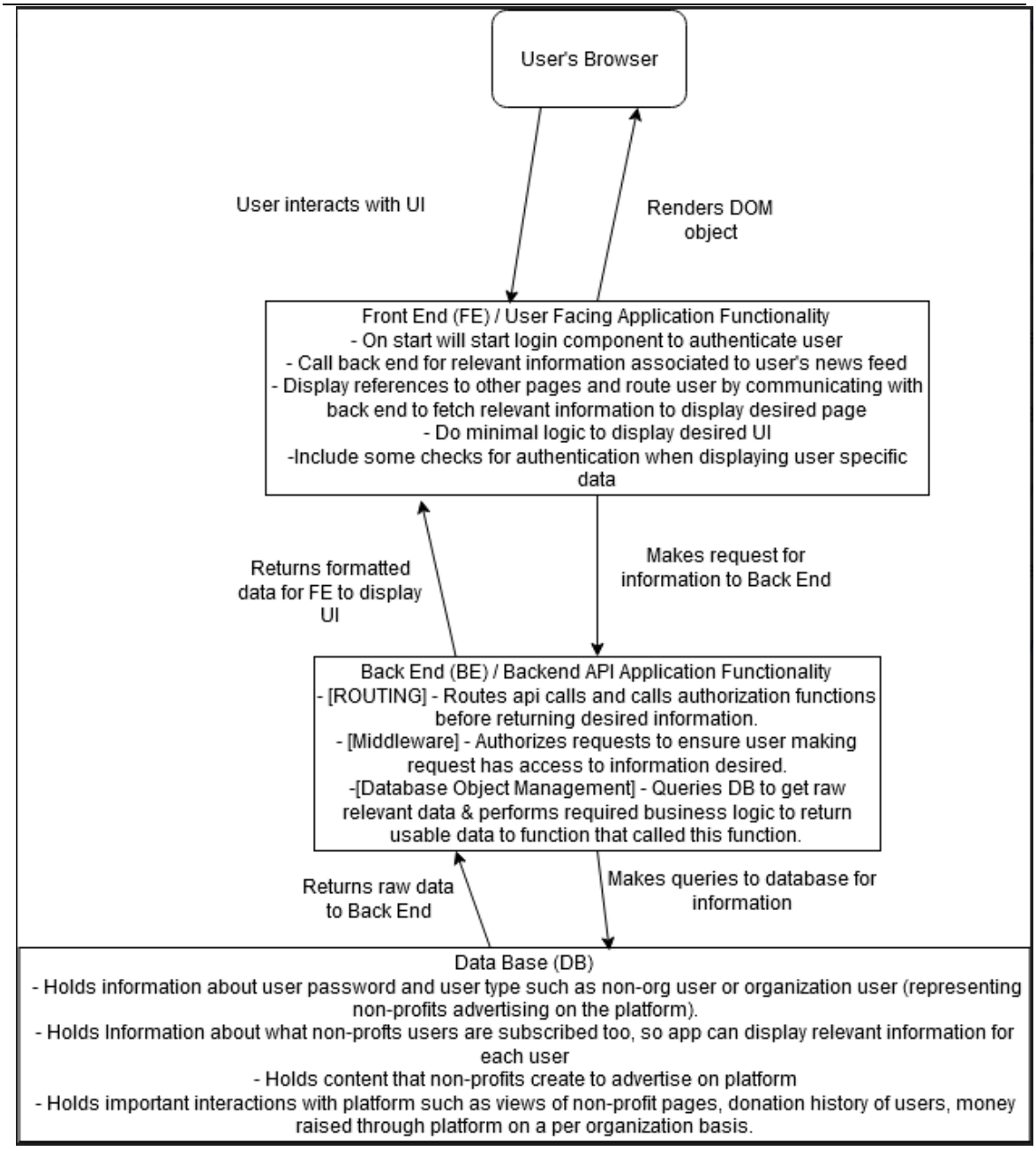
2. REFERENCE DOCUMENTS

- Team B01, “The Givers”, Project Proposal, Version 1.0, September 15, 2020
- Team B01, “The Givers”, SRS (Document number: SRS-001), Version 1.3, September 22, 2020
- Team B01, “The Givers”, SPMP (Document number: SPMP-001), Version 2.0, September 29, 2020

3. SYSTEM WIDE DESIGN DECISIONS

3.1 Software Component Architectural Design




User's Browser
User interacts with UI
Renders DOM object
Front End (FE) / User Facing Application Functionality
- On start will start login component to authenticate user
- Call back end for relevant information associated to user's news feed
- Display references to other pages and route user by communicating with back end to fetch relevant information to display desired page
- Do minimal logic to display desired UI
-Include some checks for authentication when displaying user specific data
Returns formatted data for FE to display UI
Makes request for information to Back End
Back End (BE) / Backend API Application Functionality
- [ROUTING] - Routes api calls and calls authorization functions before returning desired information.
- [Middleware] - Authorizes requests to ensure user making request has access to information desired.
-[Database Object Management] - Queries DB to get raw relevant data & performs required business logic to return usable data to function that called this function.
Returns raw data to Back End
Makes queries to database for information
Data Base (DB)
- Holds information about user password and user type such as non-org user or organization user (representing non-profits advertising on the platform).
- Holds Information about what non-profts users are subscribed too, so app can display relevant information for each user
- Holds content that non-profits create to advertise on platform
- Holds important interactions with platform such as views of non-profit pages, donation history of users, money raised through platform on a per organization basis.

3.2 Software Architecture General Description

1. The top layer is the browser-based interface that a user can connect to and begin interaction with The Givers application.
2. The second layer represents the main Flask web app that contains important user functionalities such as serving dynamic webpages, authenticating users, and HTTP routing. Other functionalities are mentioned in the diagram in section 3.1. The functionalities in this layer allow for users to have a rich experience with The Givers web application.
3. The third layer represents the database store where important information is stored such as transaction history and user login credentials. This layer is important for users to be able to access personalized content when they want it.

Once the user has accessed The Givers through a browser, the Front End will prompt for a login. Once that login has been verified the user can access various features such as searching for charities by cause or location, donating to charities, and subscribing to charities to receive news updates from them. The backend will have two APIs: one for processing payment information allowing for donations to be made and a newsfeed API that will let scrape the web for relevant articles related to a charity. The Database Object Management component will handle things like updating charity details, securing login information, and updating transactions. This information will all be stored in a PostgreSQL database.

3.3 Software Item Components

Software Item Component	Description
Browser	The user's method of interacting with the UI (either a desktop browser or a mobile browser)
Templates (Dynamic Page Views)	Files that create dynamic pages with the use of conditional html templating and scripts.
HTTP Routing	Applications handling of HTTP routing to serve appropriate content with the use of templates and some logic.
Payment Processing	Processing of user payments with external API and payment processing software service (Stripe).
User Authentication	User authentication service that interacts with database to verify user credentials and authenticate requests for user specific information.
News API	Fetching of related news with Google API and select key words.
Form Management	Manages information associated with a given form.
Database	Stores user management information and transaction history for use within web application.

Software Item Component	Description
User's Browser	The user's method of interacting with the UI (either a desktop browser or a mobile browser)
Front End	Responsible for sending data to the user's browser, routing the user to different pages, and communicating with the backend
Back End	Returns requested information to the front end
Routing	Routes API calls
Middleware	Handles authorization of requests and performs functions
Database Object Management	Queries the database to transfer data between the back end and database and vice versa
Database	Stores required data such as transactions, newsfeed subscriptions, and charity information

3.4 Component Interface Identification

Component
User's Browser <-> HTTP Routing
HTTP Routing <-> Templates
Form management <-> Payment Processing
Database <-> Payment Processing
Form management <-> User Authentication
User Authentication <-> Database
HTTP Routing <-> User Authentication
News API <-> Templates

Component	Responsibilities
Routing	Handles the calls to the payment API when users want to donate
Middleware	The payment API processes the payment information while the newsfeed API finds relevant articles on a charity to update the user.

3.5 Software Component Concept of Execution

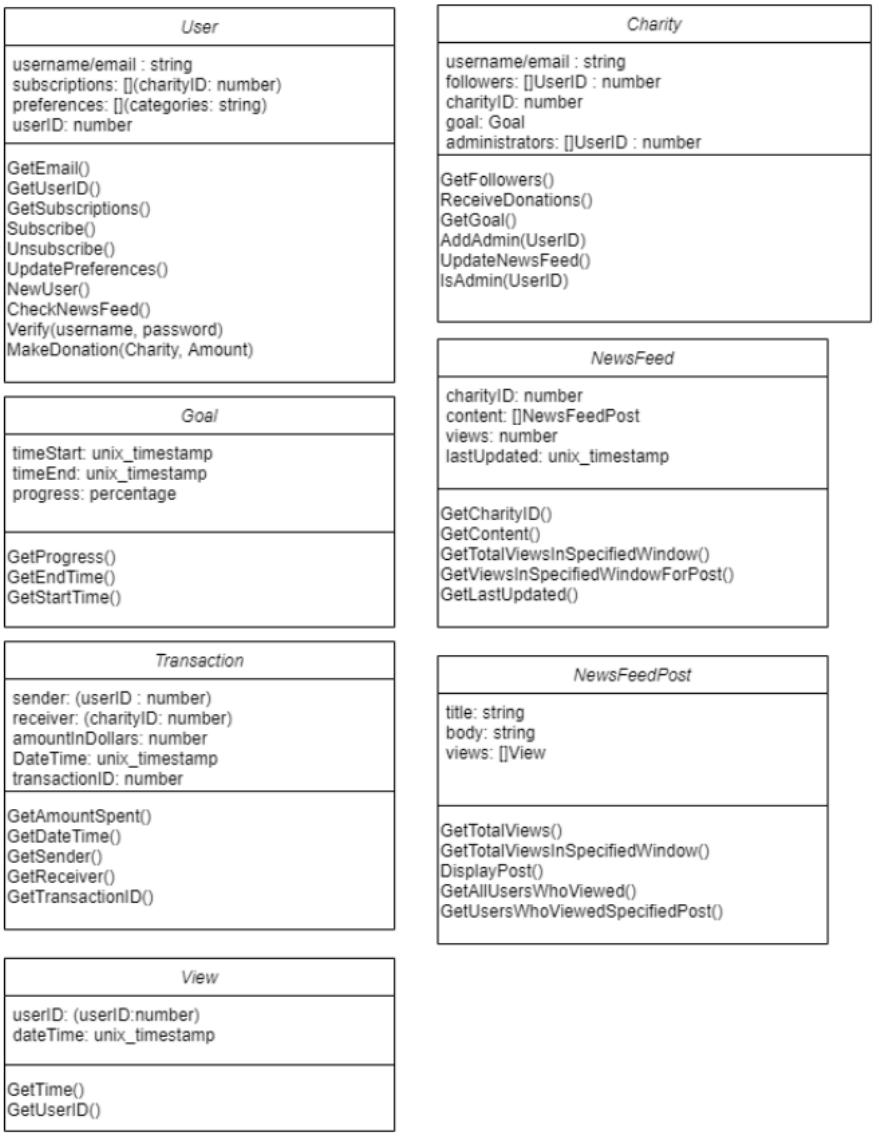
1. The database is initialized on amazon AWS using the PostgreSQL RDS service.
2. The flask web application is started on an EC2 instance on amazon AWS.
3. The flask web application starts and connects to the PostgreSQL database.
4. The flask web application attempts to connect to the News API to test the connection is working as expected.
5. The flask web application attempts to connect to the strip API service to test the service is working as expected.
6. The flask web application server listens for HTTP requests on pre-defined address.

- First the Flask application will be hosted on AWS and will need to be started up along with the PostgreSQL database
- User's Browser: The user can access The Givers web application through any browser
- Front End: The front end will prompt the user to enter their account information to login and prompts for things like searching for charities and donating to charities.
- Back End: The back end will query the Database Object Management component which will verify the user's account information.
- Routing: Once the user has entered their payment information and the amount to be donated to a charity this information will be routed as an API call.
- Middleware: Theis included the payment API that will process payment information for donations and the newsfeed API that will post relevant news articles related to a charity.
- Database Object Management: This component will send the transaction information to the database along with updating the amount of money donated to that charity.
- Database: Here the updated numbers will be stored.

4. SOFTWARE ITEM DETAILED DESIGN

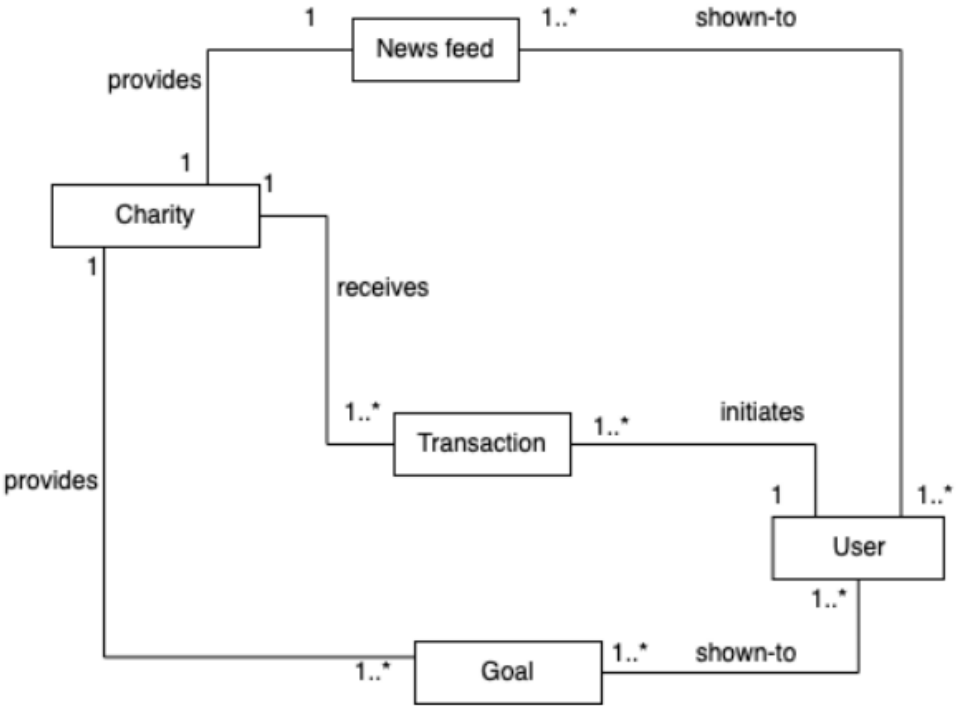
4.1 Structure

4.1.1 Software Unit Detailed Design



4.2 Static Relationship of Software Unit

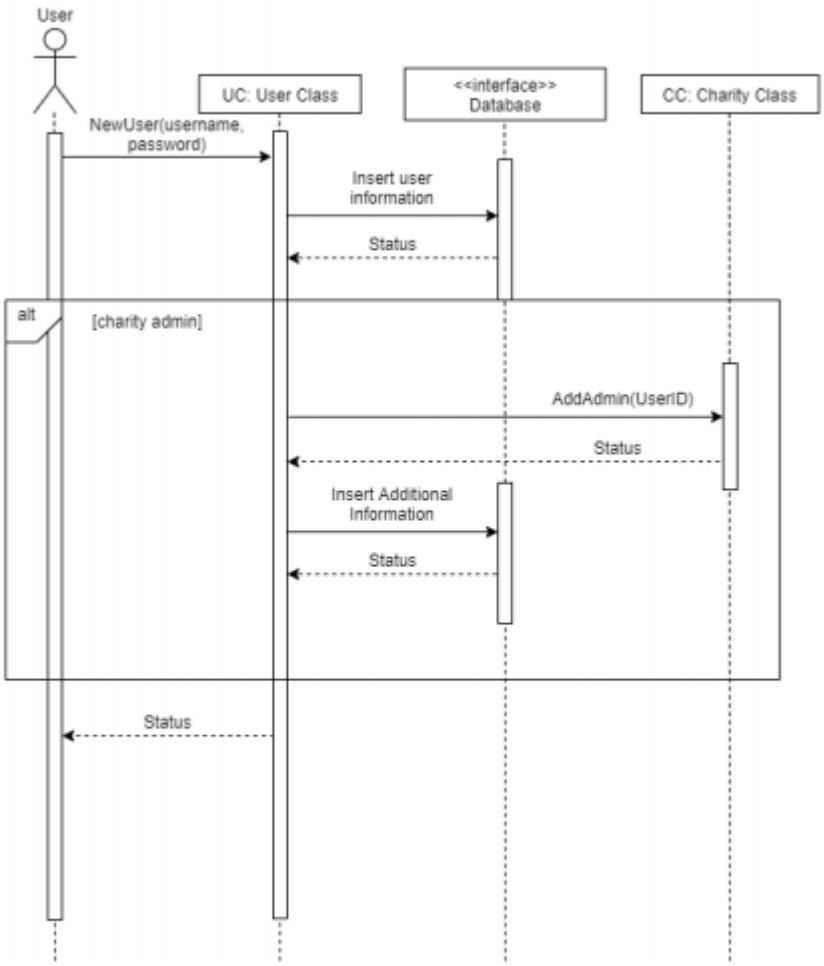
4.2.1 Run-time Object Instances



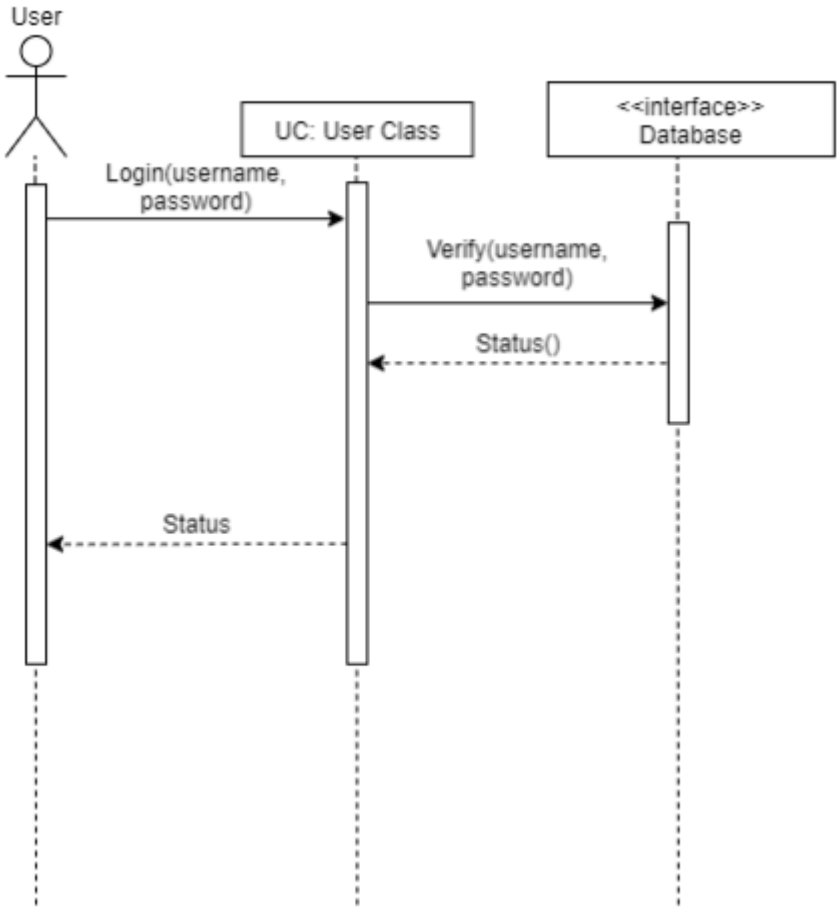
4.3 Behavior

4.3.1 Sequence Interaction Diagrams

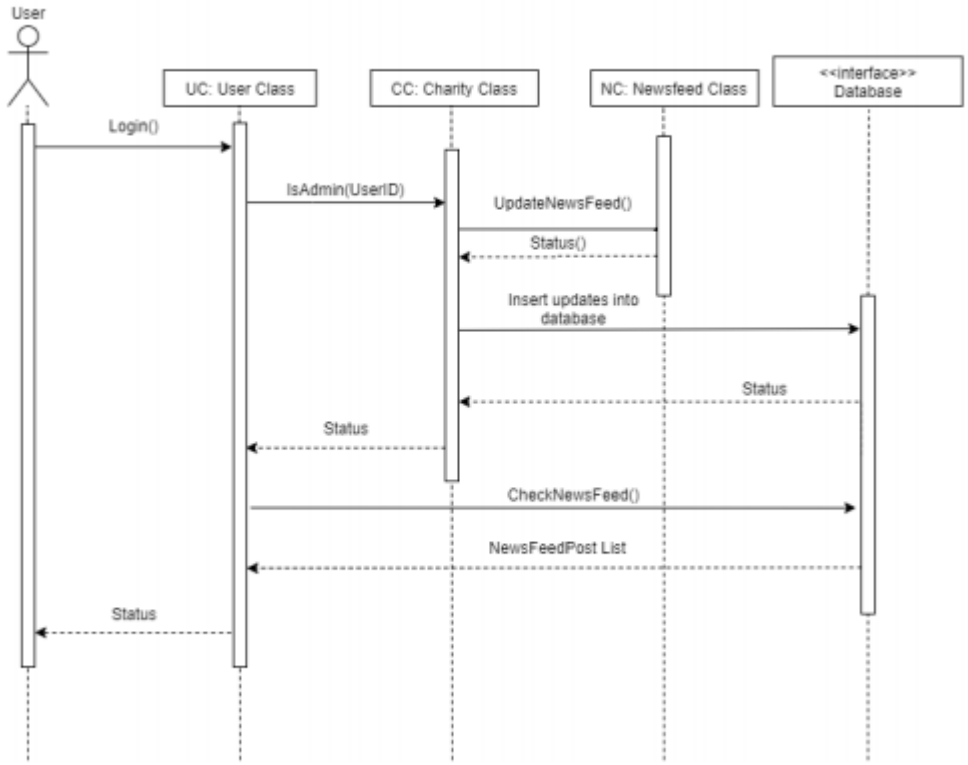
UC1: Creating Account



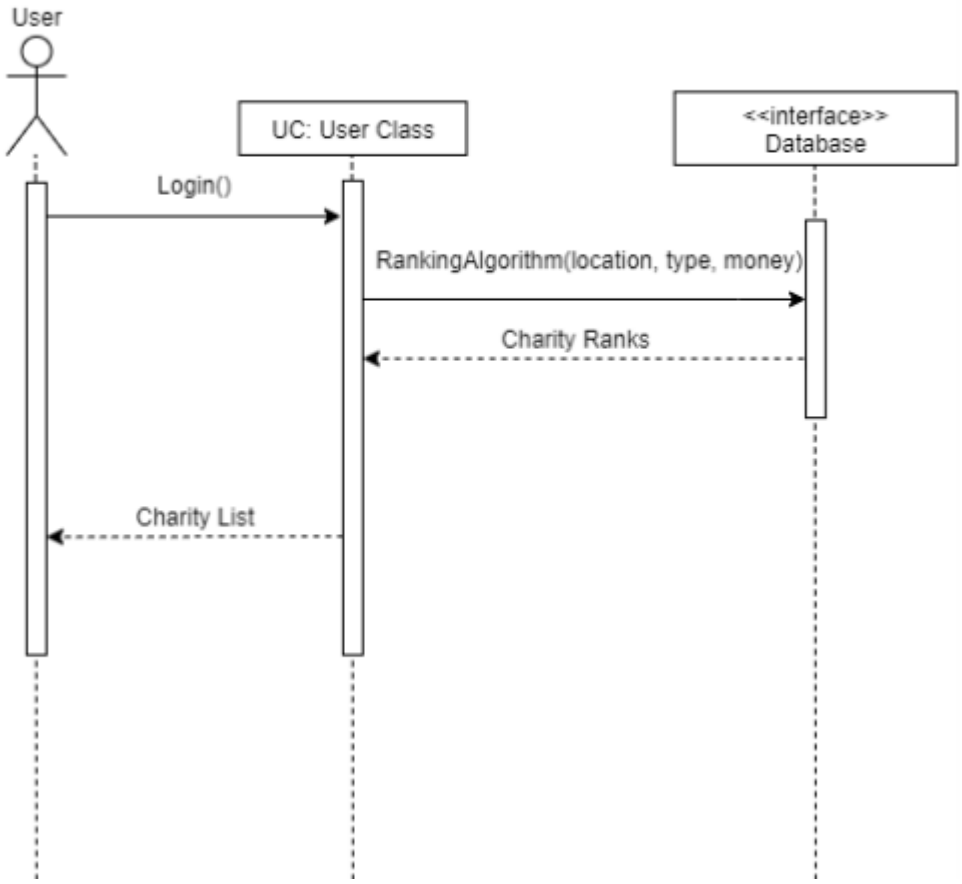
UC2: Login


User
UC: User Class
<<interface>>
Database
Login(username, password)
Verify(username, password)
Status()
Status

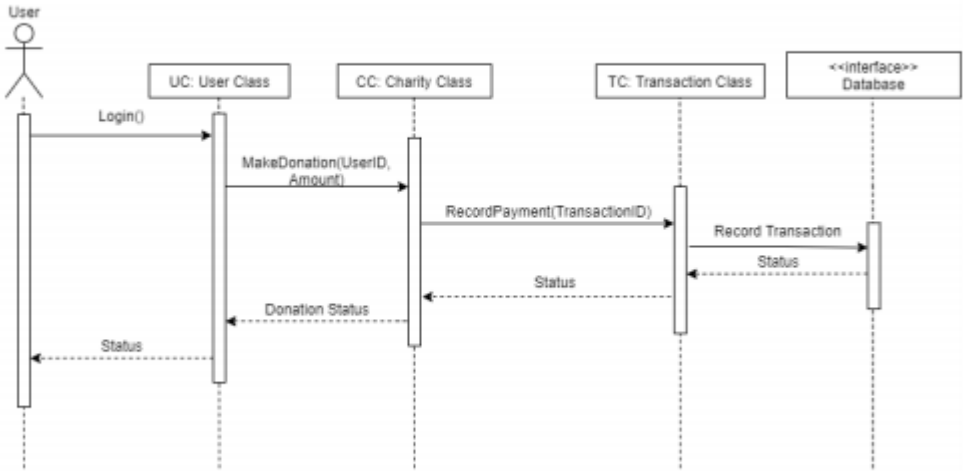
UC3: Updating Information to Users



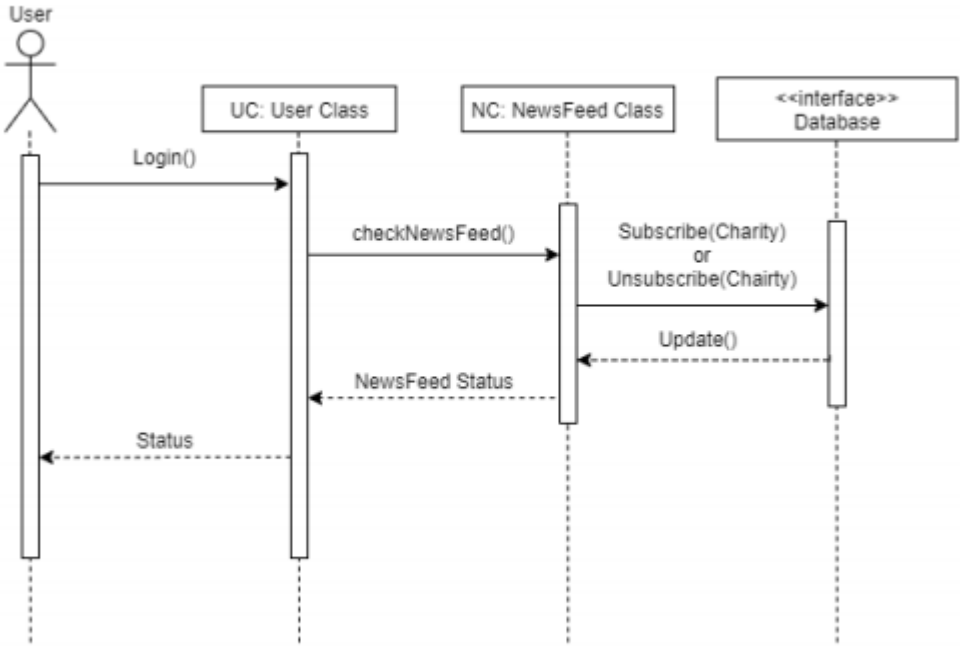
UC4: Browsing Charities by Category



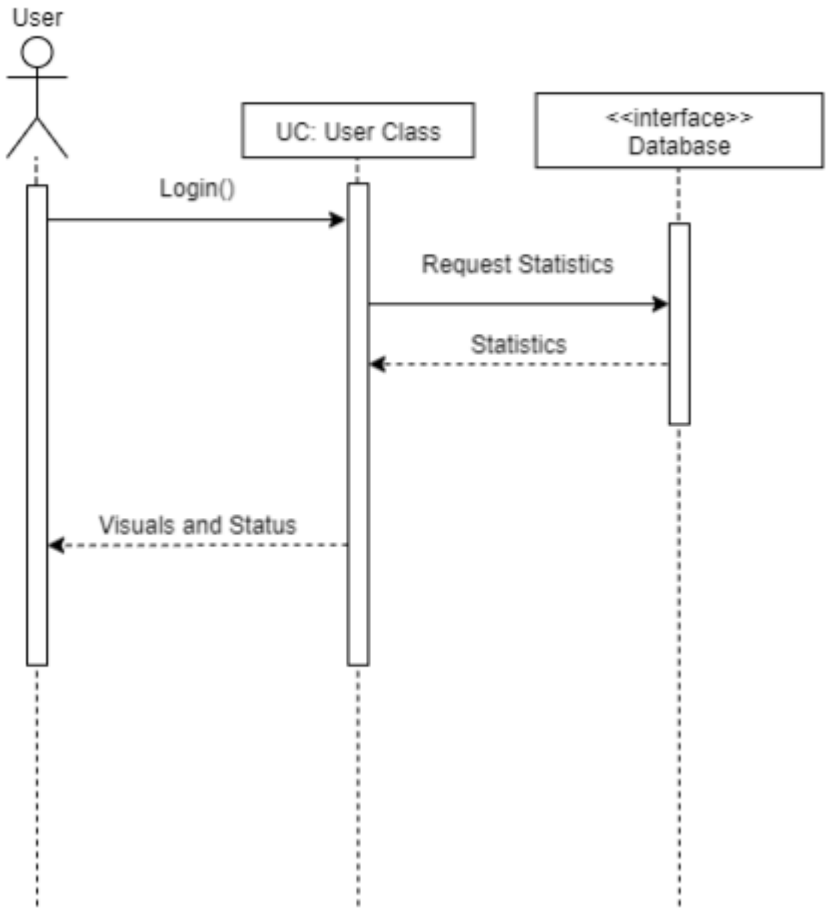
UC5: Selecting Charities to Donate



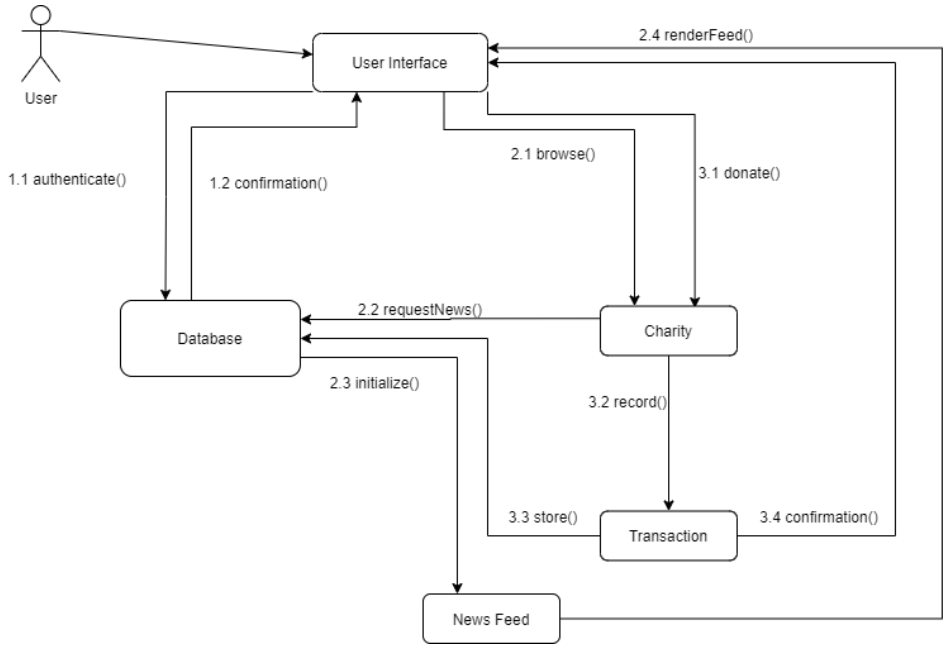
UC6: Checking Custom Newsfeed



UC7: Visualizing Donations and Interactions

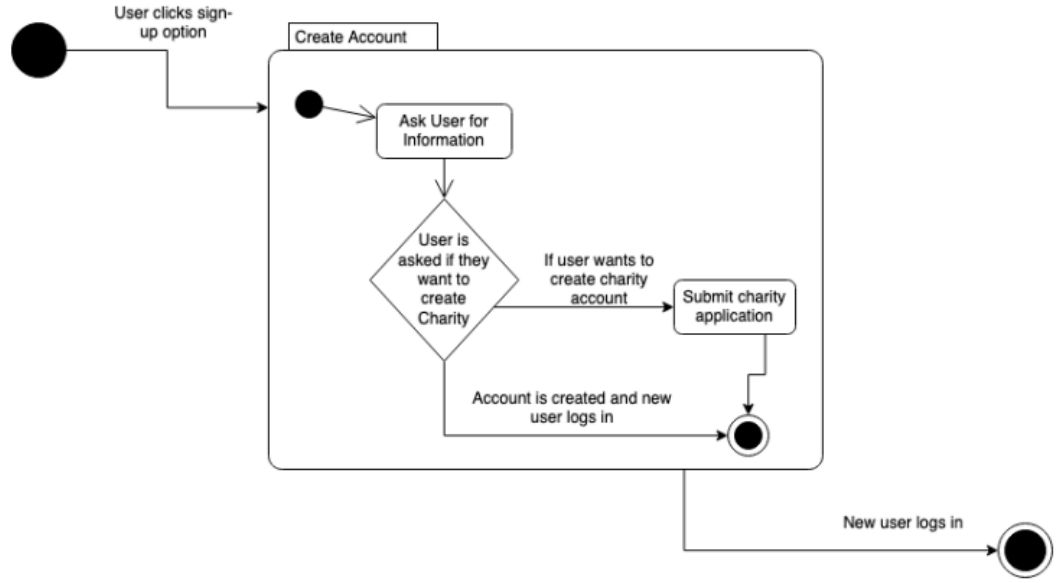


4.3.2 Collaboration Diagrams

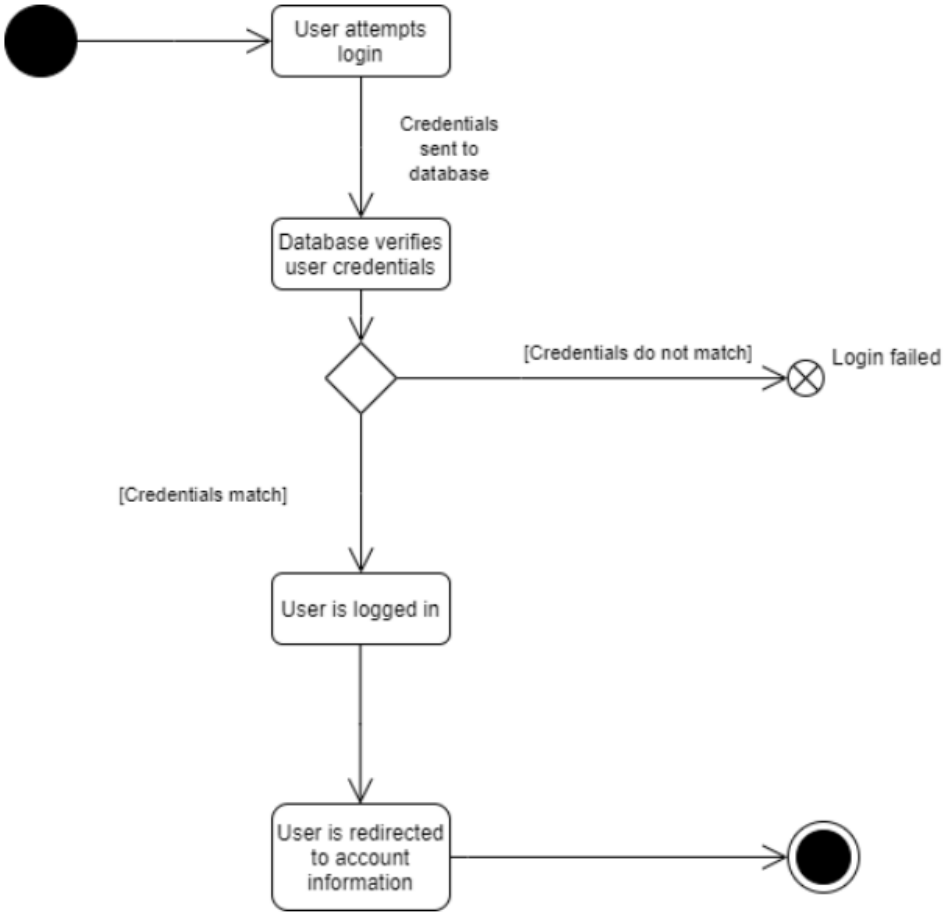


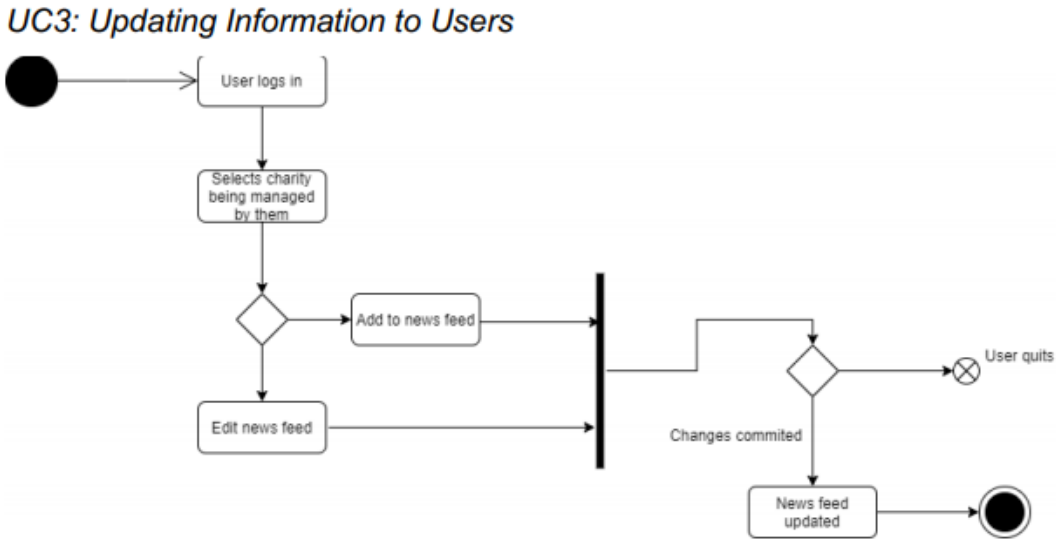
4.3.3 Activity Diagrams

UC1: Creating Account

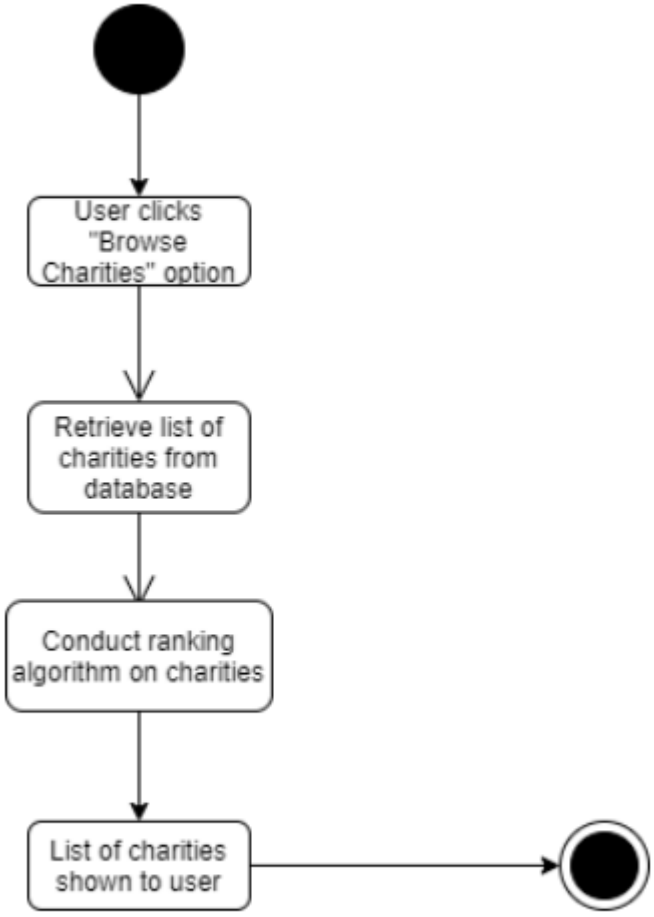


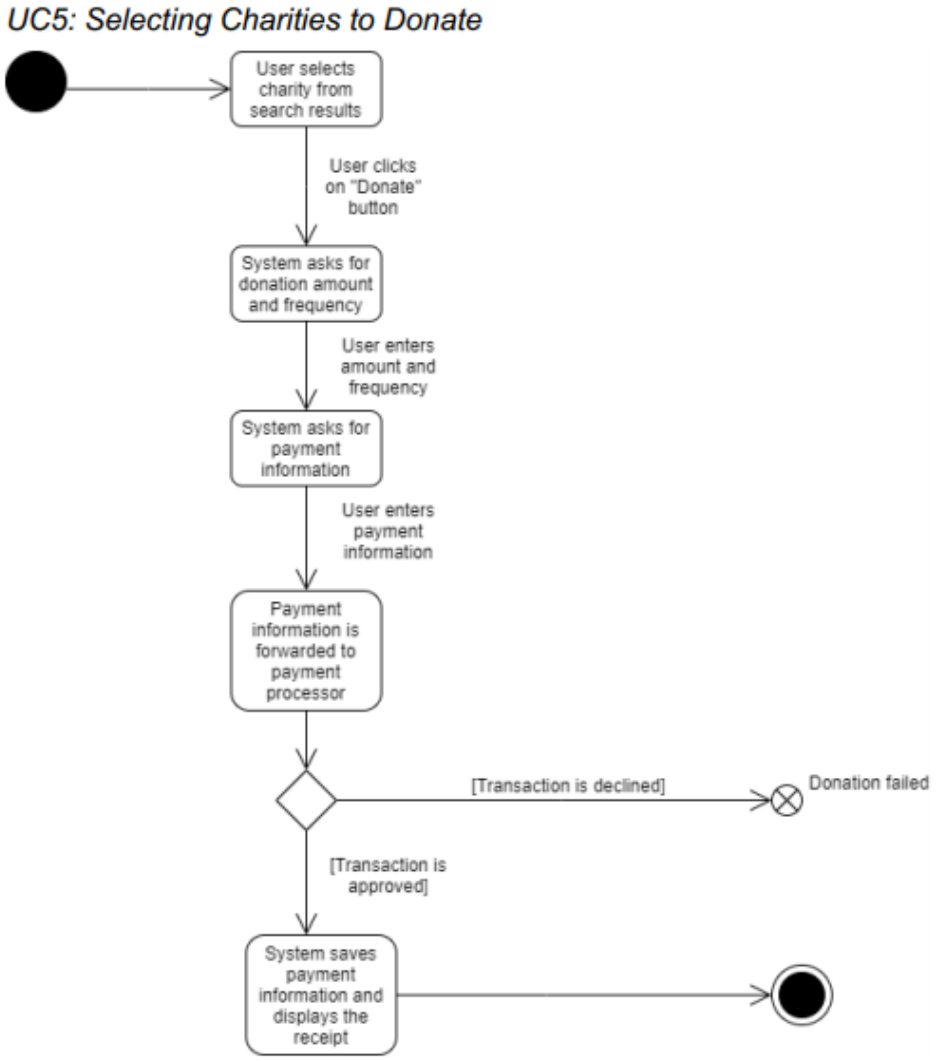
UC2: Login


User attempts login
Credentials sent to database
Database verifies user credentials
[Credentials do not match]
Login failed
[Credentials match]
User is logged in
User is redirected to account information

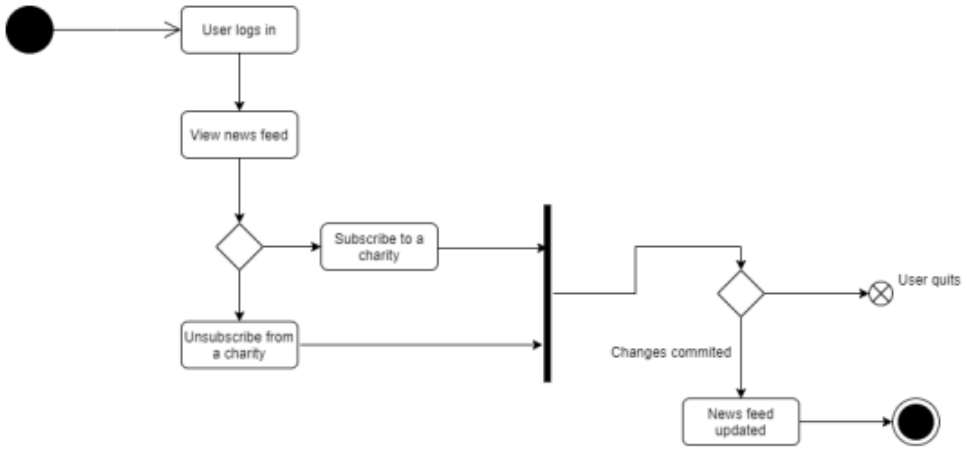

UC3: Updating Information to Users
User logs in
Selects charity being managed by them
Add to news feed
Edit news feed
User quits
Changes commited
News feed updated

UC4: Browsing Charities by Category



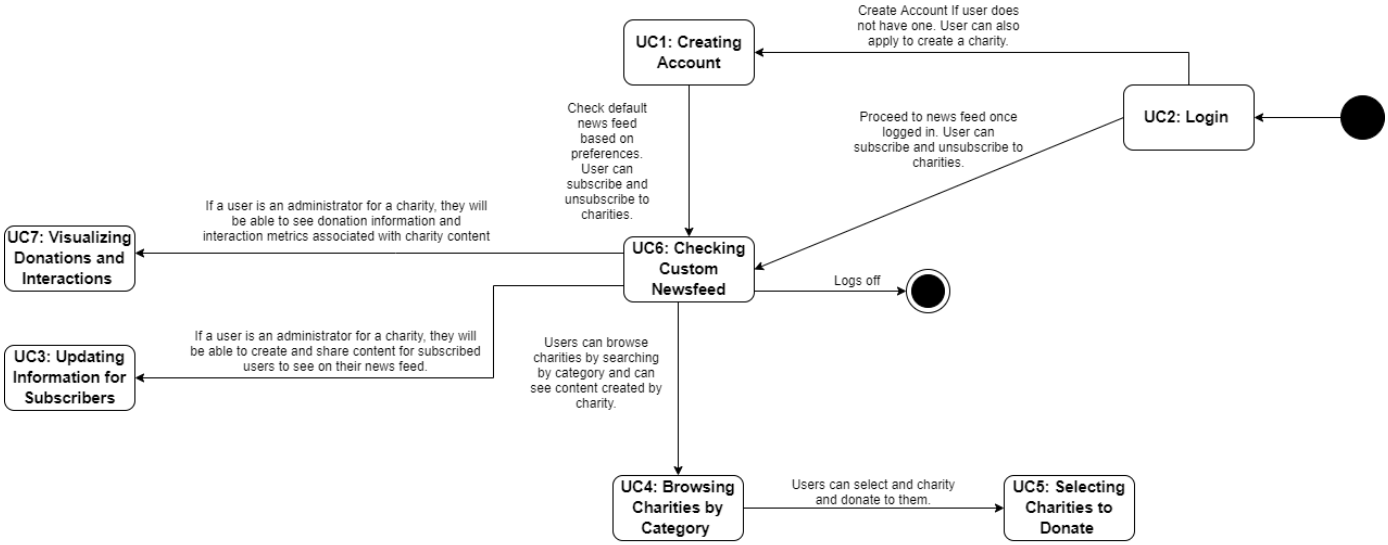

UC5: Selecting Charities to Donate
User selects charity from search results
User clicks on "Donate" button
System asks for donation amount and frequency
User enters amount and frequency
System asks for payment information
User enters payment information
Payment information is forwarded to payment processor
[Transaction is declined]
Donation failed
[Transaction is approved]
System saves payment information and displays the receipt

UC6: Checking Custom Newsfeed

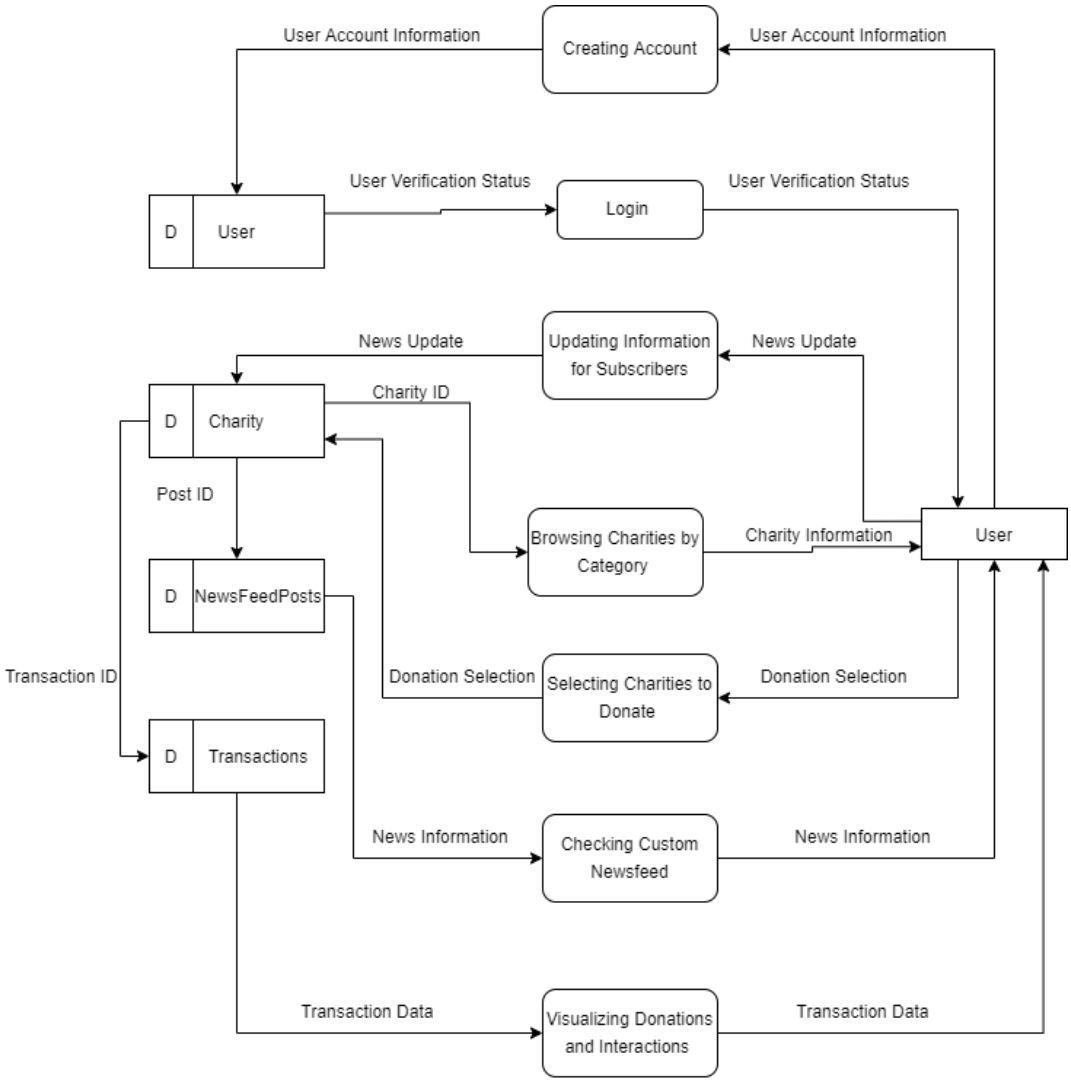


UC7: Visualizing Donations and Interactions

4.3.4 State Diagrams



4.3.5 Event Diagrams



4.4 Concept of Execution

Making an Account:

- The user will be requested for a verifiable email and a sufficient password (not too simple to be brute forced)
- The password will be hashed and salted before being stored in the database

Login:

- The user's email and password will be verified.

Charity Search:

- The Givers will enable users to search for charities based on categories such as cause (animal aid, disease prevention and treatment, etc.) along with location so users can find the exact type of charity they are looking for.

Newsfeed Subscription:

- When viewing a charity, users can opt in to hit a subscribe button that will send news updates whenever a charity updates their newsfeed
- Likewise, users can also unsubscribe from a charity

Donation:

- Users can choose to either save their payment information or reenter their payment information along with selecting a charity and the amount to be donated.
- The payment API will process this information and notify the user once the donation has gone through

Data Visualization:

- Managers of a charity can see extra statistics and graphs on that charity such as average donations with a given time period and donator retention

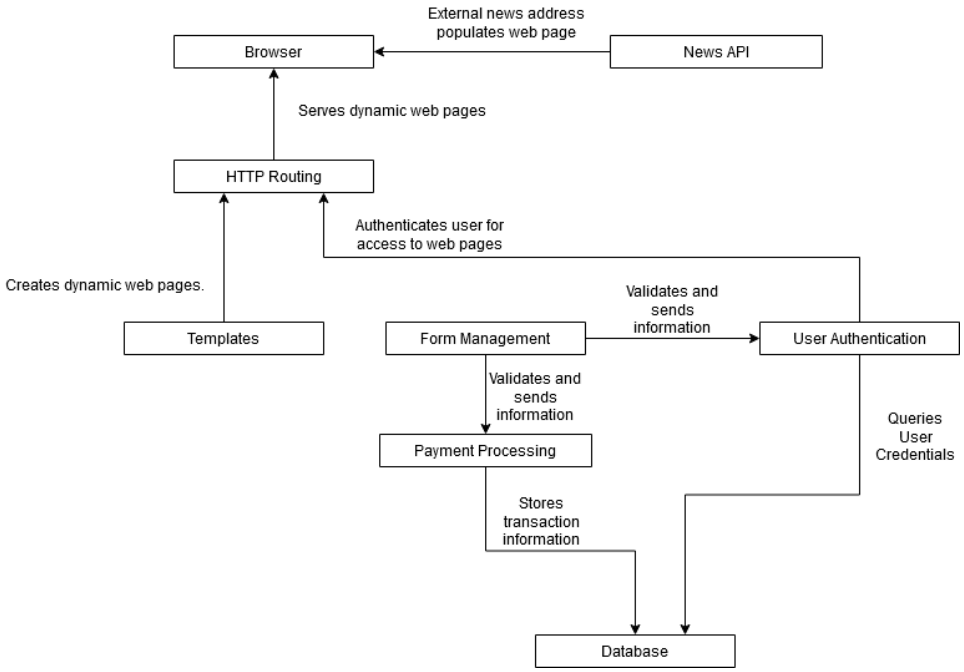
4.5 Interface Design

4.5.1 Unique identifier of Interface

Component	Description
Browser <-> HTTP Routing	The user's browser interacts with the HTTP routing of the application by sending HTTP requests to the address where The Givers is located.
HTTP Routing <-> Templates	The routing service in the application is served appropriate dynamic web pages, created by templates with some logic, to the browser.
Form management <-> Payment Processing	Information submitted through forms associated with payments is processed using the stripe API.
Database <-> Payment Processing	Confirmation of transactions received from the payment processing service (Stripe API), is recorded into the transaction history on the database.
Form management <-> User Authentication	Information submitted through forms associated with user credentials is verified and validated through the login and authentication service.
Login/ Authentication <-> Database	The login and authentication service queries the user management data in the Database to verify user credentials.
HTTP Routing <-> User Authentication	HTTP routing service ensures user specific web pages are authenticated before serving a request.
News API <-> Templates	Templates include a frame with an address of content that the user’s browser will

	fetch.

4.5.2 Interface Diagrams



5. IMPLEMENTATION ARCHITECTURE (NOT REQUIRED)

5.1 All Active and Passive Classes Assigned to Components

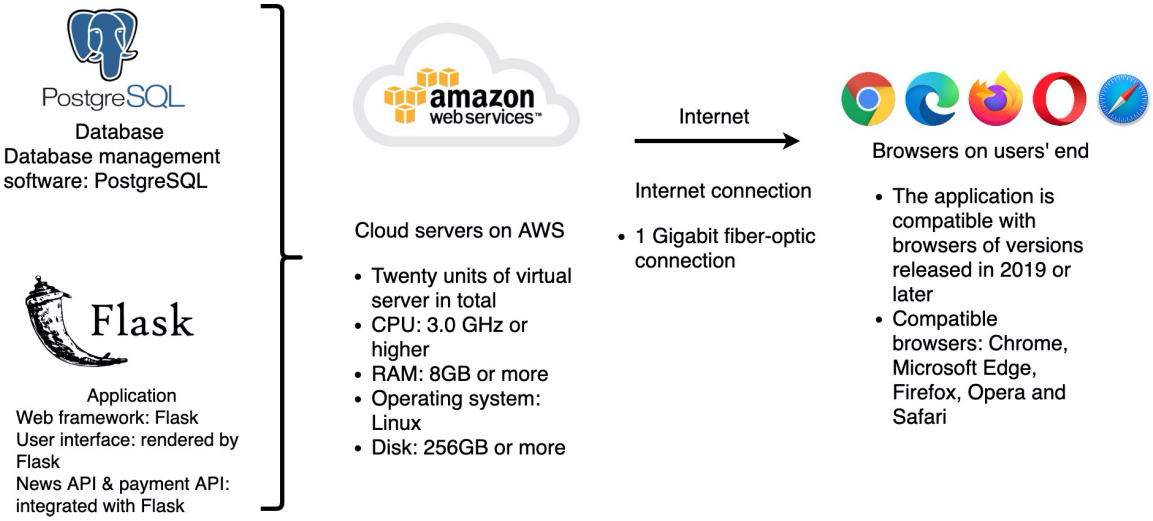
This section is not required for this application.

5.2 Diagrams of Physical Packaging of Logical Components

This section is not required for this application.

6. DEPLOYMENT ARCHITECTURE

6.1 Physical Deployment Architecture Diagram



7. DICTIONARIES

Classes

Name	Description	Methods	Attributes
User	Information on a user's account	GetEmail, GetUserID, GetSubscriptions, Subscribe, Unsubscribe, UpdatePreferences, NewUser, CheckNewsFeed, Verify, MakeDonation	username, email, subscriptions, preferences, userID
Goal	Status of fundraising goals	GetProgress, GetEndTime, GetStartTime	timeStart, timeEnd, progress
Transaction	Status of donation transactions	GetAmountSpent, GetDateTime, GetSender, GetReceiver, GetTransactionID	sender, receiver, amountInDollars, dateTime, transactionID
View	Information on the user who viewed a news feed post and the time of access	GetTime, GetUserID	userID, dateTime
Charity	Administrative features of a charity's account	GetFollowers, ReceiveDonations, GetGoal, AddAdmin, UpdateNewsFeed, IsAdmin	username, email, followers, charityID, goal, administrators
NewsFeed	Generating the content of the news feed	GetCharityID, GetContent, GetTotalViewsInSpecifiedWindowForPost, GetLastUpdated	charityID, content, views, lastUpdated
NewsFeedPost	Rendering a specific post on the news feed	GetTotalViews, GetTotalViewsInSpecifiedWindow, DisplayPost, GetAllUsersWhoViewed, GetUsersWhoViewedSpecifiedPost	title, body, views

Methods

Name	Description	Class	Arguments
GetEmail	Return the email address of the user	User	[None]
GetUserID	Return the user ID	User	[None]
GetSubscriptions	Return the list of charities that the user is subscribing to	User	[None]
Subscribe	Subscribe to a charity's updates	User	charityID
Unsubscribe	Unsubscribe from a charity's updates	User	charityID
UpdatePreferences	Update the user's preferences	User	preferences
NewUser	Create a new user account	User	userID
CheckNewsFeed	Return updates from the news feed	User	[None]
Verify	Verify the username and password entered to log into the account	User	username, password
MakeDonation	Initiate a donation transaction	User	charityID, amount
GetProgress	Return the	Goal	[None]

	percentage of fundraising goal fulfilled		
GetEndTime	Return the fundraising deadline	Goal	[None]
GetStartTime	Return the start date of the fundraiser	Goal	[None]
GetAmountSpent	Return the amount of the transaction	Transaction	[None]
GetDateTime	Return the date and time of the transaction	Transaction	[None]
GetSender	Return the user who is making the transaction	Transaction	[None]
GetReceiver	Return the charity who is receiving the donation	Transaction	[None]
GetTransactionID	Return the transaction ID	Transaction	[None]
GetTime	Return the time of accessing the news feed post	View	[None]
GetUserID	Return the user who accessed the news feed post	View	[None]
GetFollowers	Return the list of users who subscribed to the charity's	Charity	[None]

	updates		
ReceiveDonations	Record the specifics of a transaction	Charity	transactionID
GetGoal	Return the charity's fundraising goal	Charity	[None]
AddAdmin	Designate a user as the administrator of the charity	Charity	userID
UpdateNewsFeed	Add a post to the charity's news feed	Charity	newsFeedPost
IsAdmin	Return whether the user is the administrator of the charity	Charity	userID
GetCharityID	Return the ID of the charity	NewsFeed	[None]
GetContent	Return the charity's news feed posts	NewsFeed	[None]
GetTotalViewsInSpecifiedWindowForPost	Return a post's number of views to a specified window	NewsFeed	[None]
GetLastUpdated	Return the time the charity last updated its news feed	NewsFeed	[None]
GetTotalViews	Return the total number of views the charity has	NewsFeedPost	[None]

	gained		
GetTotalViewsInSpecifiedWindow	Return the total number of views to a specified window	NewsFeedPost	[None]
DisplayPost	Display a post made by the charity	NewsFeedPost	[None]
GetAllUsersWhoViewed	Return a list of all users who viewed the charity	NewsFeedPost	[None]
GetUsersWhoViewedSpecifiedPost	Returns a list of all users who viewed a specific post	NewsFeedPost	[None]

Attributes

Name	Class	Description	Type
username	User	The username represented as a string	string
subscriptions	User	A list of *charityID* variables	list
Preferences	User	A list of *categories* variables	list
userID	User	The userID represented as an integer	integer
timeStart	Goal	The start time of the goal represented as a UNIX timestamp	UNIX timestamp
timeEnd	Goal	The end time of the goal represented as a UNIX timestamp	UNIX timestamp
progress	Goal	The percentage of progress made, represented as an integer from 0 to 100	integer
sender	Transaction	A *userID* variable represented as an integer	integer
receiver	Transaction	A *userID* variable represented as an integer	integer
amountInDollars	Transaction	The amount of the transaction represented as an integer	integer
DateTime	Transaction	The time and date of the transaction, represented as a UNIX timestamp	UNIX timestamp
transactionID	Transaction	The ID of the transaction, represented as an integer	integer
userID	View	The *userID* of the	integer

		user who viewed a news feed post, represented as an integer	
dateTime	View	The time and date when the user viewed the post, represented as a UNIX timestamp	UNIX timestamp
username	Charity	The username of the charity, represented as a string	string
followers	Charity	A list of *userID* variables, which belong to the users who follow the charity	list
charityID	Charity	The ID of the charity, represented as an integer	integer
goal	Charity	An instance of the Goal class	Goal
administrators	Charity	A list of *userID* variables, which belong to the administrators of the charity	list
charityID	NewsFeed	The ID of the charity, represented as an integer	integer
content	NewsFeed	A list of *NewsFeedPost* objects	list
views	NewsFeed	The number of views the post has gained, represented as an integer	integer
lastUpdated	NewsFeed	The last time the news feed was updated, represented as a UNIX timestamp	UNIX timestamp
title	NewsFeedPost	The title of the post, represented as a	string

		string	
body	NewsFeedPost	The body of the post, represented as a string	string
views	NewsFeedPost	A list of *View* objects, which contain *userID* and *dateTime* variables	list

8. SOFTWARE ITEM COMPUTER RESOURCE UTILIZATION

The Givers is a web application that requires a browser of a version released in 2019 or later for compatibility purposes. Compatible browsers include Microsoft Edge, Chrome, Safari, Opera and Firefox. Running these browsers require a minimum of 4GB of RAM. The computer's RAM utilization cannot exceed 70% while running the app on the browser, or more RAM will need to be installed to ensure the app can be run smoothly.

The app requires a minor amount of local storage for storing cache, which will be kept under 50MB. As for CPU utilization, the calculations are needed to display charts, graphs, and other data visualizations on the front end are not intensive and should take no more than 10% of the CPU's capacity. The CPU should have enough capacity so that the app will take no more than 50% of the capacity, or there will not be enough capacity for the other processes of the operating system. If the app takes more than 50% of the CPU capacity, the CPU will need to be replaced with one of higher performance.

9. REQUIREMENTS TRACEABILITY

9.1 Software Component-Level Requirements Traceability

Every component must be traceable to a user requirement. Each requirement has a unique number which is tracked and worked on by a member on the development team. Progress should be recorded in documents and as comments in the source code for each component and any changes to the component should be noted as that may have an impact on the user experience. This would allow the requirement to be tracked forwards and backwards. Traceability is helped with the use of diagrams and charts to show the relationship of requirements and components.

10. SYSTEM DESIGN TESTING

The type of development cycle used is iterative and incremental development. The team has been divided into different groups which are focusing on different features: front-end design, back-end features, and data visualization. Meetings are held regularly where developers share updates on their progress on their

development on specific components. Each component is tested before further components are made. Components are tested with general use cases first before moving towards edge cases. Components must pass a reasonable number of test cases to be accepted and incorporated into the product.

Testing for the application consists of four types: peer review, self-check, walkthrough and inspection. During peer review, members of the development team will review each other's source code, run test cases and propose revisions. Developers will also conduct self-check to spot their own defects. During walkthrough, each developer will present and elaborate on their source code in front of others. Inspection is the most formal of all types. An experienced moderator will initiate the procedures. Reviewers will screen the source code in question according to functional requirements and take note of the defects. During the inspection meeting, the defects found by reviewers will be presented, and developers will discuss possible fixes.

When changes are made to the software design and functional requirements, the developers will update the SQA testing plan and test cases in accordance with the changes. The software quality group will conduct product testing, during which the development team will provide counsel and assistance as they see fit. Once product testing is done, the development team will invite the end user, who made the original proposal for The Givers, to conduct acceptance testing. Feedback from the end user will be noted and discussed by the development team, who will make revisions to the product accordingly.

Once the project passes testing, it will go into operations. If the project fails testing, it will return to development, where the defects will be fixed.

11. RATIONALE

Charities maintained their websites in a flawed system where you have to go to different websites and enter your payment information each time or create multiple accounts just to support multiple charities. The Givers team believes that this is a needlessly slow system. Our goal is to streamline this process by creating an application that allows a user to donate to several charities with just one account. Along with making the donation process simpler, The Givers also plans to provide charity mangers with data visualization features to help improve donator retention and allow charities to see where they can improve. Right now there is no such website or application that allows users to search for charities based on factors like cause or location along with letting you see the newsfeeds for multiple charities on a single location. These are all features the team will implement and provide. The Givers hopes to provide a solution to the problem of how charities can reach out to donors and vice versa.

12. NOTES

13. APPENDICES

13.1 Dictionaries

The dictionaries are included in Section 7 of the document, Dictionaries.

13.2 UML diagrams

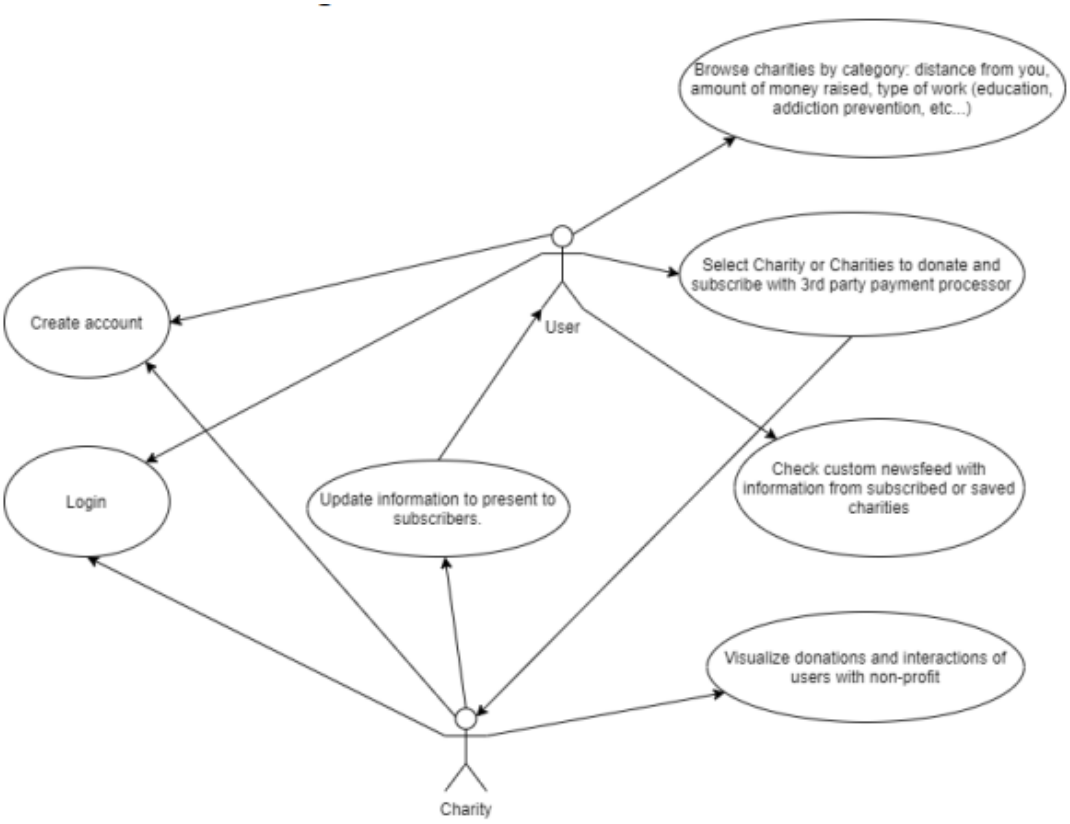
The diagrams are included in Section 4 of the document, Software Item Detailed Design.

13.3 Requirements diagrams

- Functional Requirement 1 (SR1): Account Creation
  - The system shall enable users to sign up for a new account with their personal information.
    - SR1.1: Users can select the sign-up button and fill in information such as their name and birthday.
    - SR1.2: Users can modify their account information at any time after they log in.
    - SR1.3: Users have the option to delete their account in their account management page.
- Functional Requirement 2 (SR2): Login
  - Once an account has been made the system shall enable users to login with a username and a password.
    - SR 2.1: Users can log in with a username and a password.
    - SR 2.2: Users can log out after signing in.
- Functional Requirement 3 (SR3): Updating Information for Subscribers
  - The system shall enable charity managers to add information and edit their newsfeeds.
    - SR3.1: Charity managers can provide updates to their subscribers on new goals and drives.
    - SR3.2: Charity managers can automate notifications to subscribers on when certain milestones are reached.
- Functional Requirement 4 (SR4): Browsing Charities by Categories
  - The system shall enable users to categorize charities by qualities such as location, type of cause, and amount of money raised.
    - SR 4.1: Users can browse charities by location.
    - SR 4.2: Users can browse charities by type of charitable cause.

        - Types of charitable causes: Animals, Arts & Culture, Community Development, Education, Environment, Human & Civil Rights, Human Services, Religion, Research & Public Policy.
    - SR 4.3: Users can browse charities by amount of money raised.
- Functional Requirement 5 (SR5): Selecting Charities to Donate
    - The system shall enable users to select a charity and a payment method and process the donation to that charity.
        - SR5.1: Users can select a charity.
        - SR5.2: Users can be forwarded to specific details of a charity such as the past public communications.
        - SR5.3: Users can donate to a selected charity by clicking the donate button once the user has selected a charity.
- Functional Requirement 6 (SR6): Checking Custom Newsfeed
    - The system shall enable users with accounts to subscribe to the newsfeed of a chosen charity
        - SR6.1: Users can see news updates from selected charities.
        - SR6.2: Charities can get notifications for when selected charities reach milestones.
- Functional Requirement 7 (SR7): Visualizing Donations and Interactions
    - The system shall enable charity mangers to view a graphical representation of the donations they received.
        - SR7.1: Charity managers can view graphs detailing the number of donations within a time period.
        - SR7.2: Charity managers can see charts on donator retention.

Use Case Diagram



Use Case Descriptions

UC1: Creating Account		
Description	First-time visitors can register for an account to become members of the platform. If they manage a charity, they can apply to create a charity and become an administrator for it in addition to their account. Upon acceptance from us, a charity on the platform will be created and they will become an administrator for it.	
Pre-Conditions	The user must provide the personal information required for registration.	
Flows	**Basic or Normal Flows**	1. The user opens the homepage of the software and clicks on the sign-up option. 2. The user is redirected to the sign-up page and prompted to enter the required personal information as well as an available username and password and if they manage a charity, in which they can select an existing charity in the database or set up a new charity with the required information. 3. The system verifies whether all information entered is acceptable. 4. Once the system has approved the registration, the user is redirected to his or her account information.
	Alternative Flows	None.
Post Conditions	The system has created an account for the user and/or a charity that is stored in the database.	
Special Requirements	The system should be able to store a large number of accounts in its database.	
Extension Points	None.	

UC2: Login		
Description	Once the user has registered for an account, he or she can sign in with the username and password.	
Pre-Conditions	The user needs to have registered for an account, during which he or she selected an available username and a valid password.	
Flows	**Basic or Normal Flows**	1. The user opens the homepage of the software and selects the login option. 2. The user enters his or her username and password. 3. The system verifies the user's login credentials. 4. Once the system verifies the credentials match the ones in the database, the user is redirected to his or her account information.
	Alternative Flows	None.
Post Conditions	The system has established a session in which the user's credentials are verified.	
Special Requirements	The device on which the user is accessing the platform must be able to store session information, such as cookies.	
Extension Points	None.	

UC3: Updating Information for Subscribers		
Description	The system updates the news feed of the charity selected with their latest status updates and goal statuses, such as donations received and actions taken.	
Pre-Conditions	The charity administrator must have registered for and logged into their account.	
Flows	**Basic or Normal Flows**	1. The administrator selects a charity and selects “edit or add new update” option. 2. The administrator is prompted to either select an entry to edit or insert new update, then enters new information. 3. The system retrieves from its database the latest status updates posted by the charity as well as its goal statuses. The system renders the page with the information retrieved.
	Alternative Flows	None.
Post Conditions	The system has displayed the homepage of the selected charity with its latest status updates and goal statuses.	
Special Requirements	The system must update its database with the latest data of each charity as soon as a charity posts updates to its account.	
Extension Points	None.	

UC4: Browsing Charities by Category		
Description	The system sorts charities by the mechanism selected by the user, such as distance, total amount of funds raised and type of cause.	
Pre-Conditions	The user needs to have registered for and logged into an account.	
Flows	**Basic or Normal Flows**	1. The user logs into his or her account 2. The user opens the page that lists all charities hosted by the platform 3. The user clicks on the “sort charities” button 4. In the drop-down menu, the user selects the mechanism, such as distance and the total amount of funds raised 5. The system sorts charities by the mechanism indicated
	Alternative Flows	None
Post Conditions	The charities listed have been sorted by the mechanism indicated.	
Special Requirements	The system needs to utilize an efficient algorithm to sort and categorize data for each user.	
Extension Points	None	

UC5: Selecting Charities to Donate		
Description	The user can make a donation to a charity via a third-party payment processor.	
Pre-Conditions	The user must be logged into their account and have a valid payment method, such as a credit card or a bank account.	
Flows	**Basic or Normal Flows**	1. The user selects a charity from search results. 2. The user clicks on the “donate” button and is prompted to enter the amount to donate. An option to make recurring donations, such as weekly or monthly, is also shown. 3. Once the user has confirmed the amount and frequency, if that option is selected, he or she is prompted to enter the details of the preferred payment method, such as a credit card number or bank account number. 4. The system forwards the payment information to the third-party payment processor. 5. The payment processor verifies the payment information. The user is alerted if the transaction is declined. If the transaction is approved, the payment processor sends back a confirmation to the donation platform. 6. The donation platform saves the payment method and displays it, along with the receipt, to the user.
	Alternative Flows	None.
Post Conditions	The system has saved and established a donation as per the user’s request.	
Special Requirements	The device on which the user is accessing the platform must be able to transmit data through an encrypted connection.	
Extension Points	None.	

UC6: Checking Custom Newsfeed		
Description	Once a registered user has subscribed to one or more charities' status updates, they can view all latest updates from such charities on the newsfeed on their account homepage.	
Pre-Conditions	The user must have subscribed to one or more charities' status updates, in addition to having registered for and logged into an account.	
Flows	**Basic or Normal Flows**	1. The user logs into his or her account with the credentials. 2. The system retrieves from its database the list of charities that the user has subscribed to. 3. The system retrieves the status updates of each subscribed charity and combines them into a newsfeed. 4. The system renders the user's homepage with the combined newsfeed and displays it to the user.
	Alternative Flows	None.
Post Conditions	The system has rendered the user's homepage with a newsfeed that contains status updates from all subscribed charities.	
Special Requirements	The system must support various display dimensions.	
Extension Points	None.	

UC7: Visualizing Donations and Interactions		
Description	The charity administrator can view a graphical representation of the donations the charity received.	
Pre-Conditions	The administrator must have registered for and logged into an account.	
Flows	**Basic or Normal Flows**	1. The administrator logs into his or her account. 2. The administrator selects a charity that he or she manages and clicks on the donation statistics option. 3. The system retrieves from its database the donation statistics of the charity, such as the total amount of funds it has raised and the number of donors it has. The system proceeds to generate graphical representations of the statistics and renders a page with the graphs displayed. 4. The system redirects the user to the rendered page.
	Alternative Flows	None.
Post Conditions	The system has rendered a page that displays the selected charity's donation statistics in a graphical format.	
Special Requirements	The system must support various display dimensions.	
Extension Points	None.	

13.4 Schedule Tracking

Artifact or Deliverable	Who (individual and team)	Estimated (hours)	Actual (hours)	Difference (hours)
SPMP	David Bravo	2	2	0
	Qilei Cai	3	3	0
	Tasnim Nehal	3	3	0
	Yikai Wang	3	1	2
	Summary for entire team	11	9	2

Artifact or Deliverable	Who (individual and team)	Estimated (hours)	Actual (hours)	Difference (hours)
SRS - Final	David Bravo	2	2	0
	Qilei Cai	2	3	1
	Tasnim Nehal	2	2	0
	Yikai Wang	2	2	0
	Summary for entire team	8	9	1

Artifact or Deliverable	Who (individual and team)	Estimated (hours)	Actual (hours)	Difference (hours)
SDD - Initial	David Bravo	4	3	1
	Qilei Cai	3	5	2
	Tasnim Nehal	4	4	0
	Yikai Wang	3	5	2
	Summary for entire team	14	17	3

Artifact or Deliverable	Who (individual and team)	Estimated (hours)	Actual (hours)	Difference (hours)
SDD Final	David Bravo			
	Qilei Cai			
	Tasnim Nehal			
	Yikai Wang			
	Summary for entire team			

Cumulative

Who (individual and Team)	Estimated (hours)	Actual (hours)	Difference (hours)
David Bravo	8	7	1
Qilei Cai	8	11	3
Tasnim Nehal	9	9	0
Yikai Wang	8	8	0
Summary for entire team	33	35	2

13.5 Defect Tracking

Artifact or Deliverable	Who (individual and team)	Estimated (cases)	Actual (cases)	Difference (cases)
SPMP	David Bravo	2	2	0
	Qilei Cai	2	1	1
	Tasnim Nehal	1	1	0
	Yikai Wang	3	2	1
	Summary for entire team	8	6	2

Artifact or Deliverable	Who (individual and team)	Estimated (cases)	Actual (cases)	Difference (cases)
SRS - Final	David Bravo	2	1	1
	Qilei Cai	2	1	1
	Tasnim Nehal	2	1	1
	Yikai Wang	2	2	0
	Summary for entire team	8	5	3

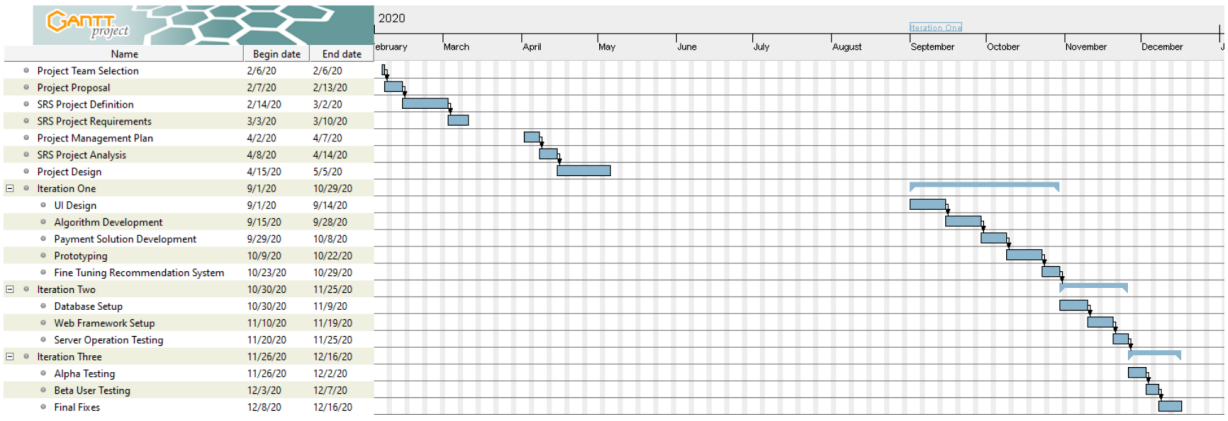
Artifact or Deliverable	Who (individual and team)	Estimated (cases)	Actual (cases)	Difference (cases)
SDD - Initial	David Bravo	4	3	1
	Qilei Cai	4	5	1
	Tasnim Nehal	4	2	2
	Yikai Wang	4	2	2
	Summary for entire team	16	12	4

Artifact or Deliverable	Who (individual and team)	Estimated (cases)	Actual (cases)	Difference (cases)
SDD - Final	David Bravo			
	Qilei Cai			
	Tasnim Nehal			
	Yikai Wang			
	Summary for entire team			

Cumulative

Who (individual and team)	Estimated (cases)	Actual (cases)	Difference (cases)
David Bravo	8	6	2
Qilei Cai	8	7	1
Tasnim Nehal	7	4	3
Yikai Wang	9	6	3
Summary for entire team	32	23	9

13.6 Project Schedule



Name	Begin date	End date
Project Team Selection	2/6/20	2/6/20
Project Proposal	2/7/20	2/13/20
SRS Project Definition	2/14/20	3/2/20
SRS Project Requirements	3/3/20	3/10/20
Project Management Plan	4/2/20	4/7/20
SRS Project Analysis	4/8/20	4/14/20
Project Design	4/15/20	5/5/20
Iteration One	9/1/20	10/29/20
UI Design	9/1/20	9/14/20
Algorithm Development	9/15/20	9/28/20
Payment Solution Development	9/29/20	10/8/20
Prototyping	10/9/20	10/22/20
Fine Tuning Recommendation System	10/23/20	10/29/20
Iteration Two	10/30/20	11/25/20
Database Setup	10/30/20	11/9/20
Web Framework Setup	11/10/20	11/19/20
Server Operation Testing	11/20/20	11/25/20
Iteration Three	11/26/20	12/16/20
Alpha Testing	11/26/20	12/2/20
Beta User Testing	12/3/20	12/7/20
Final Fixes	12/8/20	12/16/20