
## Computer Science and Engineering

---

# The Givers

# System Requirements Specification (SRS)

**Version 1.3**

Document Number: SRS-001

Project Team Number: B01, Section A

Project Team Members: David Bravo (db3454), Qilei Cai (qc542), Tasnim Nehal (tn1078), and Yikai Wang (yw3193)

# REVIEW AND APPROVALS

| Team Members | Function (Author, Reviewer, Approval) | Date | Signature |
|---|---|---|---|
| David Bravo | Author | September 22, 2020 | David Bravo |
| Qilei Cai | Author | September 22, 2020 | Qilei Cai |
| Tasnim Nehal | Author | September 22, 2020 | Tasnim Nehal |
| Yikai Wang | Author | September 22, 2020 | Yikai Wang |
| Professor Strauss | Reviewer | September 22, 2020 | Professor Strauss |
| | | | |
| | | | |

# REVISION LEVEL

| Date | Revision Number | Purpose |
|---|---|---|
| March 1, 2020 | Version 1.0 | Initial Release |
| March 10, 2020 | Version 1.1 | Addition of Sections 6-8 |
| April 11, 2020 | Version 1.2 | Fixing defects and addition of Sections 9 |
| September 22, 2020 | Version 1.3 | Fixing defects and updating dates |
| | | |
| | | |
| | | |

# TABLE OF CONTENTS

# 1. DOCUMENT PURPOSE

## 1.1 Purpose

This System Requirements Specification document aims at defining the functions, services and operational constraints of The Givers in accurate details. Multiple aspects of the project, including objectives, market analysis and functionality will be covered. The document intends to address the charities and donors considering joining the platform, as well as prospective sponsors of the project.

# 2. INTRODUCTION

## 2.1 Scope

The Givers will be a non-profit fundraising platform for charities to advocate for their causes and for donors to chip in to support the causes they resonate with. The core functionality includes an all-inclusive donation platform, custom charity ranking, charity categorization, marketing and analytics and client incentives.

The platform will streamline donations by enabling users to donate to any charity hosted on the platform via a single account registered on the platform. To narrow a multitude of hosted charities down to a selected few, charities will be ranked based on user preferences, with an emphasis on the location of the charity (local charities are highlighted to users). The platform will also enable users to categorize and view multiple charities by type and motivation.

On the marketing front, the platform will provide a news feed to keep users updated on the operations of charities, in addition to visualization on the charities' donation statistics. As an incentive to the clients, bonuses will be distributed to charities for reaching fundraising goals or making significant social impact.

## 2.2 Identification

The Givers System Requirements Specification (SRS) Version 1.3, released on September 22, 2020

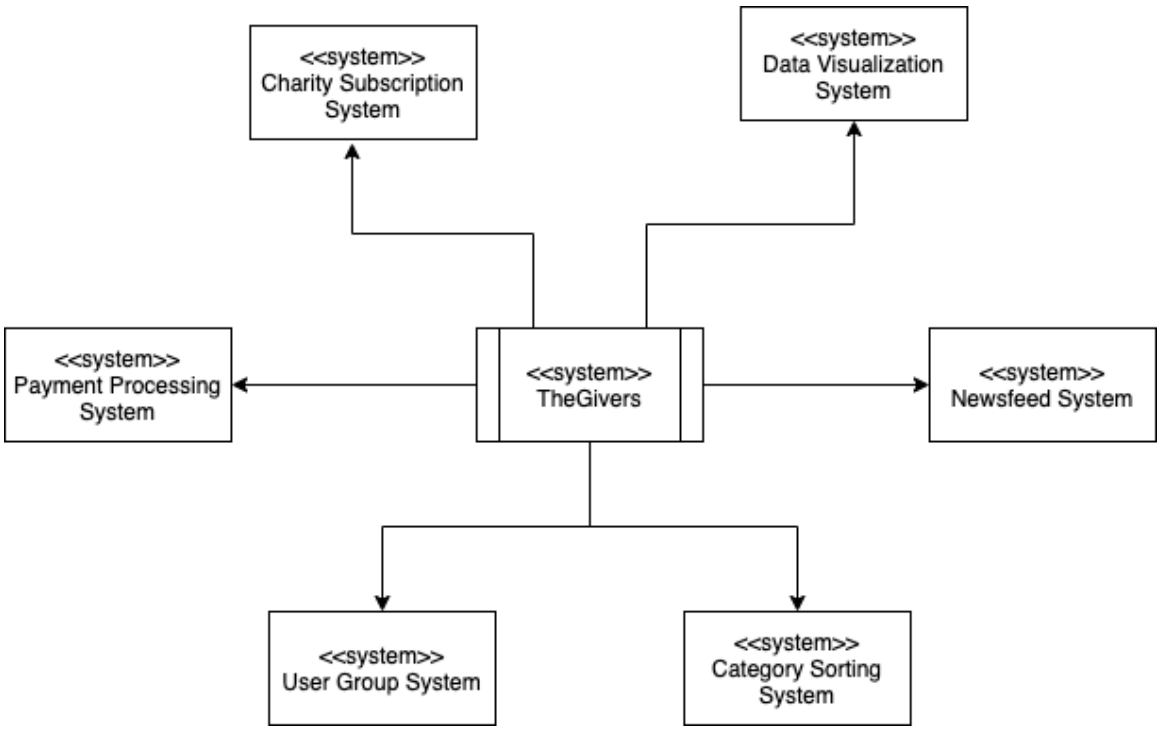
## 2.3 Bounds

The Givers platform will not provide functionality for tax form generation nor risk assessment. The system only provides functionality for user donations, public charity news feed, user categorization, relevant charity suggestions for the user, and charity goal updates.

## 2.4 Objectives

The system is a high-priority project following an agile development method. The project is based on an incremental life cycle. The first incremental release is expected to be delivered in September 2020.

## 2.5 Context Diagram



## 2.6 Additional Descriptive Items

### Product functions:

- One Donation Platform: There will be a single platform where only one account is needed to donate to several different charities rather than going to multiple charity sites.
- Custom Charity Ranking: Each charity is ranked based on user preferences, with emphasis on location (local charities are weighed higher).
- Charity Categorization: The user also has the ability to categorize and view multiple charities by type and motivation (e.g. animal rights and education advocacy).
- Marketing and Analytics: Provide news feed about charities and visuals for charity donations.
- Client Incentives: Bonuses will be given to charities for reaching certain goals such as money raised or social impact. This would encourage more charities to join the platform

### User Characteristics:

The clients will be charity organizers who will be using The Givers as a means to plug their own charity. The users are regular individuals who are not members of any charity organization but would like a single platform to donate to charities to support them.

### Constraints:

Since The Givers will be a non-profit platform there will be constraints on how the system gathers funding to maintain servers and provide bonuses to client charities. The system will also process large amounts of private data such as names and payment information meaning the system will have to maintain constant safety and security. This will require regular audit checks to make sure the system is secure and up to date.

### Assumptions and Dependencies:

This system will be a platform that can work on any web browser and will not be tied to any specific hardware meaning it will work on any mobile or computer device. AWS will be used to host the platform.

# 3. GLOSSARY

| Term | Definition |
|---|---|
| User | A person using our application to donate to charities. |
| Stakeholder | Anyone invested in our platform. |
| Non-profit | Contributions (e.g. donations) are used not for profit but for charitable causes. Revenues can also be used towards further development of The Givers platform. |

# 4. REFERENCE DOCUMENTS

- Team B01, “The Givers”, Project Proposal, Version 1.0, September 15, 2020

# 5. BUSINESS REQUIREMENTS

## 5.1 Technology

Cloud Services: We plan to deploy our projects components (Frontend, Backend API, and Database) on cloud servers. Due to the high expected overhead on maintaining physical servers to allow for our project to be readily available to mobile, tablet, and desktop users around the country, using cloud services significantly decreases the cost of maintaining servers, compared to using physical servers. As the average cost of serving one user drops, we will be able to provide our service on a larger scale with the same amount of funds, namely connecting more donors to charities. The efficiency of cloud services will raise the productivity of the project without incurring extra costs.

Encrypted Storage & Communication: Privacy is not always guaranteed when donors communicate with charities. Some of the outdated means of communications adopted by charities make donors’ privacy vulnerable to cybercriminals. The trust of customers is vital to any payment platform. Customers expect their information to be transmitted securely through the platform. Therefore, we aim at gaining customers’ trust by building a secure communications path between donors and charities. As customers’ trust of the platform accumulates so will our perceived reliability, allowing the platform to gain more frequent donors and yield more donations for the charities hosted.

## 5.2 Economics

As a non-profit organization funding will be required to pay employees, maintain server costs, and provide bonuses to charities.

This funding will be raised by acquiring investors or charging charities a subscription fee to join the system just so the system can continue running. There could also be a small fee added on to every donation which would contribute entirely to running the system.

## 5.3 Regulatory and Legal

There will be constraints on how money is distributed as this system will be a non-profit platform.

## 5.4 Market Considerations

User preferences - Users favor different charities so The Givers platform should allow categorizations for charities and utilize recommendation algorithms for users.

Lack of a single donation platform - It is cumbersome to donate to multiple charities now as each one requires reentering payment information or different login credentials and there is no application that lets you search for types of charities. This is the market The Givers is going to break into by providing a single cloud based platform that would solve all these issues.

## 5.5 Risks and Alternatives

| **Business Risk** | Insufficient active users |
|---|---|
| Description | A sufficient number of active users is critical to the success of the business model, but there may be times when the number of active users is lower than satisfactory. |
| Probability | Medium |
| How discovered | The organization will constantly monitor the number of active users within a particular time frame, such as the past week or month. |
| Responsible Party Status | The organization’s marketing department will be tasked with growing the platform’s user base. |
| Mitigation Plan | The marketing department will develop marketing campaigns to make the platform better known. If necessary, monetary incentives could |

| | |
|---|---|
| | be awarded to new users or currently active users to keep them engaged. |
| | |
| **Operational Risk** | Insufficient cash flow |
| Description | In case there is no significant number of donations made on the platform, the processing fees made from donations may be insufficient to support the organization's daily operations. |
| Probability | Medium |
| How discovered | The organization will periodically review the state of finances and analyze if the cash flow is sufficient to pay for the platform's expenses. |
| Responsible Party Status | The organization's Chief Operating Officer will work with the organization's accounting department to identify any potential shortages in the cash flow. |
| Mitigation Plan | In case of an insufficient cash flow, the Chief Operating Officer will raise the processing fees assessed on each donation to the extent that can resume a sufficient cash flow for the organization. |
| | |
| **Technology Risk** | System outage due to software or hardware errors |
| Description | Defects in the platform's code or the hardware on which the platform runs can trigger system outages. |
| Probability | High |
| How discovered | The organization will develop software that monitors the operational status of the platform, such as whether all services are online and accessible to all users. |
| Responsible Party Status | The organization's site reliability engineers will be delegated with getting the services back online. |
| Mitigation Plan | The organization will train a team of site reliability engineers exclusively responsible for such emergencies. |

| | |
|---|---|
| | During recruiting, candidates for the job are informed that they have no fixed work hours; instead, they should expect to report for duty at any time of the day in case of a system outage. At the time of a system outage, the engineers are immediately dispatched to the work site to start fixing the issue and get the services back online as soon as possible. |
| | |
| **Economic Risk** | Rising minimum wage |
| Description | In case of new regulations raising the minimum wage, the cost of staffing may exceed the organization's budget. |
| Probability | Low |
| How discovered | The organization will stay up-to-date on all new regulations announced by authorities and address the issues that may arise regarding the organization's operations. |
| Responsible Party Status | The organization's human resources department will work on addressing the issue. |
| Mitigation Plan | The human resources department will look into optimizing the organization's labor force. Redundant or wasteful use of staffing, in areas non-essential to the organization's operations, will be reduced to lower the total cost of staffing to a level within the budget. |

## 5.6 Human Resources and Training

Since our team is composed of highly skilled developers with experience developing applications, there should be minimal training needed. Furthermore, because of the team's small size, risks in conflicts are small and each team member can also tackle human resource tasks.

# 6. USER REQUIREMENTS (DESCRIPTIVE FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENTS)

## 6.1 Functional Descriptive Detailed Requirements

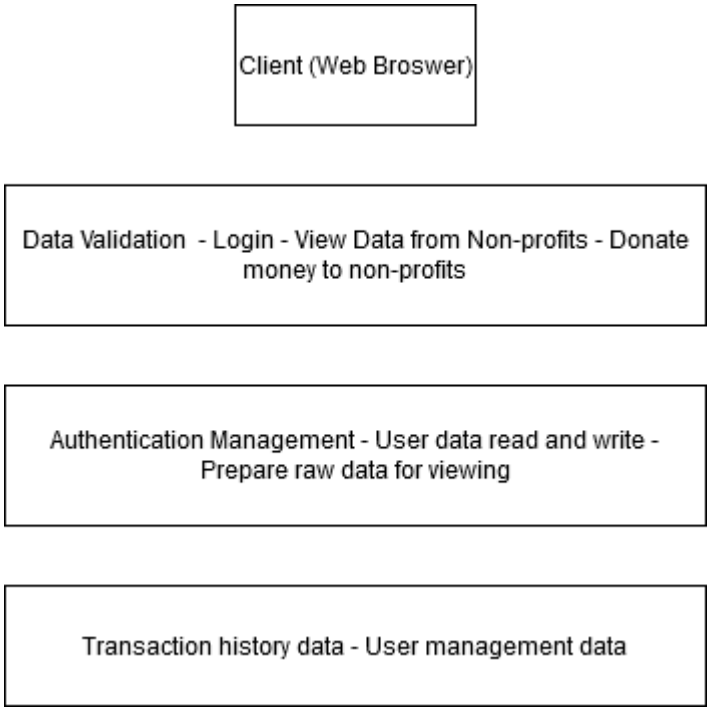
- Functional Requirement 1 (SR1): Account Creation
  - The system shall enable users to sign up for a new account with their personal information.
    - SR1.1: Users can select the sign-up button and fill in information such as their name and birthday.
    - SR1.2: Users can modify their account information 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.

## 6.2 Non-Functional Descriptive Detailed Requirements

- Non-Functional Requirement 1 (SR1): Account Creation
    - The system will securely allow the user to enter their personal information.
- Non-Functional Requirement 2 (SR2): Login
    - The system will hash sensitive data like passwords before performing a database query.
- Non-Functional Requirement 3 (SR3): Updating Information for Subscribers
    - The system will only be unavailable when site maintenance is required and will be accessible to users with little downtime otherwise allowing updates to be made regularly.
- Non-Functional Requirement 4 (SR4): Browsing Charities by Categories
    - The system will provide a ranking algorithm that will help users find charities that match their needs.
- Non-Functional Requirement 5 (SR5): Selecting Charities to Donate
    - Payments will be securely processed by a third-party payment processor API
- Non-Functional Requirement 6 (SR6): Checking Custom Newsfeed
    - The system will be delivered in a mobile-friendly version in addition to a desktop version allowing newsfeeds to be viewed on any device.
- Non-Functional Requirement 7 (SR6): Visualizing Donations and Interactions
    - The system will provide clear and concise visuals allowing for easy navigation.

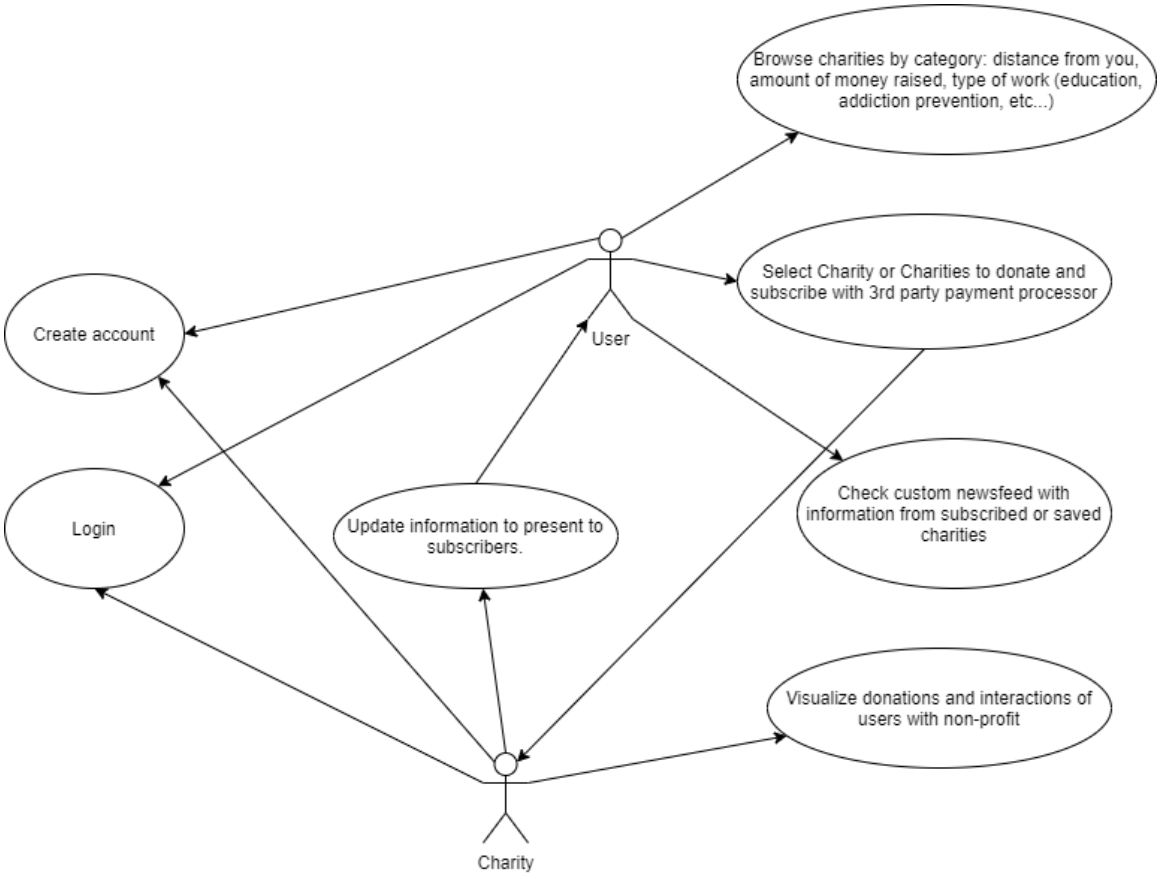
# 7. SYSTEM ARCHITECTURE



# 8. DETAILED SYSTEM REQUIREMENTS – USE CASES

## 8.1 Requirement Use Cases

### *8.1.1 Use Case Diagrams*



### *8.1.2 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.<br>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.<br>3. The system verifies whether all information entered is acceptable.<br>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.<br>2. The user enters his or her username and password.<br>3. The system verifies the user's login credentials.<br>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.<br>2. The administrator is prompted to either select an entry to edit or insert new update, then enters new information.<br>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<br>2. The user opens the page that lists all charities hosted by the platform<br>3. The user clicks on the “sort charities” button<br>4. In the drop-down menu, the user selects the mechanism, such as distance and the total amount of funds raised<br>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.<br>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.<br>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.<br>4. The system forwards the payment information to the third-party payment processor.<br>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.<br>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. | |

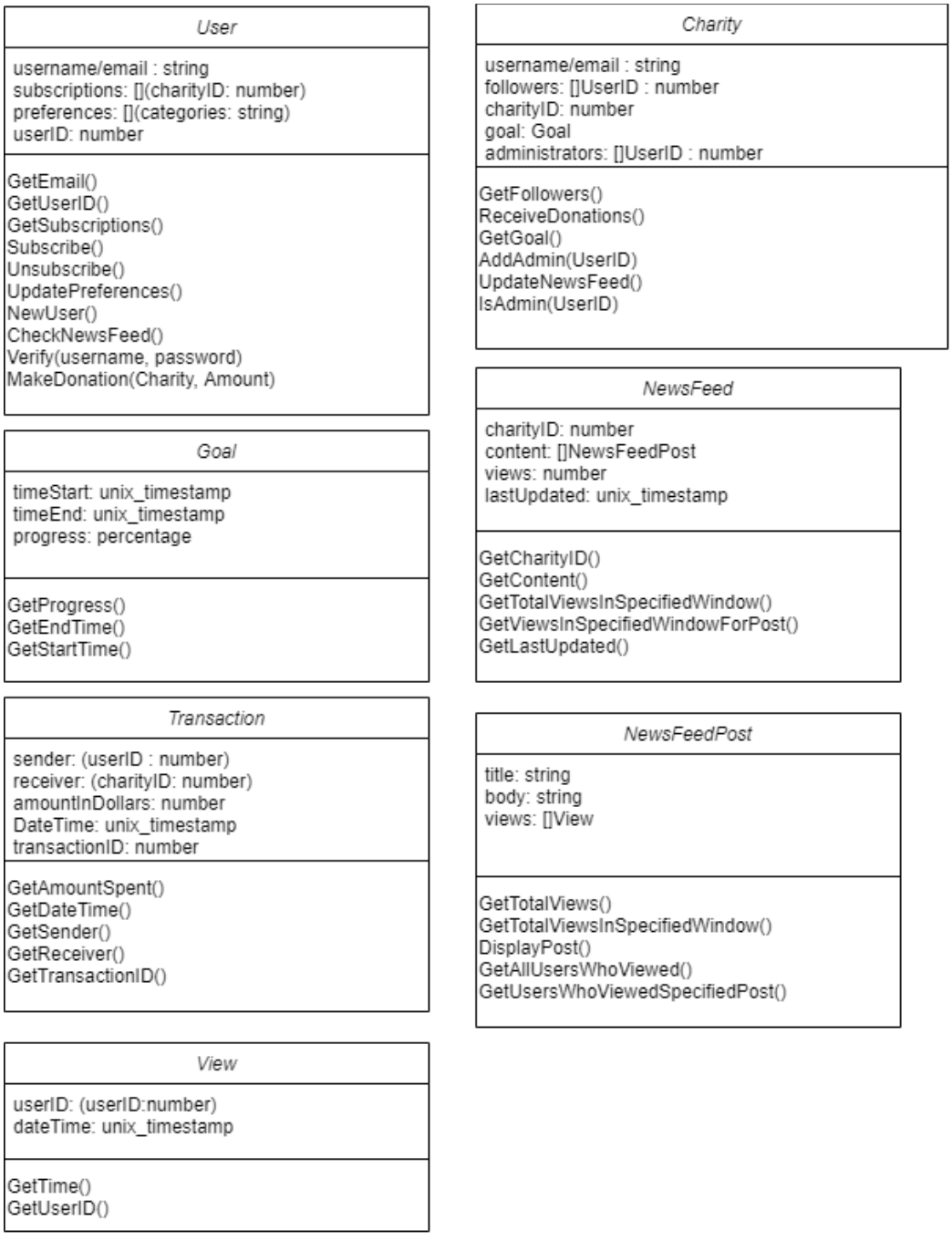
| UC6: Checking Custom Newsfeed | | |
|---|---|---|
| **Description** | Once a registered user has subscribed to one or more charities' status updates, they can view all latest updates from such charities on the newsfeed on their account homepage. | |
| **Pre-Conditions** | The user must have subscribed 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.<br>2. The system retrieves from its database the list of charities that the user has subscribed to.<br>3. The system retrieves the status updates of each subscribed charity and combines them into a newsfeed.<br>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.<br>2. The administrator selects a charity that he or she manages and clicks on the donation statistics option.<br>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.<br>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. | |

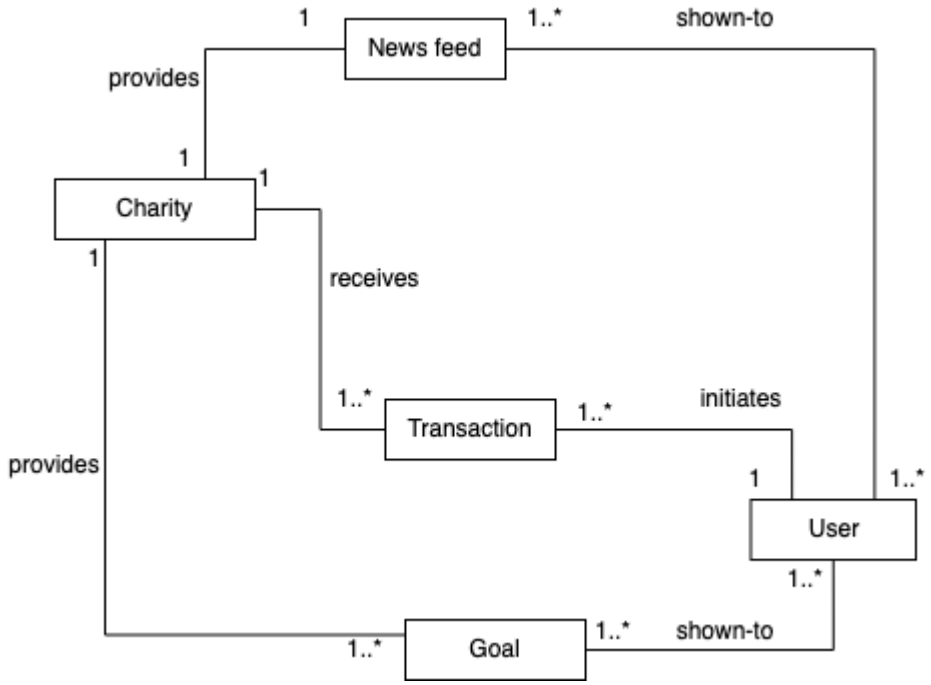
# 9. SYSTEM MODEL (UML)

## 9.1 Static - Class Diagrams

### Class Diagram



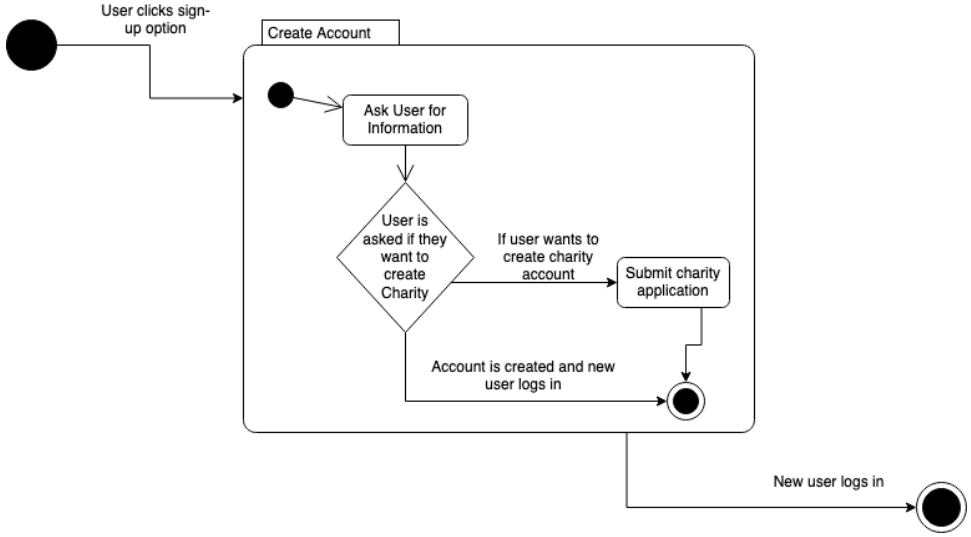
### Class Interaction Diagram



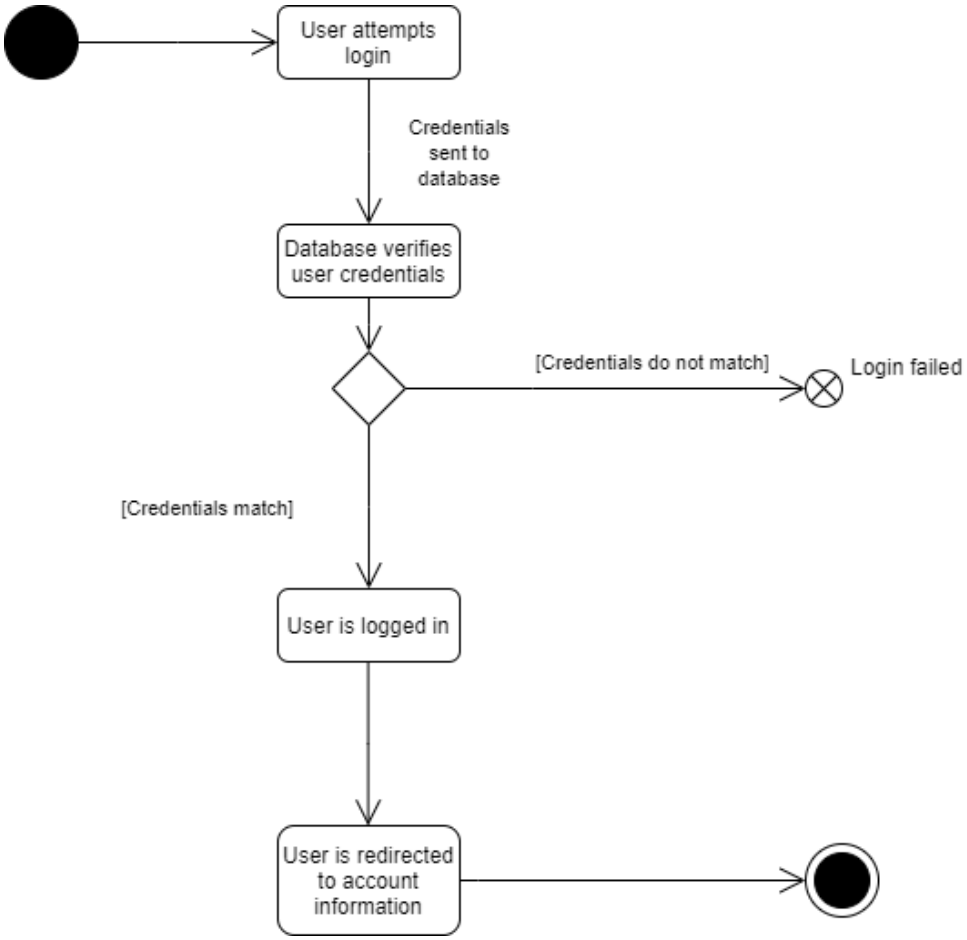

## 9.2 Dynamic - Behavioral Models

### State Diagrams

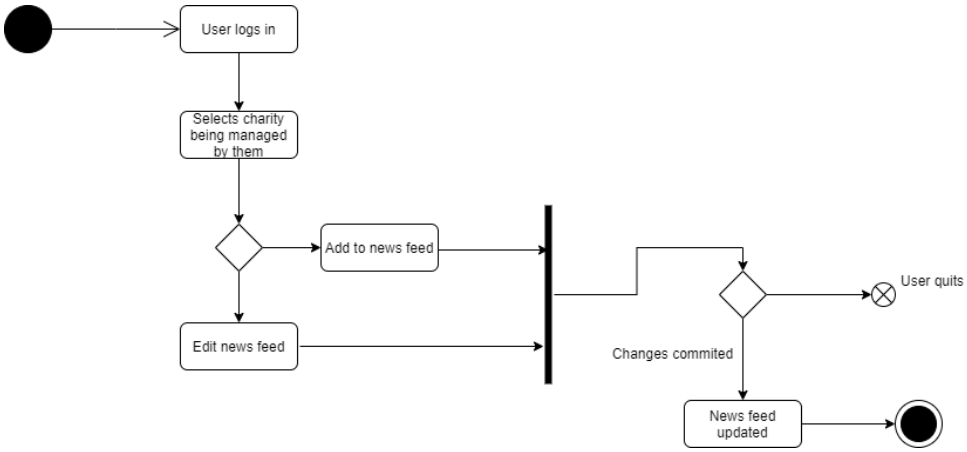
*UC1: Creating Account*



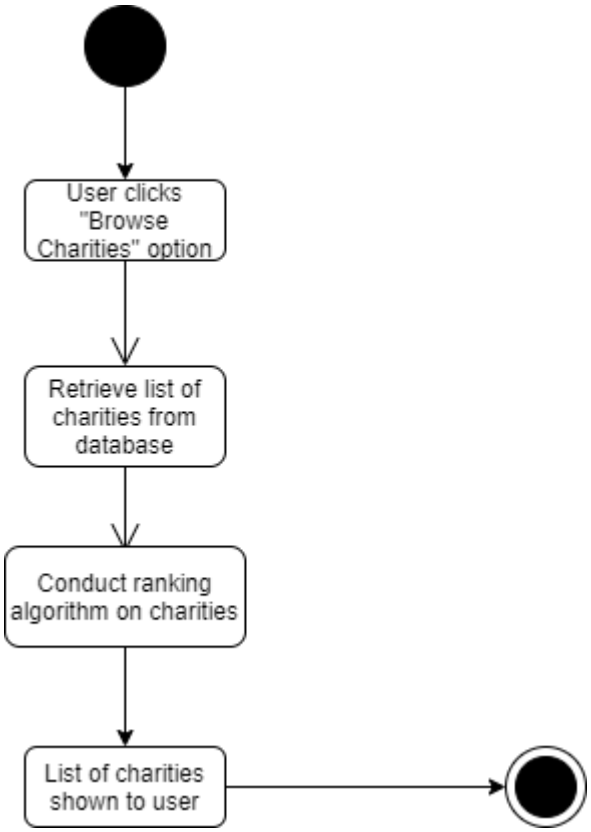
*UC2: Login*



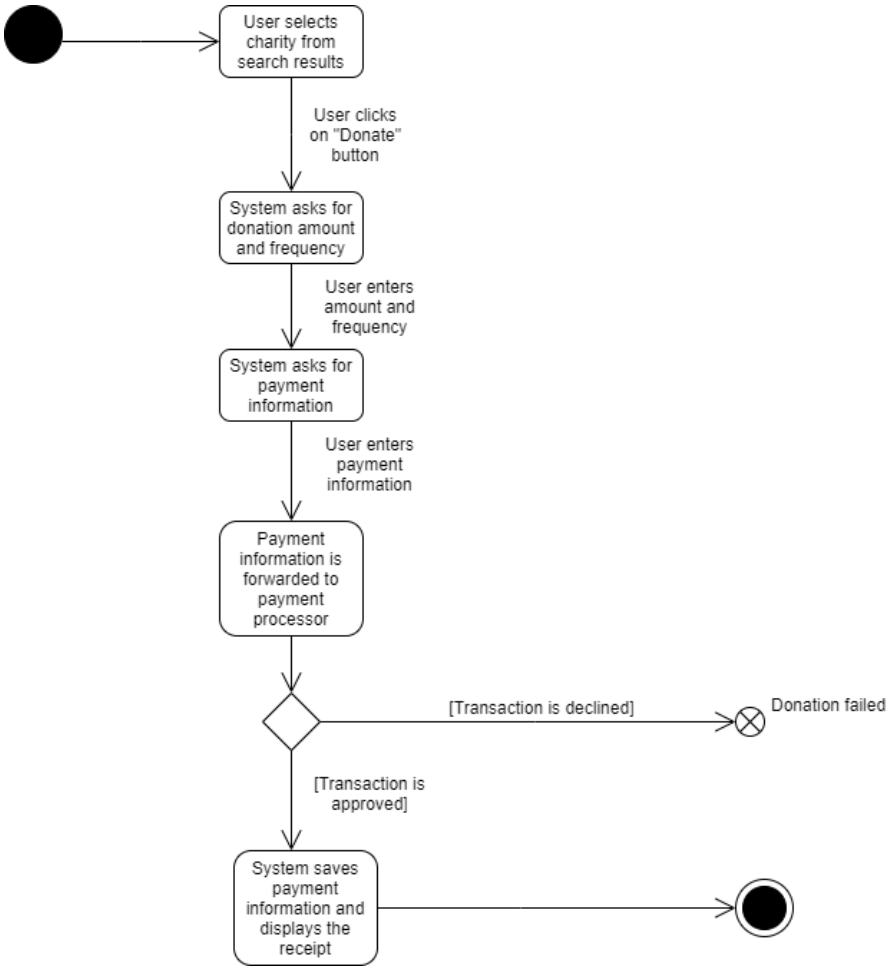

*UC3: Updating Information to Users*



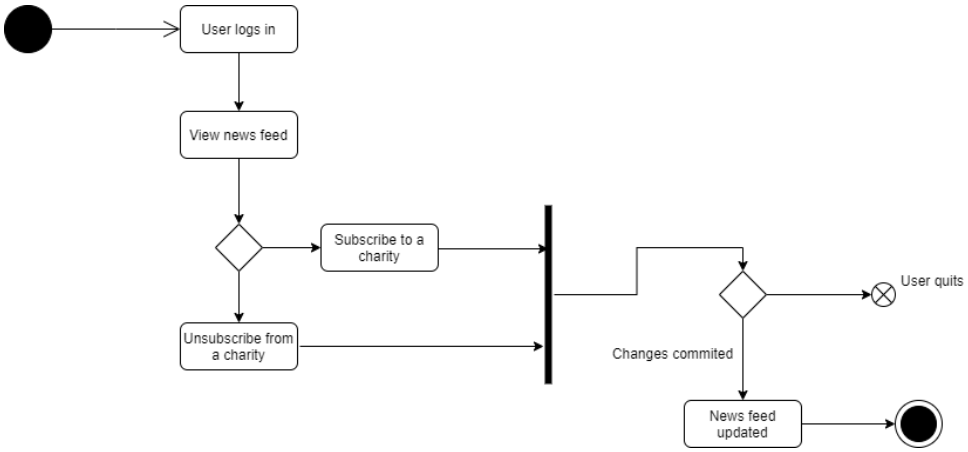
*UC4: Browsing Charities by Category*



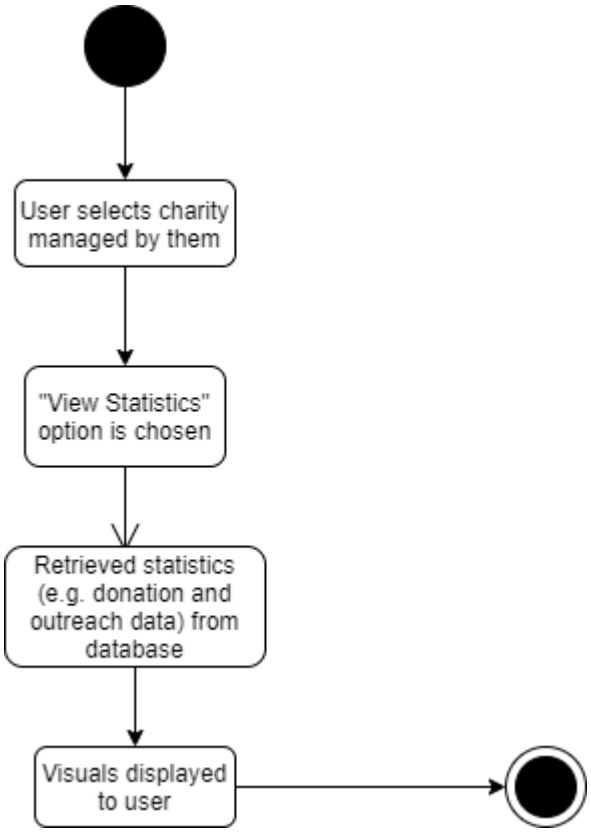
*UC5: Selecting Charities to Donate*




*UC6: Checking Custom Newsfeed*

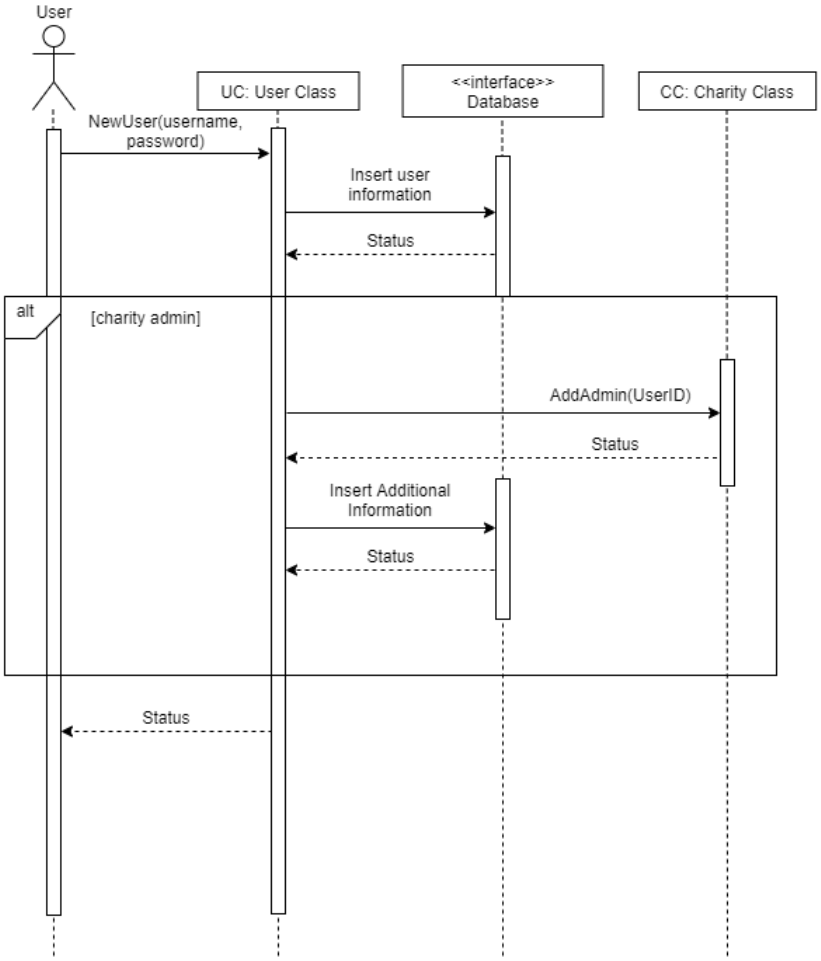


*UC7: Visualizing Donations and Interactions*

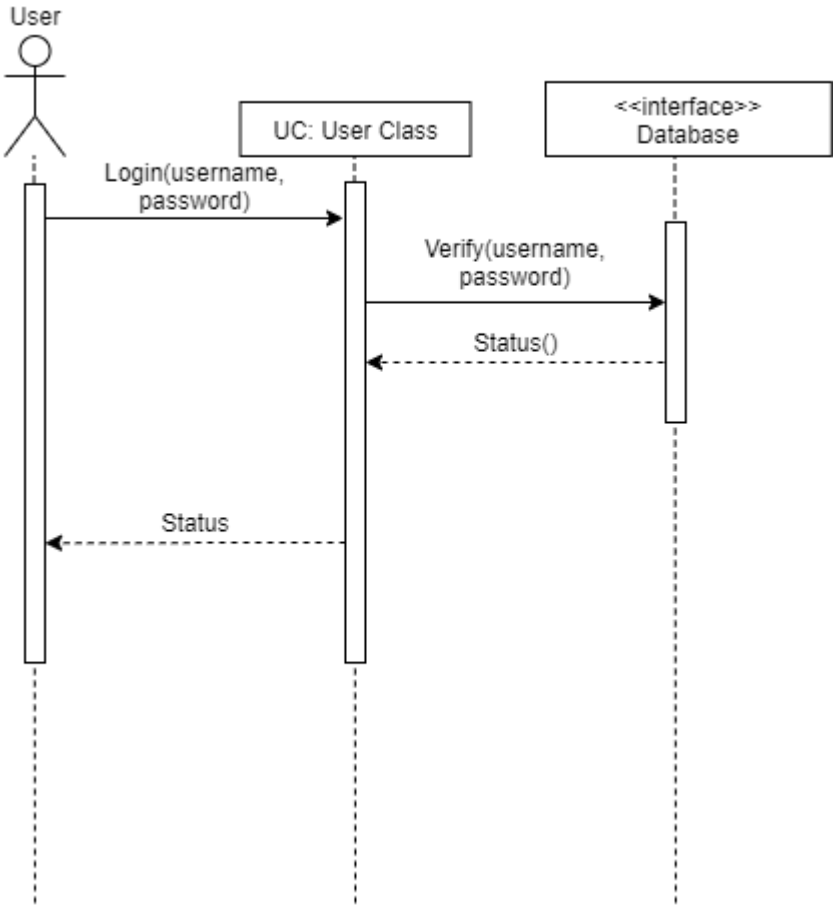


## Sequence Diagrams

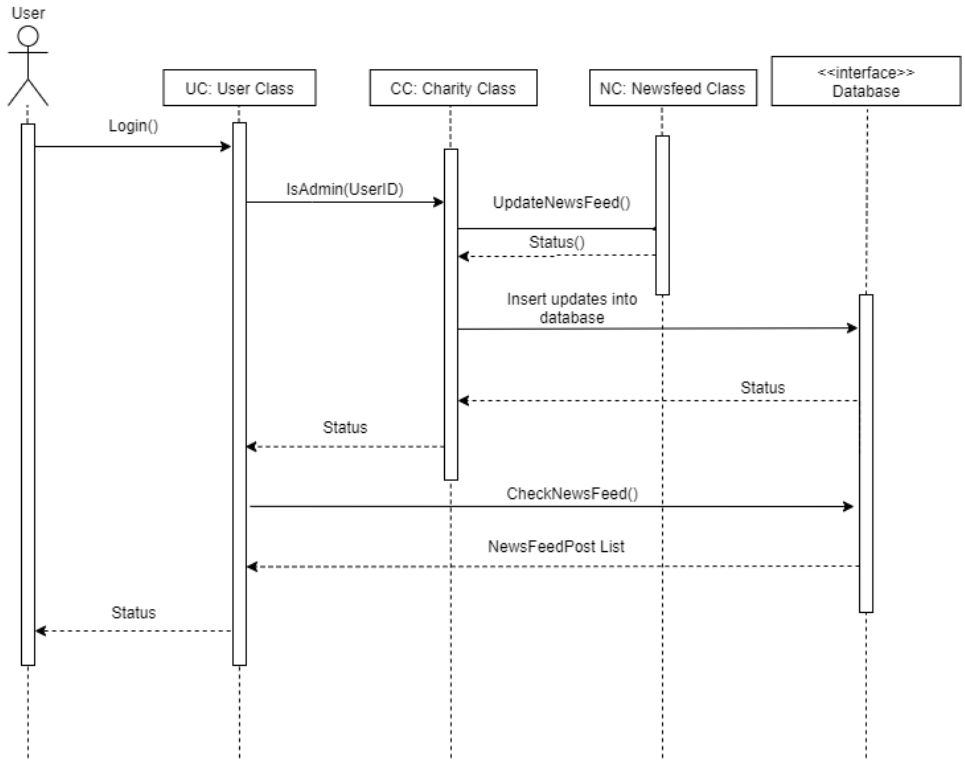
*UC1: Creating Account*



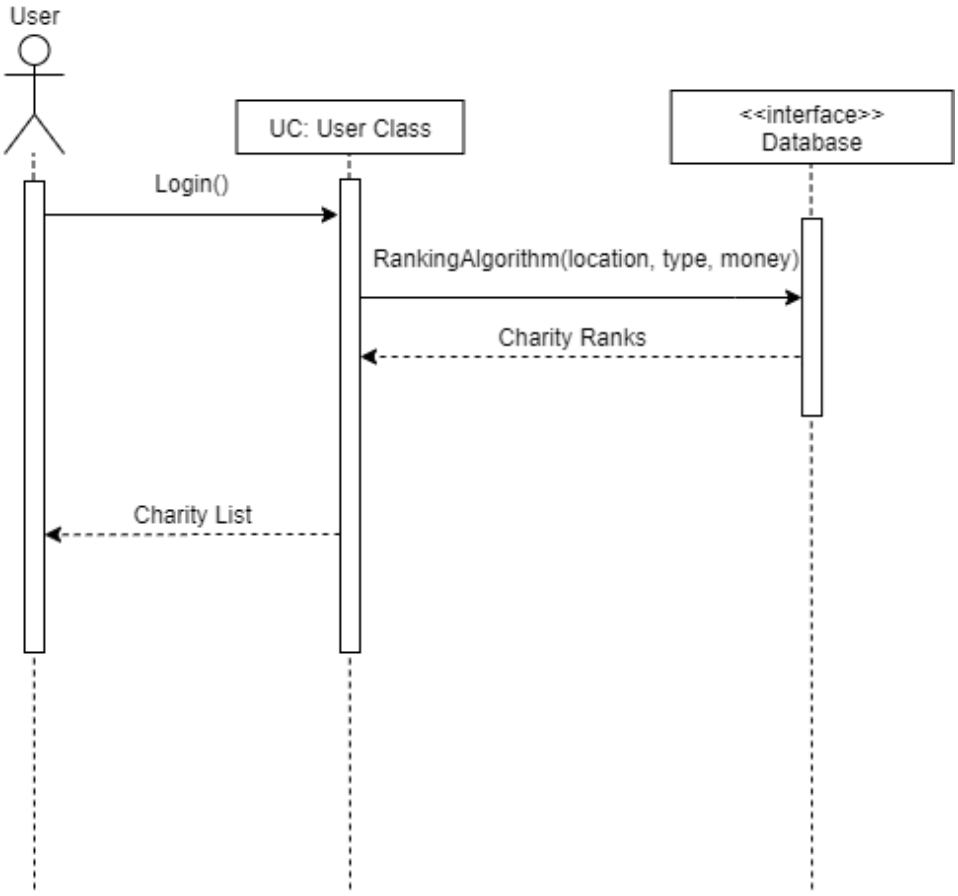
*UC2: Login*


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

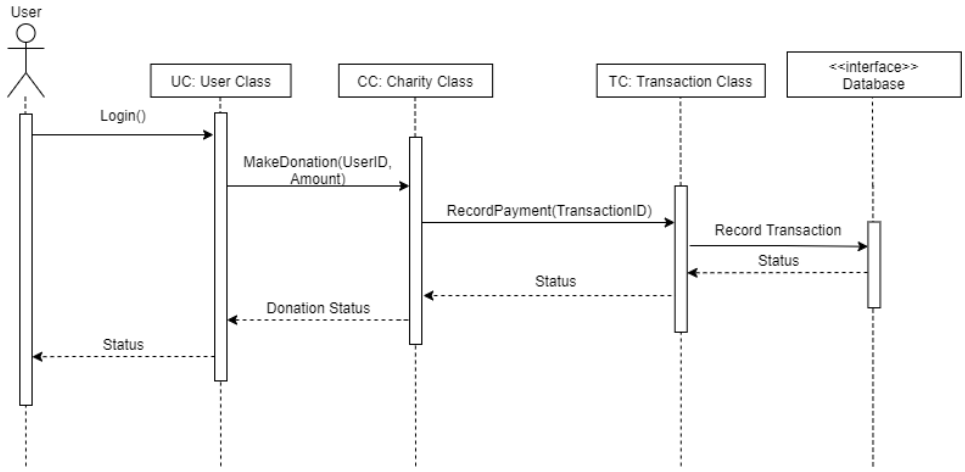
*UC3: Updating Information to Users*



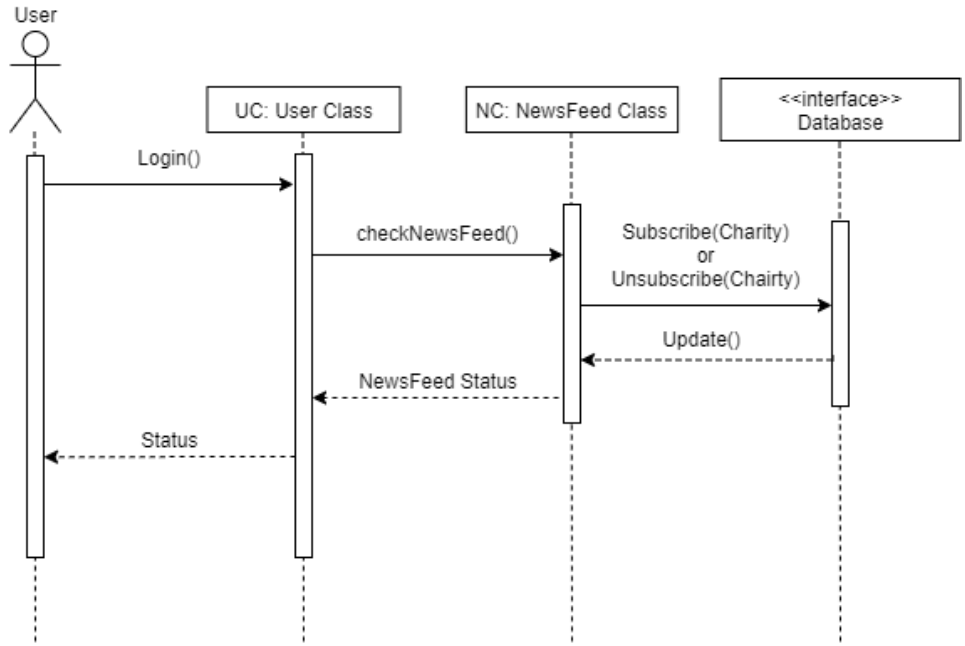
*UC4: Browsing Charities by Category*



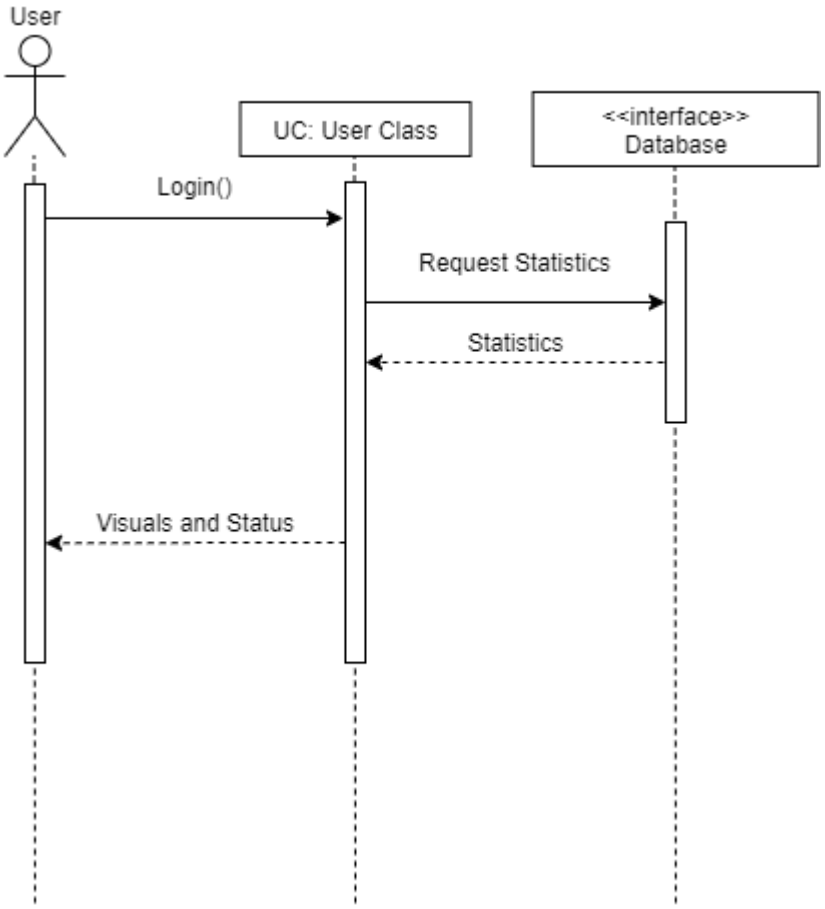
*UC5: Selecting Charities to Donate*




*UC6: Checking Custom Newsfeed*



*UC7: Visualizing Donations and Interactions*




## 10. EVOLUTION OF THE SRS

The initial version of this document outlines the specifications of the project. Over the course of the developing the project, changes might be made to the user or system requirements. Other changes might be made due to technical difficulties or ambiguities in communications. The SRS will change in accordance with the evolution of the development process. All changes will be tracked, recorded and approved prior to being incorporated into the document. The version number will be updated based off of the extent of changes compared to the previous version.

# 11. RATIONALE

# 12. NOTES

# 13. APPENDICES

## 13.1 System Test Plan Requirements

There are a few key user workflows that we want to ensure are working correctly and make sure users do not have access to services reserved for non-profit account administrators. The simulated user workflow looks like this:

- Sorting non-profits by distance from user, type of work (education, addiction prevention, etc.), amount of money raised, number of people they impacted, and by name.
- Browsing custom news feed that is based on subscriptions to non-profits.
- Adding a new subscription.

When non-profit account administrators log in, they will want to perform the following possible workflows:

- View charts and diagrams detailing donation trends within a given time period
- Update new information to share with users through the newsfeed functionality
- Use visualization tools to view the frequency and location of the users donating to the charity

The following list below indicates how we plan to test workflows and functions:

- To test the sorting methods that are exposed to users, non non-profit account administrators, they should be unit tested individually.
- To test if users can only see news from subscribed non-profits the function that produces the newsfeed should be unit testing to simulate a user subscribing and updating their newsfeed to make sure users do not see any unexpected information.

- Subscribing and unsubscribing should also be unit tested by simulating subscription and unsubscription.
- When non-profit account admins attempt to visualize data from interactions and donations in the platform, there must be some data transformations which need to be unit tested for all transformations.
- Performance testing on data visualizations and presentations should also be done to ensure the user has a good experience with the UI.
- When presenting the data, automated data creation and transformations should also be tested to ensure our presentation layer can handle odd forms of data and preprocessing is robust.
- When non-profit account admins try to update new information to share to their users, the workflow should be simulated to test that users can successfully see new content when admins publish it.
- To perform customer testing, we can mock a non-profit on the platform and ask friends or team members to test the different workflows mentioned in the platform.

Once all tests pass, acceptance and system testing is complete.

## 13.2 Qualification Provisions

The document is based on software. To ensure the document is correct, it should reflect correctly the functionality and performance of software. First, each team member conducts a desk check, or individually checks the document for review. Next, peer review is conducted so that more potential problems or conflicts are caught through collaboration. Then, defects can be identified, with the software understanding gauged, through a walkthrough review. Finally, testing can be done during formal inspections with participants to categorize and solve defects. To ensure the quality of the software, we will conduct component testing, system testing and customer testing. By testing each component and its functional and non-functional requirements as well as being tested by customers, we ensure that the system works and that the potential problems are identified. Since these will be reflected on the document, this ensures the document is correct. By evaluating feedback, we ensure that the use cases are unambiguous, complete and consistent with software since conflicts between requirements will be identified and all requirements are satisfied. Furthermore, due to component testing, requirements can be changed without a huge impact because this will be done during an early stage, and reflects modifiability as such. The impact of each change can also be traced during testing and therefore traceable, and this is also helped by the use of diagrams. Verifiability can be done if the software can be tested; therefore, this is checked during the testing phase. By extensive testing, we ensure stability by catching possible errors.

## 13.3 Requirements Traceability

**Forward Traceability**

| ID | System Requirement | Type | System Architecture | Use Cases |
|---|---|---|---|---|
| SR1 | Account Creation | Feature | Front end | UC1 |
| SR2 | Login | Feature | Database | UC2 |
| SR3 | Updating Information for Subscribers | Feature | Back end, database | UC3 |
| SR4 | Browsing Charities by Categories | Feature | Front end, database | UC4 |
| SR5 | Selecting Charities to Donate | Feature | Front end, database | UC5 |
| SR6 | Checking Custom Newsfeed | Feature | Front end, back end | UC6 |
| SR7 | Visualizing Donations and Interactions | Feature | Front end, database | UC7 |

**Backward Traceability**

| ID | System Requirement | Source Documents | Stakeholders | Elicitation Activity |
|---|---|---|---|---|
| SR1 | Account Creation | SRS, Version 1.3, September 22, 2020 | Individual donors, charity staff | Database Development Meeting |
| SR2 | Login | SRS, Version 1.3, September 22, 2020 | Individual donors, charity staff | Database Development Meeting |
| SR3 | Updating Information for Subscribers | SRS, Version 1.3, September 22, 2020 | Individual donors | Algorithm Development Meeting |
| SR4 | Browsing Charities by Categories | SRS, Version 1.3, September 22, 2020 | Individual donors | UI Design Workshop |
| SR5 | Selecting Charities to Donate | SRS, Version 1.3, September 22, 2020 | Individual donors | Payment Solution Development Meeting |
| SR6 | Checking Custom Newsfeed | SRS, Version 1.3, September 22, 2020 | Individual donors | Algorithm Development Meeting |
| SR7 | Visualizing Donations and Interactions | SRS, Version 1.3, September 22, 2020 | Individual donors | UI Design Workshop |

## 13.4 Schedule Tracking

| Artifact or Deliverable | Who (individual or Team) | Estimated (hours) | Actual (hours) | Difference (hours) |
|---|---|---|---|---|
| SRS—Domain Definition | David Bravo | 4 | 3 | 1 |
| | Qilei Cai | 4 | 3 | 1 |
| | Tasnim Nehal | 4 | 4 | 0 |
| | Yikai Wang | 4 | 4 | 0 |
| | Summary for entire team | 16 | 14 | 2 |

| Artifact or Deliverable | Who (individual or Team) | Estimated (hours) | Actual (hours) | Difference (hours) |
|---|---|---|---|---|
| SRS—Project Requirements | David Bravo | 2 | 1 | 1 |
| | Qilei Cai | 3 | 3 | 0 |
| | Tasnim Nehal | 3 | 3 | 0 |
| | Yikai Wang | 3 | 3 | 0 |
| | Summary for entire team | 11 | 10 | 1 |

| Artifact or Deliverable | Who (individual or Team) | Estimated (hours) | Actual (hours) | Difference (hours) |
|---|---|---|---|---|
| SRS—Complete | David Bravo | 3 | 3 | 0 |
| | Qilei Cai | 3 | 3 | 0 |
| | Tasnim Nehal | 3 | 2 | 1 |
| | Yikai Wang | 3 | 2 | 1 |
| | Summary for entire team | 12 | 10 | 2 |

| Artifact or Deliverable | Who (individual or Team) | Estimated (hours) | Actual (hours) | Difference (hours) |
|---|---|---|---|---|
| SRS—Version 1.3 | 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 |

## Cumulative

| Who (individual or Team) | Estimated (hours) | Actual (hours) | Difference (hours) |
|---|---|---|---|
| David Bravo | 11 | 9 | 2 |
| Qilei Cai | 12 | 12 | 0 |
| Tasnim Nehal | 12 | 11 | 1 |
| Yikai Wang | 12 | 11 | 1 |

## 13.5 Defect Tracking

| Artifact or Deliverable | Who (individual or Team) | Estimated (cases) | Actual (cases) | Difference (cases) |
|---|---|---|---|---|
| SRS—Domain Definition | David Bravo | 4 | 3 | 1 |
| | Qilei Cai | 4 | 4 | 0 |
| | Tasnim Nehal | 4 | 2 | 2 |
| | Yikai Wang | 4 | 2 | 2 |

| Artifact or Deliverable | Who (individual or Team) | Estimated (cases) | Actual (cases) | Difference (cases) |
|---|---|---|---|---|
| SRS—Project Requirements | David Bravo | 3 | 1 | 2 |
| | Qilei Cai | 3 | 2 | 1 |
| | Tasnim Nehal | 3 | 1 | 2 |
| | Yikai Wang | 3 | 2 | 1 |

| Artifact or Deliverable | Who (individual or Team) | Estimated (cases) | Actual (cases) | Difference (cases) |
|---|---|---|---|---|
| SRS—Complete | David Bravo | 4 | 1 | 3 |
| | Qilei Cai | 4 | 3 | 1 |
| | Tasnim Nehal | 4 | 2 | 2 |
| | Yikai Wang | 4 | 2 | 2 |

| Artifact or Deliverable | Who (individual or Team) | Estimated (cases) | Actual (cases) | Difference (cases) |
|---|---|---|---|---|
| SRS—Version 1.3 | David Bravo | 2 | 1 | 1 |
| | Qilei Cai | 2 | 1 | 1 |
| | Tasnim Nehal | 2 | 1 | 1 |
| | Yikai Wang | 2 | 2 | 0 |

**Cumulative**

| Who (individual or Team) | Estimated (cases) | Actual (cases) | Difference (cases) |
|---|---|---|---|
| David Bravo | 13 | 6 | 7 |
| Qilei Cai | 13 | 10 | 3 |
| Tasnim Nehal | 13 | 6 | 7 |
| Yikai Wang | 13 | 8 | 5 |

# 14. INDEX