## San Francisco State University

## SW Engineering CSC648/848 Section 4

**Hopeful Sight**

**Milestone 2 Submission Form**

**Team 3**

Koby Kern – Team Lead, Ali Almusawi – Front End, Jacob Vazquez – Backend Lead, Nathan Gabriel David – Scrum Master, Ethan Zheng – Github Master, Miguel Maurer – Frontend Lead, Miguel Melo – Front/Back End

|  |  |
| --- | --- |
| Version | Date |
| V1 | 10/14/2024 |

**Table of Contents**

1. **Data Definitions V2**
2. **Functional Requirements V2**
3. **UI Mockups and UX Flows**
4. **High Level Architecture, Database Organization**
5. **High Level UML Diagrams**
6. **Key Risks**
7. **Project Management**

**Data Definitions V2**

User

* User ID - used to identify/refer to that specific user (PK - double)
* first\_name (varchar)
* last\_name (varchar)
* Email (varchar)
* Donator - FK (Donator) N
* Donatee - FK (Donatee) N

Donator (is a user)

* has\_donated (bool)
* totalAmountDonated (float)
* Account id - FK (account)

Donatee/recipient (is a user)

* has\_applied\_for\_account (bool)
* has\_claimed (bool)
* Account id - FK (account)
* Monthly\_income(double)
* Average\_household\_income(double)
* Current\_bank\_balance(double)
* Is\_dependent(bool)
* prescription(BLOB)

Account

* Phone Number (smallint - set)
* Address (varchar)
* City (varchar)
* state(varchar)
* zip(SMALLINT)
* location(Truncated)

Donation

* Donation\_id - PK
* Donator\_ID - FK
* Donation Amount  (float)
* Region ID - FK
* Donation Date (datetime)

Region

* Region\_id (big int)
* Region\_name (varchar)
* region\_total(double)
* CSS:
  + Round bar similar to helldivers entity status

Glasses

* Name (varchar)
* Size in mm (int)
* Color (varchar)
* Prescription range (float)
* Image (BLOB)
* Inventory(int)
* description(text)

|  |  |  |
| --- | --- | --- |
| Role | Page | **Database information** |
| Donator | Home, donator page, confirmation, catalog, detail | donation amount, confirmation code, regional |
| Donatee | Prescription form | Glasses description, time sign, time sign out, glasses amount (catalog), confirmation code, regional, authentication code |
| Admin | Database page | All Access |

**Functional Requirements V2**

Requirement R.1 - Donate

R.1.1 - Enables the user to donate to the charity, which allocates glasses to people in need.

R.1.2 - The user will be able to donate to a specific region or have their donation be randomly assigned to one.

User Story Reference: 00

Priority - 1

Requirement R.2 - Get Charity Information

R.2.1 - Provides the user with information about the charity via a web page

R.2.2 - Explains the history of the charity

R.2.3 - Shares with the user the values the charity has

R.2.4 - Shows all the partners the charity has partnered with

User Story Reference: 01 and 05

Priority - 2

Requirement R.3 - Get Staff Information

R.3.1 -  Provides the user with information about all the staff working for the charity via a web page

R.3.2 - Each staff member will have a short introduction about them

User Story Reference: 02

Priority - 3

Requirement R.4 - Get Regional Donations

R.4.1 - Displays information about how much money a certain region has received through donations

R.4.2 - Each region will have a progress bar that fills up as donors donate to that specific region

R.4.3 - Bar gets filled up once the donation goal has been reached

User Story Reference: 03

Priority - 1

Requirement R.5 - Review Catalog

R.5.1 - Allows users to view all the different types of glasses that are being donated

R.5.2 - Donors will be able to see all the types of glasses that they are helping people get

R.5.3 - Those who apply for glasses will be able to view and request pairs of glasses once they’ve been approved for a pair

User Story Reference: 04

Priority - 3

Requirement R.6 - Previous Events

R.6.1 - Allows the user to view all the previous events hosted by the charity

R.6.2 - No account is required to access this information

User Story Reference: 06

Priority - 3

Requirement R.7 - Signup

R.7.1 - Allows the user to create an account

R.7.2 - Users with an account can donate to the charity

R.7.2 - Users with an account can apply to receive glasses

User Story Reference: 07

Priority - 1

Requirement R.8 - Previous Donations

R.8.1 - Users with an account that have donated in the past are able to view their previous donations

R.8.2 - This information is only accessible by the account used to donate

User Story Reference: 08

Priority - 2

Requirement R.9 - Change Authentication

R.9.1 - Allows users with an account to change their password

R.9.2 - Allows users with an account to reset their password if it is forgotten

R.9.3 - Accounts that have had their passwords changed or reset will be logged out of all other instances or devices

User Story Reference: 09

Priority - 1

Requirement R.10 - Register for Newsletter

R.10.1 - Allows users to sign up for the charity’s newsletter

R.10.2 - Those who have signed up will receive automated emails about the charity newsletter

User Story Reference: 10

Priority - 3

Requirement R.11 - Provide Social Media

R.11.1 - The application will have links to the charity’s social media on the “About Us” and home pages.

User Story Reference: 11

Priority - 3

Requirement R.12 - Change Email

R.12.1 - Will allow users with an account to change their email

R.12.2 - All services, including login, account details, and newsletter will correspond and be updated to the new one.

User Story Reference: 12

Priority - 2

Requirement R.13 - Change Payment

R.13.1 - Allows users with an account to change their payment method

R.13.2 - Old payment info will be deleted and not be saved

User Story Reference: 13

Priority - 1

Requirement R.14 - Glasses Search

R.14.1 - Allows the user to search through the catalog

R.14.2 - Glasses will be displayed as cards for easier browsing

R.14.3 - Users will be able to search glasses using a search bar

R.14.4 - Glasses will have keywords associated with its characteristics

User Story Reference: 14

Priority - 2

Requirement R.15 - Apply for Glasses

R.15.1 - Allows the user to apply for glasses

R.15.2 - User will fill out an application and be required to provide information such as monthly income, household income, bank balance, and dependency status

User Story Reference: 15

Priority - 1

**UI Mockups and UX Flows**

**A screenshot of a computer screen

Description automatically generated**

**Figma Link:**

<https://www.figma.com/design/Hbw8HVHGileaAnjGnuKLWR/M2-p2?node-id=0-1&node-type=canvas&t=aEaeNmXjE57rcucc-0>

**High Level Architecture, Database Organization**

A diagram of a company

Description automatically generated

### 1. User Table

* **Create**: Add a new user with their first name, last name, email, and optional foreign key references to their Donator or Donatee roles.
* **Read**: Retrieve a user’s details based on their User ID, including their first name, last name, email, and Donator or Donatee status.
* **Update**: Modify the user’s personal information, such as updating their first or last name.
* **Delete**: Remove a user from the system using their unique User ID.

### 2. Donator Table

* **Create**: Add a new donator entry for a user, including whether they have donated, their total donation amount, and their associated account.
* **Read**: Retrieve donator-specific details for a user, such as their donation history and account reference.
* **Update**: Update the donation information, like increasing the total donation amount or linking a new account.
* **Delete**: Remove the donator record for a user, typically when they are no longer participating as a donor.

### 3. Donatee/Recipient Table

* **Create**: Add a new recipient (donatee) entry for a user, including details like their monthly income, household income, bank balance, and whether they’ve claimed an account or prescription.
* **Read**: Retrieve a donatee’s details, such as their financial information and whether they’ve applied for or claimed their account.
* **Update**: Modify the donatee’s financial data or their claim status (e.g., if they’ve claimed their prescription glasses).
* **Delete**: Remove a donatee from the system, typically when they no longer need to be tracked as a recipient.

### 4. Account Table

* **Create**: Add a new account for a user, with information like phone number, address, city, state, zip code, and geographic location.
* **Read**: Retrieve account details based on an account ID, including the user’s contact information and location.
* **Update**: Modify the account information, such as updating the address or phone number.
* **Delete**: Remove an account from the system, typically when it is no longer active or associated with a user.

### 5. Donation Table

* **Create**: Add a new donation entry, including the donor, the donation amount, the region the donation is assigned to, and the date of the donation.
* **Read**: Retrieve donation details based on a unique donation ID, including the donor’s information, amount, and regional allocation.
* **Update**: Modify donation information, such as changing the amount or region.
* **Delete**: Remove a donation record, typically when it is no longer needed for tracking.

### 6. Region Table

* **Create**: Add a new region, including the region’s name and the total donations allocated to it.
* **Read**: Retrieve details for a specific region, such as the region’s name and total donations.
* **Update**: Modify the region’s total donation amount or change the region’s name.
* **Delete**: Remove a region from the system, typically when it is no longer relevant to the donation structure.

### 7. Glasses Table

* **Create**: Add a new pair of glasses to the inventory, including its name, size, color, prescription range, image, and stock level.
* **Read**: Retrieve details about a specific pair of glasses, such as their name, size, color, and stock level.
* **Update**: Modify the glasses inventory, for example, by updating the number of glasses available in stock or changing the description.
* **Delete**: Remove a specific pair of glasses from the inventory when they are no longer available.

·   APIs

We are using django, django rest framework and djoser for user auth. Djoser provides endpoints with session auth, token auth, JWT, and user create/update/delete.

**Auth**

POST - /auth/token/login - Create token for user logged in

POST - /auth/token/logout - Destroy token for user

POST - /auth/users/reset\_password/ - resets password for user

POST - /auth/users/reset\_username/ - resets password for user

**API**

*USERS*

GET - /api/users/ - Retrieves a list of users

POST - /api/users/ - Create new user

PUT/PATCH - /api/users/<id> - Updates existing user information

DELETE - /api/users/<id> - Deletes user and all associated information

GET - /api/users/<id> - Retrieves user detail info

*DONATEE*

GET - /api/donatee/ - Retrieves a list of donatees

POST - /api/donatee/ - Creates a new donatee

PUT/PATCH - /api/donatee/<id> - Updates information for a donatee

DELETE - /api/donatee/<id> - Deletes a donatee

GET - /api/donatee/<id> - Retrieves donatee detail info

*DONATOR*

GET - /api/donator/ - Retrieves a list of donators

POST - /api/donator/ - Creates a new donator

PUT/PATCH - /api/donator/<id> - Updates information on a donator

DELETE - /api/donator/<id> - Deletes a donator

GET - /api/donator/<id> - Retrieves donator detail info

*ACCOUNT*

GET - /api/account/ - Retrieves a list of accounts

POST - /api/account/ - Create new user

PUT/PATCH - /api/account/<id> - Updates existing user information

DELETE - /api/account/<id> - Deletes user and all associated information

GET - /api/account/<id> - Retrieves account detail info

*DONATION*

GET - /api/donation/ - Retrieves a list of donations

POST - /api/donation/ - Creates a new donation

PUT/PATCH - /api/donation/<id> - Updates information on a donation

DELETE - /api/donation/<id> - Deletes a donation

GET - /api/donation/<id> - Retrieves donation detail info

*REGION*

GET - /api/region/ - Retrieves a list of regions

POST - /api/region/ - Creates a new region

PUT/PATCH - /api/region/<id> - Updates information on a region

GET - /api/region/<id> - Retrieves region detail info

*GLASSES*

GET - /api/glasses/ - Retrieves a list of glasses

POST - /api/glasses/ - Creates a new set of glasses

PUT/PATCH - /api/glasses/<id> - Updates information on a pair of glasses

DELETE - /api/glasses/<id> - Deletes a pair glasses

GET - /api/glasses/<id> - Retrieves glasses detail info

# Django (Backend Framework)

# Usage**: Full-stack web framework for defining backend APIs and managing user sessions.**

* <https://www.django-rest-framework.org/>

# Token Authentication

# Usage**: For secure user authentication and session management.**

**High Level UML Diagrams**

 Requirement ID: 1

Title: Donate

Description: This requirement enables the user to donate to the charity, which allocates glasses to people in need.

User Story Reference: 00

Priority: ❗TOP❗

 Requirement ID: 8

Title: Signup

Description: This requirement allows a user to create an account. Once a user becomes an account they will be able to donate.

User Story Reference: 07

Priority: ❗TOP❗

Requirement ID: 10

Title: Change Authentication

Description: This requirement allows users to change their password or reset it if forgotten. Once a password is changed, all devices logged in with the previous password will be logged out.

User Story Reference: 09

Priority: ❗TOP❗

Requirement ID: 15

Title: Glasses Search

Description: This requirement allows the user to search through the catalog. Glasses will be displayed as cards for easier browsing.

User Story Reference: 14

Priority: ❗TOP❗

Requirement ID: 4

Title: Get Regional Donations

Description: This requirement provides information about donations in specific regions, displayed on a map that serves as a progress bar.

User Story Reference: 03

Priority: High

User ( Many to Many )

1. User register with one account type (Donatee and Donator)
2. User shall have one name
3. User shall have one last name
4. User shall have donate option FK
5. User shall have donate option FK
6. User shall have one email
7. User  shall have one phone number
8. User shall have a user ID (PK)

Donator ( Zero or Many )

1. User shall donate several donation amounts
2. User shall browse the catalog
3. User shall have an account id
4. User shall have several regions/donation amounts
5. A donator shall have a user  id

Recipient ( One and only one )

1. Reshall apply to one account
2. Recipient shall have one recipient id
3. A recipient shall have one income
4. A recipient shall have one balance
5. A recipient shall be weak
6. A recipient shall have one prescription
7. A recipient shall have a user id

Account ( User to Account ( Many ) )

1. An account shall have one number
2. An account shall have one address
3. An account shall have one city
4. An account shall have one zip code
5. An account shall many location regions for a percentage leading to a 100

Donation ( one and only one )

1. A donation shall have a user id (FK)
2. A donation shall have have one confirmation number
3. A donation shall have a completed regions/donation amount
4. A donation shall have one region ID (FK)
5. A donation shall have one donation id (PK)
6. A donation shall have one donation date
7. A donation shall have donation amount

Region ( One )

1. A donation amount shall have one region ID ( PK )
2. A region shall have one name
3. A region shall have multiple values
4. A region shall depend on the donation ( PK )

Glasses ( Zero or One )

1. Glasses shall have one name
2. Glasses shall have one identification number (PK)
3. Glasses shall have one color
4. Glasses shall have two separate ranges
5. Glasses shall have one or multiple images
6. Glasses shall have an inventory count
7. Glasses shall have one description

Sign up ( zero or many )

1. A user shall have more than one account
2. A user shall have have two options; donator (FK) , or recipient (FK)
3. A user shall sign up multiple times
4. A user shall have either a donator id or recipient id

Authentication ( One )

1. An authentication shall be randomized
2. An authentication should have a unique ID (PK)
3. An authentication shall be discarded
4. An authentication will sign out from one account (FK)

Glasses Search ( Many )

1. A user shall search multiple glasses
2. A user has access to the catalog
3. A user shall add one glasses to the cart
4. A user must be a recipient ( FK )

A diagram of a network

Description automatically generated

A screenshot of a computer screen

Description automatically generated

**Key Risks**

* Some members currently are taking CSC 415 which is an extremely difficult class.
  + This might affect the availability and work of our members, as they will have priority in that class over other classes.
  + Will try and complete as much as work as we can before the work in CSC 415 gets harder for them.
* A few members cannot make scheduled meetings.
  + Will try and host additional meetings so those members can get caught up.

**Project Management**

For Milestone 2, we managed to complete all the tasks by splitting up the work based on each team member's expertise in a certain technology and the tasks requirements. For example, we made sure to have the Front and Back-end leads work on Steps 3 and 4 respectively, as those steps involve a lot of front ends and back-end work to be done. We applied this same philosophy for all the steps in Milestone 2, including the Document and the Vertical SW Prototype. So far, the tool we have been using to manage each member’s task is using both Discord and Notion. Every task assigned to a member is written on each platform to ensure that everyone is on the same page. Each member’s progress is shared in each in-person and online meeting that we have and is subsequently written down either in the shared document that we have or the corresponding meeting minutes.