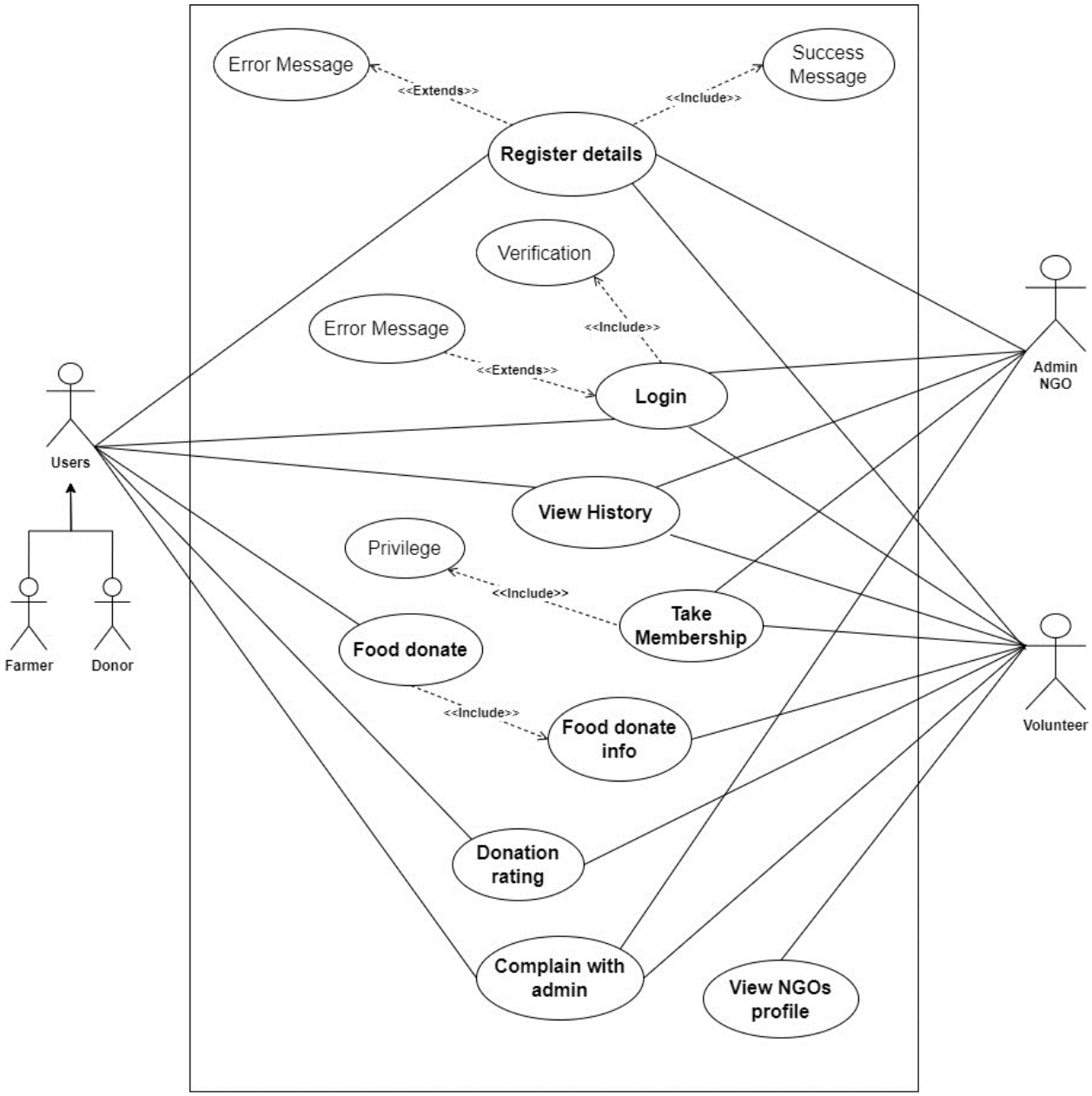
Use Case Diagram

Symbol	Name
	Actor
	Use Case
	Connector
	Generalization
	Stereotype



High-Level Use Case

Register

Use Case:	Register Details
Actors:	Donor, Volunteer, NGO, Farmer

Descriptions:	All users can input their respective details into the system. Upon submission, the system automatically registers the provided information, ensuring seamless integration and efficient data management. If the register details are correct show the success message otherwise display the error message.

Take Membership

Use Case:	Take Membership
Actors:	Volunteer, NGO
Descriptions:	A new volunteer provides the personal details, and his/her details are registered with the system. The NGO provide the membership, and then volunteers take the new membership.

Use Case:	Privilege
Actors:	Volunteer
Descriptions:	After taking a new member, a new volunteer gets the privilege of the food donation system.

Login

Use Case:	Login
Actors:	Donor, Volunteer, Farmer, NGO

Descriptions:	After registering details in the system, all the users can provide valid details and log in to the system Then successfully log in to the system. If the login details are valid display the success message otherwise error message.

Food Donate

Use Case:	Food Donate
Actors:	Donor, Farmer, Volunteer
Descriptions:	The Donor or Farmer can donate the proper food information and details with location. The system can show the donation food details in the history after posting the donated food. All the volunteers can get the donation information (Notification).

View Donation info

Use Case:	Donation Info
Actors:	Volunteer
Descriptions:	After receiving the donation info, the volunteer can view the donation details if it is possible or not possible to distribute.

Donation Rating

Use Case:	Donation Rating
Actors:	Volunteer, Donor
Descriptions:	After the food is completely donated to some people the volunteer can give the donation rating to the donor with food distributed information.

View History

Use Case:	View History
Actors:	Donor, Volunteer, NGO, Farmer
Descriptions:	All the users can view the history of food donation where who donate, where to donate or more details.

View NGO profile

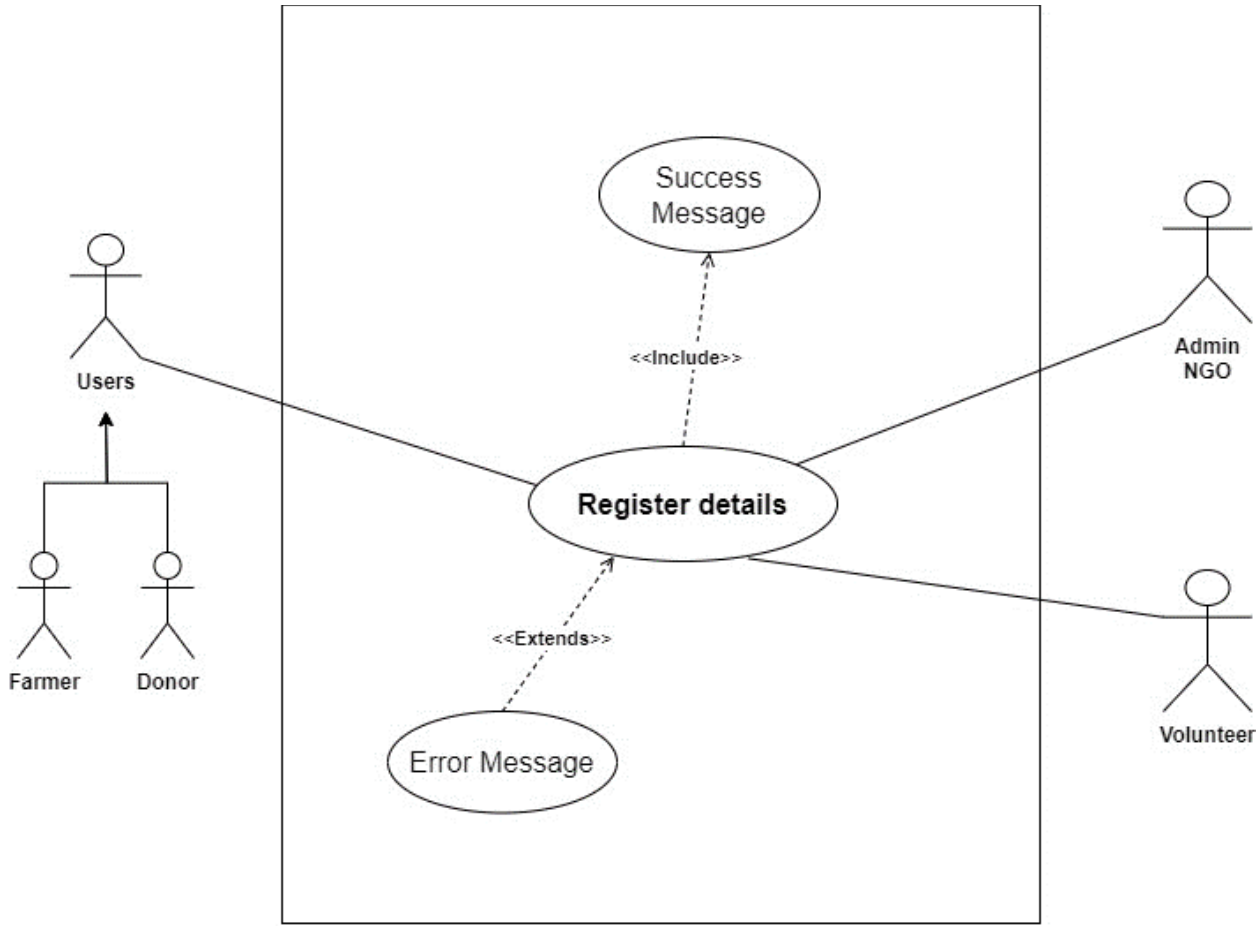
Use Case:	View NGO profile
Actors:	Volunteer, NGO
Descriptions:	The volunteer can view the NGO profile details where some information gets more details.

Complain with Administration

Use Case:	Complain with Admin

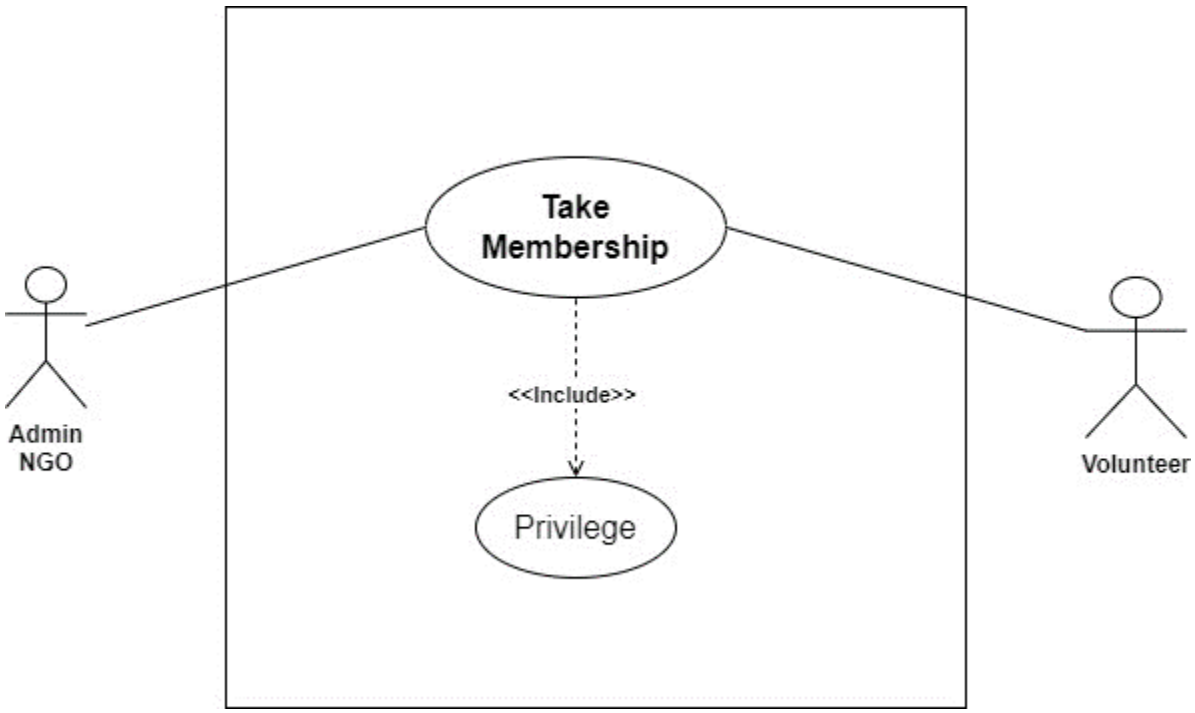
Actors:	Donor, Volunteer, Admin
Descriptions:	After contact with donors and volunteers the donors cannot be provided or donate food and the receiver cannot come to receive the donation food, they can complain to the admin

Expanded Use Case



Use Case:	Register Details
Actors:	Donor, Volunteer, NGO, Farmer
Descriptions:	All users can input their respective details into the system. The system automatically registers the provided information upon submission, ensuring seamless integration and efficient data management.

Typical Courses of Events:	
Donor, Volunteer, NGO, Farmer, Admin	**System Response**
1. A user can provide the personal details for the register in the system.	
	2. The system checks whether the provided details are valid or not.
3. Request for register in the system.	
	4. Conform register with a success message if no valid data, then show the error message.

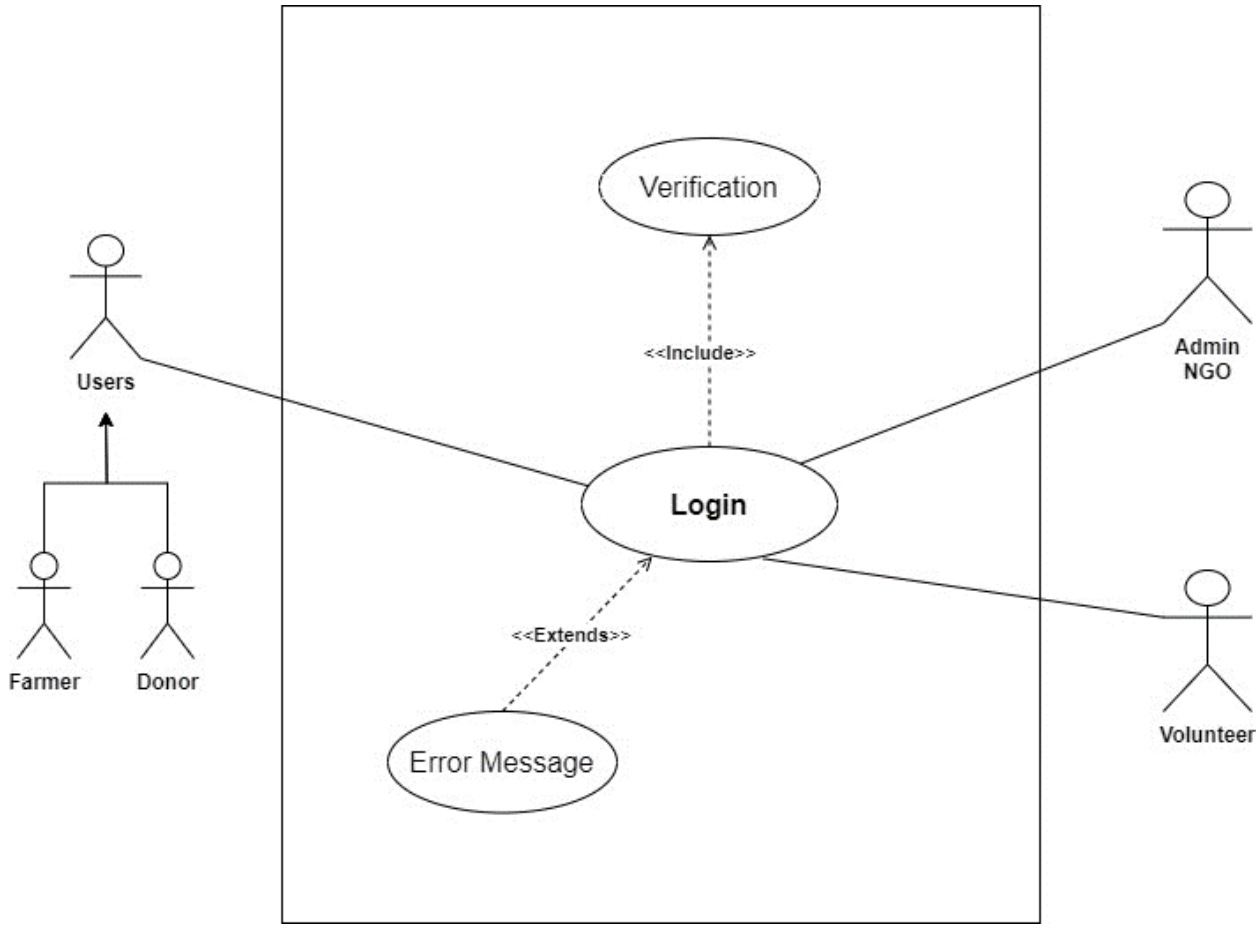


Use Case:	Take Membership
Actors:	Volunteer, NGO
Descriptions:	A new volunteer provides the personal details, and his/her details are registered with the system. The NGO provide the membership, and then volunteers take the new membership.

Typical Courses of Events:

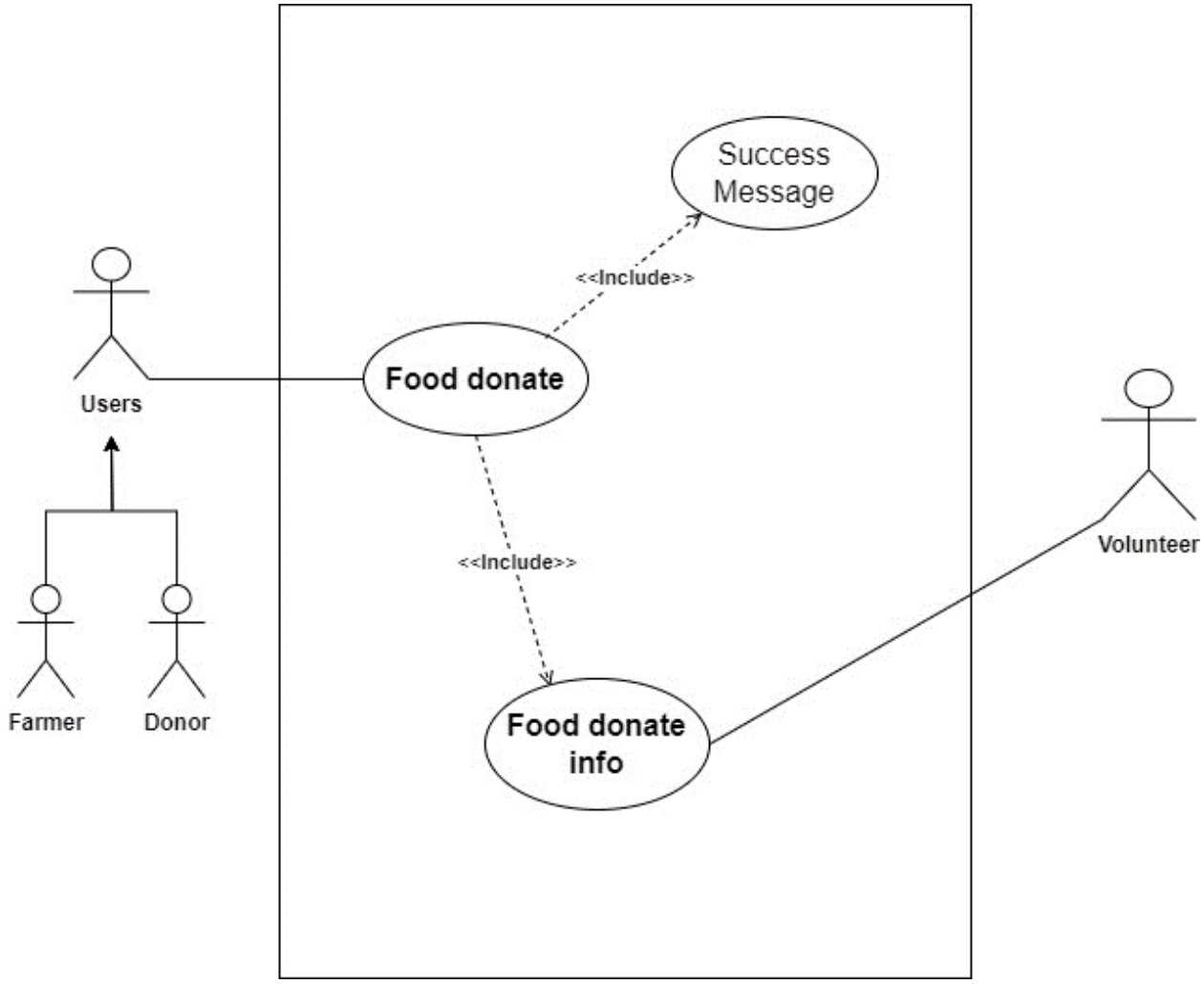
Volunteer, NGO	**System Response**
1. A new volunteer can provide the personal details for the registered membership in the NGO.	
	2. Check the Volunteer details.

3. Request to take membership in the NGO.	
	4. Confirm the registered membership and give the privilege.



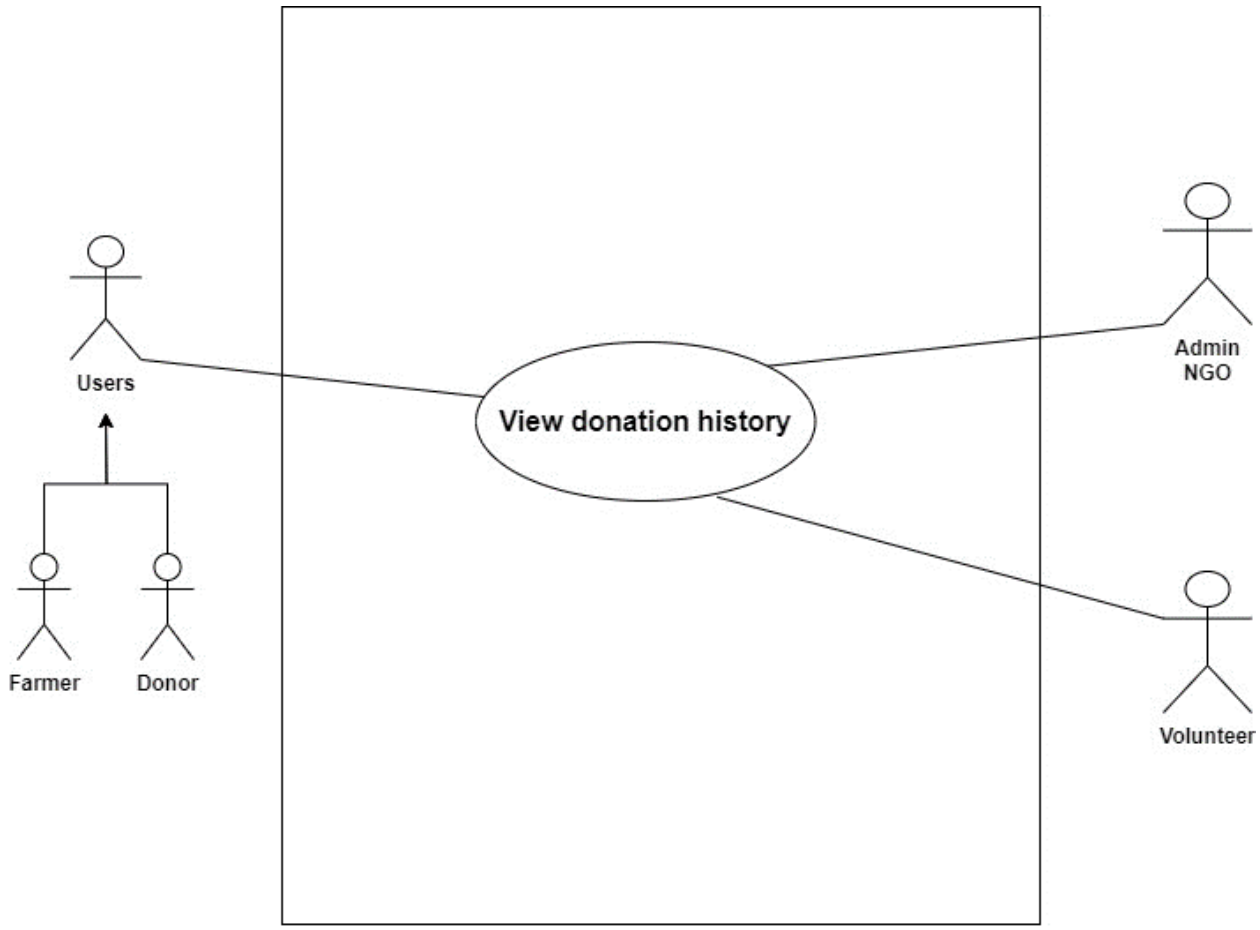
Use Case:	Login
Actors:	Donor, Volunteer, Farmer
Descriptions:	After registering details in the system, all the users can provide valid details and log in to the system Then successfully log in to the system.

Typical Courses of Events:	
Donor, Volunteer, Farmer	**System Response**
1. A new user can provide the login details in the system.	
	2. The system checks whether the provided details are valid or not.
3. Request for login in the system.	
	4. The system can navigate to the dashboard if valid data is provided to log otherwise shows the error message.



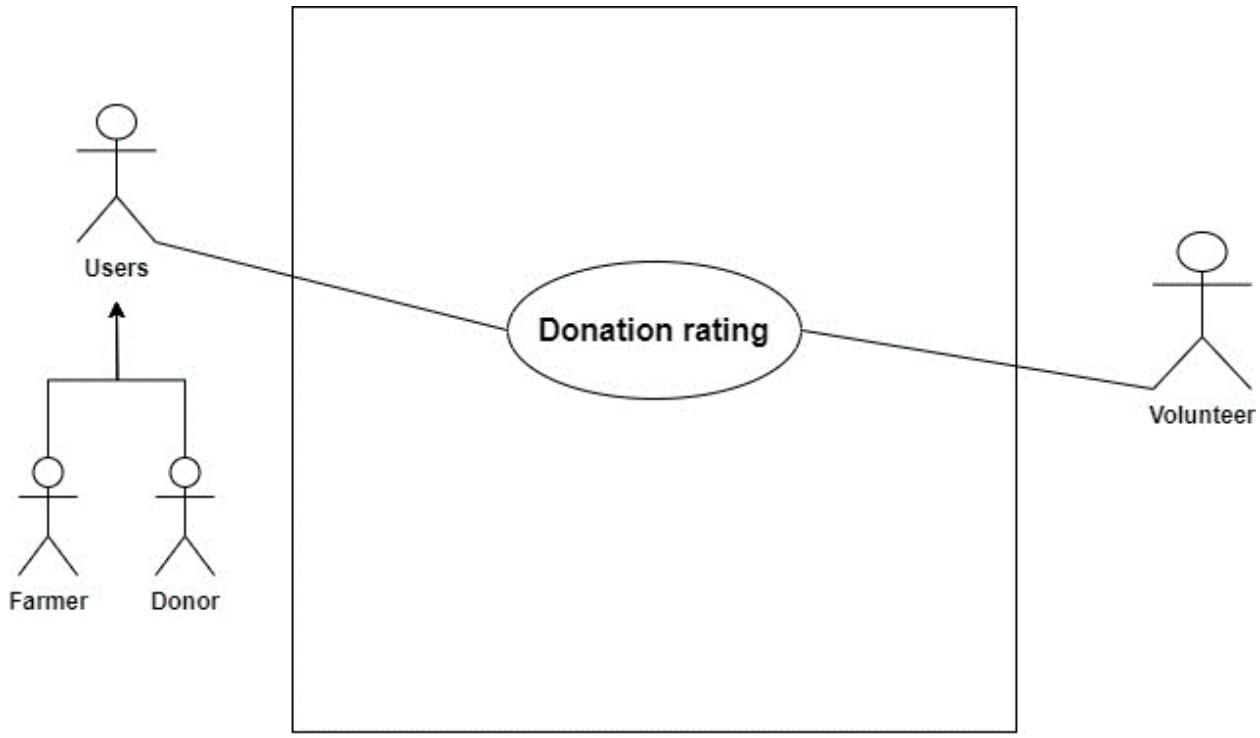
Use Case:	Food Donation
Actors:	Donor, Farmer, volunteers
Descriptions:	The Donor or Farmer can donate the proper food information and details with location. The system can show the donation food details in the history after posting the donated food.
Typical Courses of Events:	
Donor, Farmer	**Volunteers**

1. A donor can post the food details for donation.	
	2. After getting the donation information volunteers to respond to the request to accept the food.
3. Donors confirm to provide the food when contacted with the donor.	
	4. Volunteers receive the food and go to distribute it.



Use Case:	View Donation History
Actors:	Volunteer, Farmer
Descriptions:	All users can input their respective details into the system. The system automatically registers the provided information upon submission, ensuring seamless integration and efficient data management.

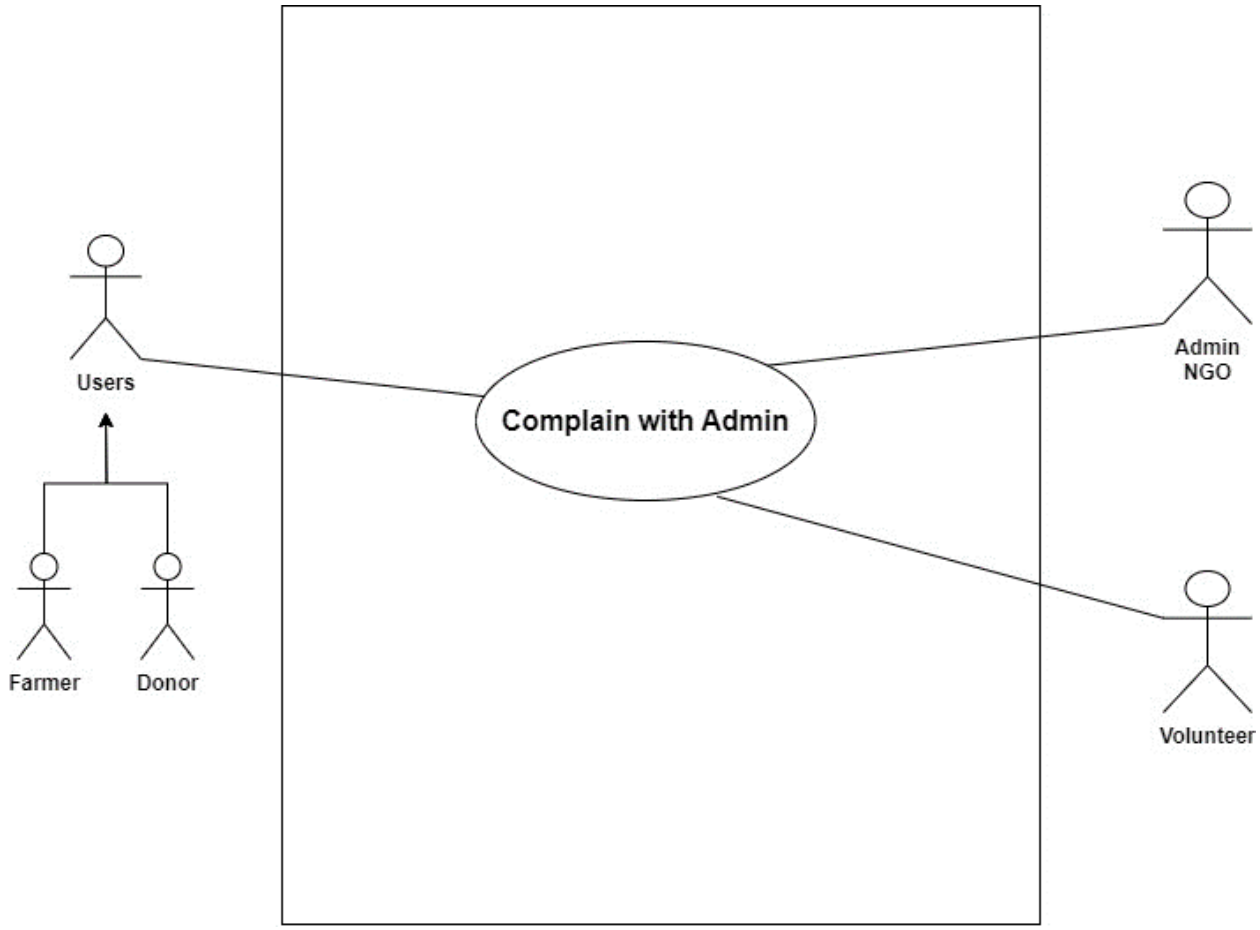
Typical Courses of Events:	
Volunteer, Farmer	**System Response**
1. A volunteer can view the donation post food details.	
	2. show the donation food details with all information like food info, donor info and location details.
3. View the donated location.	
	4. The system can show the food donated location with Google Maps.



Use Case:	Donation Rating
Actors:	Donor, Volunteer, Farmer
Descriptions:	After the food is completely donated to some people the volunteer can give the donation rating to the donor with food distributed information.

Typical Courses of Events:	
Volunteer	**Donor**
1. After the food received to distributed give the donation rating with all the information.	
	2. Get the donated rating info.
3. Not possible to accept the donation of	

food if the volunteer cannot give a response.	
	4. Volunteers do not respond to receive the food and when the expiration date is over the donor gets the expiration notification.



Use Case:	Complain with Admin
Actors:	Donor, Volunteer, Farmer
Descriptions:	All users can input their respective details into the system.

	The system automatically registers the provided information upon submission, ensuring seamless integration and efficient data management.
Typical Courses of Events:	
Donor, Volunteer, Farmer	**Admin Response**
1. The food donation time donor volunteer and farmer can complain to the admin.	
	2. Admin verifies the complaint and gives the warning.
3. Received warning.	

Context Diagram

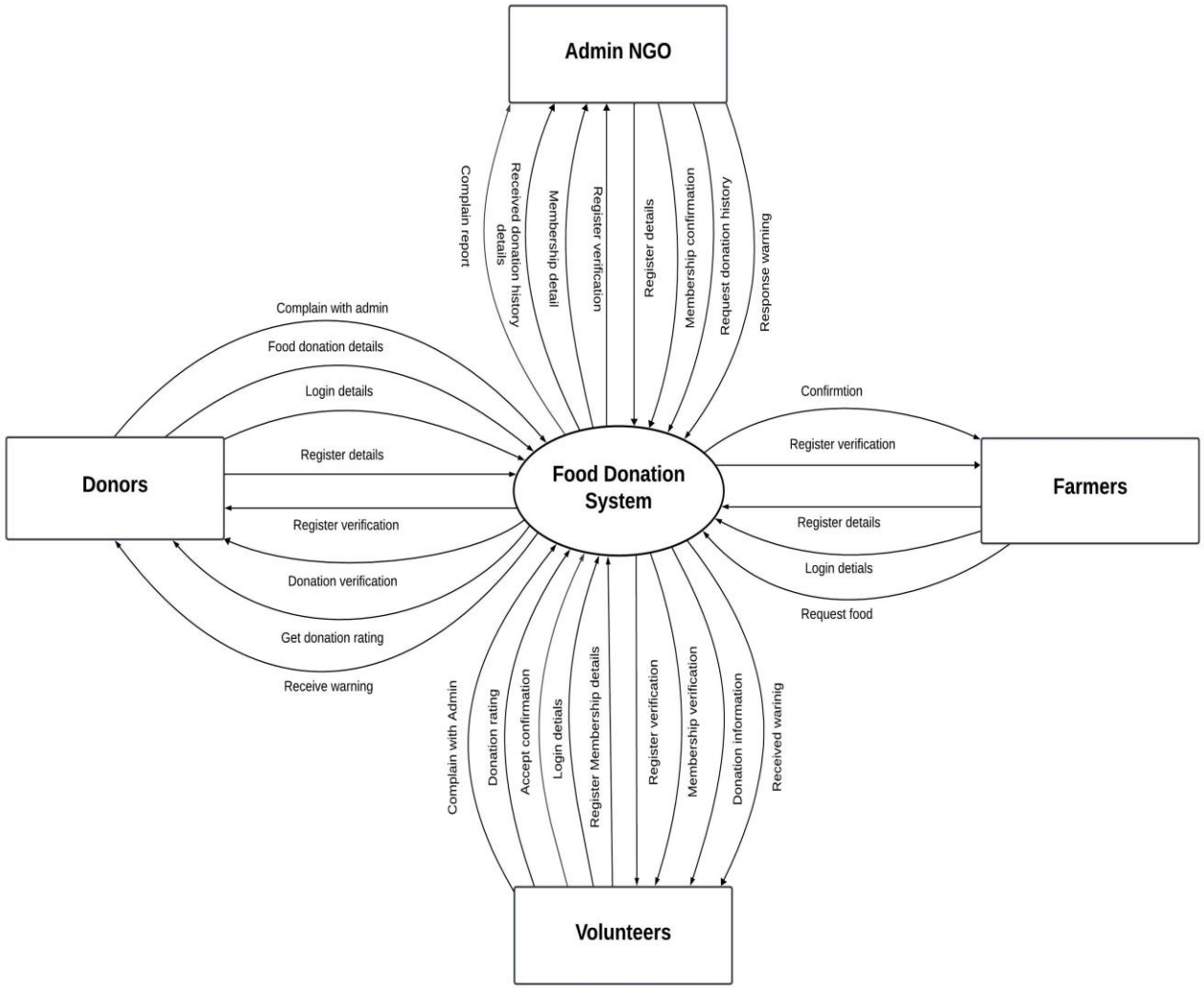


Figure 1: Context diagram

Data Flow Diagram (DFD)

Level-1

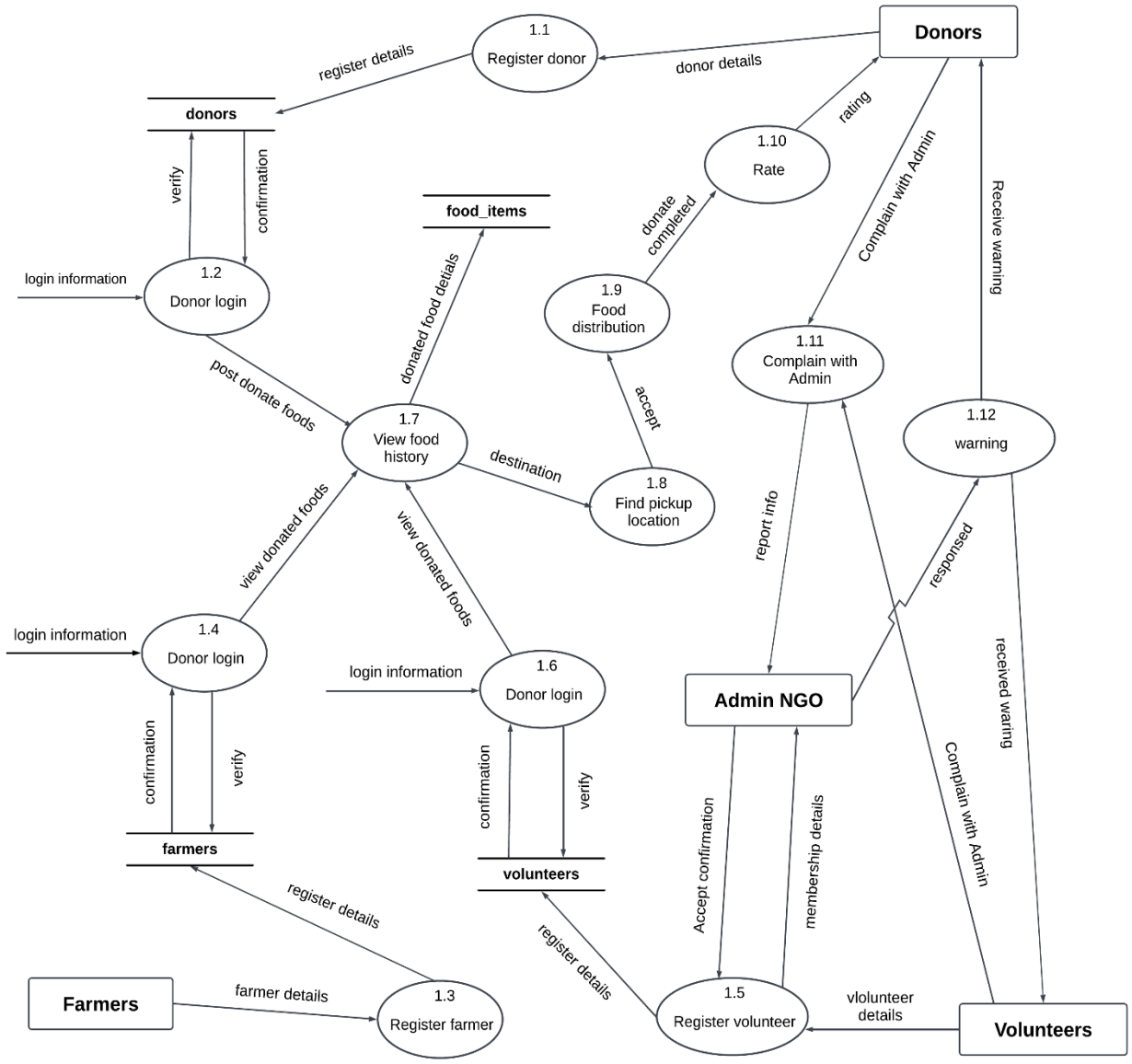


Figure 2: DFD Level-1

Level-2

Register Details

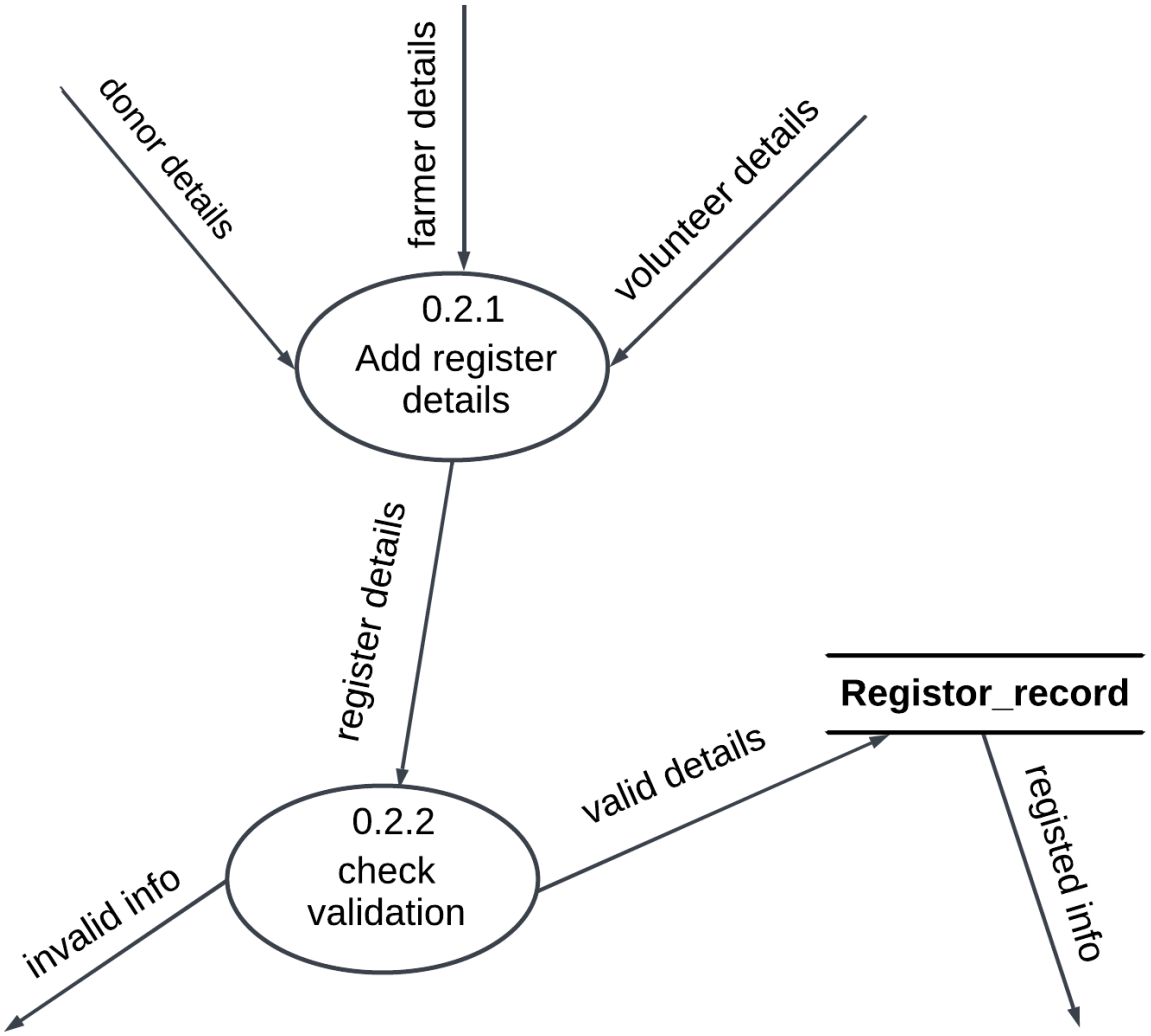


Figure 3: Register details DFD Level-2

Take Membership

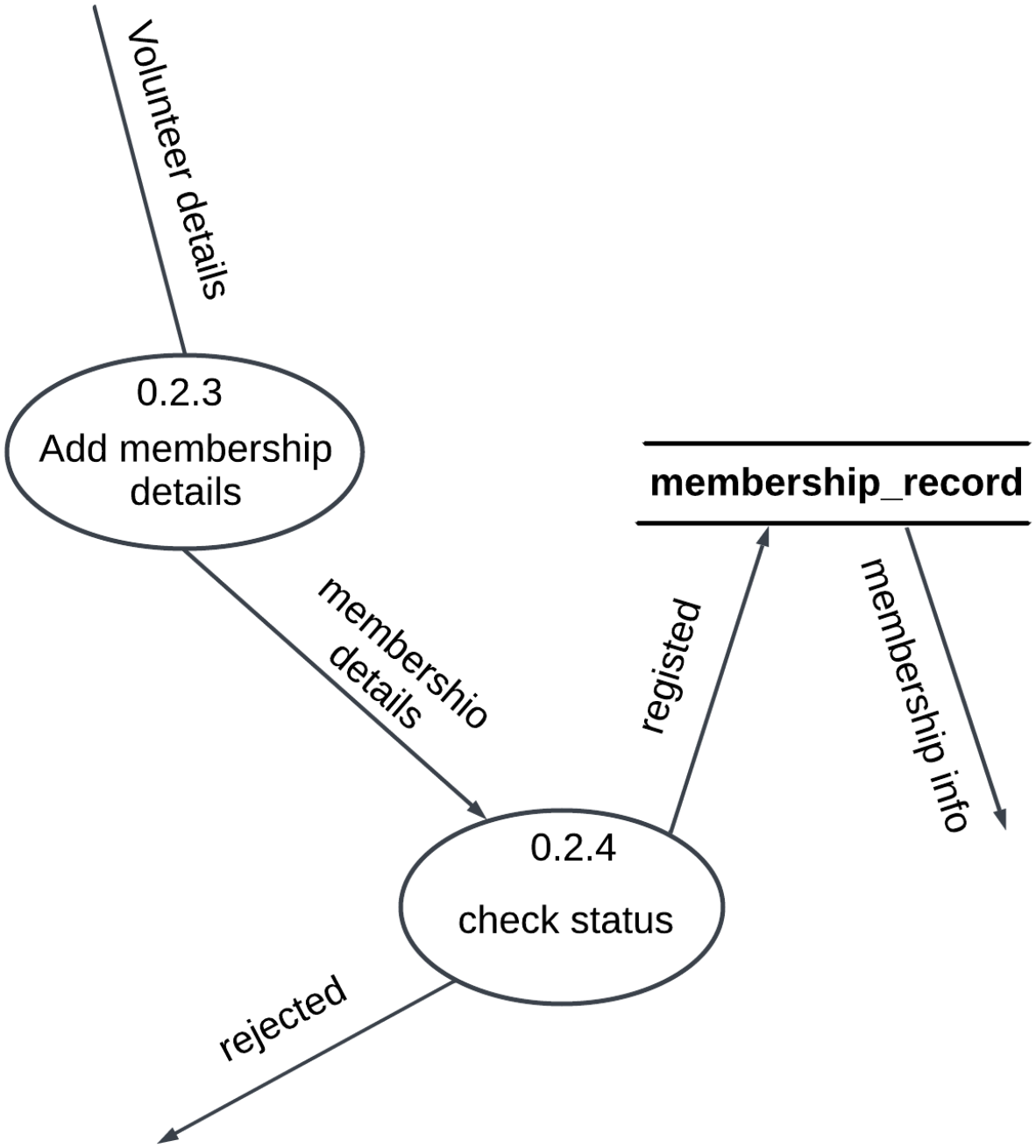


Figure 4: Take membership details DFD Level-2

Login System

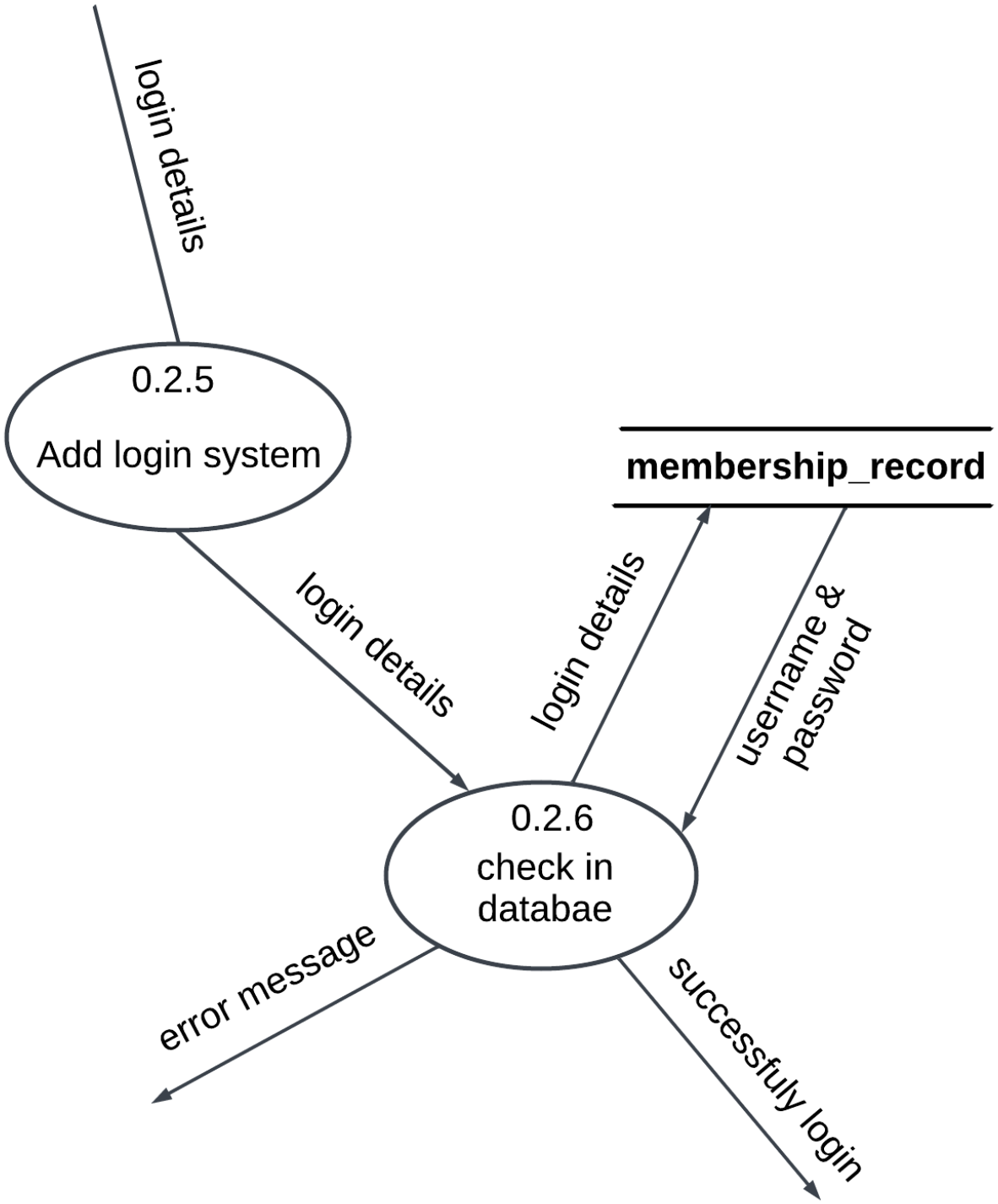


Figure 5: Login details DFD Level-2

Food Donate

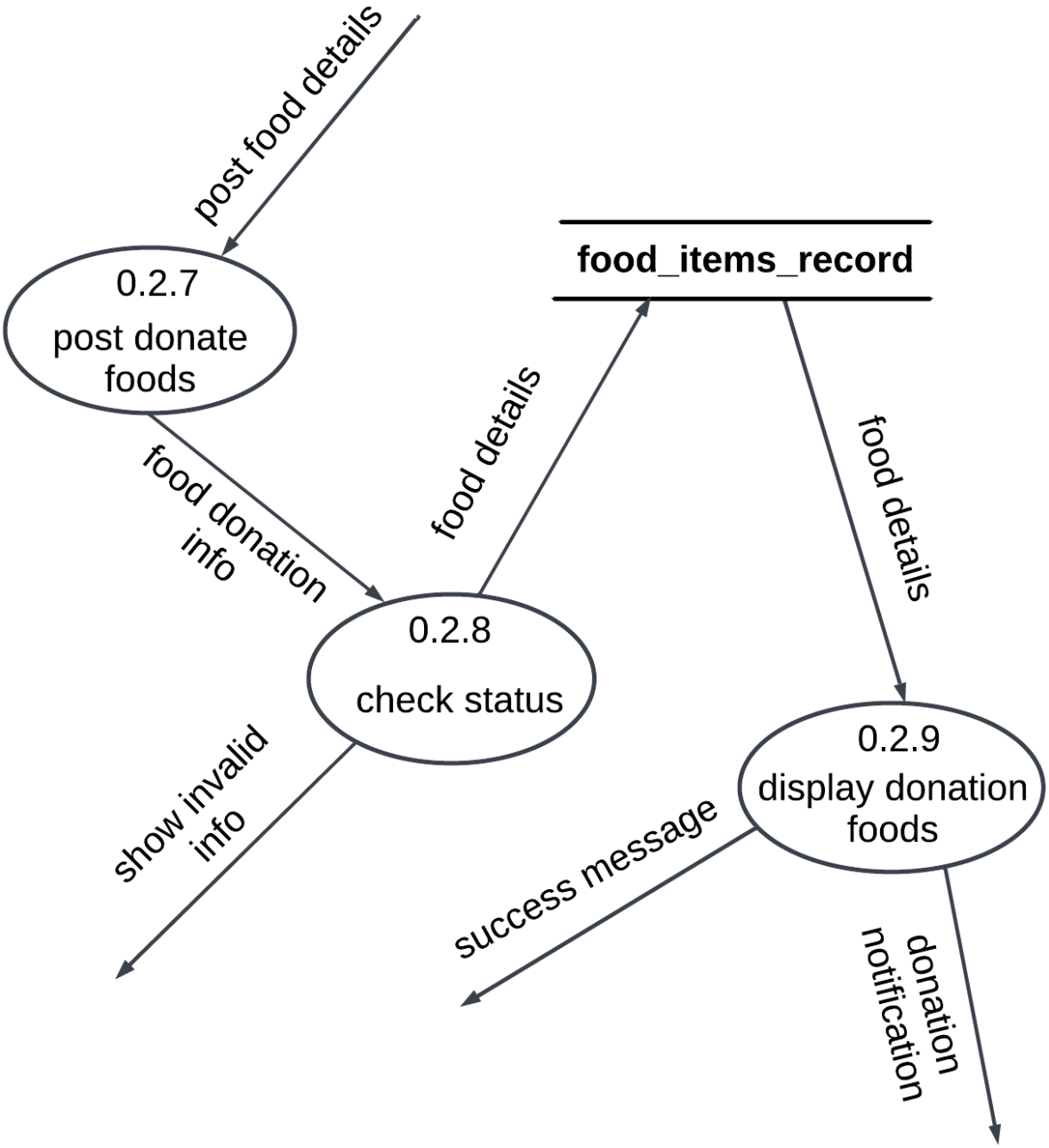


Figure 6: Food donation DFD Level-2

View Donation History

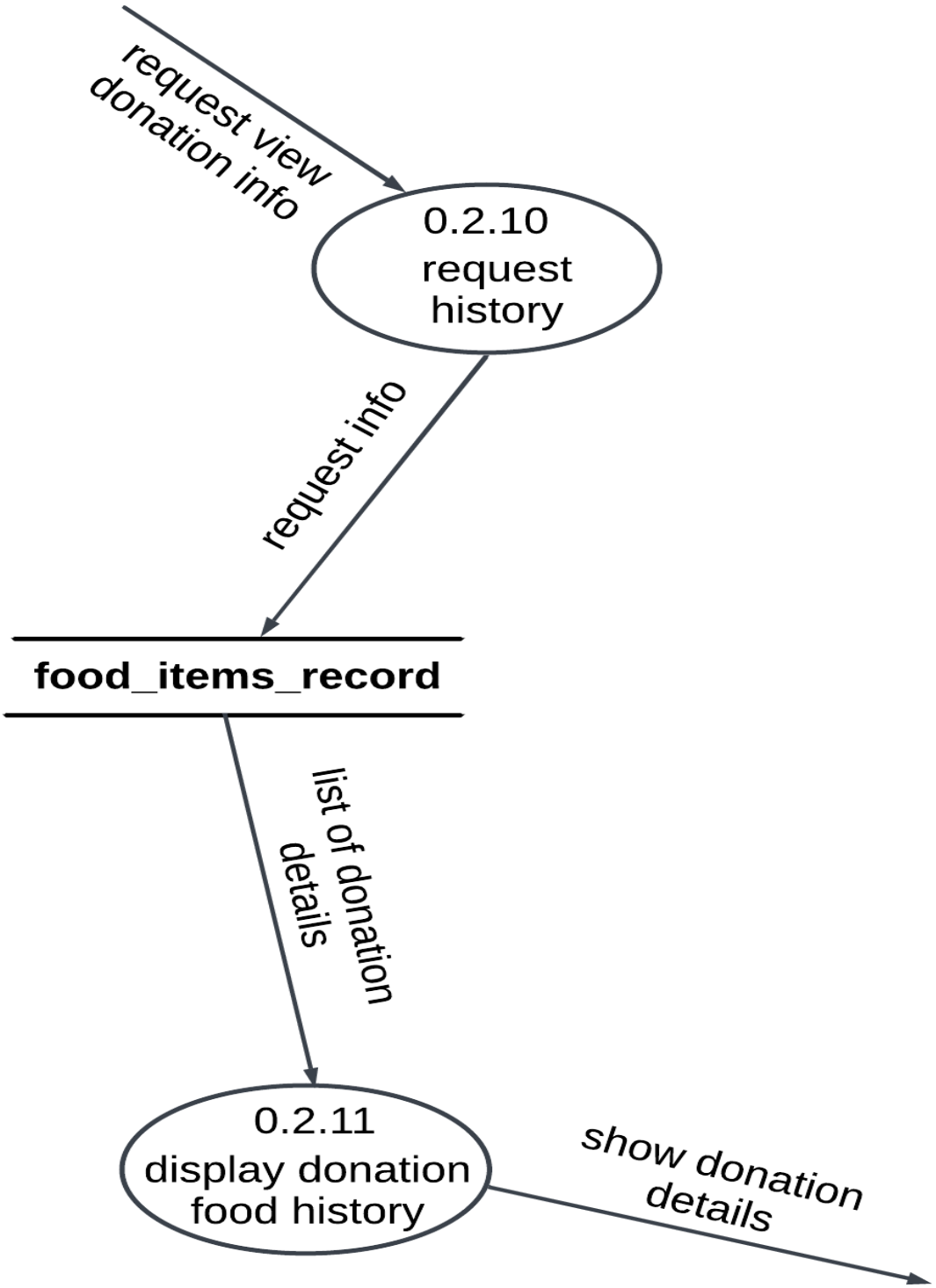


Figure 7: View Donation History DFD Level-2

Donation Rating

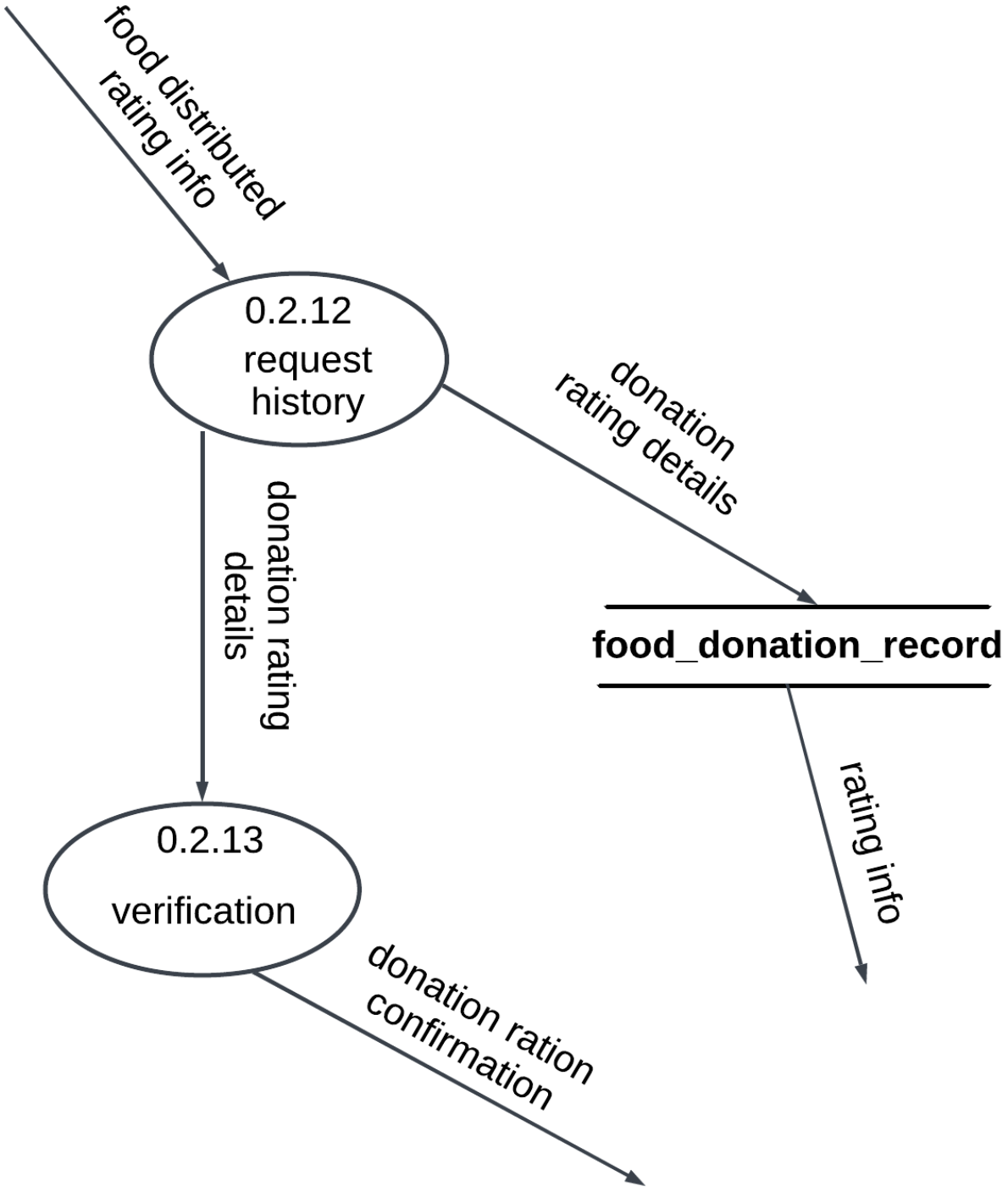


Figure 8: Donation rating DFD Level-2

Complain with Admin

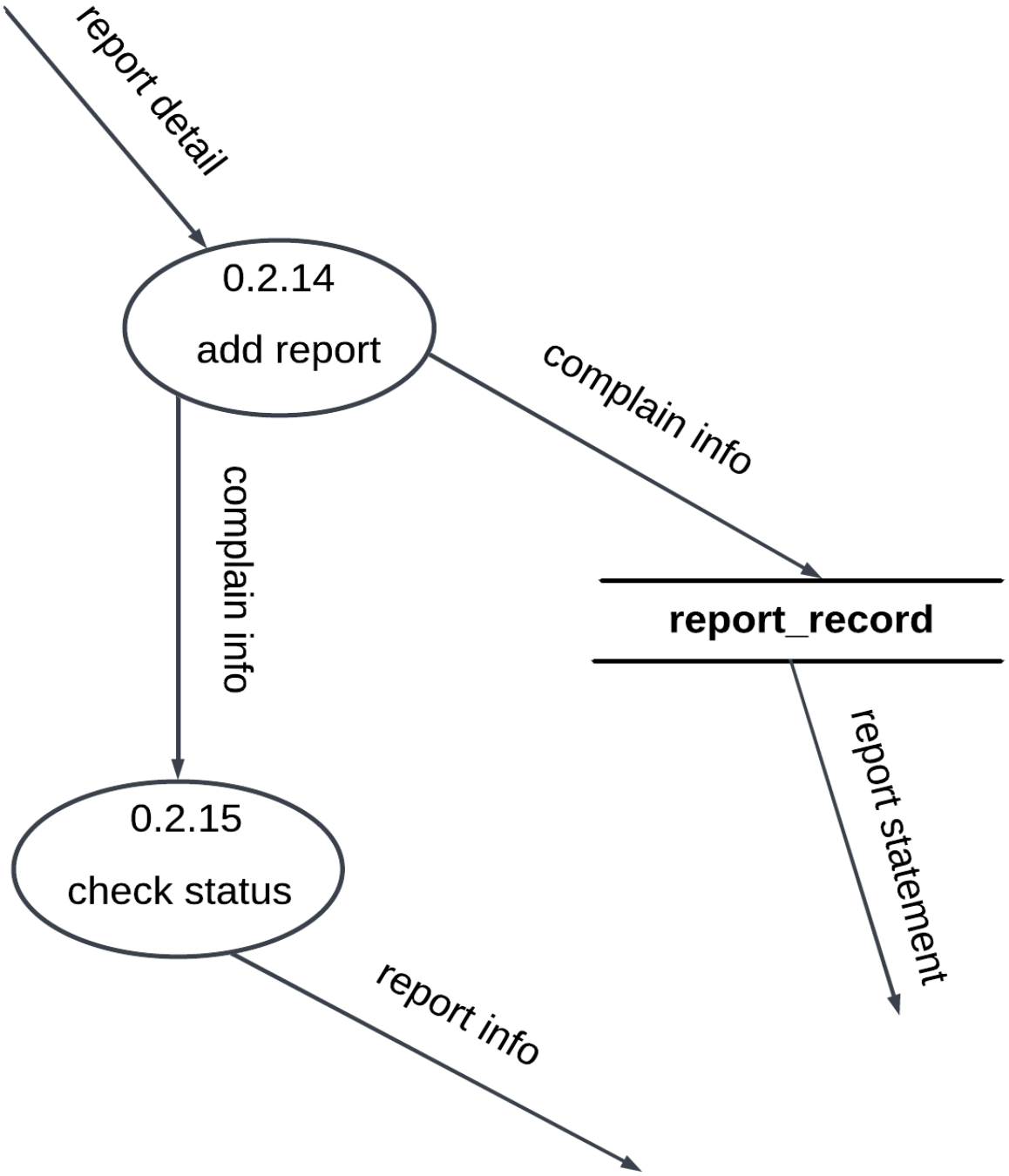


Figure 9: Complaint with admin DFD Level-2

Class Diagram

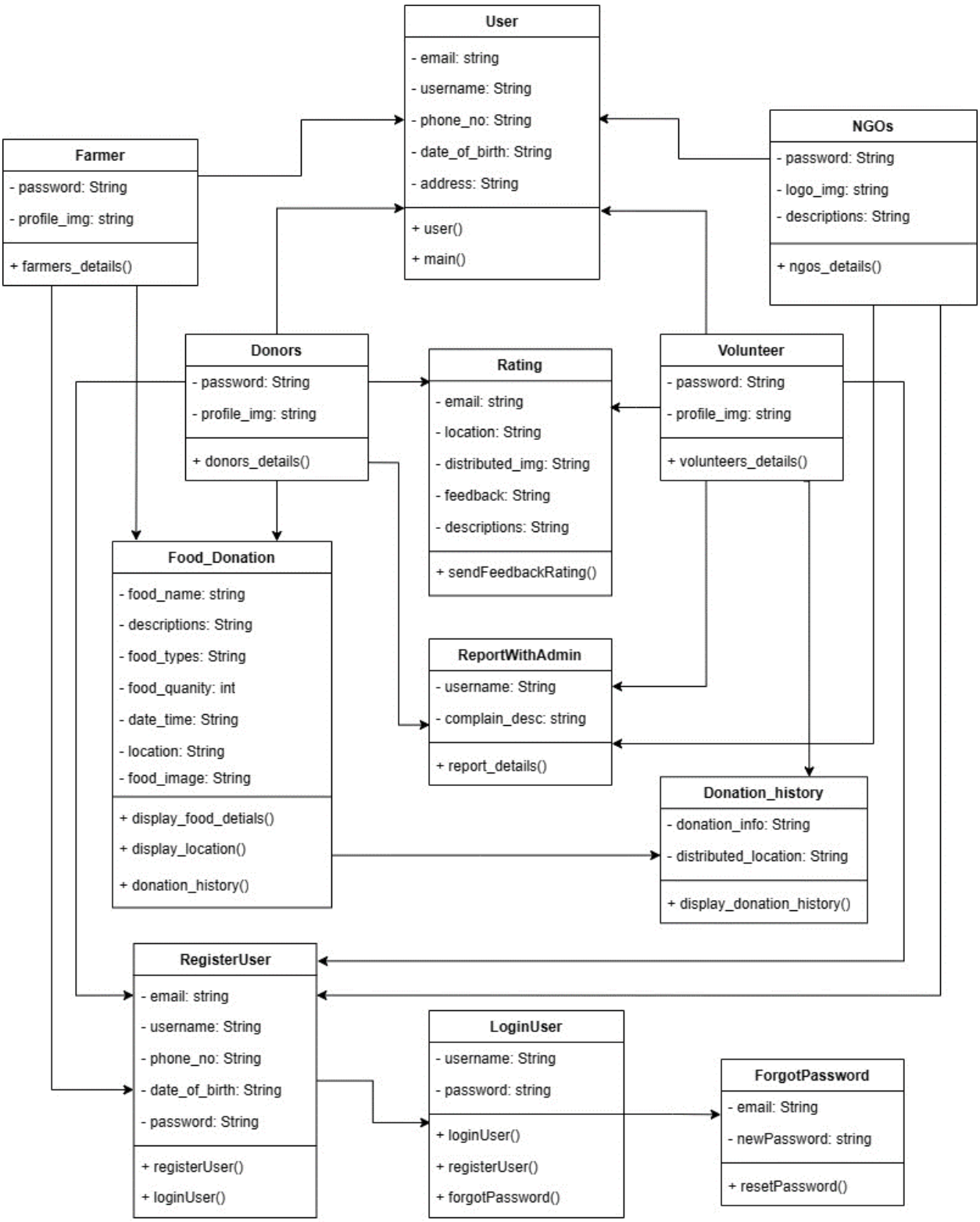


Figure 10: Class diagram of food donation application

Activity Diagram

Login Activity

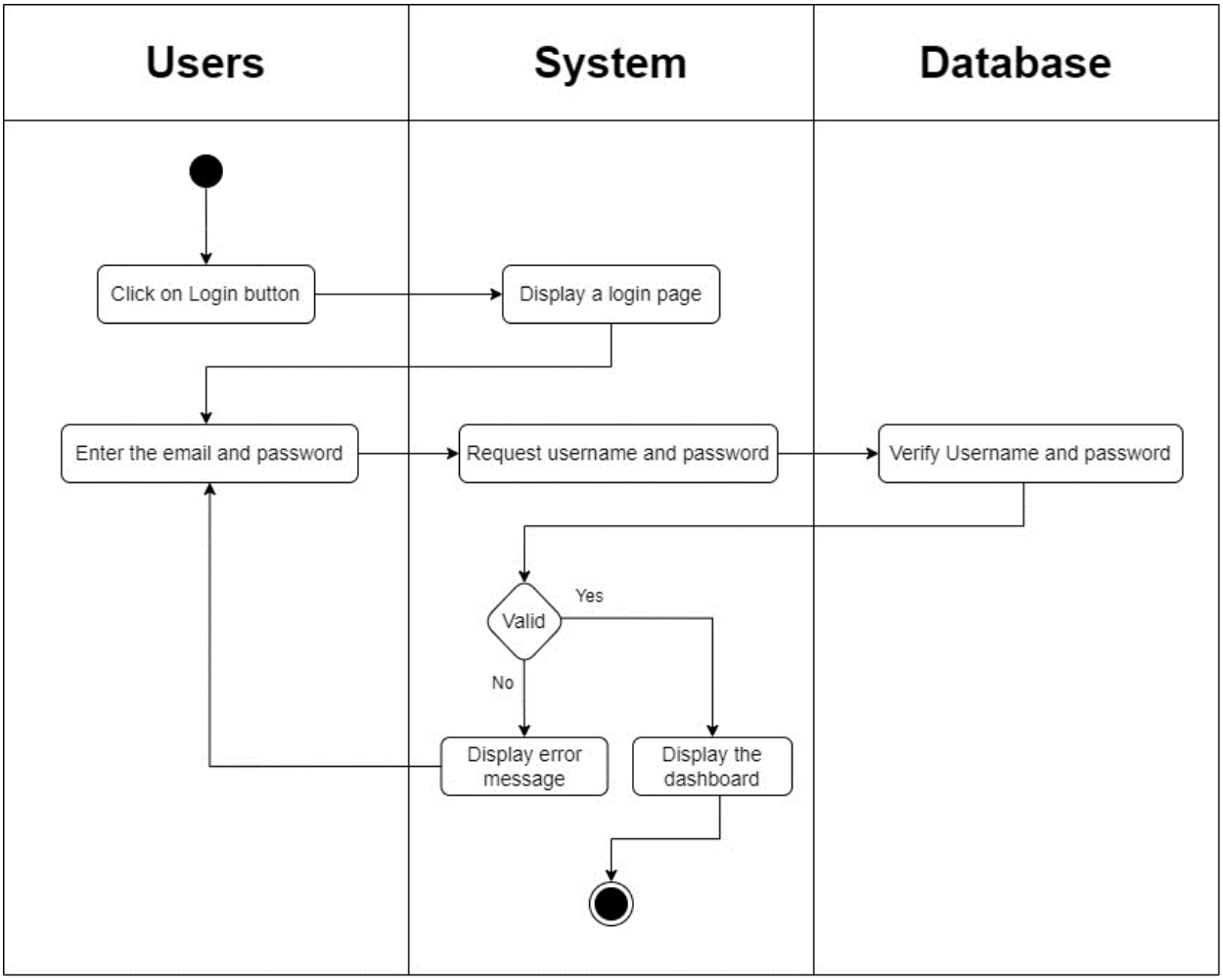


Figure 11: Login system activity diagram

Register Activity

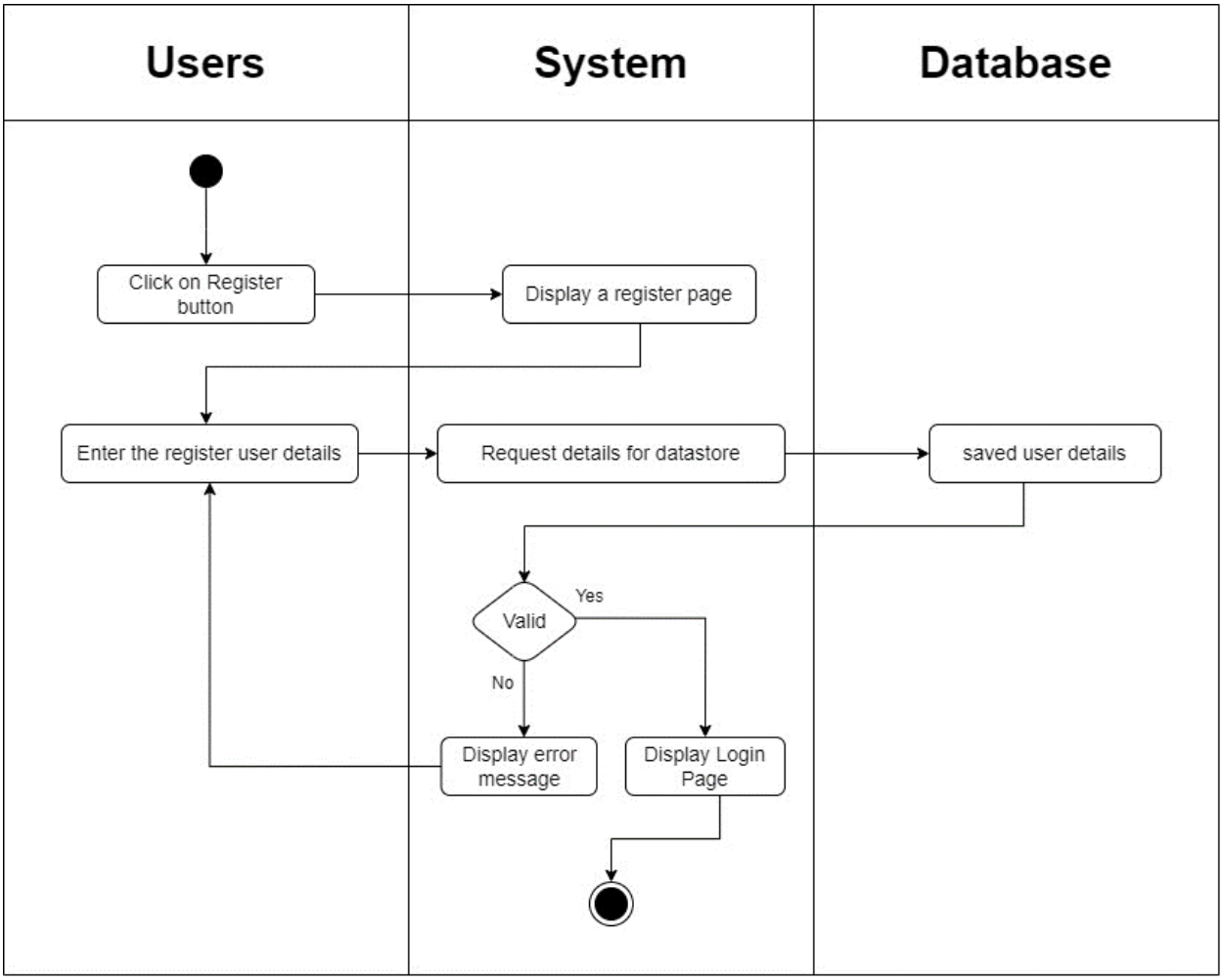
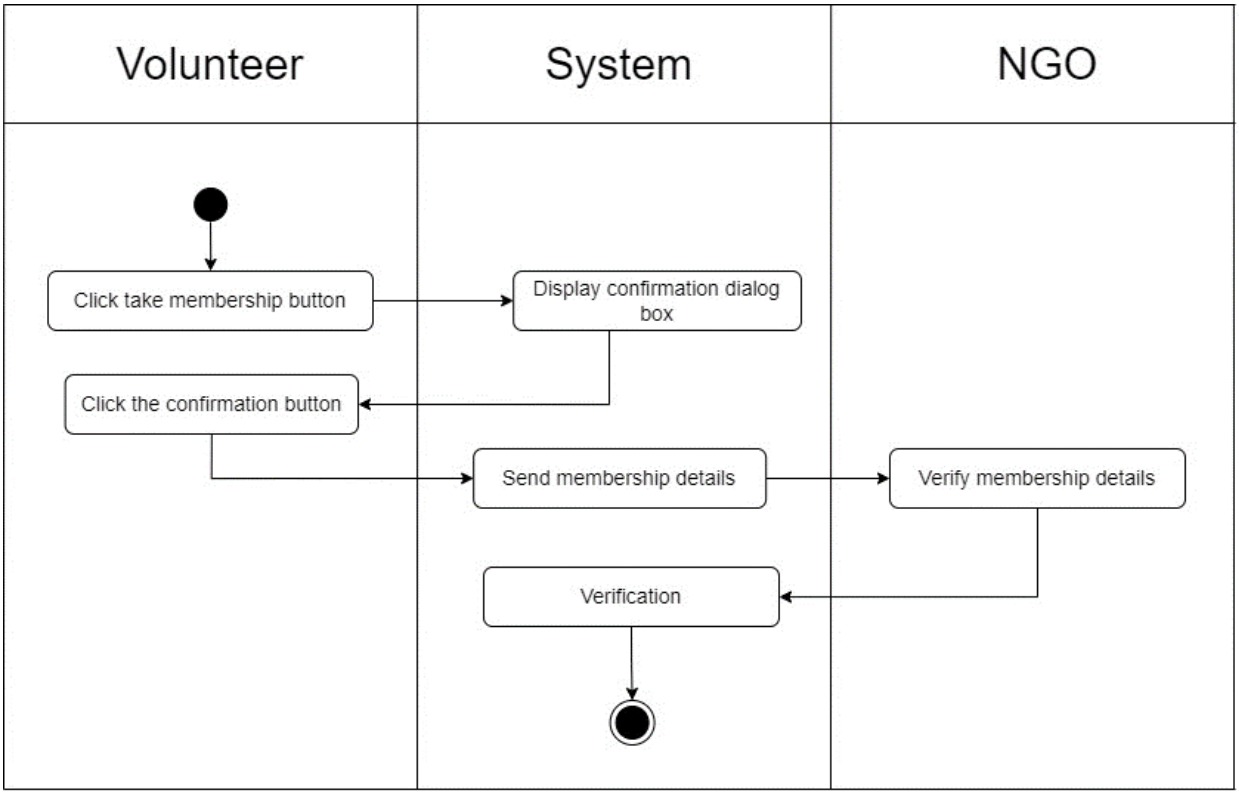


Figure 12: Register system activity diagram

Take Membership Activity



Donation Activity

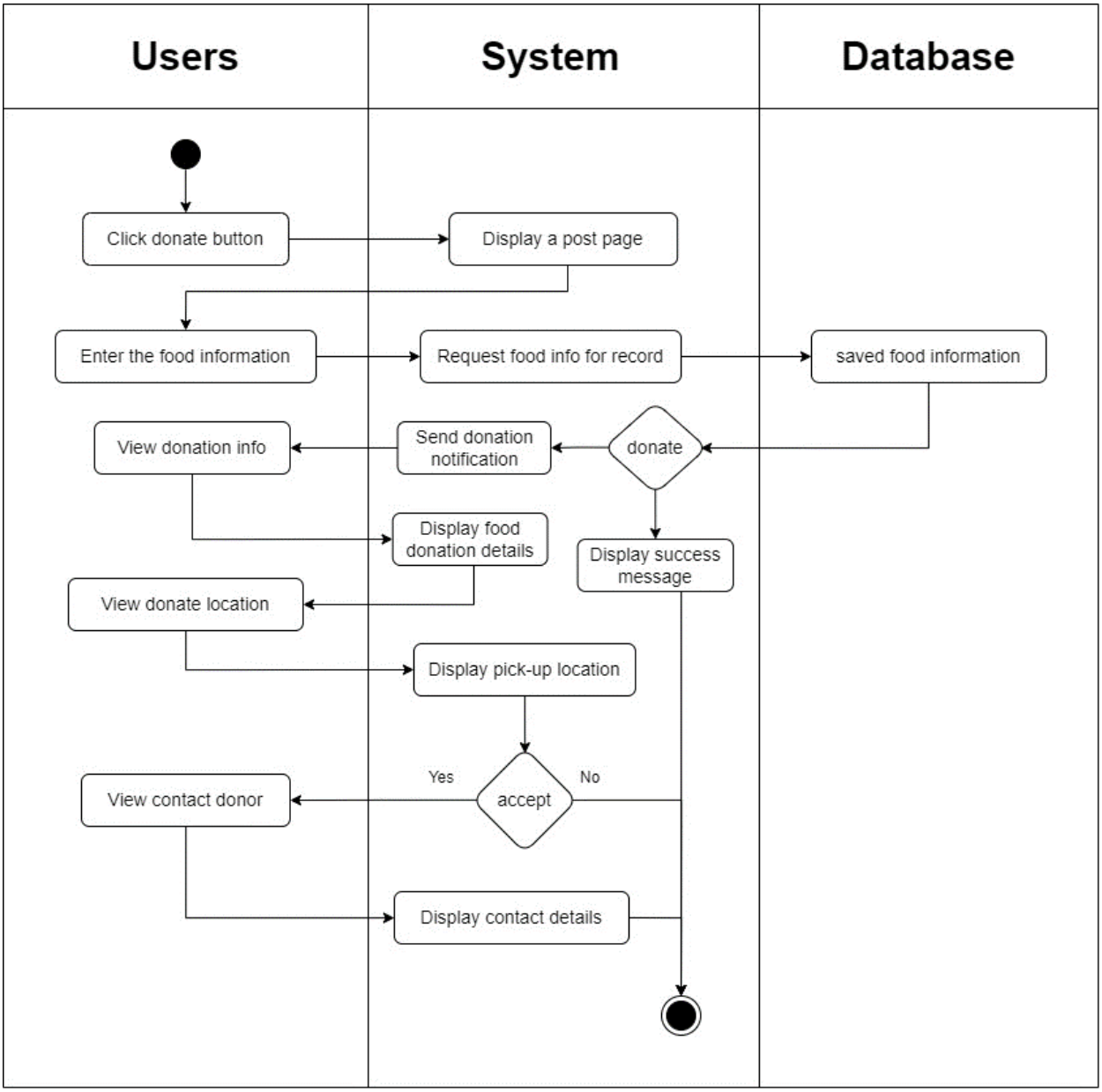


Figure 13: Food donation system activity diagram

View History Activity

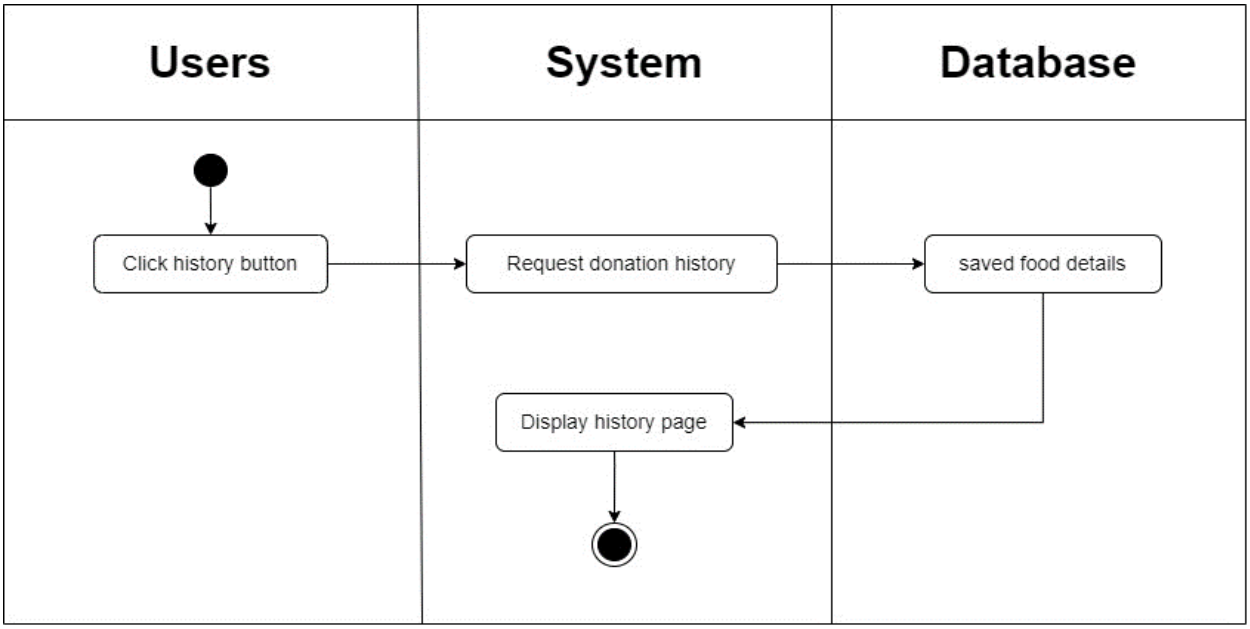


Figure 14: View the history system activity diagram

Donation Rating Activity

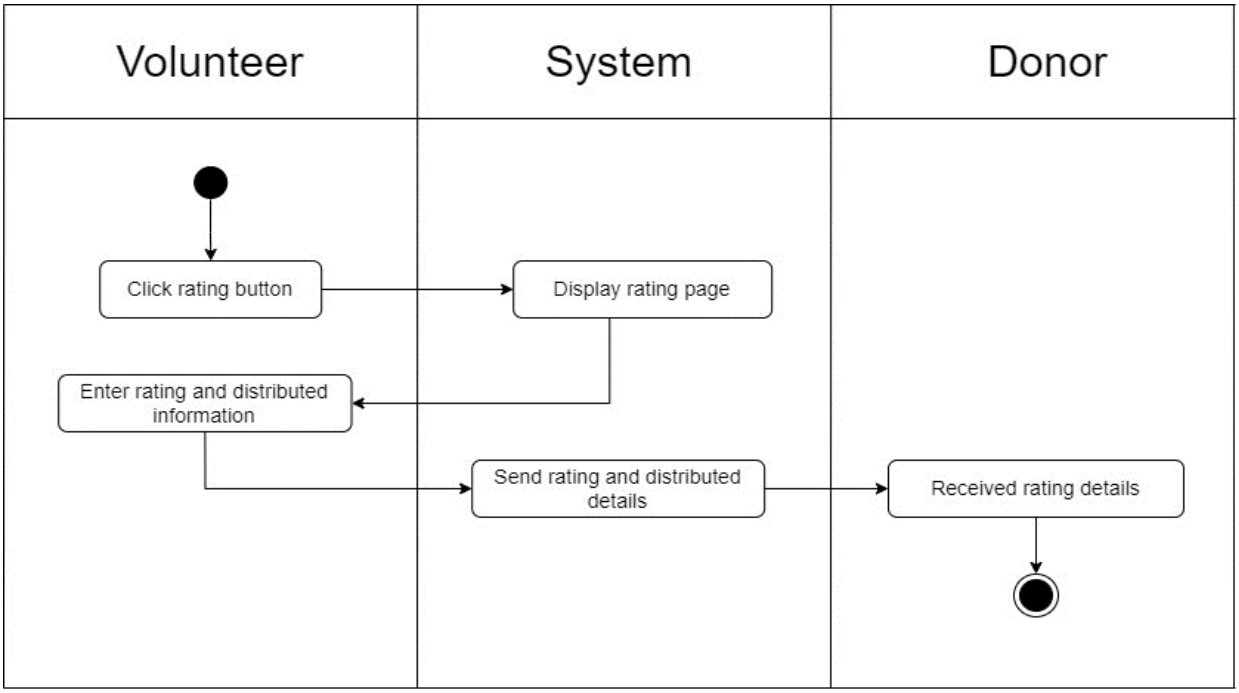


Figure 15: Rating donation system activity diagram

Complaint with Admin Activity

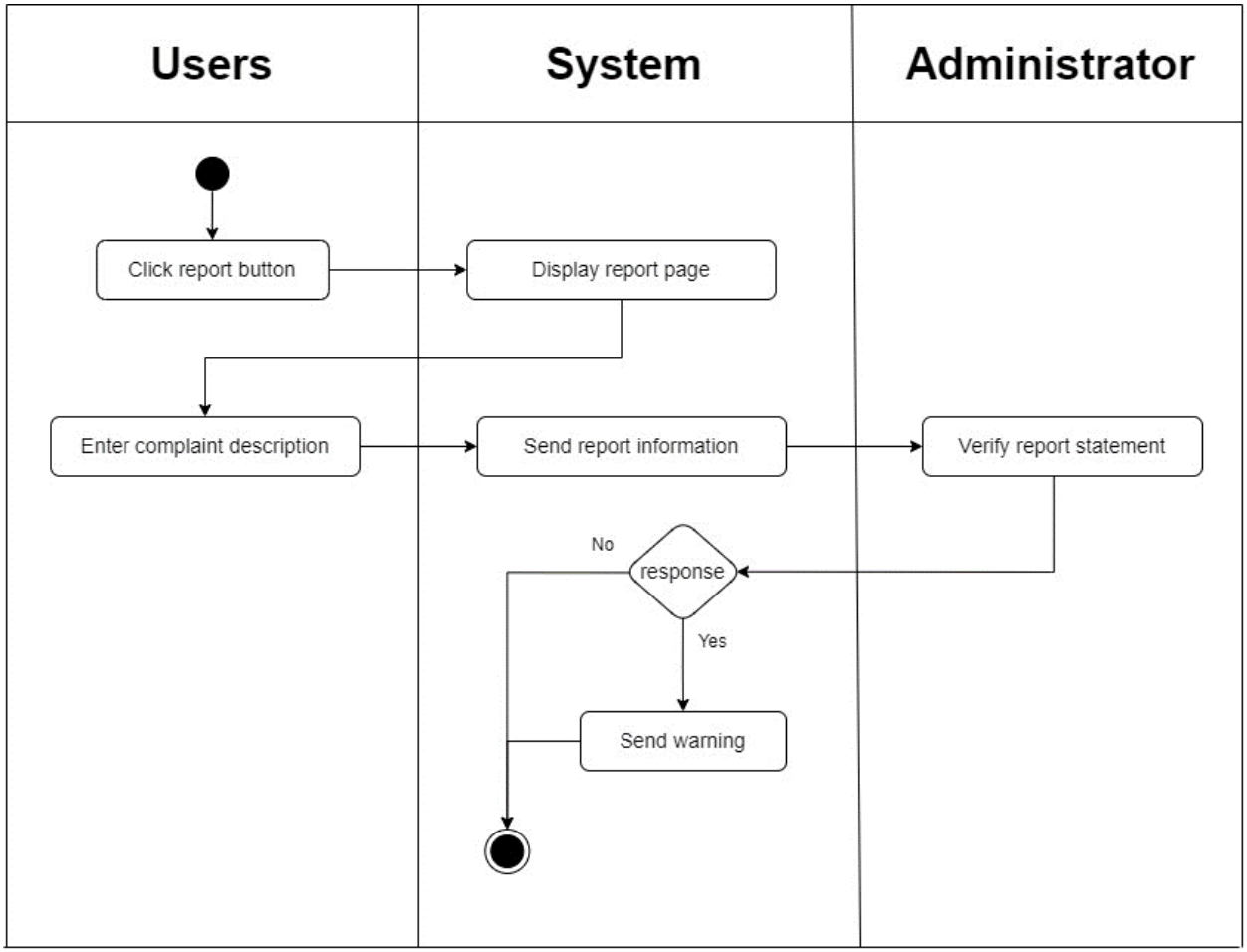


Figure 16: Complaint with admin system activity diagram

Logout Activity

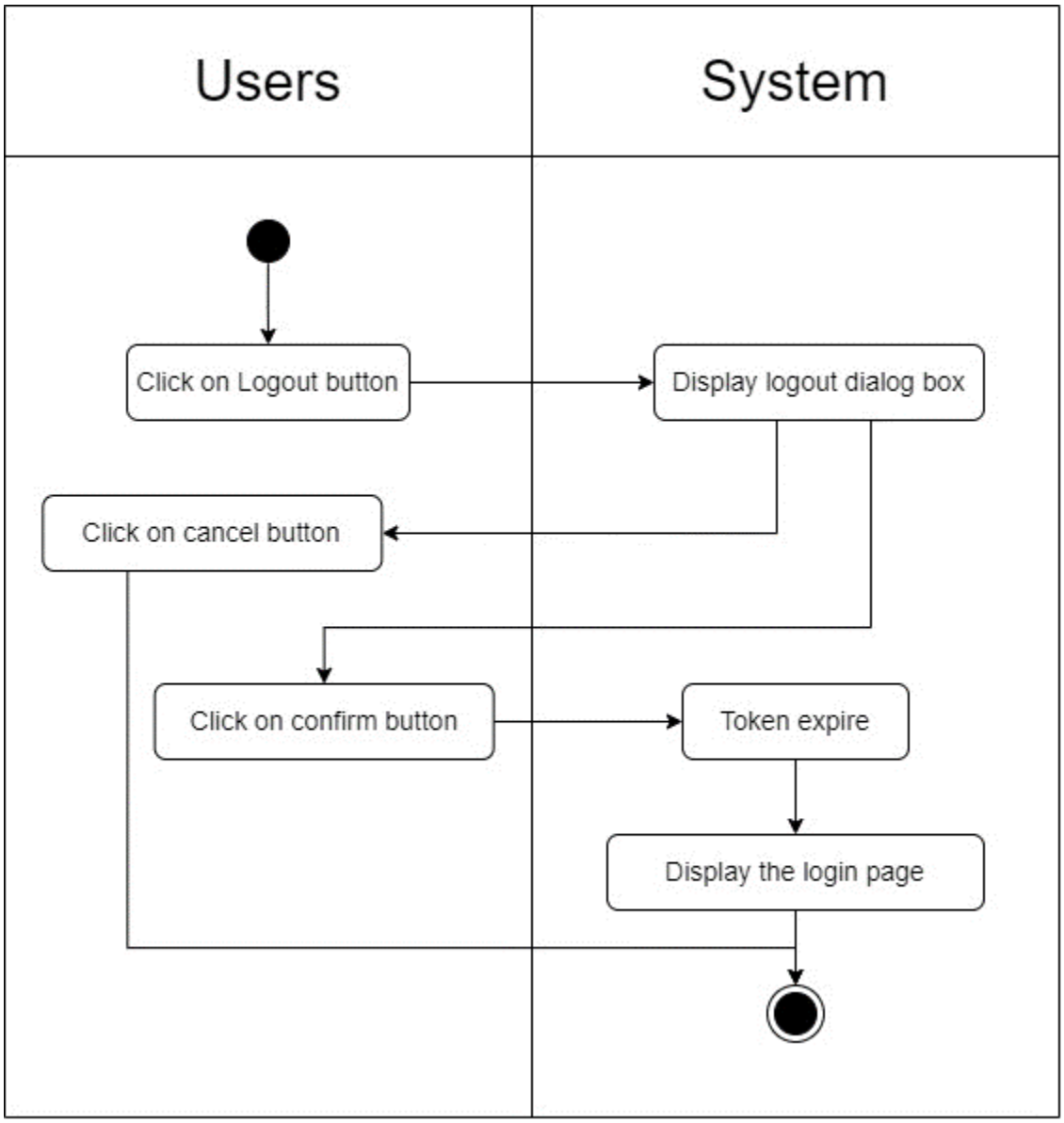


Figure 17: Logout system activity diagram

Entity Relationship Diagram (ERD)

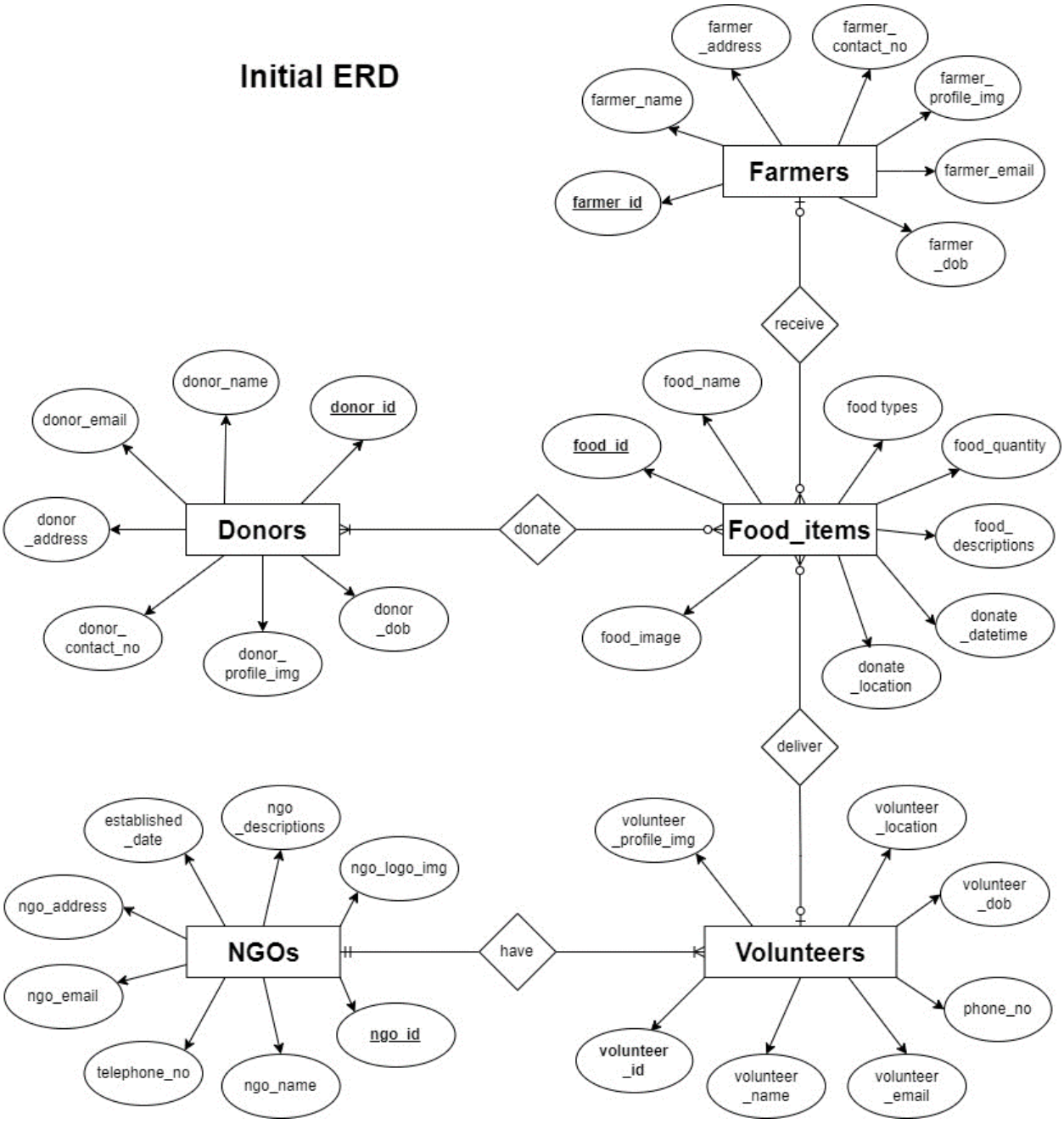
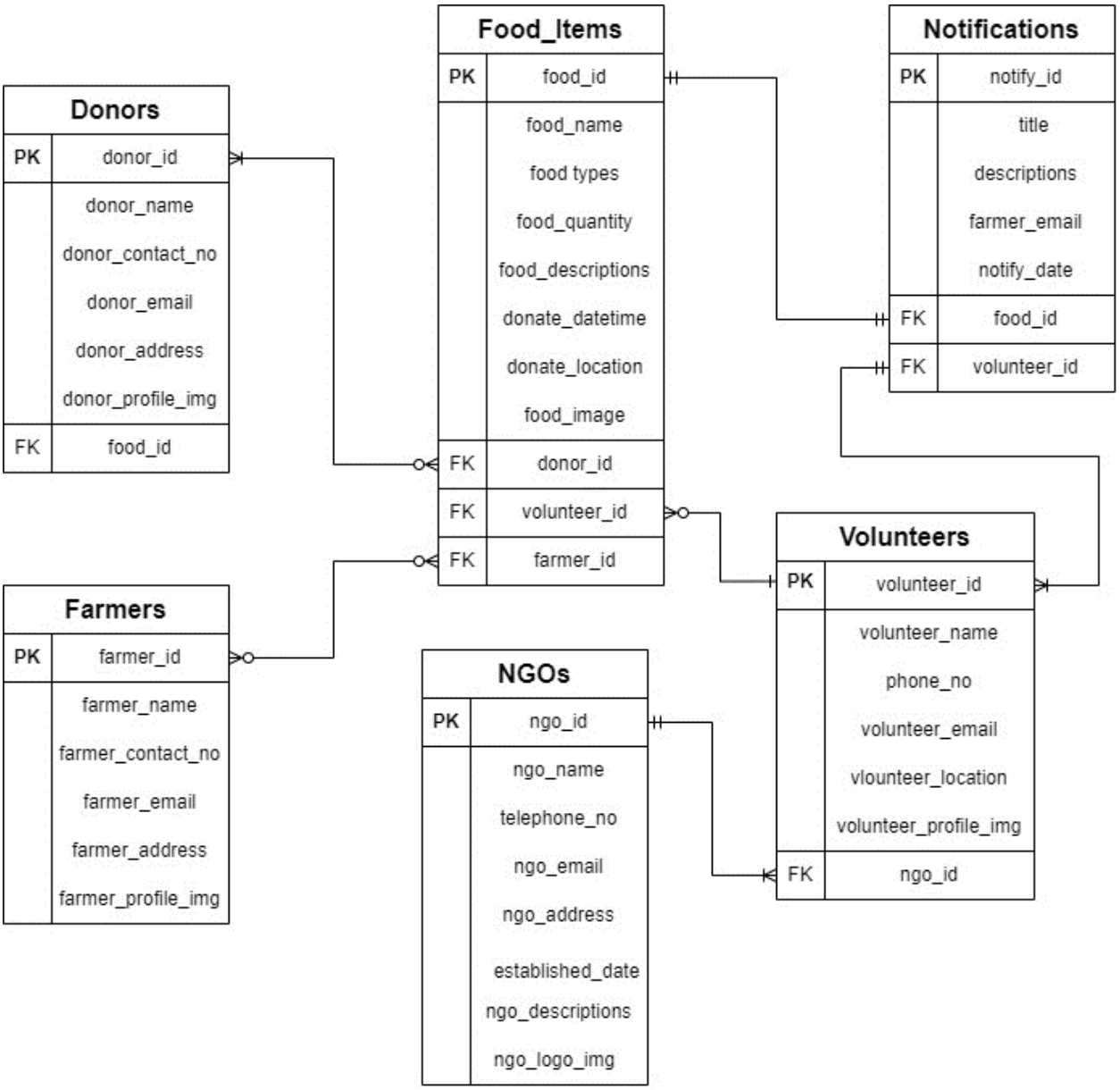


Figure 18: Initial RED


Food_Items
PK food_id
food_name
food types
food_quantity
food_descriptions
donate_datetime
donate_location
food_image
FK donor_id
FK volunteer_id
FK farmer_id
Notifications
PK notify_id
title
descriptions
farmer_email
notify_date
FK food_id
FK volunteer_id
Donors
PK donor_id
donor_name
donor_contact_no
donor_email
donor_address
donor_profile_img
FK food_id
Volunteers
PK volunteer_id
volunteer_name
phone_no
volunteer_email
vlounteer_location
volunteer_profile_img
FK ngo_id
Farmers
PK farmer_id
farmer_name
farmer_contact_no
farmer_email
farmer_address
farmer_profile_img
NGOs
PK ngo_id
ngo_name
telephone_no
ngo_email
ngo_address
established_date
ngo_descriptions
ngo_logo_img

3 Normalization

Normalization is the process of removing anomalies and reducing data redundancy. Creating tables and supporting these in a manner that is designed to protect the data and increase the database's flexibility by removing duplication and unreliable dependence is necessary. We cannot get the desired outcome when developing entities and their interactions. Anomalies can be added, updated, and removed.

The normalization process for all the attributes is done as shown in the steps given below:

3.1 Un-Normalization Form (UNF)

The process of the un-normalization is given below:

- Write all attribute names from the initial ERD with the name of the entry.
- Choose a suitable unique identifier for this entity.
- Show repeating group within {}.
- From the list, attributes are selected as a primary key and it means to be represented with an underline.

Let’s review the un-normalization form of the table again.

This is the list of UNF: -

The NGO ID is the unique identifier of the Un-normalized form so,

NGOs(<u>Nog_id</u>, ngo_name, telephone_no, nog_email, ngo_address, established_date, ngo_descriptions, ngo_logo_img, {Volunteer_id, volunteer_name, volunteer_email, phone_no, volunteer_dob, volunteer_location, volunteer_profile_img, {Food_id, food_name, food_type, food_quantity, food_descriptions, donate_datetime, donate_location, food_image, Donor_id, dnonr_name, donor_contact_no, donor_email, donor_address, donor_profile_img, Farmer_id, farmer_name, farmer_contact_no, farmer_email, farmer_address, farmer_profile_img}}**)**

The repeating group is represented inside a curly bracket as shown. We have 3 repeating out of which one is the repeating group.

3.2 First Normalization Form (1NF)

The process of the UNF to 1NF is given below:

- The repeating groups should be removed to separate the relations.
- 1NF restriction is built into the relation model.
- The advantage of 1NF is simplicity and uniform access.

All the repeating data are Foods, Donors, and Farmer's details to remove the all-repeating group.

a) **NGOs (**Nog_id, ngo_name, telephone_no, nog_email, ngo_address, established_date, ngo_descriptions, ngo_logo_img**)**

❖ **Volunteer_details (**Ngo_id***,** Volunteer_id, volunteer_name, volunteer_email, phone_no, volunteer_dob, volunteer_location, volunteer_profile_img, {Food_id, food_name, food_type, food_quantity, food_descriptions, donate_datetime, food_location, food_image, Donor_id, dnonr_name, donor_contact_no, donor_email, donor_address, donor_profile_img, Farmer_id, farmer_name, farmer_contact_no, farmer_email, farmer_address, farmer_profile_img}**)**

The repeating groups should be removed to separate the relations.

b) **Volunteer_details (**Ngo_id***,** Volunteer_id, volunteer_name, volunteer_email, phone_no, volunteer_dob, volunteer_location, volunteer_profile_img**)**

c) **Food_details(**Ngo_id***,** Volunteer_id*, Food_id, food_name, food_type, food_quantity, food_descriptions, donate_datetime, donate_location, food_image, Donor_id, dnonr_name, donor_contact_no, donor_email,

donor_address, donor_profile_img, Farmer_id, farmer_name, farmer_contacta_no, farmer_email, farmer_address, farmer_profile_img**)**

List of **1NF** tables:

NGOs (Nog_id, ngo_name, telephone_no, nog_email, ngo_address, established_date, ngo_descriptions, ngo_logo_img**)**

Volunteer_details (Ngo_id***,** Volunteer_id, volunteer_name, volunteer_email, phone_no, volunteer_dob, volunteer_location, volunteer_profile_img**)**

Food_details(Ngo_id***,** Volunteer_id*, Food_id, food_name, food_type, food_quantity, food_descriptions, donate_datetime, donate_location, food_image, Donor_id, dnonr_name, donor_contact_no, donor_email, donor_address, donor_profile_img, Farmer_id, farmer_name, farmer_contact_no, farmer_email, farmer_address, farmer_profile_img**)**

3.2 Second Normalization Form (2NF)

While converting the table to 2NF we should look at the table with a compositive key and review whether each non-key attribute is a dependency on part of the key or all the keys. If the attributes are partially dependent on a composite key, then we must remove the partial key and dependents to a new table.

The process of the 1NF to 2NF is given below:

- The relation is in 2NF.
- The table must be in 1NF.
- Tables should not have a partial dependency.
- All non-key attributes are fully functional and depend on the primary key because it is located within 1NF and not only a part of the primary key.
- A separate Relation should be defined for any attributes that are fully dependent on one attribute of the composite identification.

- Data redundancy is generated by partial functional dependency on an identifier; hence they should be avoided.

Steps to transform into 2NF.

- Identified the functional dependencies in 1NF.
- Each determinant should represent the new relation's primary key.
- Declare as non-key attributes those attributes that are dependent on a specific determinant respecting that determinant.

Identifying Practical Functional Dependency

List every possibility of the composite determinant's primary key and its parts (primary key)

The NGOs table is not a partial dependency, so it is automatically 2NF because this table is not part of the key but the **Volunteers** and **Food_Details** tables are partial dependencies, so it is normalizing the 2NF.

a) **NGOs (**Nog_id, ngo_name, telephone_no, nog_email, ngo_address, established_date, ngo_descriptions, ngo_logo_img**)**

Symbolic representations:

Again, check the **Volunteer_details** table:

❖ **Volunteer_Details (**Ngo_id***,** <u>Volunteer_id</u>, volunteer_name, volunteer_email, phone_no, volunteer_dob, volunteer_location, volunteer_profile_img**)**

Ngo_id*, Volunteer_id ⟶

Ngo_id* ⟶

Volunteer_id ⟶ volunteer_name, volunteer_email, phone_no, volunteer_dob, volunteer_location, volunteer_profile_img

b) **Volunteer_details** (Ngo_id***,** Volunteer_id*)
c) **Volunteers** (Volunteer_id, volunteer_name, volunteer_email, phone_no, volunteer_dob, volunteer_location, volunteer_profile_img)

Again, check the **Food _details** table:

- **Food_details(**Ngo_id***,** Volunteer_id*, Food_id, food_name, food_type, food_quantity, food_descriptions, donate_datetime, donate_location, food_image, Donor_id, dnonr_name, donor_contact_no, donor_email, donor_address, donor_profile_img, Farmer_id, farmer_name, farmer_contact_no, farmer_email, farmer_address, farmer_profile_img**)**

Ngo_id *, Volunteer_id*, Food_id ⟶

Ngo_id*, Food_id ⟶

Volunteer_id*, Food_id ⟶

Ngo_id*, Volunteer_id* ⟶

Ngo_id* ⟶

Volunteer_id* ⟶

Food_id ⟶ food_name, food_type, food_quantity, food_descriptions, donate_datetime, food_location, food_image, Donor_id, dnonr_name, donor_contact_no, donor_email, donor_address, donor_profile_img, Farmer_id, farmer_name, farmer_contacta_no, farmer_email, farmer_address, farmer_profile_img.

d) **Food_details (**Ngo_id *, Volunteer_id*, Food_id***)**

e) **Foods (**Food_id, food_name, food_type, food_quantity, food_descriptions, donate_datetime, donate_location, food_image, Donor_id, dnonr_name, donor_contact_no, donor_email, donor_address, donor_profile_img, Farmer_id, farmer_name, farmer_contacta_no, farmer_email, farmer_address, farmer_profile_img**)**

List of the **2NF** tables:

NGOs (Nog_id, ngo_name, telephone_no, nog_email, ngo_address, established_date, ngo_descriptions, ngo_logo_img**)**

Volunteer_details (Ngo_id***,** Volunteer_id*)

Volunteers (Volunteer_id, volunteer_name, volunteer_email, phone_no, volunteer_dob, volunteer_location, volunteer_profile_img)

Food_details (Ngo_id *, Volunteer_id*, Food_id***)**

Foods (Food_id, food_name, food_type, food_quantity, food_descriptions, donate_datetime, donate_location, food_image, Donor_id, dnonr_name, donor_contact_no, donor_email, donor_address, donor_profile_img, Farmer_id, farmer_name, farmer_contact_no, farmer_email, farmer_address, farmer_profile_img**)**

3.3 Third Normalization Form (3NF)

The process of the **2NF** to **3NF** is given below:

Create one relation for each determinant in transitive dependency.

- The table must be in 2NF.
- Inside this new table, a primary key must be selected.
- Identifying dependencies between non-prime key attributes in a table is step one.

- The interdependent non-prime key attributes from the original table should be normalized to create a new table.
- In the original table, this primary key is converted to a foreign key.

The entity has no transactive dependency that is already 3NF. The **Volunteer_details** and **Food_details** do have not any non-key attributes, so it has no transactive dependency so automatically 3NF.

a) **Volunteer_details** (Ngo_id***,** Volunteer_id*)
b) **Food_details (**Ngo_id *, Volunteer_id*, Food_id***)**

A transitive dependency happens when two non-primary key attributes are related to one another. Check the transactive dependency of another table because it has more than two or more non-key attributes. The table has more than two non-kye attributes so that checks the transitive dependency as given below:

NGOs (Nog_id, ngo_name, telephone_no, nog_email, ngo_address, established_date, ngo_descriptions, ngo_logo_img**)**

Volunteers (Volunteer_id, volunteer_name, volunteer_email, phone_no, volunteer_dob, volunteer_location, volunteer_profile_img)

Foods (Food_id, food_name, food_type, food_quantity, food_descriptions, donate_datetime, donate_location, food_image, Donor_id, dnonr_name, donor_contact_no, donor_email, donor_address, donor_profile_img, Farmer_id, farmer_name, farmer_contact_no, farmer_email, farmer_address, farmer_profile_img)

Check the transactive dependency:

For **NGOs: -**

- **NGOs (**Nog_id, ngo_name, telephone_no, nog_email, ngo_address, established_date, ngo_descriptions, ngo_logo_img**)**

Ngo_id → ngo_name →

Ngo_id → telephone_no →

Ngo _id → nog_email →

Ngo _id → ngo_address →

Ngo _id → established_date →

Ngo _id → ngo_descriptions →

Ngo _id → ngo_logo_img →

c) NGOs (Nog_id, ngo_name, telephone_no, nog_email, ngo_address, established_date, ngo_descriptions, ngo_logo_img**)**

The NGOs table has no transactive dependency because it was impossible to give the values of other data, so it is also automatically 3NF.

For **Volunteers**: -

- **Volunteers** (Volunteer_id, volunteer_name, volunteer_email, phone_no, volunteer_dob, volunteer_location, volunteer_profile_img)

Volunteer_id → volunteer_name →

Volunteer_id → volunteer_email →

Volunteer_id → phone_no →

Volunteer_id → Duration →

Volunteer_id → volunteer_dob →

Volunteer_id → volunteer_location →

Volunteer_id → volunteer_profile_img →

d) **Volunteers** (Volunteer_id, volunteer_name, volunteer_email, phone_no, volunteer_dob, volunteer_location, volunteer_profile_img)

The Volunteers table has no transactive dependency because it was impossible to give the values of other data, so it is also automatically 3NF. The non-key doesn't give the non-key attributes.

Foods (Food_id, food_name, food_type, food_quantity, food_descriptions, donate_datetime, donate_location, food_image, Donor_id, dnonr_name, donor_contact_no, donor_email, donor_address, donor_profile_img, Farmer_id, farmer_name, farmer_ contact _no, farmer_email, farmer_address, farmer_profile_img)

The Food_id can be given the Food items detail, Donor_id and Farmer_id. So: -

Food_id → food_name, food_type, food_quantity, food_descriptions, donate_datetime, donate_location, food_image,

Then,

Food_id → food_name →

Food_id → food_ type →

Food_id → food_ quantity →

Food_id → food_ descriptions →

Food_id → food_ datetime →

Food_id → donate_location →

Food_id → food_image →

Now,

Food_id gives the Donor_id and Farmer_id can give the Model then Model gives: -

Were,

Food_id → Donor_id → dnonr_name, donor_contact_no, donor_email, donor_address, donor_profile_img,

Donor_id → dnonr_name →

Donor_id → donor_contact_no →

Donor_id → donor_email →

Donor_id → donor_address →

Donor_id → donor_profile_img →

Again,

Food_id → Farmer_id → farmer_name, farmer_contact_no, farmer_email, farmer_address, farmer_profile_img,

Donor_id → farmer_name →

Donor_id → farmer_contact_no →

Donor_id → farmer_email →

Donor_id → farmer_address →

Donor_id → farmer_profile_img →

e) **Foods (**Food_id, food_name, food_type, food_quantity, food_descriptions, donate_datetime, donate_location, food_image, Donor_id*, Farmer_id***)**
f) **Donors (**Donor_id, dnonr_name, donor_contact_no, donor_email, donor_address, donor_profile_img**)**
g) **Farmers (**Farmer_id, farmer_name, farmer_contact_no, farmer_email, farmer_address, farmer_profile_img**)**

The Volunteer_details table is not created because all the attributes are found in the Food_details table. The final entities and attributes are given below:

List of **3NF**

Food_details (Ngo_id *, Volunteer_id*, Food_id***)**

NGOs (Nog_id, ngo_name, telephone_no, nog_email, ngo_address, established_date, ngo_descriptions, ngo_logo_img**)**

Volunteers (Volunteer_id, volunteer_name, volunteer_email, phone_no, volunteer_dob, volunteer_location, volunteer_profile_img)

Foods (Food_id, food_name, food_type, food_quantity, food_descriptions, donate_datetime, donate_location, food_image, Donor_id*, Farmer_id***)**

Donors (Donor_id, dnonr_name, donor_contact_no, donor_email, donor_address, donor_profile_img**)**

Farmers (Farmer_id, farmer_name, farmer_contact_no, farmer_email, farmer_address, farmer_profile_img**)**

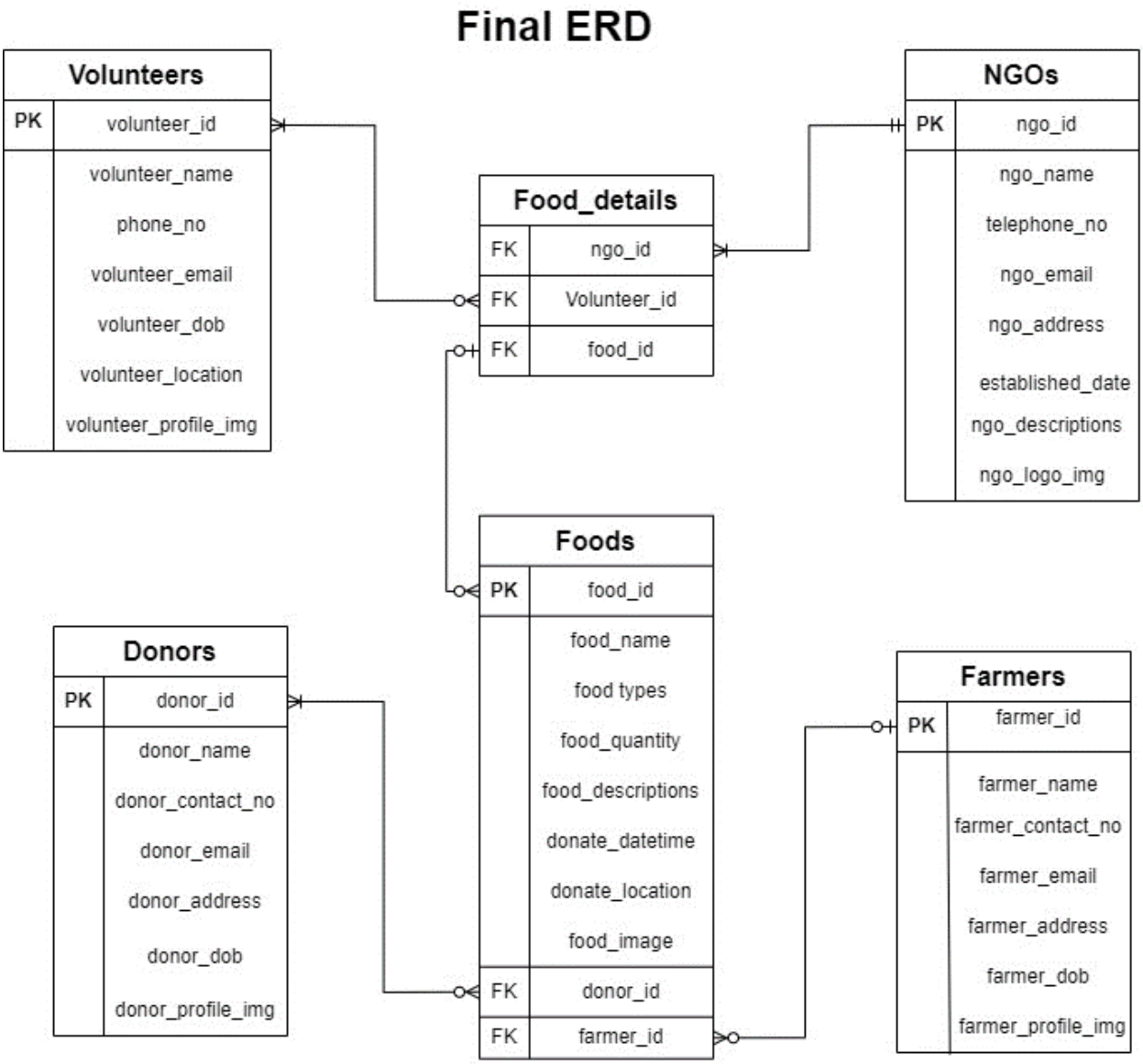
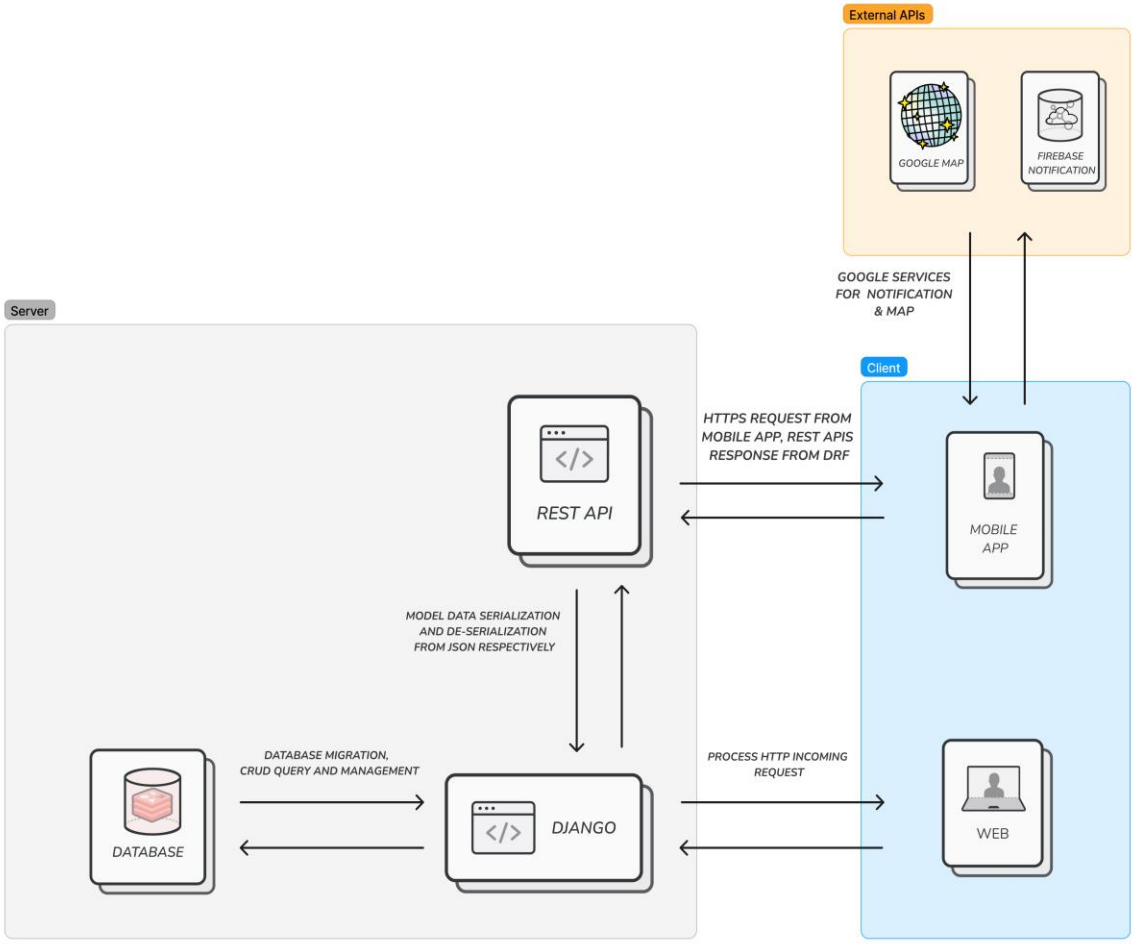


Figure 19: Final ER-Diagram

System Architecture



Wireframe

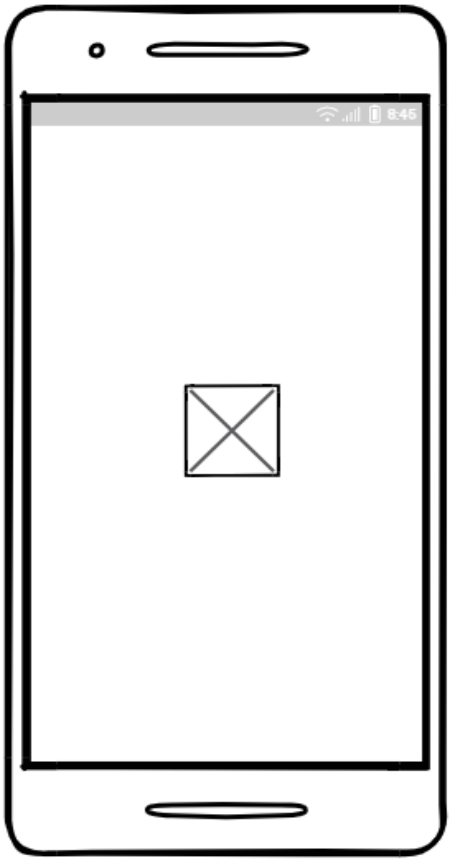


Figure 20: Splash screen mobile UI

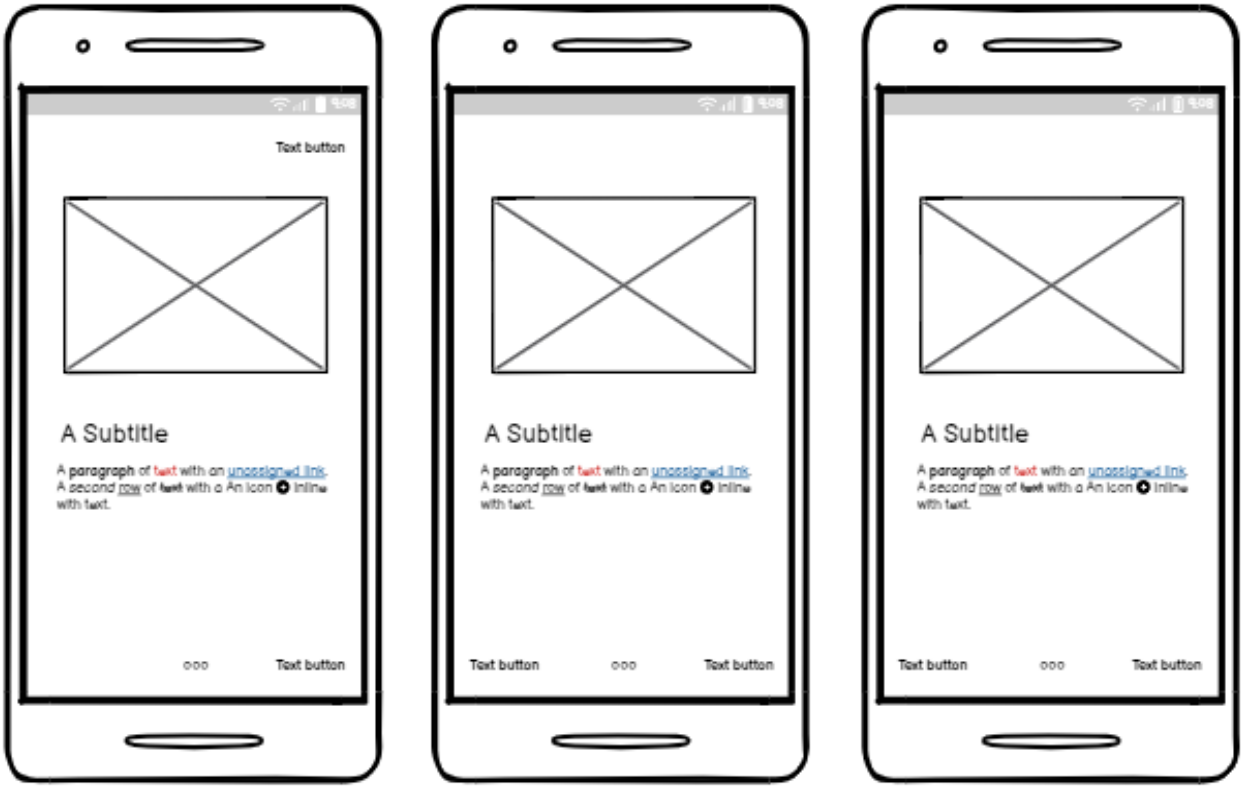


Figure 21: Walkthrough screen mobile UI

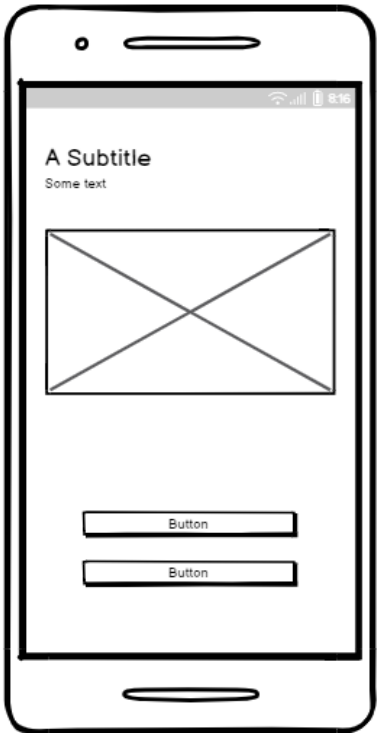


Figure 22: Welcome screen mobile UI

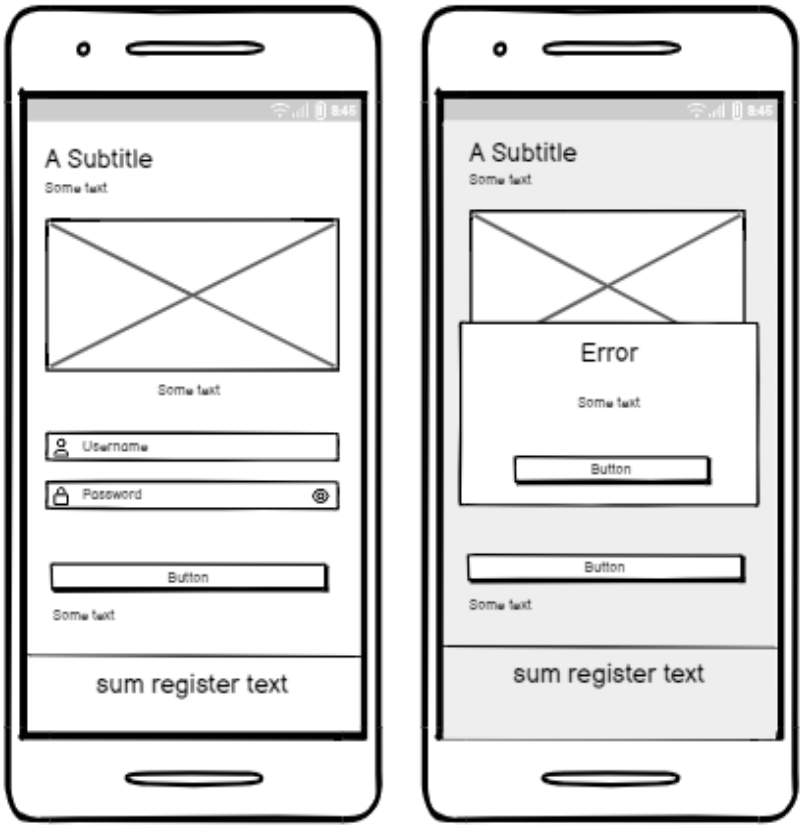


Figure 23: Login screen mobile UI

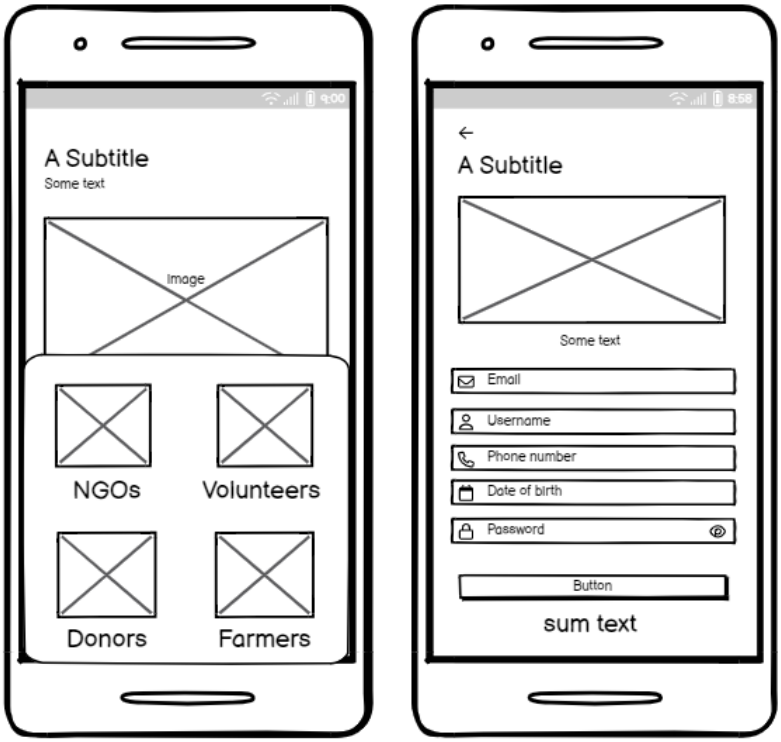


Figure 24: Register screen mobile UI

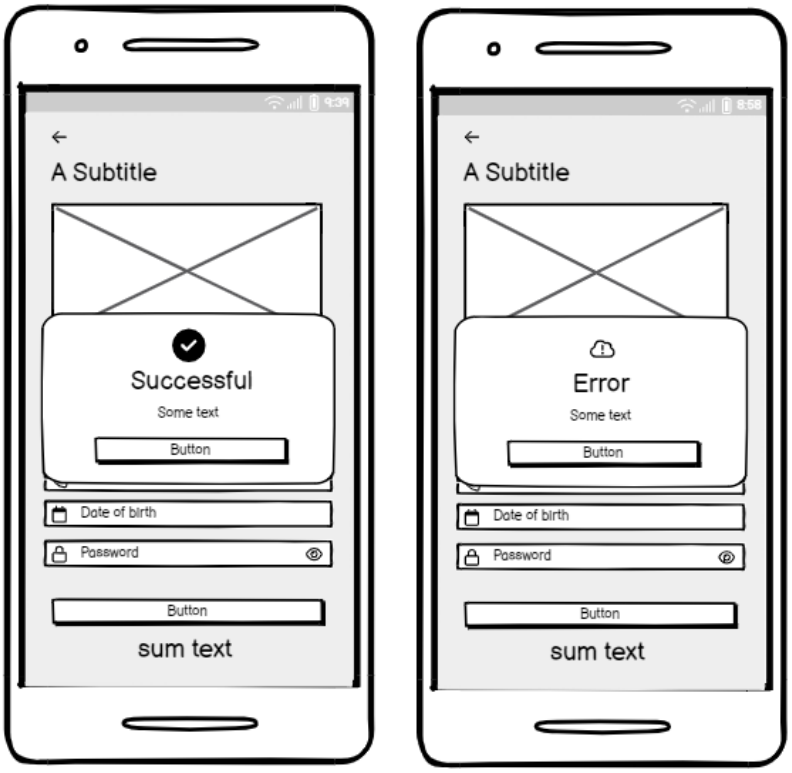


Figure 25: Register response message mobile UI

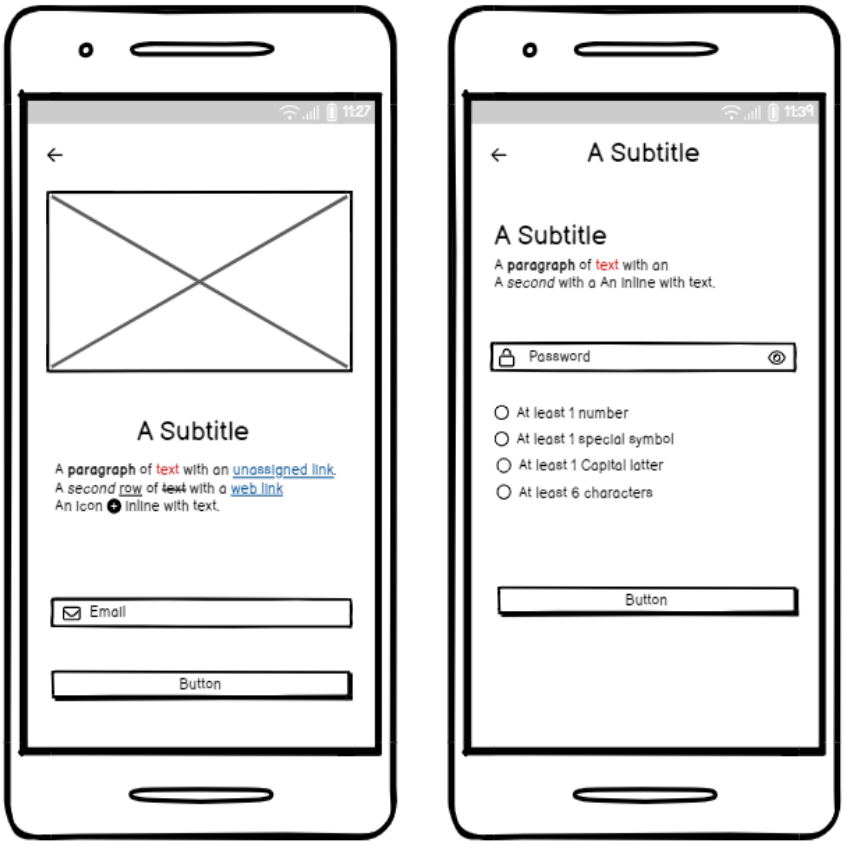


Figure 26: Forgot password screen mobile UI

Dashboard for Donors

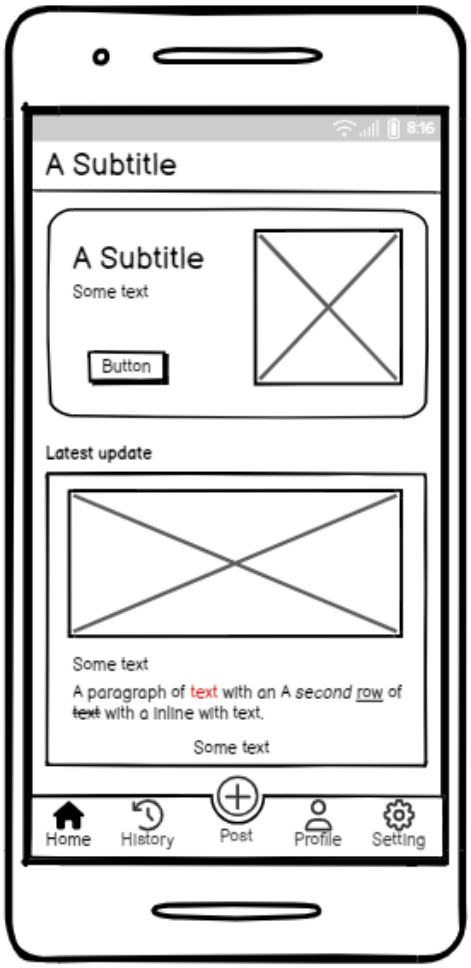


Figure 27: Donor Home screen mobile UI

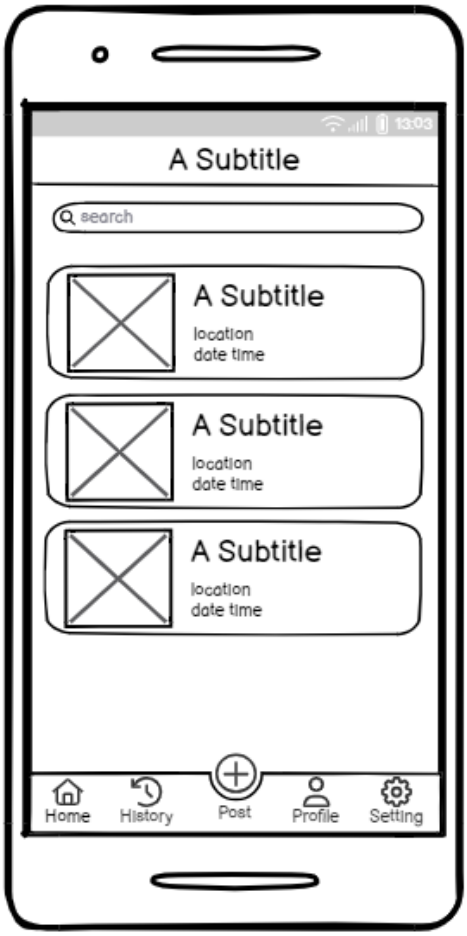


Figure 28: Donor History screen mobile UI

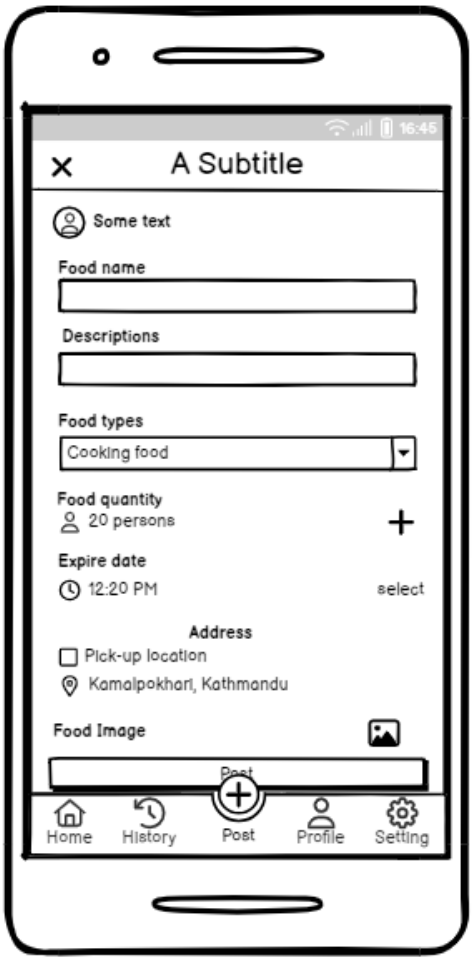


Figure 29: Donor Post/Donation screen mobile UI

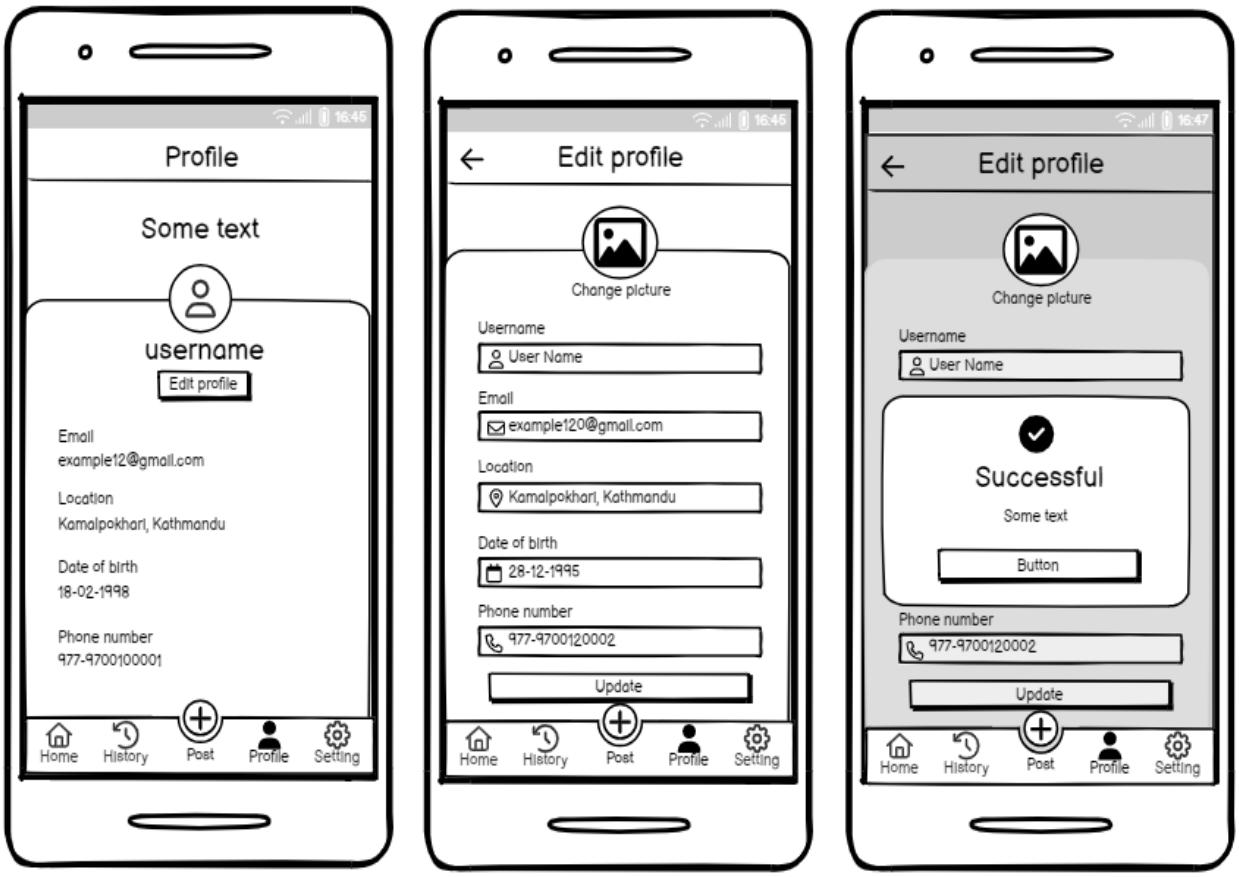


Figure 30: Donor Profile screen mobile UI

Figure 31: Donor Setting screen mobile UI

Dashboard for Volunteer and Farmer

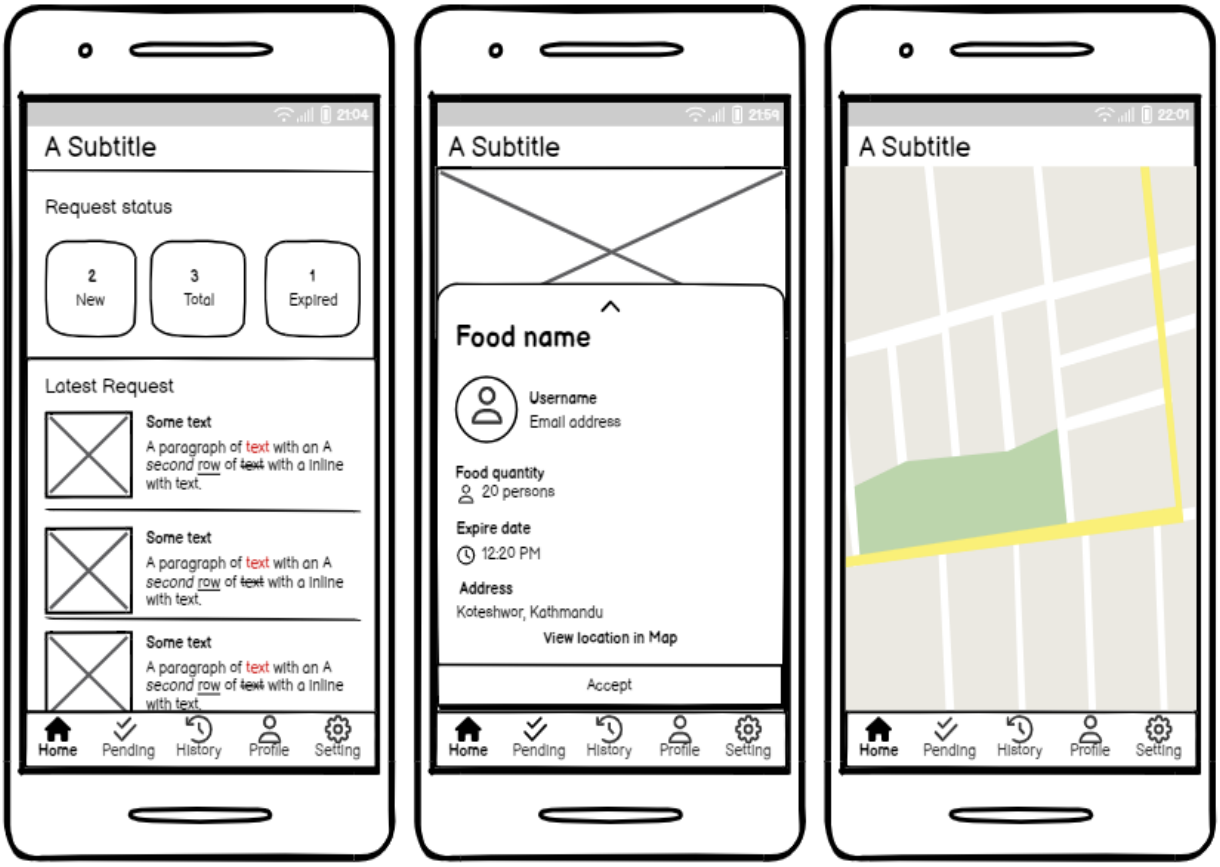


Figure 32: Volunteer Home screen UI

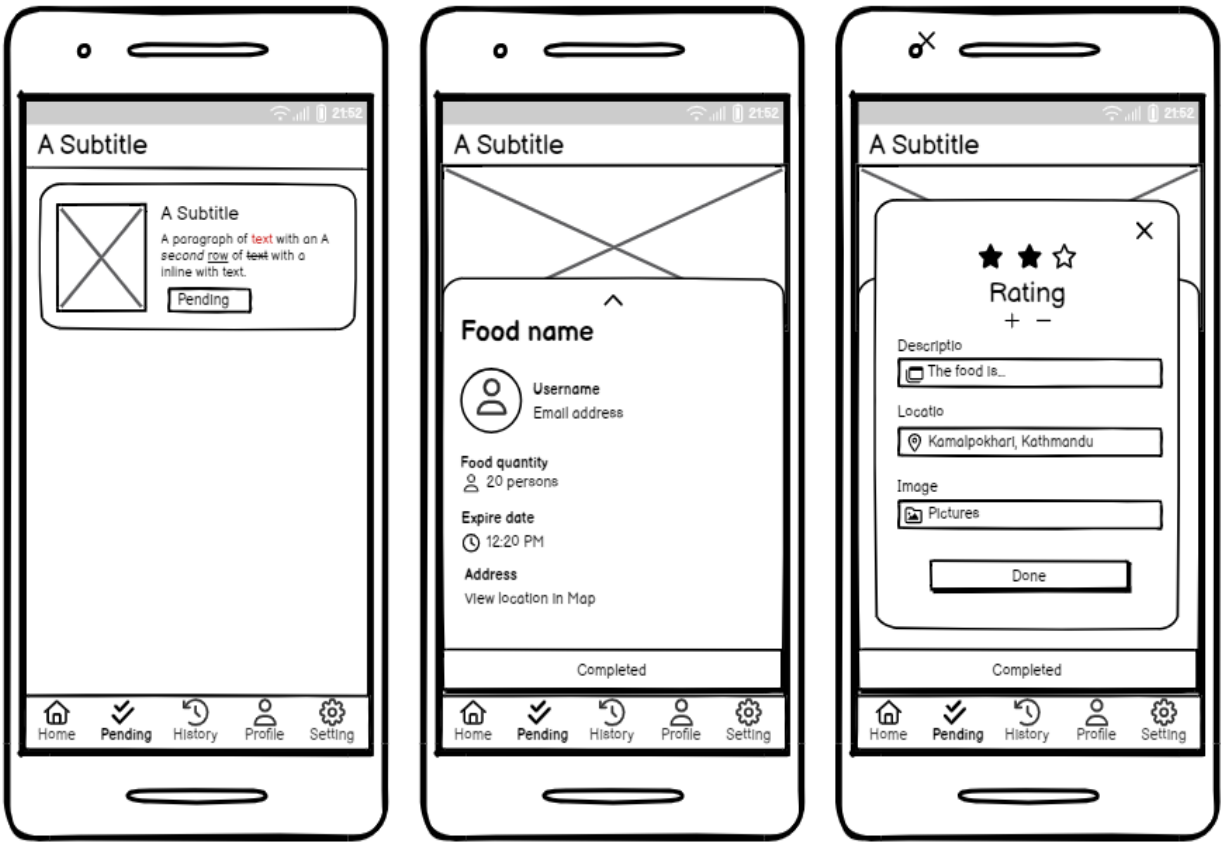


Figure 33: Volunteer Pending screen UI.

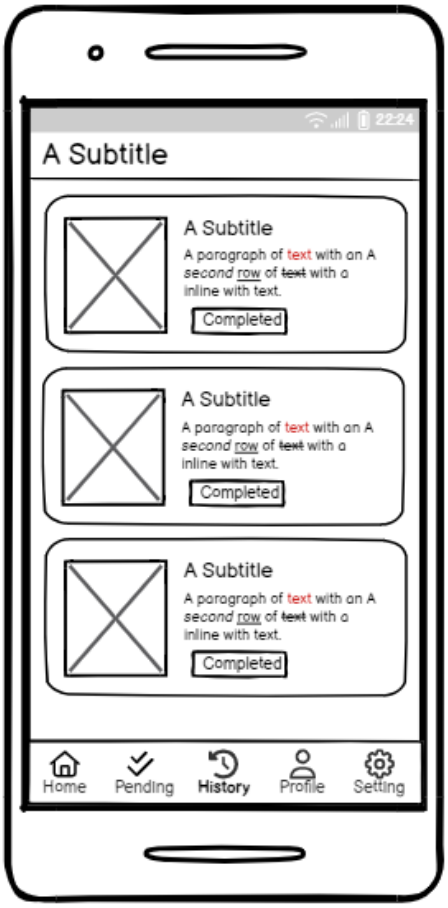


Figure 34: Volunteer History screen

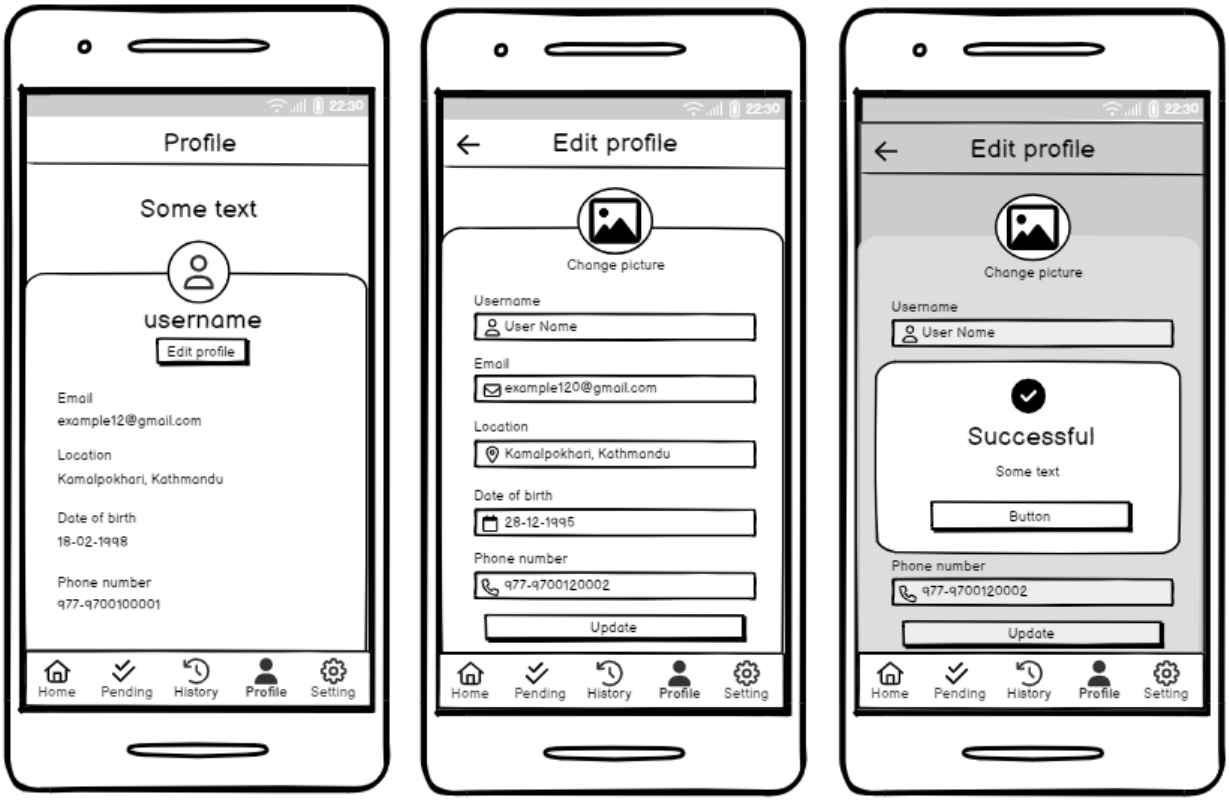


Figure 35: Volunteer Profile screen UI

Figure 36: Volunteer Setting screen UI

NOG Dashboard

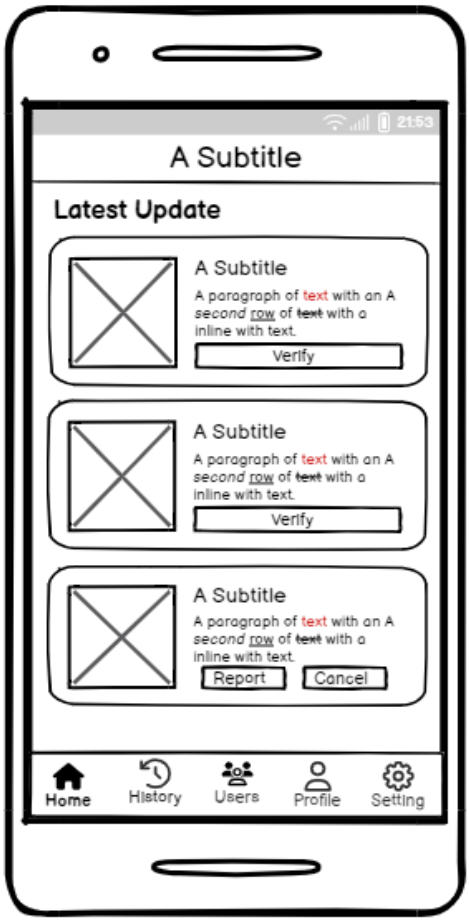


Figure 37: NGO Home screen UI

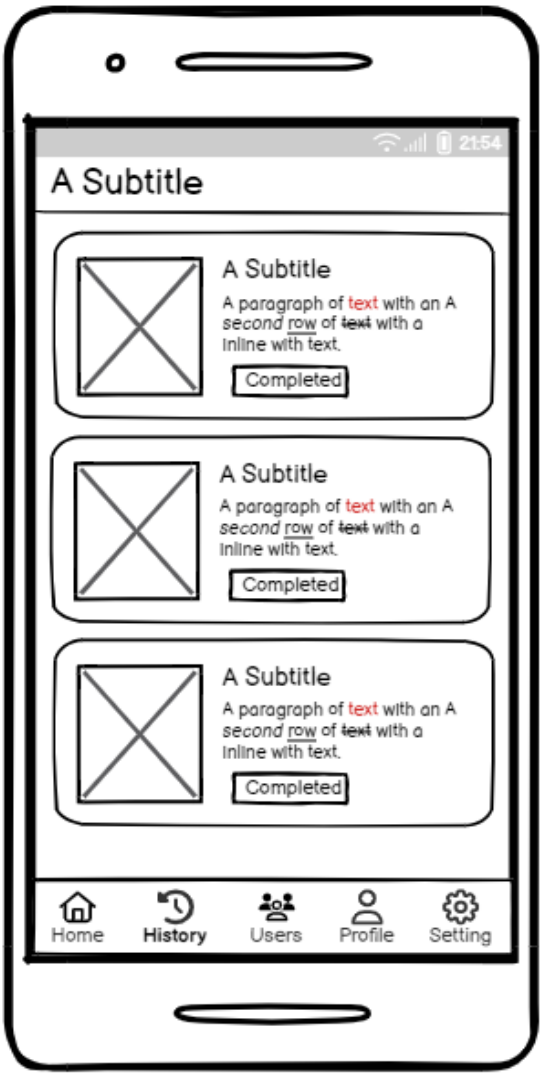


Figure 38: NGO History screen UI

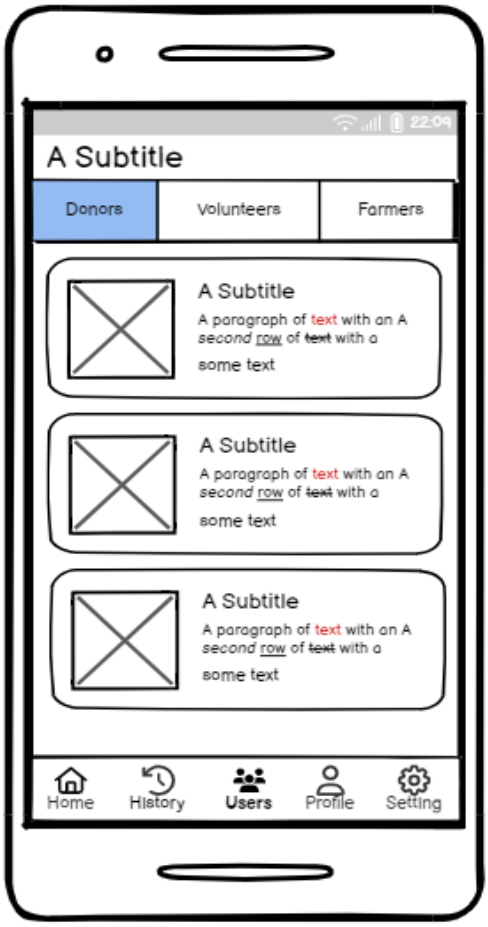


Figure 39: NGO Users screen UI

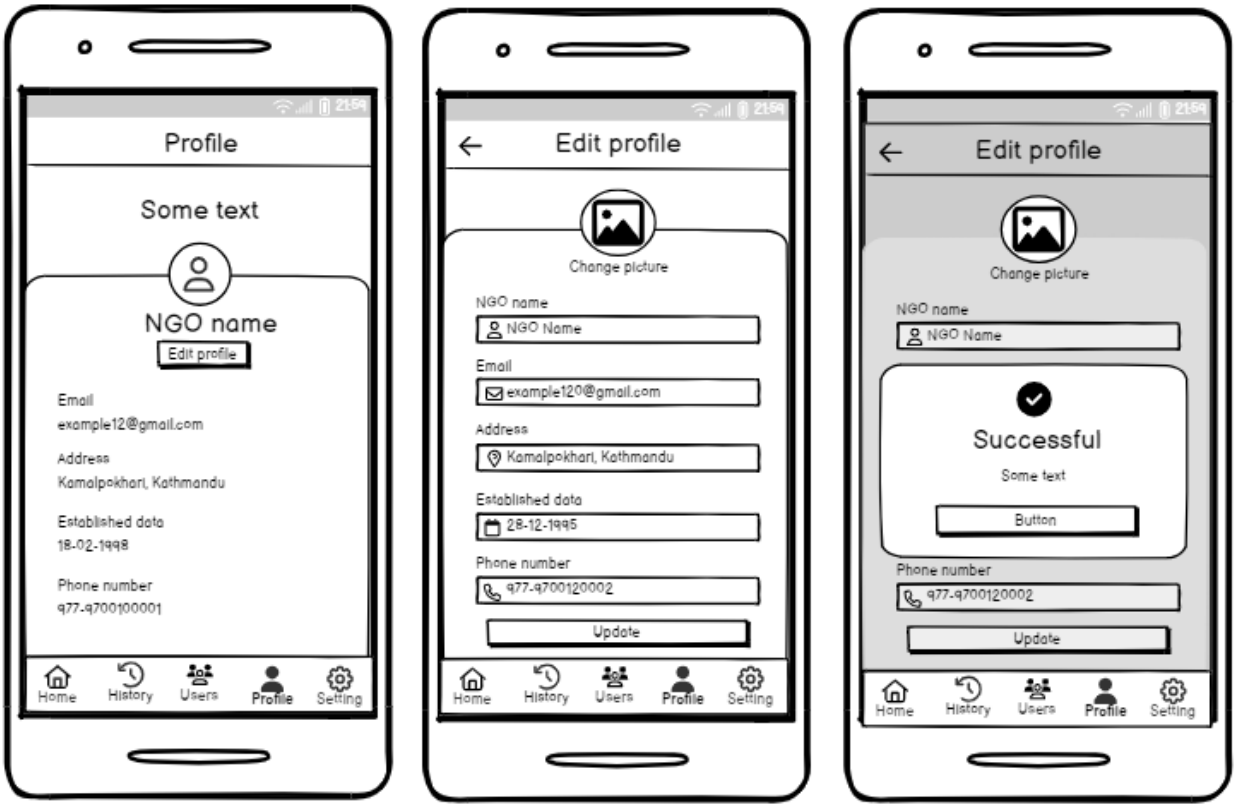


Figure 40: NGO Profile screen UI

Figure 41: NGO Setting screen UI