COMP 9900 PROJECT REPORT
CHARITY CONNECT
GROUP PENTAQUARK

Group Member:

Zou Yizhang	Scrum Master /Back-end	z5197979	z5197979@ad.unsw.edu.au
Yang Zhenyu	Back-end	z5241974	z5241974@ad.unsw.edu.au
Wang Yijie	Front-end	z3473653	z3473653@ad.unsw.edu.au
Li Weiran	Front-end	z5223041	z5223041@ad.unsw.edu.au
Zhou Yuhao	Front-end	z5227282	z5227282@ad.unsw.edu.au

Submission Date: 31st July 2021

COMP9900 Pentaquark Report

1 Overview

1.1 Background

In modern society, more and more social/political activities are moving online due to the pandemic in recent years. We are facing the situation that online platforms like websites, web-based applications are in tremendous need and its importance are gradually growing. Even portable devices could complete most of our daily tasks.

Traditional philanthropy system requires involvement of third-party institutions such as The Red Cross. These organizations work as medium for sponsors to be familiar with the philanthropy process and to be able to target their suitable charity groups. However, the involvement of intermediate organizations makes the philanthropy process more complicated in a way, as the sponsor/charity organizations would have to contact with the intermediate organizations first, providing all the required authentication materials to register and then proceeding to the next steps.

Another issue in traditional philanthropy system is that not all the charity groups or sponsor companies are familiar with this type of intermediate organizations which might cause them a huge amount of time looking or searching for intermediate organizations to provide helps/ask for helps. Traditional philanthropy pays attention on charity and ignore the sponsors. Moreover, existing philanthropy method do not address the issue through the specific requirements of people in need. Due to the global trend of online activities and communication as well as the shortcoming of traditional philanthropy, we intend to build an online platform for sponsors and charity groups to search, connect and communicate with each other.

For sponsors, our charity-connect platform is a website for all sponsors who are willing to participate in charity activities. This platform could show the need list of registered charities which allows sponsors to conveniently setup connection and receive messages from their designated charity groups, encouraging more people/companies to enroll in the philanthropy activities and attracting more charitable organizations to promote their plans.

For charity groups, our charity-connect website allows all charity groups to announce their need lists. These lists could be clothes, foods, accommodation, or some special goods, not only money, which makes the connection and communication more precise and effective. Meanwhile, the website builds the connections between sponsors and charity groups so that they could send messages to each other. All the sponsors/charity lists on the website are accessible for all the public users, which promotes the publicity of our website and then further push the establishment of online charity-sponsor connect activities.

1.2 System Architecture

The following diagram shows our system architecture. We choose a three-layer model to develop our web application, divided into presentation layer, business layer and data layer. The following section describes our project on these three layers.

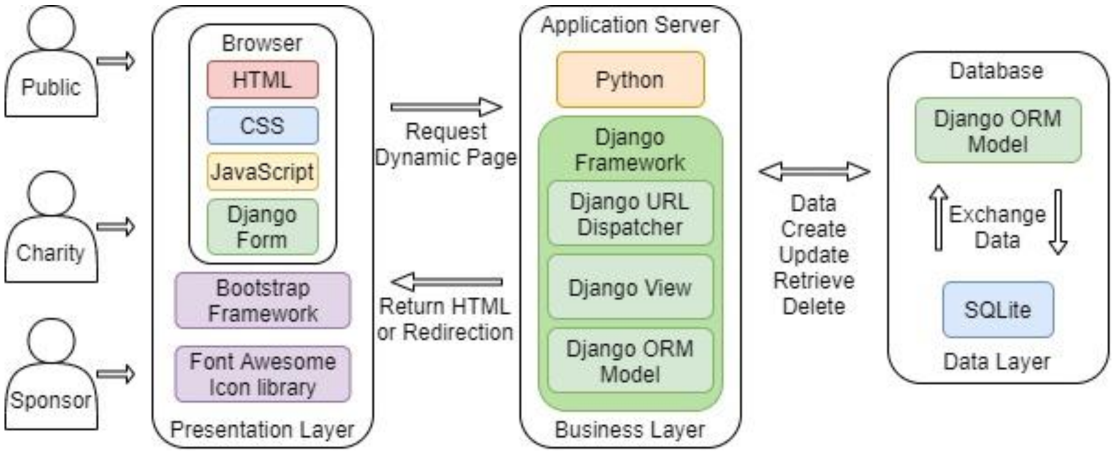


Figure 1 System Architecture with Three Layers

1.2.1 Business Layer

For the business layer, we chose to use the Django framework to create applications, build websites and data models, and use python as the development language to build the functionality.

For the Django framework, the MTV model (see the figure below) is used to keep the components loosely coupled.

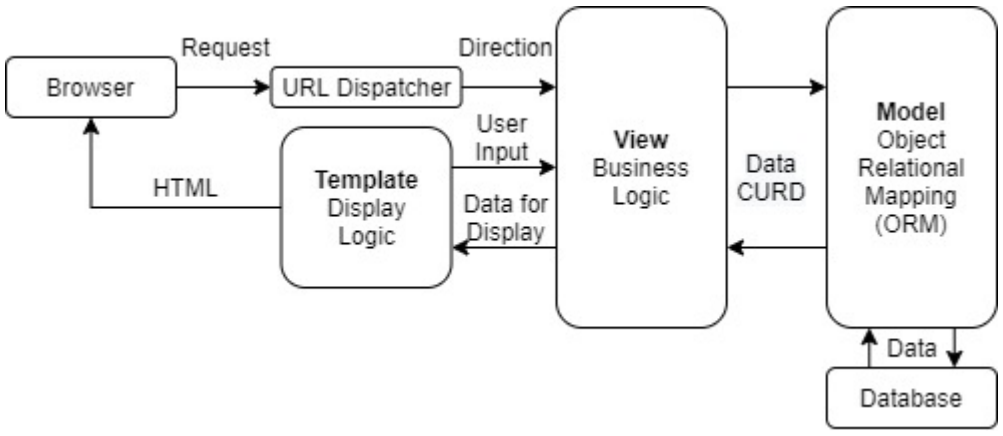


Figure 2 Django MTV Concept

The model layer is used to structure and manipulate the data of Web application, this will describe in data layer section

The view layer is used to encapsulate the logic responsible for processing a user's request and for returning the response. For this project, we have divided 8 types of view functions recorded in ***cc/views.py***, which correspond to the following functions: home page (**base(), signup(), signin(), signout()**), charity or sponsor list page (**charity_list(), sponsor_list()**), profile and edit page(**details(), edit()**), connection request or reply page (**connect(), reply_message()**), sponsor recommendation list page (**recommendation()**), charity search page (**search()**), top 10 sponsor page (**top()**) and message list page (**message_box(), show_message()**). Each of these view functions has a correspond URL pattern recorded in Django URL dispatcher ***cc/urls.py***. The view function receives requests processed by the URL dispatcher and passes the returned data or redirect requests back to the front end after being processed by Django.

The template layer is used to provides a designer-friendly syntax for rendering the information to be presented to the user, this will describe in presentation layer section.

1.2.2 Data layer

For the data layer, we chose SQLite as the database, as Django's default database, is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine with small size and is sufficient for this project.[3] For this project, the database model will be defined in ***cc/models.py***, containing key and value setting information for each data table. Django will use the ORM model to interact with the SQLite database. The details of the database and its operations will be described in the Data Model and Implementation section.

1.2.3 Presentation layer

For the presentation layer, we used Bootstrap framework, because this framework can adapt to multiple devices, and provides a powerful UI library to create a more beautiful and unified web application.[2] For HTML rendering, we use dynamic template rendered by Django correspond to each view function with GET or POST method, where the data transmissions are using forms defined in cc/forms.py, and JSON type data processed and rendered by Django in HTML. For the HTML styles, details, and icons, we used Bootdey snippets library and FontAwesome icon library.

2 Data Model and Implementation

2.1 Data schema

Charity-connect program adopt SQLite as database, which has a higher degree of compatibility with the back-end develop tools Django. Another reason for choosing SQLite is that the portability and flexibility of it are more suitable for small projects. Overview, there are seven tables are created in the database, which are user, charity, provide, sponsor, need, message, and connect. In the system, charity, provide, sponsor, and need have relationship with user table. These five tables record all independent information related with user. Other two tables are built for user inter-actions.

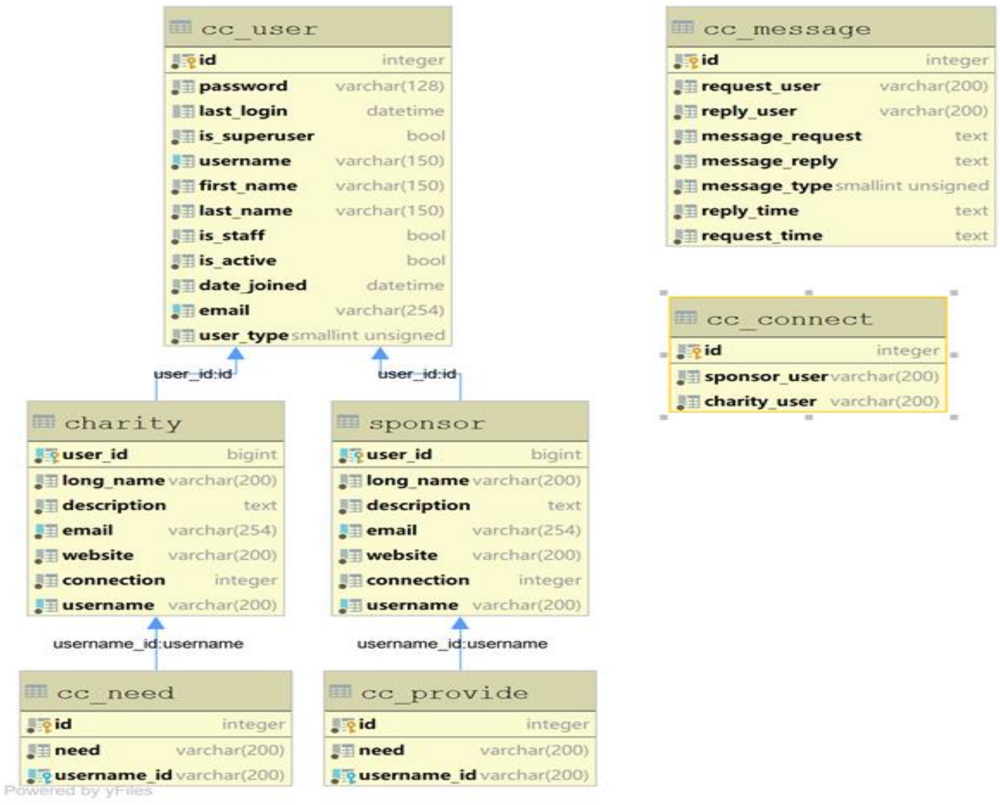


Figure 3 Database Architecture

2.2 Description of Tables

2.2.1 cc_user (user)

User table stores abstract information of both charity and sponsor user.

Username is also a varchar field, which is unique. Furthermore, the **id** is also a unique integer. These two features could be regarded as fields for identification.

Password is a varchar field in this table which is be encrypted. The encoding process will be finished by Django.

Email is the special Email field, which could identify irregular email address and remind user.

Username + Password will be used in authenticate system to check user identity.

User_type is selected by different type user. {1:Charity,2:sponsor}.

These above four features need to be filled in by the user.

Date_joined is a datetime field which record the registration time of user.

Last_login shows last login time with user, it is also a datetime field.

is_superuser and **is_staff** are two field with default False values. These two features will determine who could access Django back-end.

is_active is set with value True. Manually adjusting this feature to False will freeze the user.

These two features are auto-updated.

2.2.2 charity

Charity table is used to record the details of all user with type {1:Charity} in cc_user table.

The Charity table inherits some properties of cc_user table through one-to-one field function.

user_id, username, email maintain same with the cc_user table.

long_name is a char field, it is different from username, which could be duplicate.

Website is a special URL field, which could identify irregular website and remind user.

Connection records the connection times of this charity to others.

Description shows the shortest text content to introduce the charity.

These above four features should be filled in by charity.

2.2.3 sponsor

Sponsor table also inherits part features of cc_user and details information of type {2:sponsor} users.

The sponsor table has the mirror structure of Charity user. All fields in sponsor are same as Charity. The only difference is the information should be filled in by sponsor and display in sponsor list.

2.2.4 cc_provide and cc_need

These two tables are easy to understand with few columns.

Username_id linked to each **username** in sponsor or charity table. As we know, **username** is unique which point a valid user.

Need represents an item, which could be sponsor provided item or charity wanted.

In this table, **username_id** could use multiple times due to one user may have many items in their wanted list.

2.2.5 cc_message

This is a first table which records inter-actions between users. This table records all messages in one try connection.

Once a sponsor finds a suitable charity or a charity find a good sponsor, they could send a connection request to the designated person.

Request_user means the requester in once connection who send information firstly, it is a char field.

Message_request is a text field stores the message sent by requester.

Request_time is a string which record the request time (Sydney time).

Each time, requester send a request with the message. The replier could receive this and select agree or disagree. And the replier also could send a reply message.

Reply_user represents the replier name, which could be used identify replier.

Message_reply is also a text field which is sent by replier.

Reply_time is the Sydney time when the replier answers the request.

Message_type represents the status of this connection. {1: unread; 2:agree; 3:disagree}

2.2.6 cc_connect

This is the last table records the successful connection.

In cc_message table, once a message status changed from 1:unread to 2:agree, the cc_connect will record this connection.

Sponsor_user is the sponsor user in this successful connection, whether it is the requester or the responder.

Charity_user is the charity user in this successful connection, whether it is the requester or the responder.

2.3 data implementation

2.3.1 sign up

Sign up function will create data in **cc_user, charity,** and **sponsor** tables.

Once user input **username, password, email,** and **user_type** in the front-end form.

Firstly, the database will store **username, password, email, user_type** in the **cc_user** table.

Secondly, the database will inherit **username, password** and **email** to **sponsor,** or **charity** table based on the **user_type.** Both these operations will record the encoded password.

2.3.2 sign in

Sign in page will only use data from **cc_user.**

Once the front page receives the **username** and **password**, the database will verify these two fields in **cc_user** table. Because **username** is unique, and these two fields are one-to-one relationship in the database.

2.3.3 edit

Edit page will update **long_name, description, website** and **wanted items**. This information could be changed by user.

Once user form new information about these fields. Database will get the current **user_type** in the **cc_user table.** Then Database will write the new **long_name, description** and **website** properties to **charity** or **sponsor** tables based on the **user_type.**

If the operator is charity, the **wanted items** will updated to **need table**. On the contrary, **provide table** will be updated when the user is sponsor. According to **username** in **need** or **provide table,** it is easy to filter current wanted items list. **New wanted items** is acquired by user form in edit page.

Each time, the database checks the **new wanted items** with **current wanted items** in **need** or **provide table.** Saving the same wanted items in these two items list, deleting disappeared wanted items and adding the new wanted items to these two tables.

2.3.4 details

The details page shows the information about designated user.

Each time, the detail page will get the **user_type** from **cc_user table** according to **username**.

Based on **user_type**, database will decide to call data from **charity** or **sponsor table**. It is also call data according to **username**, database will return the full detail information about this user.

In details page, there are other messages should be displayed. Successful connected partner, this information could get from **cc_connect**.

Each time, database filter **username** in **charity** or **sponsor table**, it will return all successful connected partner with this user.

2.3.5 charity list / sponsor list

This page displays the whole user with same type.

In the charity list page, it will return all charity information from **charity table** separated by page, which is same as the sponsor list page acquire data from **sponsor table**.

Once a page number received from front-end, slicing the corresponding elements in list.

2.3.6 top 10 sponsor

In **sponsor table**, there exist a field records successful connection times with this sponsor.

Each time, top 10 sponsor will show the first ten sponsor information with all sponsor information called from **sponsor table**, which ordered by connection times.

2.3.7 connection request

Once a requester send connect request to another user.

Database will get **current user_type** from **cc_user**, and it will also acquire **replier user_type** according to **replier user_name.**

If these two user types are same, website will not allow to send request.

Once two user types are different, database will record this message.

Current username -> request_user; formed request text information -> message_request.

Current Sydney time -> request_time; message_type -> default 1.

Other fields related to replier will be blank.

2.3.8 response request

Once a user checks the mailbox, it could see the unread message which store in **cc_message** with **message_type** is 1.

The user could form reply text and choose to agree or disagree.

The reply text will store in **cc_message** according to message **id**. And the message _type should also change from 1 to 2 or 3.

Once a **message_type** updated to 2, which means agree. This message will be stored in **cc_connect table**. Firstly, database recognize current **user_type**. If it is {1:charity}, store a new connection with **charity_use**r field filled in current **username**. At the same time, record the **request_user** in **cc_message** to the new table **cc_connect** as the **sponsor_user**. If it is {2:sponsor}, this will be the opposite of before.

2.3.9 search

For the implementation of the search function, since we are using a combination query like the book search system, we can extract any combination of name, description and need (possibly empty string) from the request. The charities that contain the extracted keywords in name, description and need fields are matched in a case-insensitive manner, and all charities are matched for the empty string. Since we have set foreign key relation by username for table **Charity** and table **Need** in the data model, we can optimize the SQL JOIN expression using ***select_related()*** in the Django ORM model, which improves the query efficiency. Finally, we use the same operation as charity/sponsor list to return the query data as search list based on the number of pages requested.

2.3.10 recommendation

For the implementation of the recommendation function, we extract the recommendation level from the request, and the username of the currently logged-in charity, since we need to recommend sponsors according to different recommendation levels. Based on the data model, we find the current user's needs in Need table and match the entries in the **Provide** table according to each requirement of the current user and count and sort the number according to the username to get the sorted **Sponsor**'s username and the number of matched needs. Finally, the filtered data is returned as a list of recommendations based on the number of connections and the number of pages requested using the same operation as the charity/sponsor list.

3 Functionalities

3.1 Home page

On each page, there are fixed sidebars and menu bars for users to click and view the corresponding page. Different users see different content. Public user access, the user cannot view the Recommendation page, Search page and Message Box. After login, the user can view the Message Box, and the message Box bar always displays the number of unread messages of the current user. Only Charity users can access the Recommendation page. The Search page can only be accessed by Sponsor users. At the same time, the left menu bar always shows the Current User login status. Click on a username to jump to the current user's profile page. Such a design can let the current user at any time to view their own data, very convenient. For public users, clicking "public" cannot complete the function of jumping to other pages.

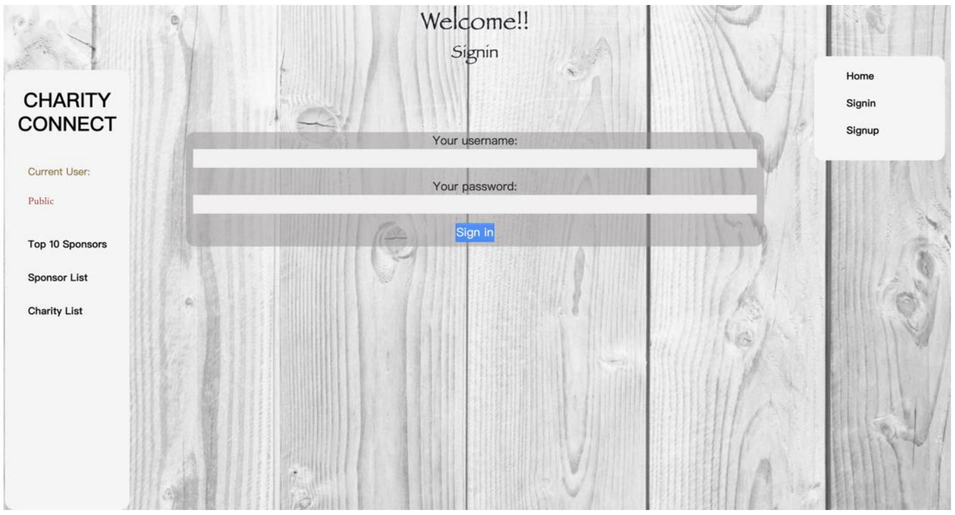


Figure 4 Home page before login

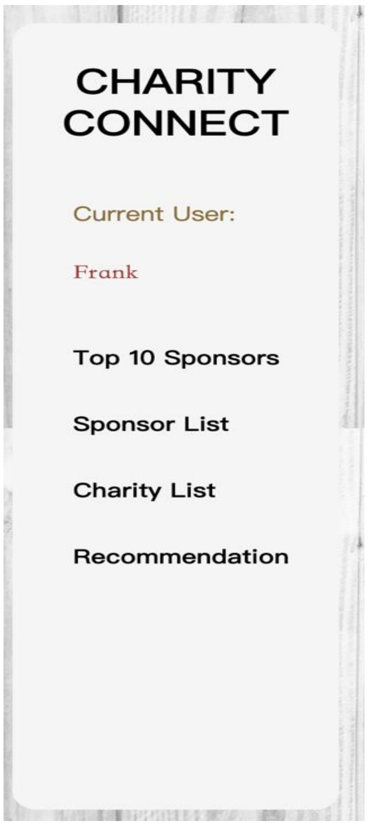


Figure 5 Side bar after signin as Charity

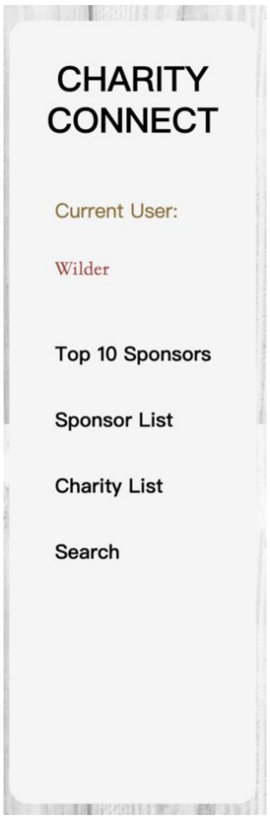


Figure 6 Side bar after signin as Sponsor

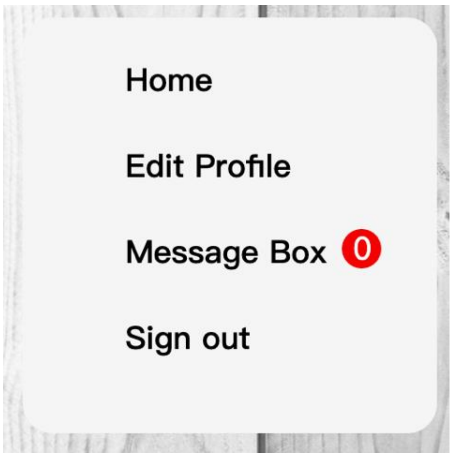
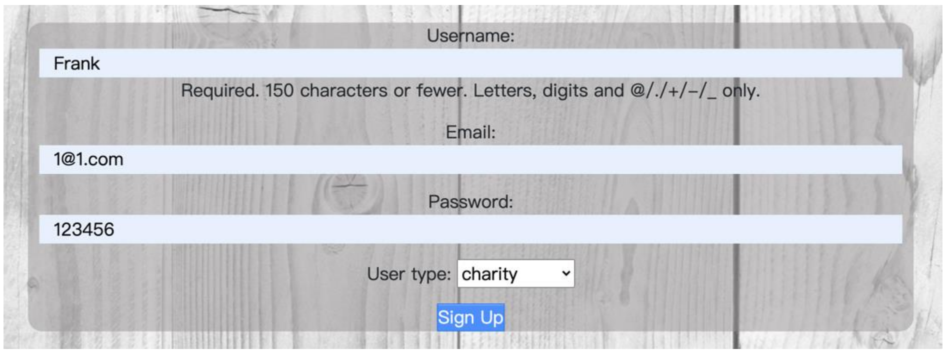


Figure 7 Menu bar after signin

3.2 Sign up and sign in

Registration: Public access users can register as Charity or Sponsor users by selecting from the drop-down menu. If the username and email address already exist, the registration will be failed. If the registration is successful, the login page will be displayed.


Username:

Frank

Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Email:

1@1.com

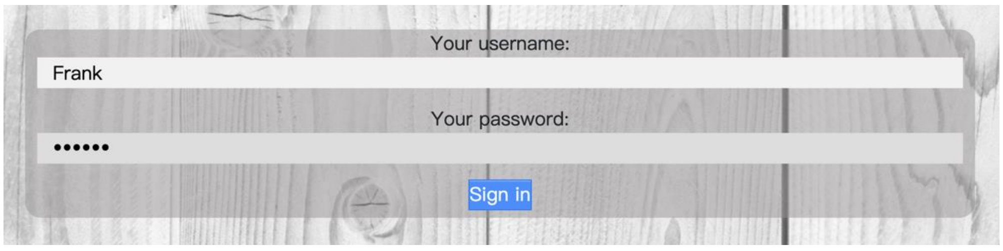
Password:

123456

User type: charity

Figure 8 Sign up

Login: The user logs in to the website using the username and password. If the username and password are correct, the login is completed, and the home page will be displayed. Note: The default password of a user who has logged in to the back-end database is 123456.


Your username:

Frank

Your password:

••••••

Figure 9 Sign in

3.3 Edit Profile

By clicking the 'Edit Profile' button in the upper right corner column, the page will jump to the user profile page. Users can change their full name, user description, user website on this page. If the user enters a website address that does not conform to the normal URL format, a pop-up message will indicate that the input error. Charity users can select the items they need and describe their needs in addition to the three items in the options bar. It is convenient to extract and storage of other requirements to the back-end database, the page prompts you to separate other items with commas. Sponsor users can select the items they can provide, and in addition to the three items in the options bar, they can also describe the other items they can provide. It is convenient to extract and storage of other requirements to the back-end database, the page prompts you to separate other items with commas. Click the Save button to save the change of user profile. A pop-up window indicates that the data has been saved successfully, and it will jump back to the previous page after a few seconds.

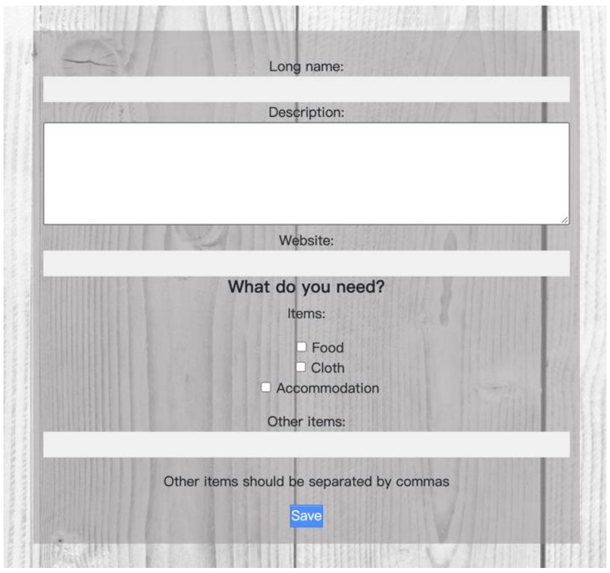
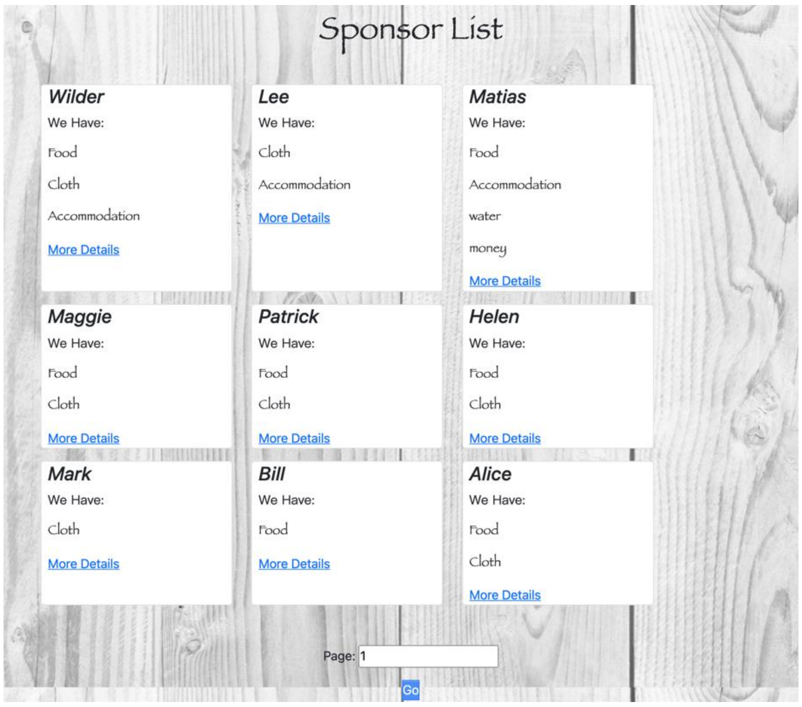


Figure 10 Edit profile

3.4 Sponsor list & Charity list

The Sponsor List page will display the primary information about all current Sponsors users, including the user's long name and the items they can provide. Each page has a 3*3 card layout, and 9 sponsor user cards can be viewed. The current user can use the page turning function to go to another page. Each Sponsor card has a More Details button, click the button will jump to the user

Details page.


Sponsor List

Wilder
We Have:
Food
Cloth
Accommodation
More Details

Lee
We Have:
Cloth
Accommodation
More Details

Matias
We Have:
Food
Accommodation
water
money
More Details

Maggie
We Have:
Food
Cloth
More Details

Patrick
We Have:
Food
Cloth
More Details

Helen
We Have:
Food
Cloth
More Details

Mark
We Have:
Cloth
More Details

Bill
We Have:
Food
More Details

Alice
We Have:
Food
Cloth
More Details

Page: 1
Go

Figure 11 Sponsor List

The Charity List page will display the primary information about all current Charity users, including the user's long name and the items they need. Each page has the same layout as the Sponsor list page. The current user can use the page turning function to go to another page. Each Sponsor card has a More Details button, click the button will jump to the user Details page.

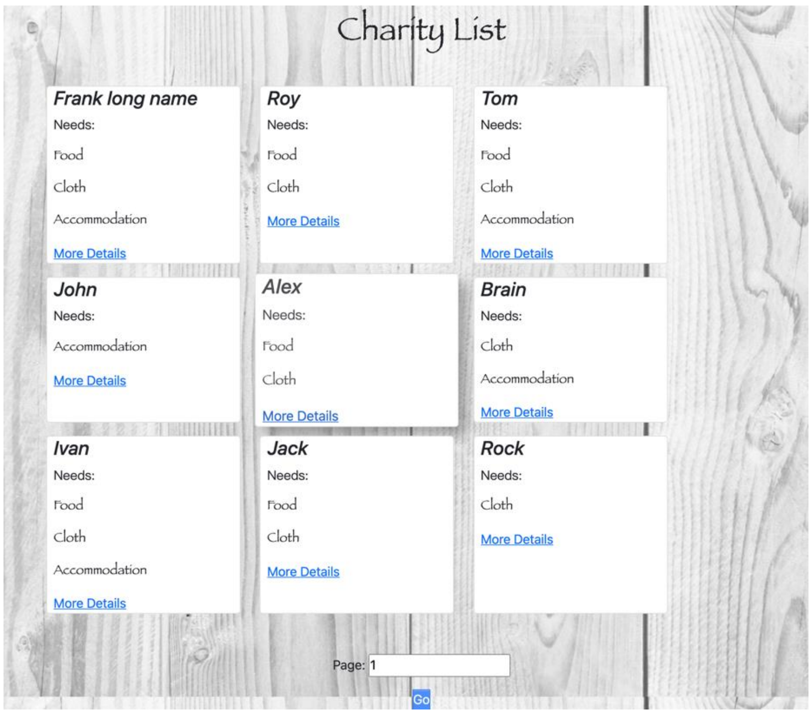
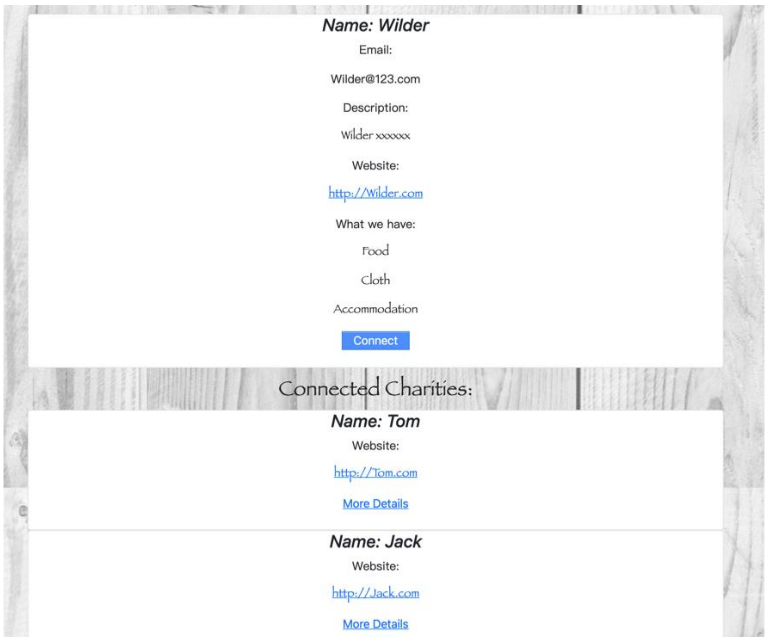


Figure 12 Charity List

3.5 Details

The Details page displays detailed information about the user to be viewed, including all the information edited and saved in the Edit Profile screen. The website is a URL link, click the link to jump to the corresponding website. According to project requirements, the Charity user details page displays Sponsors users who have successfully connected to the current user in order of number of successful connections from top to bottom. Each Sponsor user card includes the Sponsor name and the redirect website. The list features a Novelty function. The More Details button can be used to further jump to the Details page of the connected Sponsor user. The same as the details page of the charity user, the charity users who have successfully connected with the current user are displayed at the bottom of the Sponsor user details page. Each charity user card includes the charity name and the redirect website. These charity users will be ranked from top to bottom according to the number of successful connections. Users can use the More Details button to further jump to the Details page connected to Charity. The connected Charities displayed in the Sponsor user details page is another novelty. The user clicks the OK button to jump back to the previous page.


Name: Wilder

Email:

Wilder@123.com

Description:

Wilder xxxxxx

Website:

http://Wilder.com

What we have:

Food

Cloth

Accommodation

Connected Charities:

Name: Tom

Website:

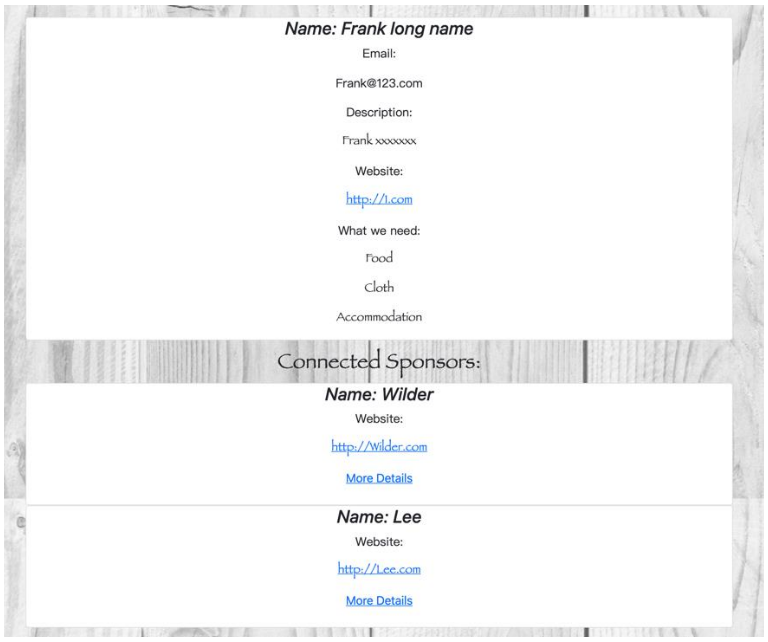
http://Tom.com

Name: Jack

Website:

http://Jack.com

Figure 13 Sponsor Details


Name: Frank long name

Email:

Frank@123.com

Description:

Frank xxxxxxx

Website:

http://1.com

What we need:

Food

Cloth

Accommodation

Connected Sponsors:

Name: Wilder

Website:

http://Wilder.com

More Details

Name: Lee

Website:

http://Lee.com

More Details

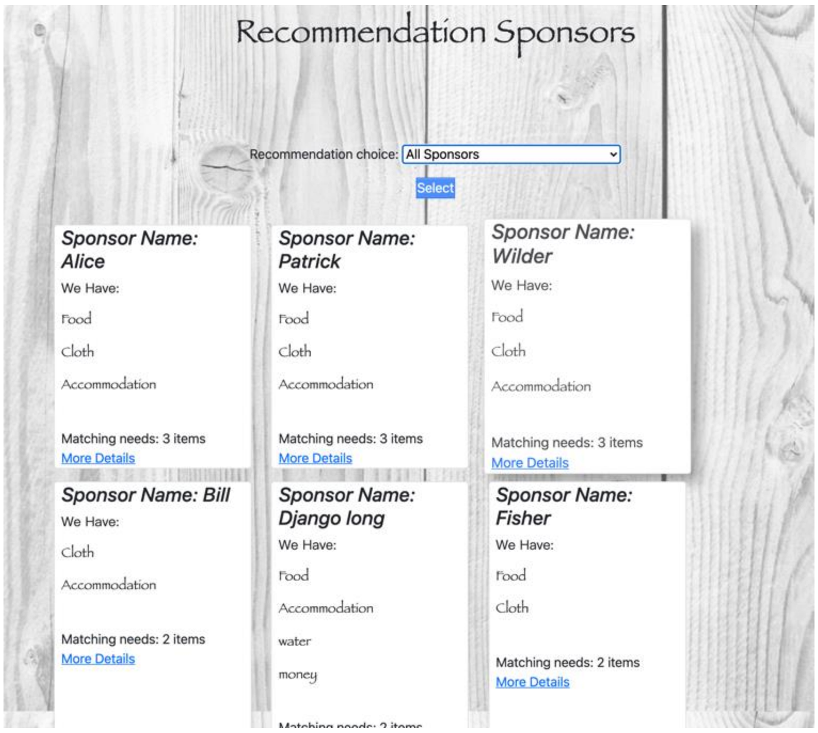
Figure 14 Charity Details

3.6 Recommendation page

Only Charity users can access Recommendation Page. Recommendation Page and Sponsor list page are similar. The page has a 3*3 nine-grid layout, and each page displays 9 sponsor cards. The difference is that in addition to showing the Sponsor user's name and the items that can be provided, each card also shows the number of items that match the items required by the currently logged in Charity user. Click the More Details button to jump to the details page of the current Sponsor user.

On the Recommendation page, users can select different Recommendation sponsors through the drop-down menu at the top. The list contains three options: "All Sponsors", "At least one connection sponsors," and "Zero connection sponsors." Similarly, Recommendation Page has a page-flipping function that allows the current user to jump to another page.

Recommendation Page can make recommendations based on the items needed by the charity users who logged in, and charity users can choose sponsors according to their own preferences. Such a design gives charity users the right to make choices, which meets the actual needs of operation.



Recommendation Sponsors

Recommendation choice: All Sponsors

Select

Sponsor Name: Alice
We Have:
Food
Cloth
Accommodation
Matching needs: 3 items
More Details

Sponsor Name: Patrick
We Have:
Food
Cloth
Accommodation
Matching needs: 3 items
More Details

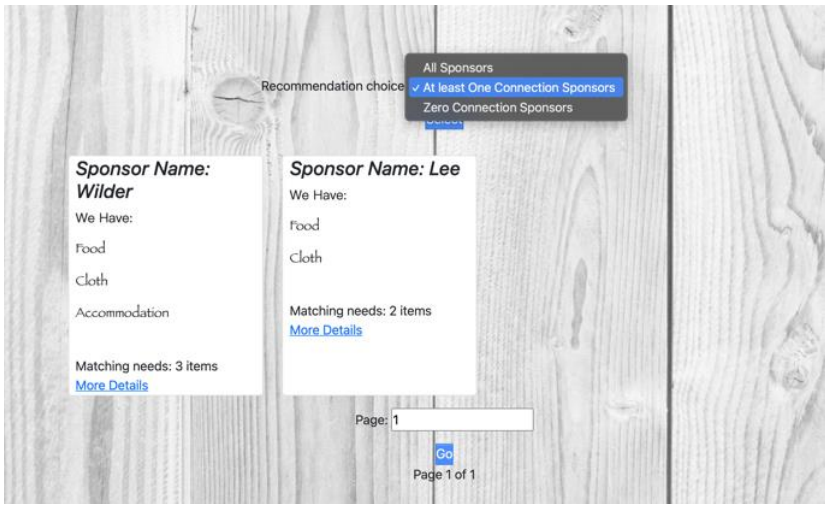
Sponsor Name: Wilder
We Have:
Food
Cloth
Accommodation
Matching needs: 3 items
More Details

Sponsor Name: Bill
We Have:
Cloth
Accommodation
Matching needs: 2 items
More Details

Sponsor Name: Django long
We Have:
Food
Accommodation
water
money
Matching needs: 2 items

Sponsor Name: Fisher
We Have:
Food
Cloth
Matching needs: 2 items
More Details

Figure 15 All recommendation sponsors



Recommendation choice: ✓ At least One Connection Sponsors

Sponsor Name: Wilder
We Have:
Food
Cloth
Accommodation
Matching needs: 3 items
More Details

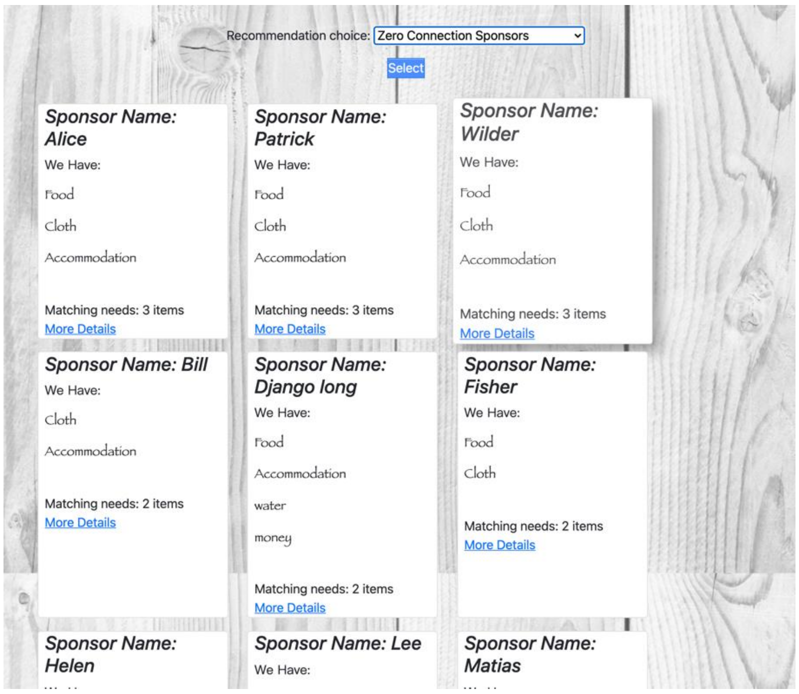
Sponsor Name: Lee
We Have:
Food
Cloth
Matching needs: 2 items
More Details

Page: 1

Go

Page 1 of 1

Figure 16 At least one connection Sponsors


Recommendation choice: Zero Connection Sponsors

Select

Sponsor Name: Alice
We Have:
Food
Cloth
Accommodation
Matching needs: 3 items
More Details

Sponsor Name: Patrick
We Have:
Food
Cloth
Accommodation
Matching needs: 3 items
More Details

Sponsor Name: Wilder
We Have:
Food
Cloth
Accommodation
Matching needs: 3 items
More Details

Sponsor Name: Bill
We Have:
Cloth
Accommodation
Matching needs: 2 items
More Details

Sponsor Name: Django long
We Have:
Food
Accommodation
water
money
Matching needs: 2 items
More Details

Sponsor Name: Fisher
We Have:
Food
Cloth
Matching needs: 2 items
More Details

Sponsor Name: Helen

Sponsor Name: Lee
We Have:

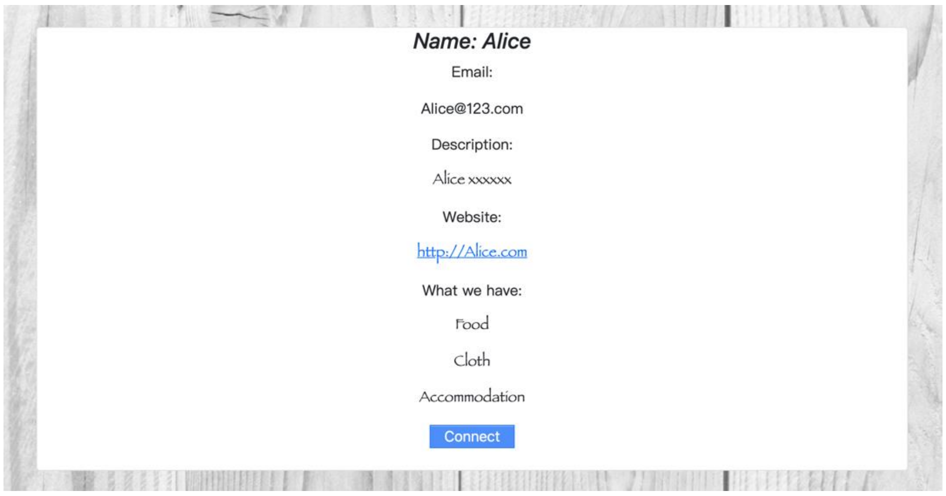
Sponsor Name: Matias

Figure 17 Zero connection sponsors

3.7 Connect

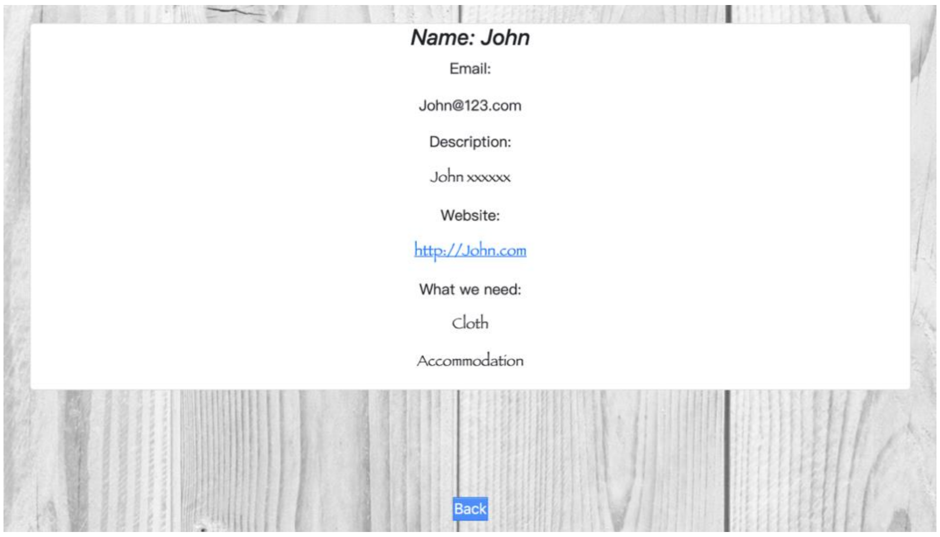
According to the project requirements, the Sponsor user and the Charity user can send a message to complete the connection function. On the user details page, the current user can initiate a connection request and send messages to the other user. Users can click the 'Connect' button on the Details page to go to the Connect request page. Users of the same class cannot send messages to each other. For example, a charity user cannot send a connection request message to a charity user. Therefore, when viewing the details page of other users of the same class, there will be no 'connect' button. On the Connect page, the recipient of the current message is displayed, along with a dialog box for editing the message. Clicking the 'Send' button will prompt a popup indicating that the message has been sent successfully, and a few seconds later return to the details page of the user contacted.

At this point, the list of unsolved messages for the outbox of the current user's massage box will show this message. After the connected user responds, the message will be moved to Outbox's solved message list. For the connected user, the message will be displayed in the Massage Box's Inbox's list of unsolved messages. Once the reply is complete, the message will be moved to the Solved messages list in the Inbox. The details of Message Box will be covered in section 3.8.



Name: Alice
Email:
Alice@123.com
Description:
Alice xxxxxx
Website:
http://Alice.com
What we have:
Food
Cloth
Accommodation

Figure 18 Details page with 'connect' button



Name: John
Email:
John@123.com
Description:
John xxxxxx
Website:
http://John.com
What we need:
Cloth
Accommodation

Figure 19 Details page without 'connect' button

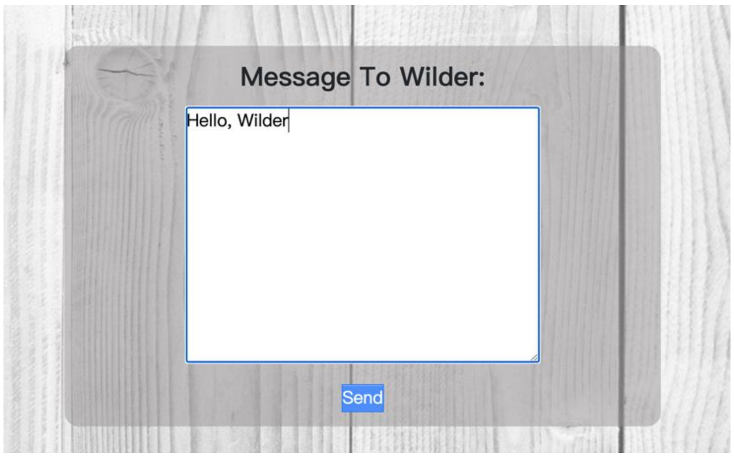


Figure 20 Send request message

3.8 Message box

The Message Box appears in the menu bar only after the user has successfully logged in. The number in the red circle to the right of the Message box represents the number of unread messages for the currently logged in user.

The user enters the Message Box Page by clicking on the Message Box. In the Message Box page, the entire interface is divided into two blocks. The two blocks are divided into the messages received by the current user (Inbox) and the messages sent by the current user (Outbox). Sending and receiving messages are divided into two parts: solved message and unsolved message. The purpose of this distinction is to make it easier for users to view and respond to messages. Clicking on solved Message or Unsolved Message in the left column will display the corresponding content in the message card on the right. The Solved Message columns in Inbox and Outbox show the sender of each message (who sent the message or whom I sent the message to), the status of the connection corresponding to the current message (agree or reject), and the Australian time when the message was sent and replied to. Time stamps are another innovation in our project design, which is designed to distinguish between multiple messages sent to and from the same contact. In addition, users can get more timely responses. The Unsolved Message columns in Inbox and Outbox do not show connection status. The reason is that the sending user or receiving user has not yet replied to the message and decided whether to agree to the connection.

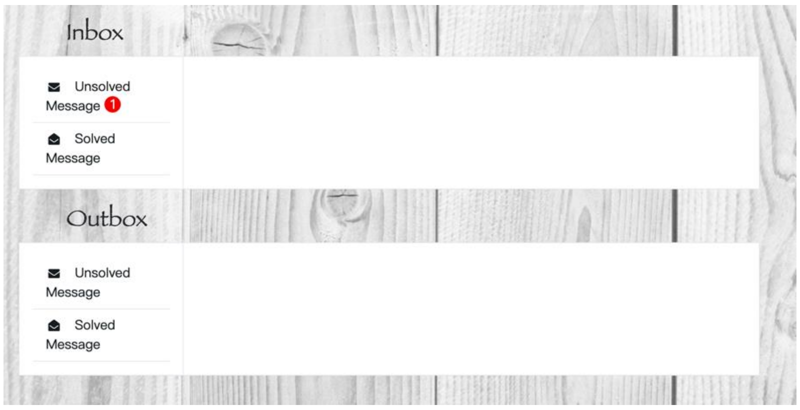
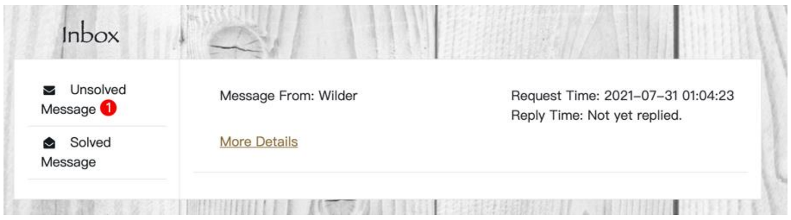


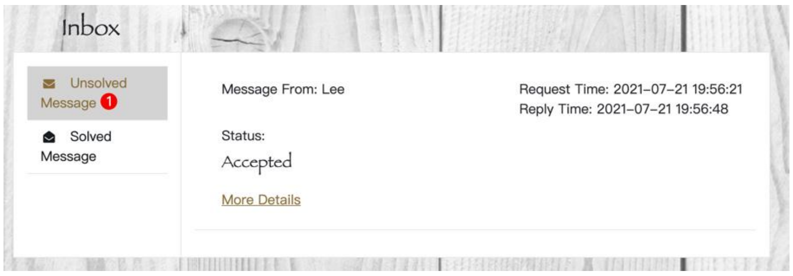
Figure 21 Message Box



Message From: Wilder

Request Time: 2021-07-31 01:04:23
Reply Time: Not yet replied.

Figure 22 Unsolved message list in inbox

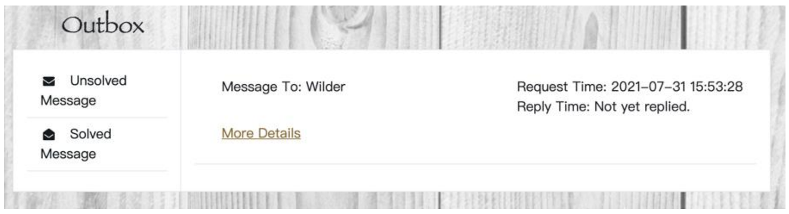


Message From: Lee

Request Time: 2021-07-21 19:56:21
Reply Time: 2021-07-21 19:56:48

Status:
Accepted

Figure 23 Solved message list in inbox


Outbox

Unsolved Message

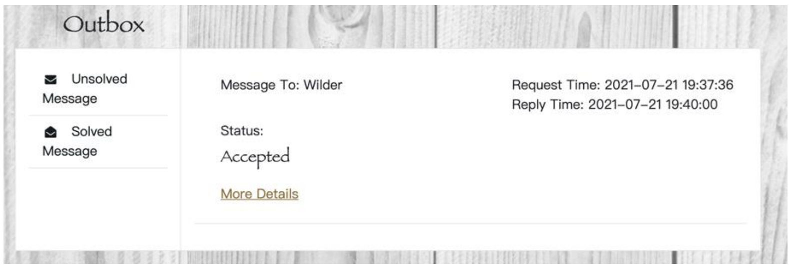
Solved Message

Message To: Wilder

Request Time: 2021-07-31 15:53:28
Reply Time: Not yet replied.

More Details

Figure 24 Unsolved message list in outbox


Outbox

Unsolved Message

Solved Message

Message To: Wilder

Request Time: 2021-07-21 19:37:36
Reply Time: 2021-07-21 19:40:00

Status:
Accepted

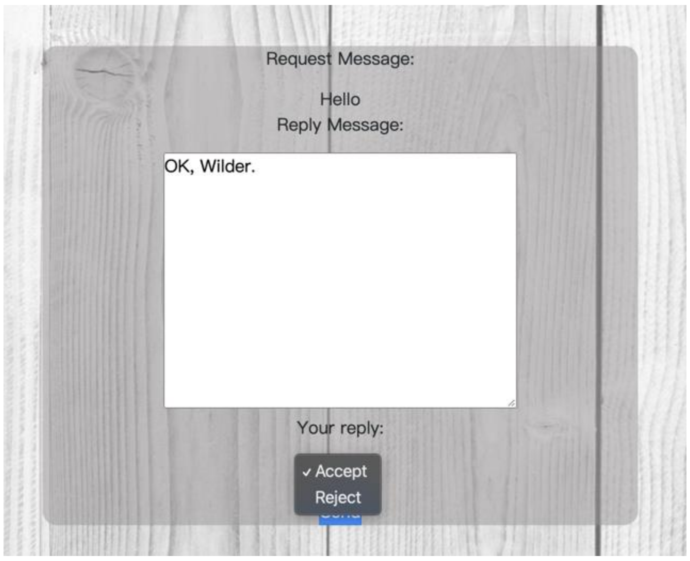
More Details

Figure 25 Solved message list in outbox

3.8.1 reply message

After the user clicks the More Details button of the unsolved message in Inbox column, the page will jump to the Reply Message page. Because the current user can only reply to these messages and cannot make any changes to all replied messages, the message sent by current users cannot be changed as well. Such design can show the logical when users use this website. In the Reply Message interface, the request message and reply Message input boxes are displayed. The following drop-down menu allows the user to choose the option (agree or reject) for this connection. As with the Connect page, when the user has finished editing and selecting the reply message, the user can click the Send button to receive a notification to indicate that the message was sent successfully, and then jump back to the Message Box page after a few seconds.

Messages that complete the reply will move from the Inbox's unsolved message list to the Inbox's Solved message list. Users can click the More Details button in their Inbox's solved message list to view the detail of this received message and their responses. If the user replies with agree, the number of successful connections for both parties increase by 1 in the database. The added connection will be displayed on the details page of both parties. At the same time, with the increase in the number of successful connections of Sponsor users, the ranking and display of top 10 sponsors and recommendation sponsors will be affected. The details of the show Message function are described in 8.2.


Request Message:

Hello

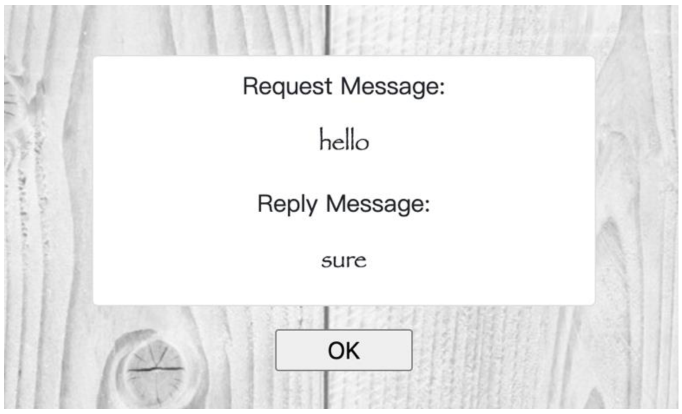
Reply Message:

OK, Wilder.

Figure 26 Send reply message

3.8.2 show message

In addition to the Unsolved message from the Inbox, clicking the More Details button on the message card in the Message Box will redirect the page to the Show Message page. The show message page displays the request message and reply message. In addition, if the user clicks on the unsolved message of the Outbox, the page will display 'not yet replied'. Clicking 'OK' button after viewing this page will take back to the Message Box page.


Request Message:

hello

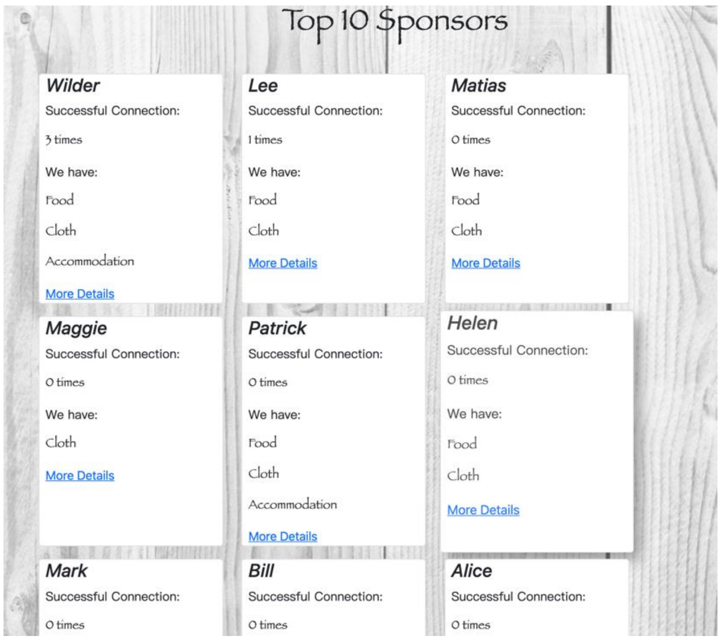
Reply Message:

sure

Figure 27 Show message details

3.9 Top 10 Sponsor

The Top 10 sponsor page is accessible to all users, including public users. It shows the sponsor users in the top 10 list of connection, sorted in descending order by the number of connections completed. Like the sponsor list page, each sponsor user card displays the sponsor user's long name and the items that the sponsor user can provide. The number of successful connections between the sponsor users and Charities will be displayed as well. By click the More Details button at the bottom of the card to jump to details page of this sponsor user.



Top 10 $ponsors

Wilder
Successful Connection:
3 times
We have:
Food
Cloth
Accommodation
More Details

Lee
Successful Connection:
1 times
We have:
Food
Cloth
More Details

Matias
Successful Connection:
0 times
We have:
Food
Cloth
More Details

Maggie
Successful Connection:
0 times
We have:
Cloth
More Details

Patrick
Successful Connection:
0 times
We have:
Food
Cloth
Accommodation
More Details

Helen
Successful Connection:
0 times
We have:
Food
Cloth
More Details

Mark
Successful Connection:
0 times

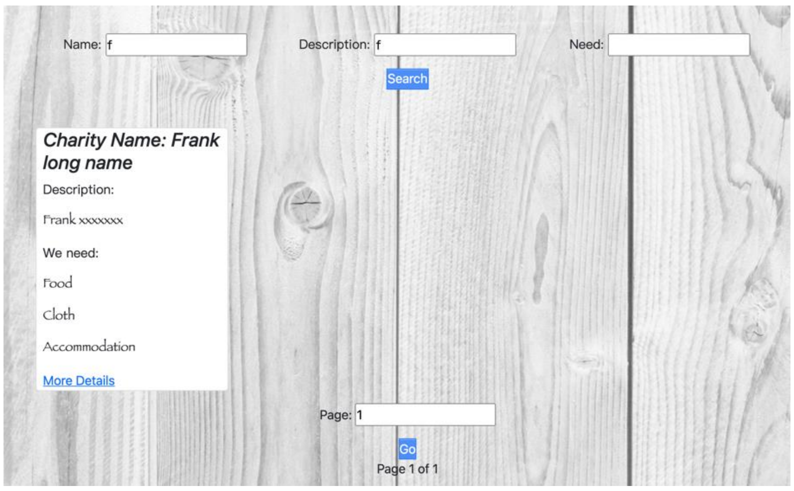
Bill
Successful Connection:
0 times

Alice
Successful Connection:
0 times

Figure 28 Top 10 sponsors

3.10 Search

Only sponsor users can access the search page. The current sponsor user can search for the charity users based on the charity user's long name, description and items they need. If the one of the keyword search fields is empty, the tag will not be searched. The simultaneous search of the three columns can improve the search efficiency of Sponsor users. If all search fields are empty, all charities will be displayed. In the charity card of search results, include charity long name, required items, and the first 20 characters of description. The purpose of this operation is to allow the current user to directly confirm that the search for charity description was successful. Same as the charity list page, it is 3*3 grid layout and display nine charity cards, the current user can move forward to another page through the page flip function.


Name: f Description: f Need:

Search

Charity Name: Frank long name

Description:

Frank xxxxxxx

We need:

Food

Cloth

Accommodation

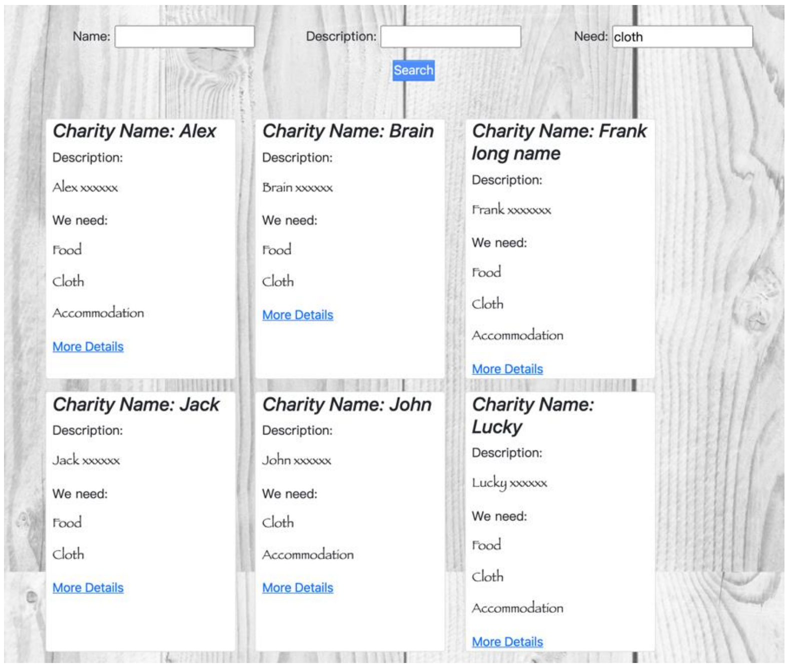
More Details

Page: 1

Go

Page 1 of 1

Figure 29 Search sample 1


Name: Description: Need: cloth

Search

Charity Name: Alex

Description:

Alex xxxxxx

We need:

Food

Cloth

Accommodation

More Details

Charity Name: Brain

Description:

Brain xxxxxx

We need:

Food

Cloth

More Details

Charity Name: Frank long name

Description:

Frank xxxxxxx

We need:

Food

Cloth

Accommodation

More Details

Charity Name: Jack

Description:

Jack xxxxxx

We need:

Food

Cloth

More Details

Charity Name: John

Description:

John xxxxxx

We need:

Cloth

Accommodation

More Details

Charity Name: Lucky

Description:

Lucky xxxxxx

We need:

Food

Cloth

Accommodation

More Details

Figure 30 Search sample 2

3.11 Third-Party Functionalities/Libraries

3.11.1 Front-end

3.11.1.1 Bootstrap

Bootstrap is a free and open-source CSS framework which contains design templates based on CSS and JavaScript. These templates include typography, forms, buttons, navigation, and other interface components [7].

Bootstrap is released under the MIT license and is copyright 2018 Twitter [8], which provides free download and Bootstrap usage permission. The license also allows Bootstrap to be used in packages or projects by modifying the source code [8].

3.11.1.2 FontAwsome

FontAwsome (2021) is an open-source website with large number of fonts and icons to be used for free. In our project we have implemented the message icons from FontAwsome into our message box to improve aesthetic.

FontAwsome icon is licensed by the CC BY 4.0 license and applies to all icons packaged as .svg and .js file types, which allows to be used for commercial projects and open-source projects [9].

3.11.1.3 Bootdey

Bootdey is an open-source website with multiple Bootstrap templates that are free of charge and users are permitted to obtain copies of its software or associated documentation files [10]. Our message box is implemented based on the inbox templates but with our innovation to satisfy our specific requirements [10].

Bootdey is licensed by MIT license which means Bootdey could be used for free, and templates are permitted to be modified and implemented in projects according to the MIT license [10].

3.11.2 Back-end

3.11.3.1 Django

Django is a Python-based free and open-source web framework that follows the model–template–views (MTV) architectural pattern.[4] It is a high-level Python web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development.[1] We chose Django as the framework because it has a simple backend management, built-in user authentication interface for rapid development, while having a relatively smooth learning curve and can be easily deployed to CSE machines. This project uses several Django libraries such as forms, models, Q, auth.

Django is licensed under the BSD license. It allows developer redistribution and use in source and binary forms, with or without modification under BSD license.[1]

3.11.3.2 SQLite

SQLite is a C-language library that implements a small, fast, self-contained, high-reliability, full-featured, SQL database engine. SQLite is the most used database engine in the world.[3] It is also the default database for the Django data model, which makes it easier to manipulate the database for our project.

SQLite is open source, meaning that develop can make as many copies of it as you want and do whatever you want with those copies, without limitation.[3]

3.11.3.3 Python libraries

Python's standard library is very extensive, offering a wide range of facilities.[5] This project we use lots of python standard libraries such as math, os, datetime. We also use additional libraries ***pytz***, ***pytz*** brings the Olson-tz database into Python. This library allows accurate and cross platform time zone calculations using python.[6]

Python and its libraries are licensed under the Zero-Clause BSD license. It allows developer get permission to use, copy, modify, and/or distribute software for any purpose without fee.[5]

4 Implementation Challenges

4.1 Front-end

4.1.1 List page formatting

In our charity-connect platform, we allow public users to access to our sponsors as well as charity lists. This would promote our website to more users or organizations but would require a neat, clear, and user-friendly interface. To achieve that, we have implemented a card-like interface for every element in the lists. However, the card-like interface is hard to be arranged in a neat and clear way like the issue with gaps between each card, issue on how to arrange the number of cards per row etc.

To solve the formatting issue of our lists (Top 10, sponsor and charity lists), a container is implemented. This container specifies the position of our lists in the website window and stipulates the height/width of the whole list. And then we implement a 3 times 3 configuration. To solve gaps between cards, we setup the maximum width and the margin to the right for every card.

4.1.2 Message inbox styling

For the message inbox in our platform, we divide it into two subcategories: Inbox and Outbox. And then we further divide each subcategory into solved and unsolved cases which allows users to follow up their message stream more efficiently. However, this type of arrangement will end up with massive messages congesting in a single page and cause visual chaos when users checking their message box. On the other hand, from aesthetic perspective, this is not user-friendly.

Therefore, we then consider using an inbox style from email application with switching tabs that formats the content of solved and unsolved cases into different pages. This changes the way users interacts with the website and reliefs the redundance of information congestion.

Users would be able to browse their message history in Outbox and check newest in-coming message in Inbox which drastically boost the efficiency of handling connection and communication process between sponsors and charity groups.

4.2 Back-end

4.2.1 Merge and transmit forms

Since we use form to pass data from the front-end to the back-end, we can only pass one area of the form in one click, and in the search and recommendation pages, we need to pass both the search content or recommendation level and the requested page number in the request, and this problem has bothered us for a long time.

The solution we give is to set up a click event in JavaScript, collect the corresponding forms when a click occurs, and add an invisible form to the page to integrate the collected multiple forms into a single form to transmit, then extract and split the form in the view function.

4.2.2 Optimize data model

Since we use Django's built-in user authentication interface, but the built-in unique user type does not meet the needs of our project's multiple user types. We need to have multiple user tables.

Our solution is to keep the default user authentication and use Django's one-to-one field to override and extend the default user table so that each user is one-to-one correlated to the charity and sponsor tables, reducing the complexity of the model.

Due to the large number of database searches when using the search and recommendation functions, the time complexity in running these two view functions is high and the response time is a bit slow. Therefore, we need to speed up the query efficiency of the database to accelerate the access speed of these two functions.

The solution we give is to optimize the query efficiency after using JOIN statements in SQL by resetting the foreign keys of the data model and using the ***select_related()*** method of Django ORM operations.

4.2.3 Multi-connection

As we added novel function which is allow charities to communicate and connect with sponsors multiple times, we added timestamps to the message list so that each connection could be separated, but users could exist in different time zones and cause confusion.

Our solution was to introduce the pytz library, which handles time based on the unified Olson-tz database when request connection or reply message.

5 User Manual

Capstone Project development is done through Django on both the front and back ends.

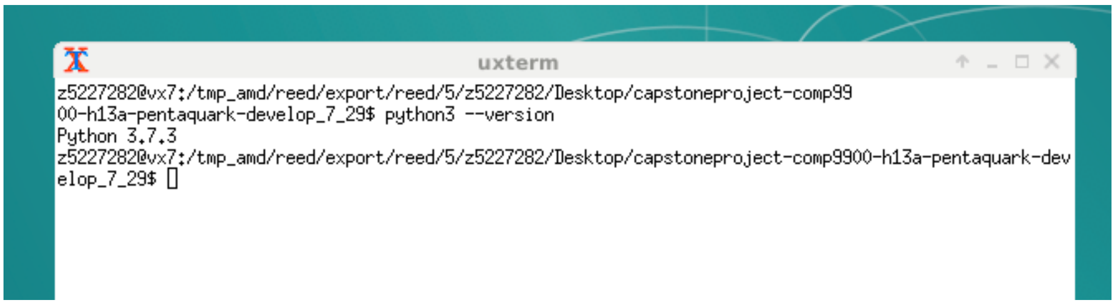
Note: This Web application was tested only on a 16-inch screen monitor.

Incompatibilities may occur with other sizes. In addition, the screenshots of the web pages displayed in the Functionality module are from the MacOS operating system to display better font effects. The User Manual module is tested through the CSE Vlab.

Django is a full-featured server-side web framework written in Python. Django web applications can run on almost any computer that can run Python3: Windows, Mac OSX, Linux/Unix, etc. Almost any computer has the capabilities needed to run Django during development.

This project development environment test was completed in THE CSE Vlab.

We recommend that you use the latest version of Python 3, our project is based on Python 3.7.3.



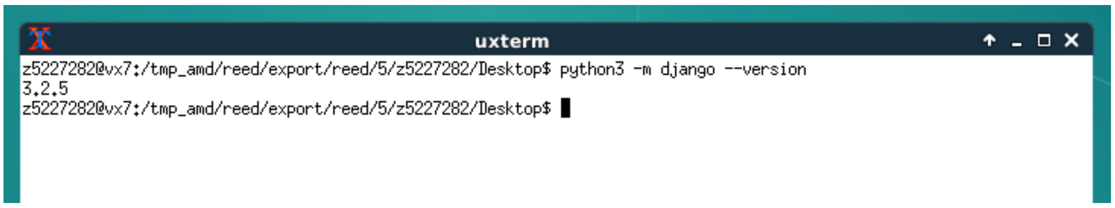
```
z5227282@vx7:/tmp_amd/reed/export/reed/5/z5227282/Desktop/capstoneproject-comp99
00-h13a-pentaquark-develop_7_29$ python3 --version
Python 3.7.3
z5227282@vx7:/tmp_amd/reed/export/reed/5/z5227282/Desktop/capstoneproject-comp9900-h13a-pentaquark-dev
elop_7_29$ 
```

Figure 31 Python3 version

Download Django from your terminal by entering the following code (python3 is installed by default here):

>>>python3 -m pip install Django

Django version 3.2.5 for the CSE Vlab automatic installation (Django version 3.2.4 or higher)

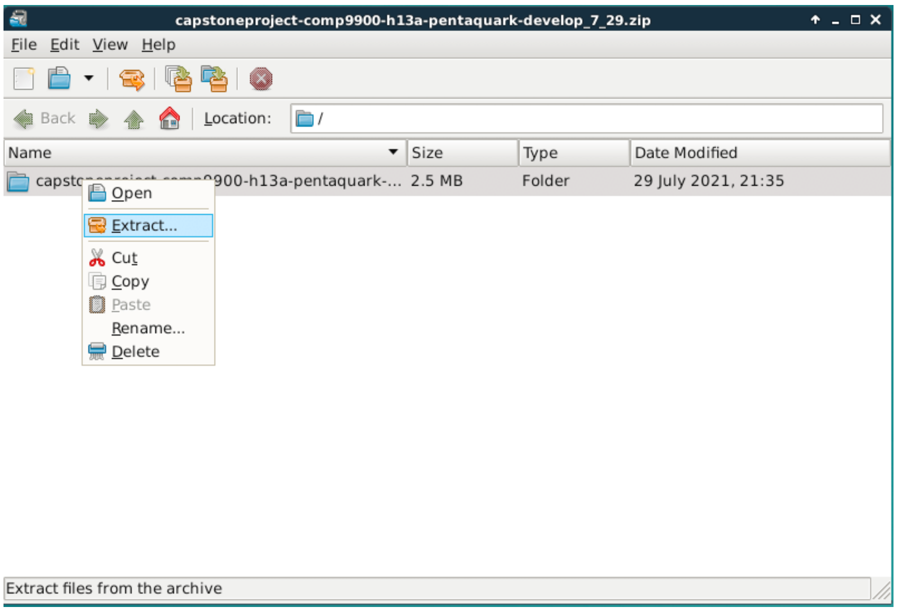


```
z5227282@vx7:/tmp_amd/reed/export/reed/5/z5227282/Desktop$ python3 -m django --version
3.2.5
z5227282@vx7:/tmp_amd/reed/export/reed/5/z5227282/Desktop$ █
```

Figure 32 Django version

Steps to open the project file:

1. Download the "CapstoneProject - Comp9900-H13A-Pentaquark" ZIP file to your desktop. Double-click the zip file and unzip the "CapstoneProject - Comp9900-H13a-Pentaquark" folder (as shown in the compressed folder) to the current path. After the extraction, the contents of the folder are shown in the figure.



Name	Size	Type	Date Modified
capsto...900-h13a-pentaquark-...	2.5 MB	Folder	29 July 2021, 21:35

Extract files from the archive

Figure 33 Extraction Zip File

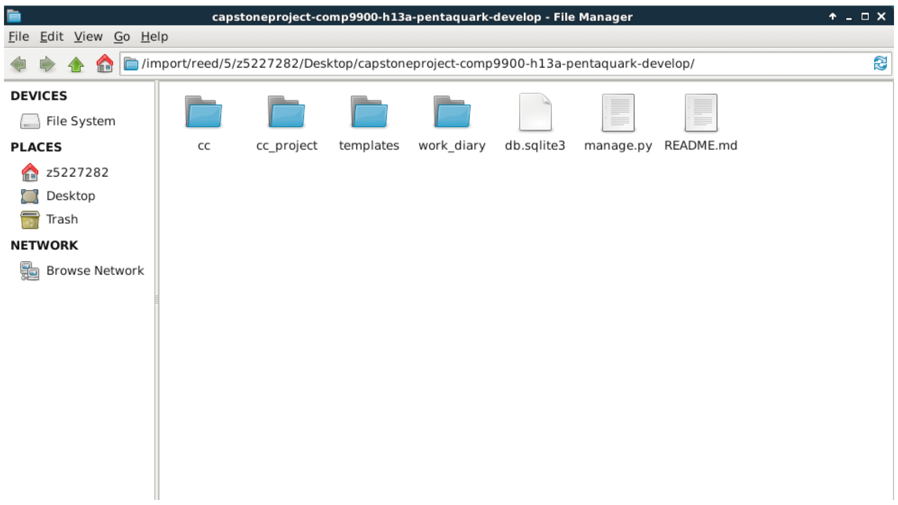


Figure 34 Content of folder after extraction

2. Open the terminal and change the directory to capstoneproject-Comp9900-H13a-Pentaquark. Since the name suffix of the downloaded file was not determined at that time, you can enter the code below in the terminal to obtain the correct file path:

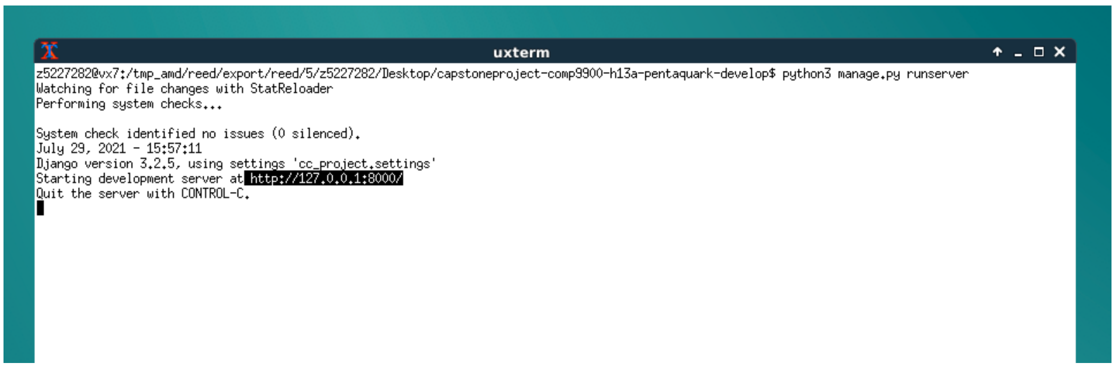
>>>cd "cap" + "TAB button"

3. Run the following command in Terminal in the "capstoneproject-comp9900-h13a- Pentaquark "folder:

>>>python3 manage.py runserver

4. Then the terminal should display the following content, and now you can visit our website

http://127.0.0.1:8000/



```
z5227282@vx7:/tmp_amd/reed/export/reed/5/z5227282/Desktop/capstoneproject-comp9900-h13a-pentaquark-develop$ python3 manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
July 29, 2021 - 15:57:11
Django version 3.2.5, using settings 'cc_project.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

Command of runserver & website

This document was tested using the following browsers:

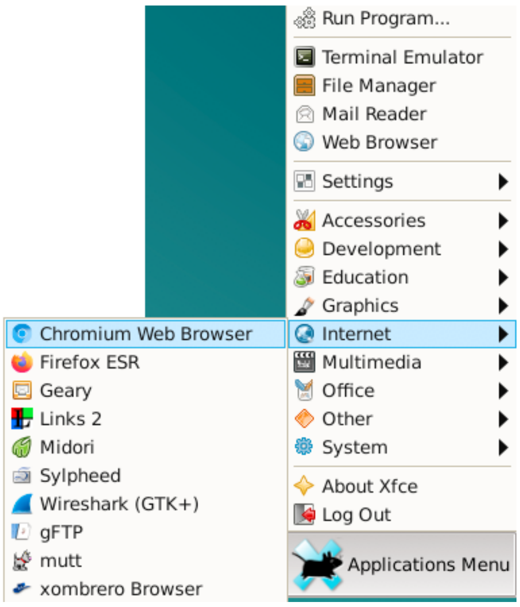


Figure 35 Internet browser

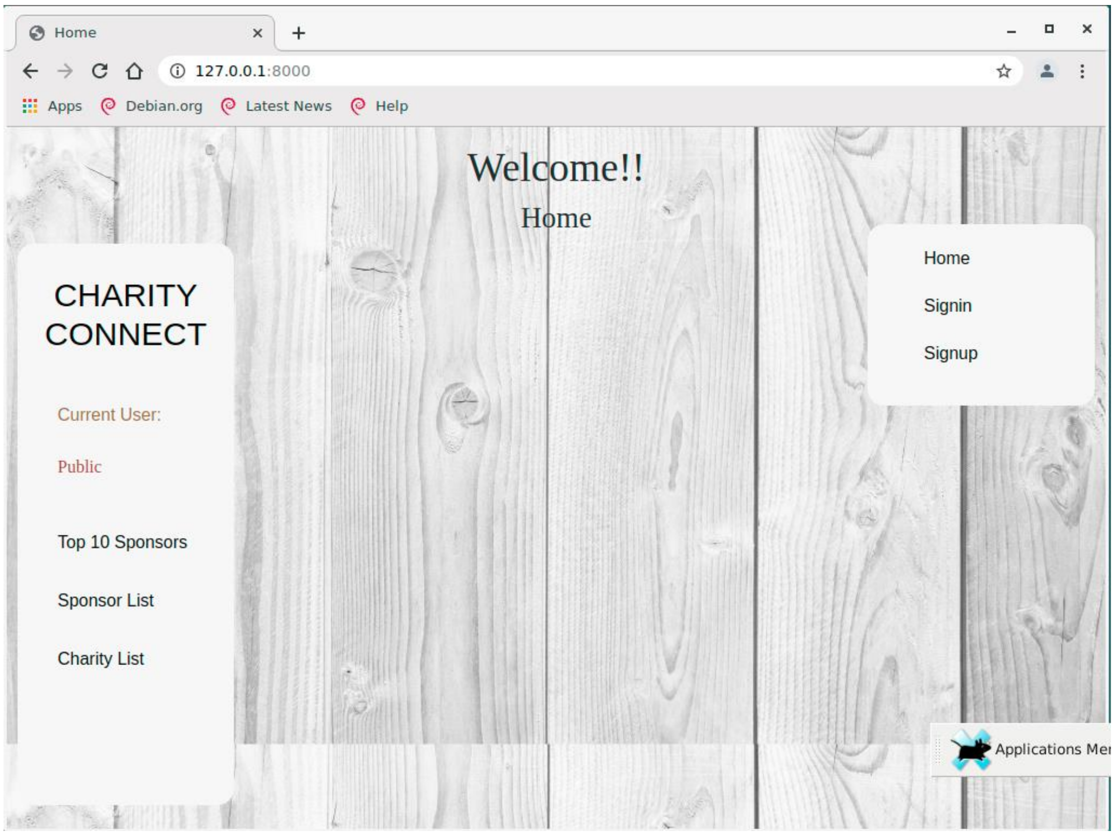


Figure 36 Home Page

For our website, each page has a fixed sidebar and menu bar for the user to click and view the corresponding jump to the page, just like figure above. Different users can see different columns. For public users, the Recommendation page, Search page, and Message Box cannot be viewed. After login, the user can view the Message Box, and the message Box bar always displays the number of unread messages of the current user. The Recommendation page is accessible only to Charity users. The Search page can only be accessed by Sponsor users.

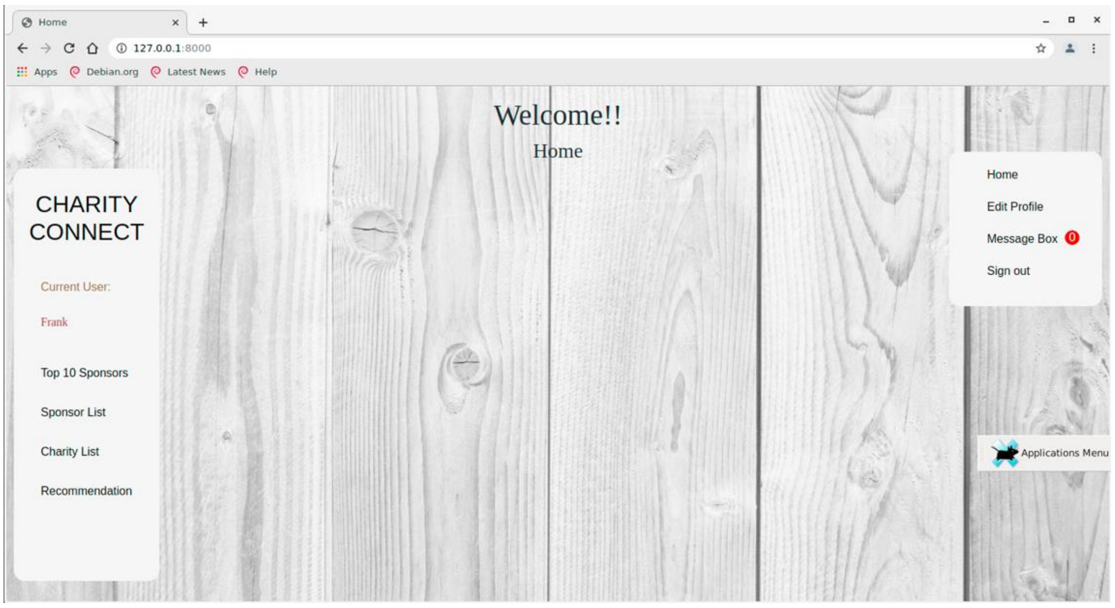


Figure 37 Home Page after sign in

The following are the usernames of all the dummy users available for testing:

Charity: Wilder, Lee, Matias, Maggie, Patrick, Helen, Mark, Bill, Alice, Smith, Fisher, Django

Sponsor: Frank, Roy, Tom, John, Alex, Brain, Ivan, Jack, Rock, Neil, Lucky

Note: All the default password of dummy user in database for test are 123456.

If there is a problem with database loading, it can be resolved by entering the following code in the terminal:

```
>>>python3 manage.py makemigrations
```

```
>>>python3 manage.py migrate
```

If you cannot open the file properly or have other problems, please contact Yuhao Zhou via z5227282@ad.unsw.edu.au or any member of this project team.

6 Reference

[1] Docs.djangoproject.com. 2021. Django documentation. [online] Available at: <https://docs.djangoproject.com/en/3.2/> [Accessed 30 July 2021].

[2] Mark Otto, a., 2021. Bootstrap Introduction. [online] Getbootstrap.com. Available at: <https://getbootstrap.com/docs/5.0/getting-started/introduction/> [Accessed 30 July 2021].

[3] Sqlite.org. 2021. About SQLite. [online] Available at: <https://sqlite.org/about.html> [Accessed 30 July 2021].

[4] En.wikipedia.org. 2021. Django (web framework) - Wikipedia. [online] Available at: <https://en.wikipedia.org/wiki/Django_(web_framework)> [Accessed 30 July 2021].

[5] Docs.python.org. 2021. Python documentation. [online] Available at: <https://docs.python.org/3/library/> [Accessed 30 July 2021].

[6] PyPI. 2021. pytz. [online] Available at: <https://pypi.org/project/pytz/> [Accessed 30 July 2021].

[7] En.wikipedia.org. 2021. Bootstrap (front-end framework) - Wikipedia. [online] Available at: <https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework)> [Accessed 31 July 2021].

[8] Mark Otto, a., 2021. *License FAQs*. [online] Getbootstrap.com. Available at: <https://getbootstrap.com/docs/4.0/about/license/> [Accessed 31 July 2021].

[9] Fontawesome.com. 2021. *Font Awesome*. [online] Available at: <https://fontawesome.com/license/free> [Accessed 31 July 2021].

[10] Bootdey, L., 2021. *License | Bootstrap snippets | Bootdey*. [online] Bootdey.com. Available at: <https://www.bootdey.com/license> [Accessed 31 July 2021].