# Architecture Specifications

## Shelter Connect

Team TBD - Joseph Zhang, Ryker, Schwartzenberger, Simon Bang, Sunwoo Kang

**1. Website Shell/NavBar/Footer**

This is HTML and CSS that will comprise the basis of our web application. Included in this is also the site navigation functionality, referred to as a NavBar from here on. The NavBar and Footer will contain the links to different pages on the website, as well as provide links to our social media pages in the footer. For basic layout and design, Bootstrap v4.1 (https://getbootstrap.com/docs/4.1/getting-started/introduction/) will be used to construct the website layout and nevigation bar functionality. We chose Bootstrap because it is something our developers already have experience in and will adapt itself well to the design we have laid out already. A design/aesthetic reference can be found here: https://info-461-tbd.github.io/wiki/Requirements.pdf

Properties:

- Navbar (<navbar>)
  - The navbar itself, which will be at the top of all of our website pages.
- WebsiteLogo (<img href = '/index.html'>)
  - The logo will be on the left side of the navbar. When clicked, it can lead the user back to the home page.
- About (<a href = '/about'>)
  - The link will lead the user to the about page.
- Register (<a href = '/register'>)
  - The link will lead the user to the registration page.
- SignIn (<a href = '/account'>)
  - The link will lead the user to the sign in page.
- FooterInfo (<p>)
  - Static information about our website and contact information.

Functionality:

- direct()
  - Parameter: none
  - Return: new page
  - Description: when the user clicks on a link in the navbar, the user will be taken that specific page.

Inputs:

- A mouse click from the user on one of the links.

Output:

- The user is taken to the page specified by the link they clicked on.

**2. Database**

The data storage and accessibility needs of our application will be provided by a NoSQL database, hosted using Firebase (https://firebase.google.com/docs/database/). It will be used as the canonical source of information about the Organizations that register an account with our app and the donation requests created by those organizations. Data that is created/input inside the application will be stored in this database. We chose Firebase over other database options by discussing our back-end development experience as a group, which revealed that two of us had experience using Firebase in a project previously. Firebase also handles account registration, email verification, and other administrative tasks that are outside the scope of this project.

- **Account:**

  Properties:
  - uid (int)
    - The unique ID for each “Account” entry
  - Name (varchar(100) / string)
    - The name of the organization.
  - Photo (varchar(50) / string)
    - The profile picture uploaded or chosen by the user
  - Email (varchar(50) / string)
    - Email address of the user, used to sign in to the account
  - Phone (varchar(16) / string)
    - A phone number to the organization that can facilitate donation requests and their fulfillment.
  - Address (varchar(100) / string)
    - A physical address where donations can be dropped off.
  - Zip (varchar(12) / string)
    - The zip code tied to the address above.
  - Organization_desc (varchar(200) / string)
    - A short text description of the organizations, it’s mission, and goals.

  Functionality:
  - Email Verification, profile pictures, Authentication will be handled internally by Firebase: https://firebase.google.com/docs/auth/web/manage-users

  - firebase.auth().createUserWithEmailAndPassword(email, password)
    Creates an account with given email and password. This function also handles account validation.

  - firebase.auth().signInWithEmailAndPassword(email, password)

Signs in the user if email and password match the entry in the database. Throws an error if not.

- firebase.auth().currentUser;
  get the currently signed-in user, null if no user currently signed in.

- currentUser.updateProfile(profileData, success(), fail())
  Updates the user's using the JSON-formatted profileData param.

- firebase.auth().signOut()
  Signs the currently signed-in user out.

- auth.sendPasswordResetEmail()
  Sends an email to the users email with instructions to reset their password.

- currentUser.delete()
  Deletes the currentUsers information from our database and disables use of the account on the website.

Inputs:

1. An organization fills out our registration form to create an account on our site
2. An organization submits a valid username and password to sign in
3. An organization updates their profile information on the site
4. An organization presses the sign out button
5. An organization forgets their password and triggers a password reset
6. An organization presses the delete account button and confirms their choice

Outputs:

1. An account is created in the database and account verification steps are taken before the user can sign in.
2. The user is authenticated and redirected to their account page
3. Information is updated and stored in the database
4. User is signed out and redirected off their account page, if applicable
5. A password reset email will be sent to the user
6. The user account is deleted

- **Request (database entry)**

  Properties:

  - Id (int)
    - The unique id assigned to each new "Request" entry
  - Date_added (date / string)
    - The date the request was submitted on the site.
  - Date_expired (date / string)

    - The date when the request will be marked as inactive and no longer displayed on the site
- Request_text (varchar(300) / string)
    - A short text description of the specifics of the donation request.
- Donation_type (varchar(50) / string)
    - An indication of what
- Ptr_user_account (foreign key / int)
    - A pointer to the UID of the Organization account that created the request
- Active (boolean)
    - An indication of if this post is still active (and thus displayed on the site)

Functionality:

- CreateRequest(date_added, date_expired, request_text, donation_type, ptr_user_account)
  Creates a new entry in the request section of the database with the given params.

- getRequestDetail(id)
  Returns the full data for the request with id provided. Throws error if not found.

- getList()
  Returns an Array containing all active donation requests.

- closeRequest(id)
  Marks the request as expired or closed in the database so it will not be displayed on the site any longer.

Inputs:

1. Organization creates a new donation request
2. User navigates to the website homepage
3. User clicks on a specific request in the list

Outputs:

1. Donation request information is stored in the database and subsequently displayed on the site
2. A list of all active requests are displayed
3. User is taken to the IndividualPost page for that request, where the information about it is shown in more detail.

**3. IndividualPost**

Stores data about a post and display a short blurb about each listing. If the parent organization is viewing it, an “edit” button will be present that will allow the organization to update information about the post.

Properties:

- Title (String)
    - The title of the requested good, provided when an organization makes a new post on the OrganizationAccountPage.
- Description (String)
    - The description of the requested good, provided when an organization makes a new post on the OrganizationAccountPage.
- Photo (<img>)
    - The example photo of the requested good, provided when an organization makes a new post on the OrganizationAccountPage.
- ContactForm (<form>)
    - Email (<input type="text">)
        - The email the inquirer can be reached at.
    - Subject Line (<input type="text">)
        - The subject for the message the inquirer is sending.
    - Message Body (<input type="text">)
        - The message's contents that the inquirer will fill out.
    - Submit (<button>)
        - Button that will submit the contact form's information to the database and send the email to the organization.
- ownedByUser (boolean)
    - A boolean flag that will flag true if the listing is being viewed by the organization that posted it, and false otherwise.

Functionality:

- editPost()
    - Parameters: none
    - Return: new IndividualPost
    - Description: Allows the organization to edit their own post: edit certain fields, all fields, or delete the post entirely.
- submitForm()
    - Parameters: none
    - Return: boolean
    - Description: Returns true if the form data is sent to a valid email. Otherwise, returns false.

Input:

1. Users will input their email address if they are trying to reach out to the organization.
2. Users will input a subject line for their email.
3. Users will input their email message body to send to the organization.
4. Users will click the submit button to send the email.

5. Organizations can click the edit button to open up all of the forms again to edit the content.

Output:

1. The organization will receive new emails from the user.
2. The organization can edit their old posts.

**4. postingsList**

The postings list will hold all of the posts made by the organizations and allow users to filter the posts by different criteria. If a user clicks on a post they are brought to the individual post view.

Properties:

- Title (String)
    - The title of the current filter set on the list of posts
- Filter drop-down menu (ul)
    - Option: Newest to Oldest (li)
    - Option: Oldest to Newest (li)
    - Option: By Organization (li)
        - List of all Organization in a list to select from (li)
- Post New Request (button)
    - Input will be a mouse click on the button, and expected output should be that the organization will be redirected to the individual post page

Functionality:

- reDirect()
    - Parameters: individual post id
    - Return: page of individual post
    - Description: if a post is clicked on, the user will be redirected to that post's individual page

Inputs:

1. A click on a post
2. A click on the filter drop down menu
3. A click on the newest to oldest filter option
4. A click on the newest to oldest filter option
5. A click on the by organization option
6. A click on a specific organization in the "by organization" filter option

Outputs:

1. Redirect to the individual view of the post
2. A drop down menu appears with the following options: newest to oldest, oldest to newest, by organization
3. A list of all posts in order from most newly created to latest created

4. A list of all posts in order from most latest created to most newly created
5. Another drop-down menu appears with all of the organizations
6. A list of all posts created by that specific organization

**5. Organization Registration**

Organizations will have to register on to the site before they can post requests. This page will verify the information and make sure that all the necessary fields are filled out for future communication and accountability.

Properties:

- Organization name (<form>)
    - Input line for the organization's name.
- Public phone number (<form>)
    - Input line for the organization's phone number.
- Public email (<form>)
    - Input line for the organization's email address.
- Street address (<form>)
    - Input line for the organization's main building street address.
- City (<form>)
    - Input line for the organization's city address.
- State (<form>)
    - Input line for the organization's state address.
- Zip (<form>)
    - Input line for the organization's zip code.
- Password (<form>)
    - Input line for the organization's password for their account.
- Confirm password (<form>)
    - Input line to confirm the organization's password that was previously input.
- Website (<form>)
    - Input line for the organization's website, if applicable.
- Organization description (<form>)
    - Input line for the organization to enter in information about themselves so that non-organization users can learn more about the organization.
- Terms of Agreement (String)
    - A static string explaining the terms of agreement that a user must understand before registering their organization.
- Agreement of Terms (<button>)
    - A button that verifies if the user has read the terms of agreement.
- Contact name (<form>)
    - Input line for the organization's personal contact's name.
- Contact email (<form>)

    - Input line for the organization's personal contact's email.
- Electronic signature (<form>)
    - Input line for the organization's personal contact to electronically sign.
- Submit button (<button>)
    - A button that will process all of the inputted information to our database.

Functionality:

- confirmPass()
    - Parameters: password, confirmPassword
    - Return: boolean
    - Description: if the passwords correctly match up, then this function will return a true boolean which the organization cannot register without.
- agreementOfTerms()
    - Parameters: none
    - Return: boolean
    - Description: if the user presses the Agreement of Terms button, then this function will return a true boolean which the organization cannot register without.
- register()
    - Parameters: organizationName, organizationPhoneNumber, organizationEmail, streetAddress, city, state, zip, password, website, description, contactName, contactEmail, elecSignature
    - Return: new Organization Account
    - Description: all of the information will be taken and stored in the database, if the proper fields and requirements are met.
- error()
    - Parameters: passBoolean, agreementBoolean, requiredFieldBoolean
    - Return: error statement
    - Description: if any of the booleans passed in are false, the error will be thrown. The passwords must be matching, the agreement of terms button must be clicked, and the required fields must all be filled in properly. If not, users will be shown an error message, where they will re-enter the information.

Input:

1. A string of the organization's name.
2. A string of the organization's public phone number.
3. A string of the organization's public email.
4. A string of the organization's street address.
5. A string of the organization's city address.
6. A string of the organization's state address.
7. A string of the organization's zip code.
8. A string of the organization's name.
9. A string of the organization's password.

10. A string of the organization's password confirmation.
11. A string of the organization's website.
12. A string of the organization's description.
13. A string of the organization's personal contact's name.
14. A string of the organization's personal contact's email.
15. A string of the organization's personal contact's electronic signature.
16. A click on the submit button.

Output:

1. Redirect the information to our database, where it will be stored in the new account of the organization.
2. Redirect the user to the home page of the website, where they can see new functions for organizations.
3. Throws error if the conditions of error() are not met.

**6. Organization Login**

The organization login page will take input from the user to verify if the username and password are valid in our database. If so, it will lead the user to the home page, where they can further use our website.

Properties:

- Email (<form>)
    - Input line for the organization's account email.
- Password (<form>)
    - Input line for the organization's account password.
- SignInButton (<button>)
    - A button that triggers the Login() function.

Functionality:

- Login()
    - Parameters: email, login
    - Return: Access to organization's information
    - Description: If the information matches up in our database, the user will then be redirected to the home page where they will now have access to their account information and ability to post requests.
- error()
    - Parameter: username, password
    - Return: error statement
    - Description: If the information does not match up in our database, an error will be thrown and will notify the user that either their email or password is wrong. They will have to re-enter their email and password.

Inputs:

1. A string of the organization's email.
2. A string of the organization's password.
3. A click on the "sign in" button.

Outputs:

1. Redirect user to homepage with access to their organization's information.
2. Error if the verification information is incorrect.

**7. Organization Account Page**

Lists all information about the organization as well as a list of all the posts made by the organization.

Properties:

- Organization Logo (img)
  - The logo of the organization
- Title (String)
  - The name of the organization
- Description (String)
  - A body of text containing the description of the organization
- Email (email)
  - The email of the organization
- Phone Number (link)
  - The phone number of the organization
- Address (link)
  - The address of the organization
- postingsList (postingsList)
  - A postingsList with the filter set to that specific organization
- Edit (button)
  - Appears for users of the organization. Clicking it will result in making the description, email, phone number, and address into text boxes containing the current text which they can edit
- userOfOrganization(boolean)
  - Returns true if the user is from the organization and allows them to access the edit button

Functionality:

- editOrganization()
  - Parameters: none
  - Return: new Organization Profile
  - Description: Allows the organization to edit their own profile information: edit certain fields or all fields.

Inputs:

1. A click on the edit button by a user of the organization
2. A click on the link address of the organization
3. A click on the email address of the organization
4. A click on the logo of the organization
5. A click on the title of the organization
6. A click on the phone number of the organization

Outputs:

1. The description, email, phone number, and address into text boxes containing the current text which they can edit
2. Redirects to the google maps search of the address
3. Opens up a mail:to address
4. Redirects the user to the organization's website (if they have one, if not no action occurs)
5. Redirects the user to the organization's website (if they have one, if not no action occurs)
6. If on mobile, opens up dialer with the number

**8. About Page**

Lists information about the website and its creators; lets visitors find out how to contact the developers and learn what the website is all about.

Properties:

- missionStatement (String)
    - A body of text of what the devs' mission statement is.
- problemStatement (String)
    - A body of text of what the problem we are trying to solve is.
- solutionStatement (String)
    - A body of text of how we are trying to solve this problem.
- devloperPictures (<img>)
    - 4 images that show each of the four developer's faces.
- developerInfo (String)
    - A line of text for each developer that shows our names and roles.

Functionality:

- No functionality.

Inputs:

- No inputs.

Outputs:

- No outputs.