Calvin Chen, Jimmy Tran, Wo Bin Chen, Saurav Kharb
10/22/2019
INFO 442 - Co-op Software Dev

Homework 3 - Requirements

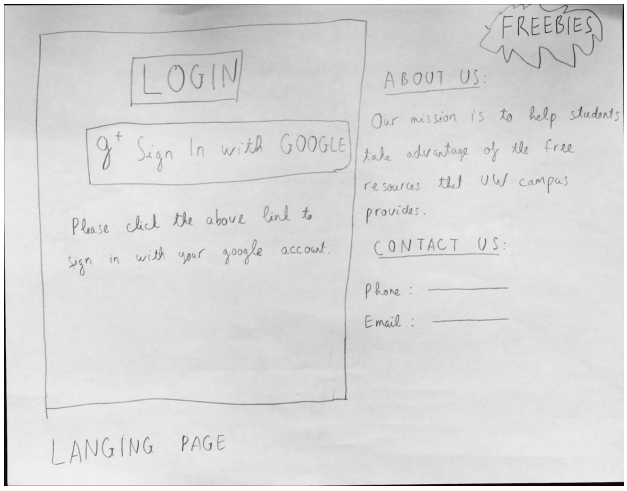
Overall

- Website is built primarily using HTML, CSS, and JavaScript.
- The website needs to be responsive for different devices such as computers, mobile devices or tablets.
- If the website server is down, it should display an appropriate error message.
- The website uses Firebase as the database to store login and post information.

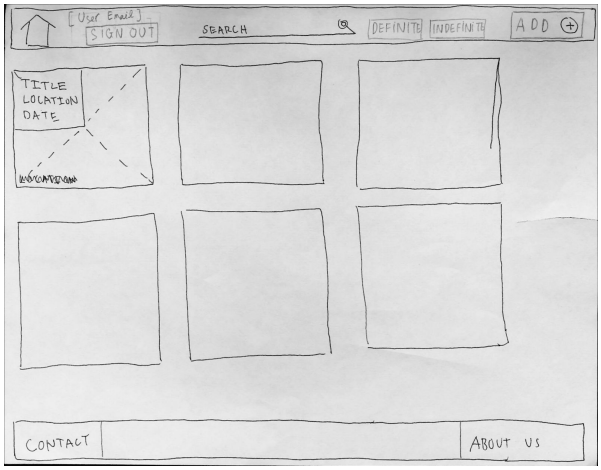
Log-in/Landing Page

- Login Page will be the landing page
- The log in/landing page will be split into two halves, vertically down.
    - On the left side of the page, the top middle will show a ‘login’ text and below it will it will have a google sign in button. The button will be aligned in the middle on the left side. The button will say “g+ Sign in with google”. Below will give instructions on how to log in and also aligned with the google sign in button. “Please click the above link to sign in with your google account”
    - On the middle/top right side will display text that says ‘about us’. Under that will display a brief amount of text that discusses who we are. Below that will display text that says ‘contact us’, providing our phone number and email
- Log In Button will require integration of google sign-in (and is handled by google themselves)
- If the User presses on the google sign in button, a new window opens up showing which google account you want to log in with
- User will click an account that is on the list and the window will close
- User will be logged in under their google account.
- If intended email is not listed on the list of accounts, an option to add account button is located on the bottom
- If there are no errors during the login process, the user is signed in and directed to the dashboard.
- If the user clicks on the sign out button, the user will be signed out and the log in will appear back in their original position.

- Signed out user will be be directed back to the landing page.



Dashboard



- The dashboard will consist of three different parts and the user is logged in: Upper/Top Navigation, Middle Post Section, and Bottom Footer

- **Upper/Top-Navigation**
    - **Home Button**
        - Home button will be an icon of a home
        - If the user clicks on the icon, then the page will automatically refresh
        - Automatically refreshing the website/application will render new post by definitive posts

- **Display email name:**
  - On the right side of the home button will have text that says ‘welcome, [insert email here]!
  - Email will be acquired from the google authentication and shows who the user is logged in as
- **Sign Out Button**
  - The sign out button will be located right after the welcome message
  - The button must display ‘Sign Out’
  - If the user clicks on the sign out button, user will be directed back to the landing/login page.
- **Search Functionality:**
  - The search bar has to located at the top of the website.
  - On the right of the search bar, there should be a clickable search icon that submits the search query
  - The user has to input any characters in search.
  - The system has to be able to query for strings in titles, descriptions, tags, and locations from every posts in the database
  - Clicking the search bar allows for plain text from the user that will filter out results to be shown on the dashboard
  - If the user presses enter, this will trigger a filtering function that will filter based on the search query.
  - By pressing enter or clicking the search icon and if the input is invalid, the system has to display to the user an empty dashboard and that there were no search results found.
  - If the input is valid, the system shows the queried results in order from newest to latest (left to right and descending)
- **Radio Buttons (Definite/Indefinite):**
  - There will be two radio buttons located after the search bar, lined up horizontally
  - The first button will be the definite and the second will be the indefinite
  - The definite button will be automatically filled as default
  - When the user clicks on one of the buttons, the button will fill, and will follow the grid display (check grid display). The dashboard will display the respective type (definite/indefinite) that is selected
  - Only one button can be selected. When the definite button is on, the indefinite button is off. When the indefinite button is on, the definite button is off.
- **Creating Post Button**

        - On the top right side of the website, will be a right aligned add button. This button will say Add with a '+' icon next to it
        - If the button is clicked, a pop-up screen will overlay the dashboard with the creating post form.
        - Creating Post will have a detailed explanation below
- **Post Section:**
    - Each post will be a tile in a grid format(check grid display)
    - Each tile will have inside it:
        - Top right hand corner vertically aligned with title, location, date
        - The tile should will be filled with a picture
    - The size of each tile will scale up depending on the browser size (Having it response - check responsiveness)
    - Scrolling down on the page will display more post into the grid along with the original dimensions of each tile.
    - Posts are organized from left to right, descending from newest to latest
- **Footer**
    - Located at the bottom of the page
    - The footer includes a copyright, which includes a year, project name, and copyright symbol
    - There should be a link for the "About Us" page
    - **About Us**
        - Contains information about the creators and project
    - The footer includes contact information (email)

Interacting with Post:

- Users can interact with a post by clicking on a post on their dashboard. Once they click on a post, a modal box appears describing relevant information (description, start/end time etc) about that post.
- There are primarily two ways a user can interact with a post.
    - The first is an option to upvote or downvote a post. This is enabled by the presence of two arrow buttons (Up arrow icon for upvote, down arrow icon for down vote). We do this to help maintain quality of posts, allow filtering search results by vote count in our dashboard, and create trends by interesting/disinteresting posts by measuring vote activity.
        - The user can only vote an individual post once
        - The user can only either upvote or downvote an individual post
        - Users can change their vote by clicking the opposite arrow
        - When the user upvotes or downvotes, the total displayed votes will change (i.e. the total votes increase when you upvote and decrease

when you downvote). This will be located on the bottom right of each tile. If the post does not have any upvote downvote, then there will be nothing to display but the arrows itself.
Upvote/downvote can be positive or negative integers. It will be the sum of total votes combined

- The second option is to exit out of the modal. A user can do this by clicking on the 'X' button on the modal, and then replying to the 'Are you sure' prompt. This will allow users to exit out of an individual post, and prevent accidental exits.
- If the user is the creator of the post, they can edit or delete the post (check edit/delete post section)

Grid Display/Responsiveness

We will be utilizing a Grid System to display posts on our dashboard. A grid system helps create a series of containers, rows, and columns to layout and align content. This will allow our dashboard to be responsive to different screen widths. So whether a user on a mobile phone flips their screen, or if the website is used on different devices, the grid system will keep the posts well defined and resized accordingly. For this project, we intend to use Bootstrap, an open source CSS framework that utilizes powerful mobile-first flexbox grid to build layouts of all shapes and sizes. We will leverage Bootstrap's 12 column layout system to switch between the number of columns displayed based on the screen width. This grid system will ensure a bug free scrolling/post viewing experience for users with almost any device.

- Uses flexbox to build the layout
- Uses Bootstrap's 12 column layout system to switch number of columns displayed based on the screen width (i.e a phone screen will display less items compared to a wider computer monitor)

Create Post

Users are able to create posts to share any resources by clicking the "Add Post" button on the top right corner of the page from anywhere in the website (except the landing/login page).

When creating a post, a modal pops up and the user provides information for the post. The rendering of module looks like the picture below. The "CREATE POST" appear in the center of the modal.

The modal will prompt the user to add their title, description, location, date (optional), time (optional), tags and picture (optional)

- Prompt user to add a title, a description, a location. These fields will require text input from the users. The information will be stored as a “string” in the database. These are required fields.
- One section with radio buttons that will make the user choose two options: definite and indefinite. User will only pick one options.
    - The definite will be picked at default and user will need to enter the required fields of date and time (start and end) as text.
    - If the user use indefinite, they don’t have to enter date and time.
    - All the textfields will be stored in Firebase as String.
- There will also be **one** image that is uploaded by the user. The picture will require a fileSelector to allow users to select an image from their end of the device. If the user does not provide the picture, a default picture will be assigned to the user. Below is the default picture.

(Source: https://twitter.com/uw) (default picture)

- Also, the picture that upload need to be have the size 300pix x 300pix (this is an optional feature that can be changed based on the engineer choice and user feedback later).
- The user may also add **one** optional tags (such as “food”, “free”, “drinks”) to their posts. Adding tag will help organize search results based on tag categories. The tag will be stored as tag in Firebase.
- Among all of these fields, title, description and location are required to be entered. Date, time, a tag and a picture are optional to enter by the users.
- If any required fields appear to be blank, an error message will be shown prompting the user to re-enter the missing fields.

There also a submit button at the bottom on the modal. Clicking submit without any information in the required fields will prompts the user to fill in the missing fields.

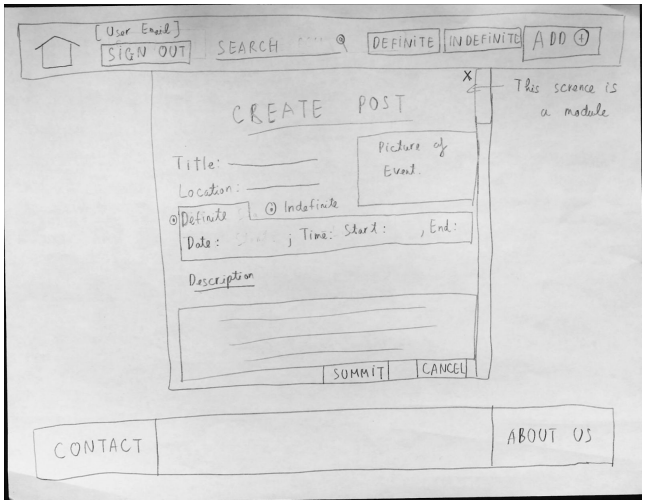
After successfully submitting:

- The modal will disappear and the original screen(dashboard) behind will be shown.
- The dashboard will be updated with the new post.
- The database in Firebase will also need to be updated with the new post
- The new post will be located on the top-left of the Middle Post Section.
- If the user is on the “Contact” or “About Us” page before clicking on the “Add +” button (creating the post):
    - she/he will be transferred to the dashboard with definite selected if the new post is definite.
    - she/he will be transferred to the dashboard with indefinite selected if the new post is indefinite.

If the user clicks “cancel” button (at the bottom-right of the modal next to the summit button) or the ‘X’ located on the top right of the modal:

- Users are prompted to confirm if they want to close by a pop-up indicating “yes” or “no”.
- If they click “yes”, the modal disappears and the original screen(dashboard) behind will be shown.
- The dashboard will show all the origin post without any new post. However, if other users have created new posts or deleting posts or editing posts, the dashboard will update accordingly.
    - If a new post is added, the edited post will move one position to the right or down on the dashboard.
    - If a new post is deleted, the edited post will move one position to the left or up on the dashboard.
- If they click “no”, the yes/no prompt will disappear and the pop-up modal will be shown back again.

The database in Firebase will store all the fields as String and images in png or jpg. After the creation of a new post, the database will be updated accordingly.



Editing Post

After the user clicking on a post, the module that shows all the information about the post will will be popup in front of the user.

The post will contain the information: title, location, definite or indefinite, description and an optional image. If the post has definite property, it also has two more required fields: date and time (start and end). If it is indefinite, it will just show the indefinite property.

Also, the post will have a field where the user email will show up at the bottom left corner of module.

- The user email can be used to check if the user is the one who create the post by storing String and compare them with the user email String in the top-left corner of the web page.
- If the post is created by the user, the button “Edit” will show to the user on the right bottom corner of the module like in the picture. It is located next to the “Delete” button.

After the user clicks on the “Edit” button, the module will change into the module that can let the user change the text on all the fields that were entered by the user earlier.

There are two cases to consider:

- Case 1:

    - If the post was created by “definite” type which means it records the date and time of the event, the user can edit these fields: title, location, date, time (start and end), description.
    - The user can also delete one image from the image that is stored in the post if the user uploads it during the creation of the post.
    - The user can also upload an image from her/his local device.
    - If no image is uploaded the default picture will be shown in the box space.
- Case 2:
    - If the post was created by “indefinite” in the radio button, then the user can edit these fields: title, location, description as text.
    - The user can also delete one image from the image that is stored in the post if the user uploads it during the creation of the post.
    - The user can also upload an image from her/his local device.
    - If no image is uploaded the default picture will be shown in the box space.

After the user has finished editing the post, the user can click on the submit button at the bottom of the module:

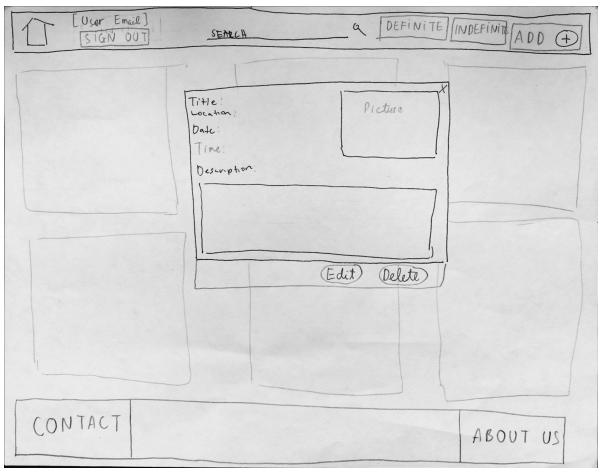
- If any of the required fields are missing, an error message will show up to prompt the user enters these fields.
- After successfully submitting, the modal will disappear and the original screen(dashboard) behind will be shown.
- The dashboard will be updated with the new post.
- The database in Firebase will also need to be updated with the new post
- The edit post is located on the same position of the Middle Post Section.
- If there are any changes to the database like some users add new posts or delete posts, the position of the edited post will change accordingly:
    - If a new post is added, the edited post will move one position to the right or down on the dashboard.
    - If a new post is deleted, the edited post will move one position to the left or up on the dashboard.

If the user clicks “cancel” button (at the bottom-right of the modal next to the summit button) or the ‘X’ located on the top right of the modal:

- Users are prompted to confirm if they want to close by a pop-up indicating “yes” or “no”.
- If they click “yes”, the modal disappears and the original screen(dashboard) behind will be shown.

- The dashboard will show all the origin post without any new post. However, if other users have created new posts or deleting posts or editing posts, the dashboard will update accordingly.
    - If a new post is added, the edited post will move one position to the right or down on the dashboard.
    - If a new post is deleted, the edited post will move one position to the left or up on the dashboard.
- If they click “no”, the yes/no prompt will disappear and the pop-up modal will be shown back again.

The database in Firebase will stored all the fields as String and images in png or jpg. After the creation of a new post, the database will be updated accordingly.



Delete Post

After the user clicking on a post, the module that shows all the information about the post will will be popup in front of the user.

The post will contain the information: title, location, definite or indefinite, description and an optional image. If the post has definite property, it also has two more required fields: date and time (start and end). If it is indefinite, it will just show the indefinite property.

Also, the post will have a field where the user email will show up at the bottom left corner of module.

- The user email can be used to check if the user is the one who create the post by storing String and compare them with the user email String in the top-left corner of the web page.
- If the post is created by the user, the button “Delete” will show to the user on the right bottom corner of the module like in the picture. It is located next to the “Edit” button.

After the user clicks on the “Delete” button, users are prompted to confirm if they want to delete by a pop-up indicating “yes” or “no”:

- If yes, the modal disappears and the original screen(dashboard) behind will be shown with the update without the deleted post.
- The dashboard will be updated with the new post.
- The database in Firebase will also need to be updated with the new post
- At the same time, the record of the post in the database in Firebase will be permanently deleted.
- If no, the yes/no prompt will disappear and the pop-up modal will be shown back again.

The database in Firebase will stored all the fields as String and images in png or jpg. After the creation of a new post, the database will be updated accordingly.

Database

The database will be a non-relational database due to the inherent structure of our data. Particularly, we will use Firebase Realtime Database. The data is structured in the form of a JSON tree. The main branches of the tree would include *users* and *posts*. The *users* branch would include information about each user that has signed up for our service. The *posts* branch would include data points related to each individual post. (see below for details)

User

Data points obtained from Google Sign-in Authorization web client

Post

- Title
- isDefinite
- Start time

- End time
- Location
- Description
- Picture
- Tag
- Upvotes
- Downvotes