

Deployment Document

11/4/2017

Lecture time: 9:30 A.M.

Team Member	Email
Siyuan Xu	siyuanx@usc.edu
Sina Karachiani	karachia@usc.edu
Jessica Zhu	jessicyz@usc.edu
Jeffrey Hernandez	jeffredh@usc.edu

Pre-Requisites

1. Download Node.js
2. Download Ionic/Cordova
3. Go to the project folder in the command line

For Android

1. Download Android Studio
2. Enter `ionic cordova platform add android` into the command line
3. Enter `ionic cordova build android` into the command line, depending on your device. `ionic cordova build` works if you want to build for multiple OS's
4. For an emulator, enter `ionic cordova run android` into the command line
 - a. If it says you don't have the license for a certain version, download the API version they ask for in Android Studio
 - b. At the current time, the most recent Android API does not have a stable Developer Preview release, so you will need to change the API on the emulated device
 - i. Go to your AVD manager by selecting Tools > Android > AVD Manager
 - ii. Select the pencil under Actions next to the device you want to emulate
 - iii. Download and select an API that is less than 27
5. To test on your android device, connect your android device to your computer and enter `ionic cordova run android --device` into the command line for debug mode
6. To run the production version, enter `ionic cordova run android --prod --release`
 - a. To simply build for production, enter `ionic cordova build android --prod --release`
7. The APK can be found under `platforms/android/build/outputs/apk`. Open the file in your device and install the app
 - a. **NOTE: If the package is corrupt, simply use the debug version, which is created in step 3**

Or, if you only want a working app, download and run the project's apk that we will provide.

For iOS

Running it on an emulator:

1. Enter `ionic cordova platform add ios` into the command line
2. Enter `ionic cordova build ios` into the command line
3. Enter `ionic cordova emulate ios` into the command line

Code signing problem

1. Enter `ionic cordova platform add ios` into the command line
2. Enter `ionic cordova build ios` into the command line. `ionic cordova build` works if you want to build for multiple OS's
3. Open the `.xcodeproj` file in `platforms/ios/` in Xcode
4. Connect your phone via USB and select it as the run target
5. Click the play button in Xcode to try to run your app
 - a. If there is a code signing issue, just search for the tutorials online

Server Setup

Create an MySQL database instance on Amazon Web Services, and connect it to MySQLWorkbench.

[Here](#) is a link to a guide on how to perform the above step if you're unfamiliar with Amazon Web Services.

Once the AWS instance is connected to your MySQLWorkbench, run the file `setup.sql` to setup the database schema.

Detailed Design

10/22/2017

Lecture time: 9:30 A.M.

Team Member	Email
Siyuan Xu	siyuanx@usc.edu
Sina Karachiani	karachia@usc.edu
Jessica Zhu	jessicyz@usc.edu
Jeffrey Hernandez	jeffredh@usc.edu

Hardware Requirements:

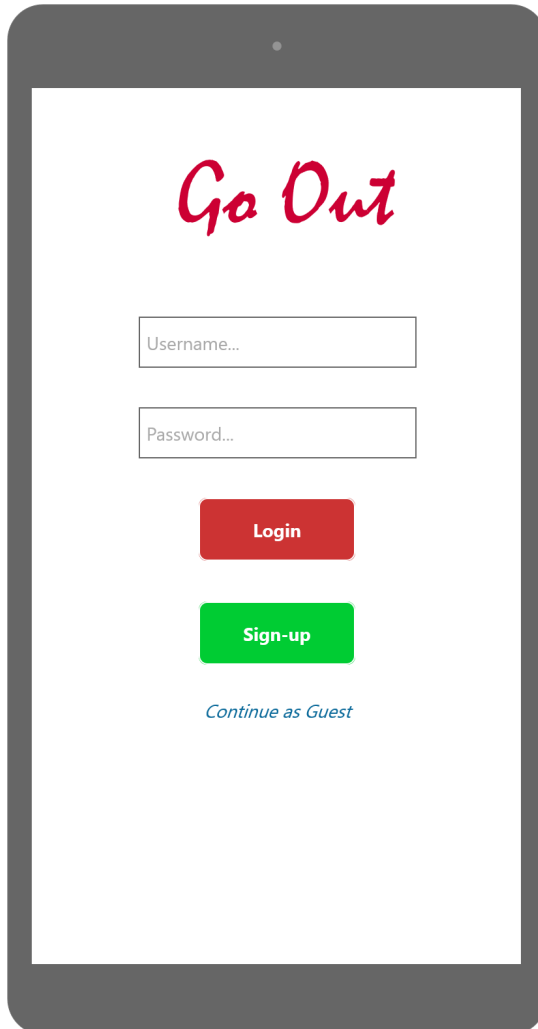
- Smartphone (iOS and Android), since the app is cross-platform

Software Requirements:

- Web server
- Framework: Ionic
- Google Maps API
- Database: MySQL
- Java (Servlet)

GUI

Log-in Page



The Log-in Page UI mockup is displayed within a smartphone frame. At the top, the app logo "Go Out" is written in a red, cursive font. Below the logo, there are two white input fields with thin black borders. The first field is labeled "Username..." and the second is labeled "Password...". Under these fields, there are two buttons: a red button with the text "Login" in white, and a green button with the text "Sign-up" in white. At the bottom of the page, there is a link that says "Continue as Guest" in a blue, italicized font.

Go Out

Username...

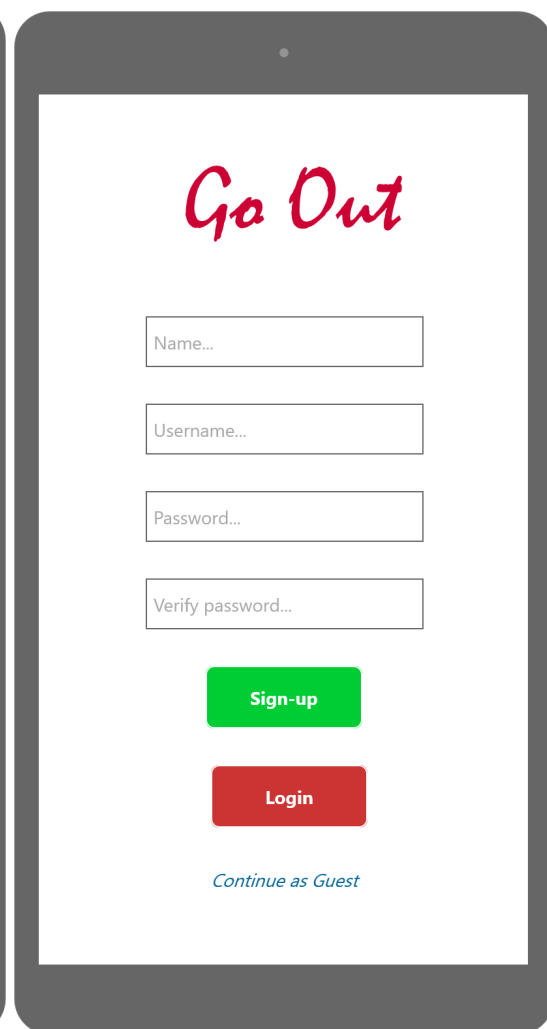
Password...

Login

Sign-up

Continue as Guest

Sign-up Page



The Sign-up Page UI mockup is displayed within a smartphone frame. At the top, the app logo "Go Out" is written in a red, cursive font. Below the logo, there are four white input fields with thin black borders. The first field is labeled "Name...", the second is labeled "Username...", the third is labeled "Password...", and the fourth is labeled "Verify password...". Under these fields, there are two buttons: a green button with the text "Sign-up" in white, and a red button with the text "Login" in white. At the bottom of the page, there is a link that says "Continue as Guest" in a blue, italicized font.

Go Out

Name...

Username...

Password...

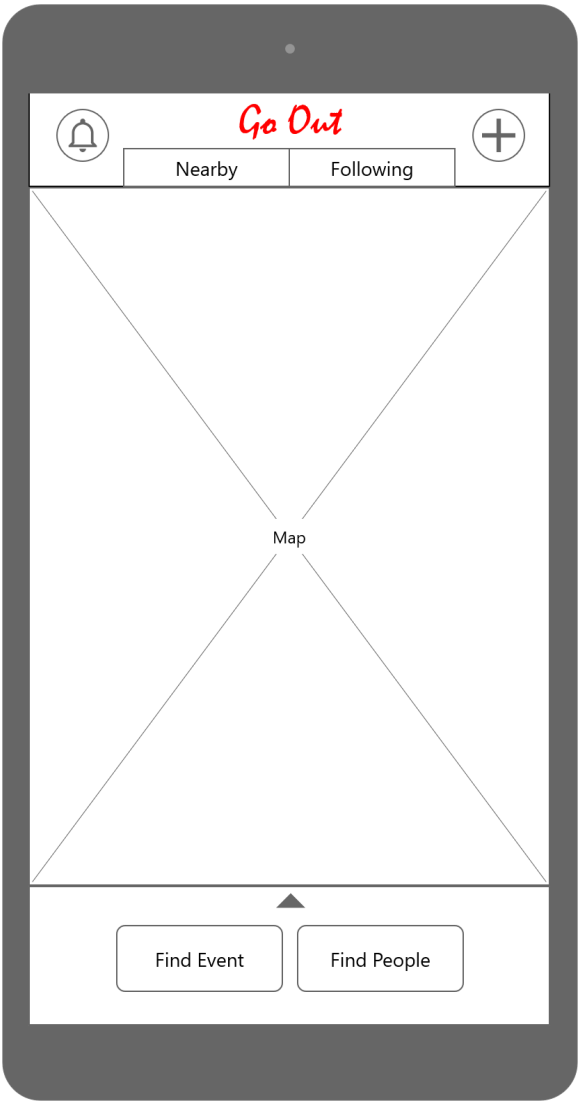
Verify password...

Sign-up

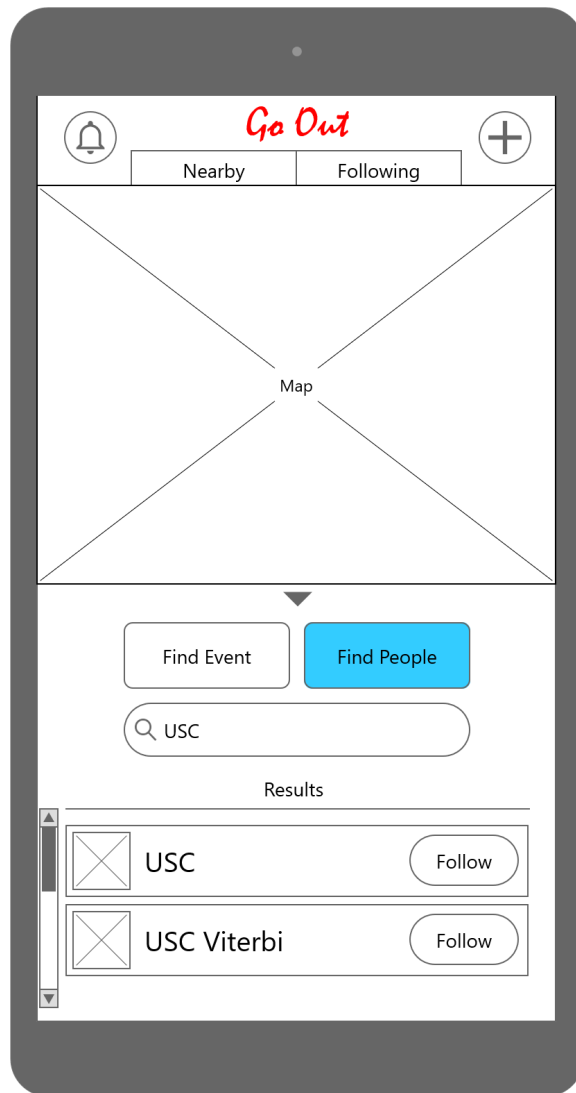
Login

Continue as Guest

Home Page




Home Page Functionalities



Event Info

Back




GO OUT

Title

Title of the event

Date and Time

03 / 10 / 2014



Location

Address of Event

Description

Description of the Event

Privacy

Who is able to see the event
(public/private)

Event Add/Edit

Back

Save


GO OUT

Title

Title of the event

Date and Time

03 / 10 / 2014



Location

Address of Event

Description

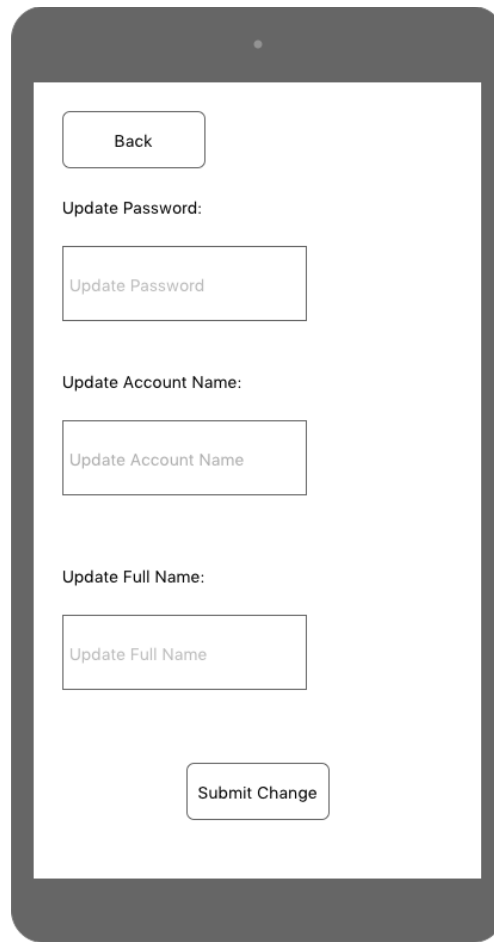
Description of the Event

Privacy

☒ Public

☐ Private

User Settings

A mobile app interface for user settings. It features a dark grey header bar with a small white dot in the center. Below the header is a white content area. At the top of the content area is a rounded rectangular button labeled "Back". Below this is the label "Update Password:" followed by a rectangular input field with the placeholder text "Update Password". This is followed by the label "Update Account Name:" and another rectangular input field with the placeholder text "Update Account Name". Below that is the label "Update Full Name:" and a third rectangular input field with the placeholder text "Update Full Name". At the bottom of the content area is a rounded rectangular button labeled "Submit Change".

Back

Update Password:

Update Password

Update Account Name:

Update Account Name

Update Full Name:

Update Full Name

Submit Change

Your Page

Back

Your Page

Your Events:

Item1Delete

Item2

Interested Events:

Item1

Item2

Users Following:

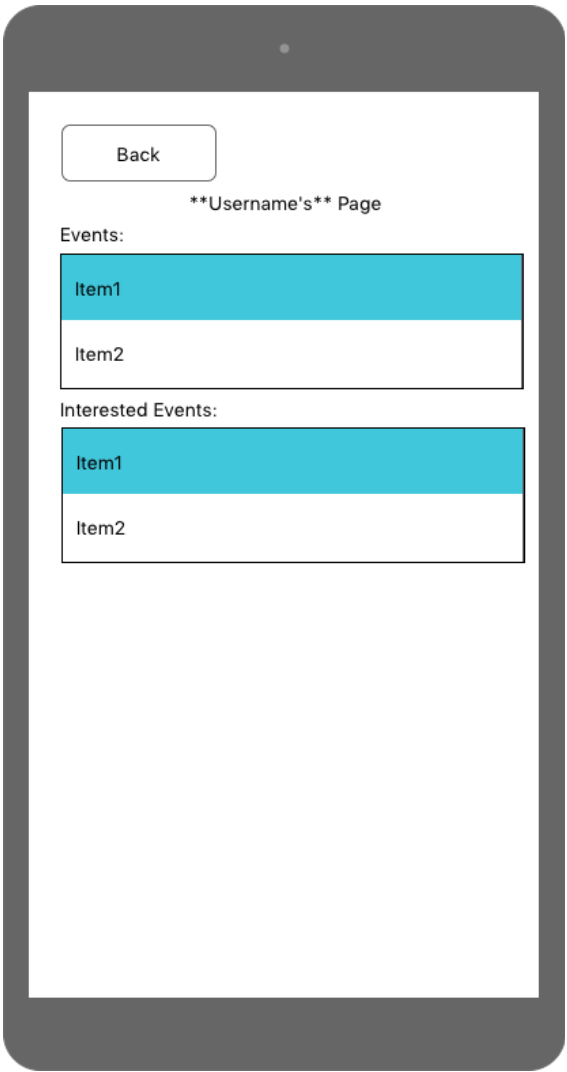
Item1

Item2

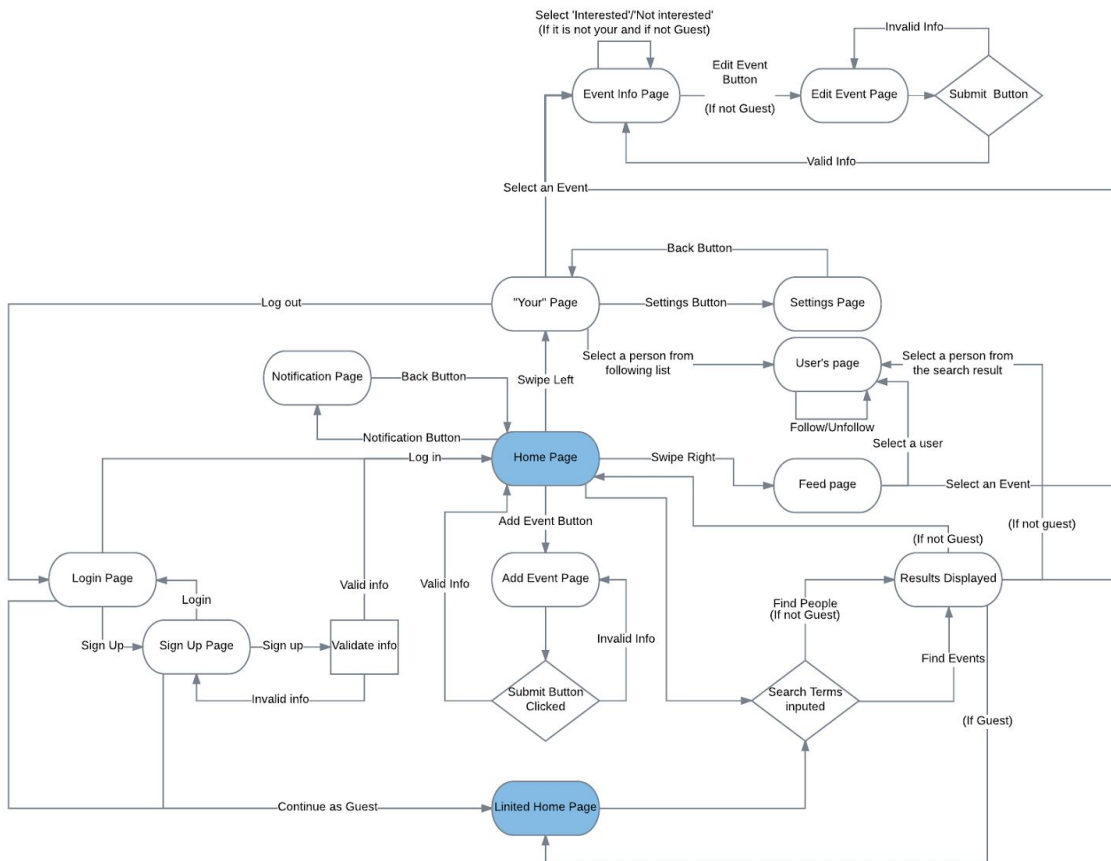
Account Settings

Log Out

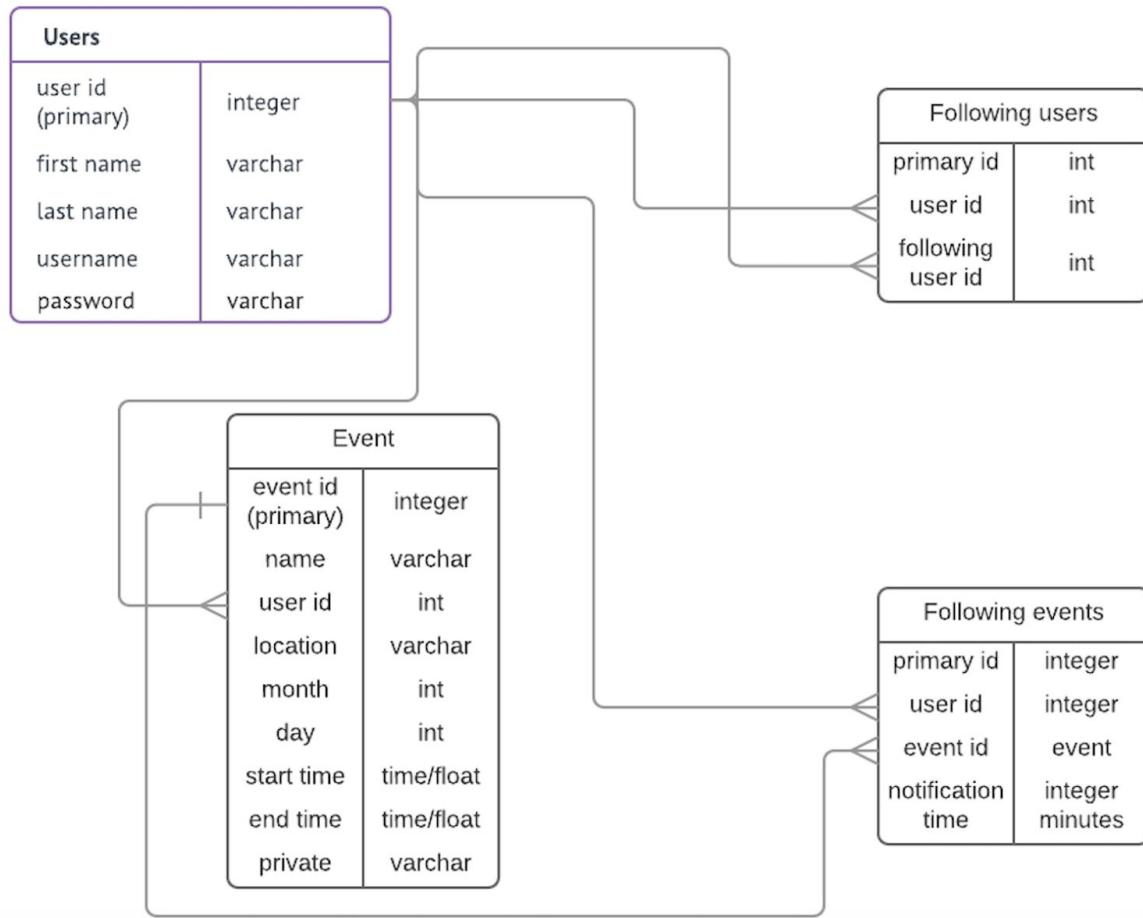
Following User's Page



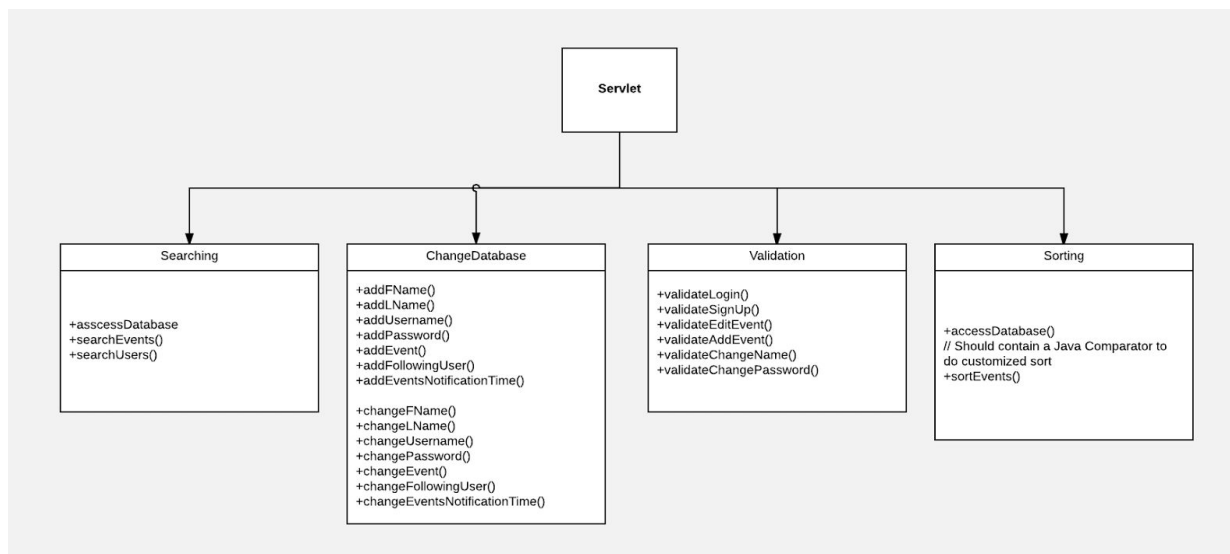
GUI Flow-Chart

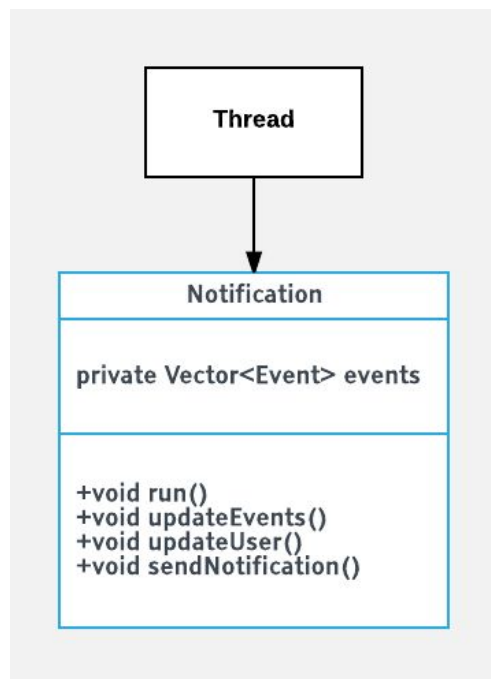
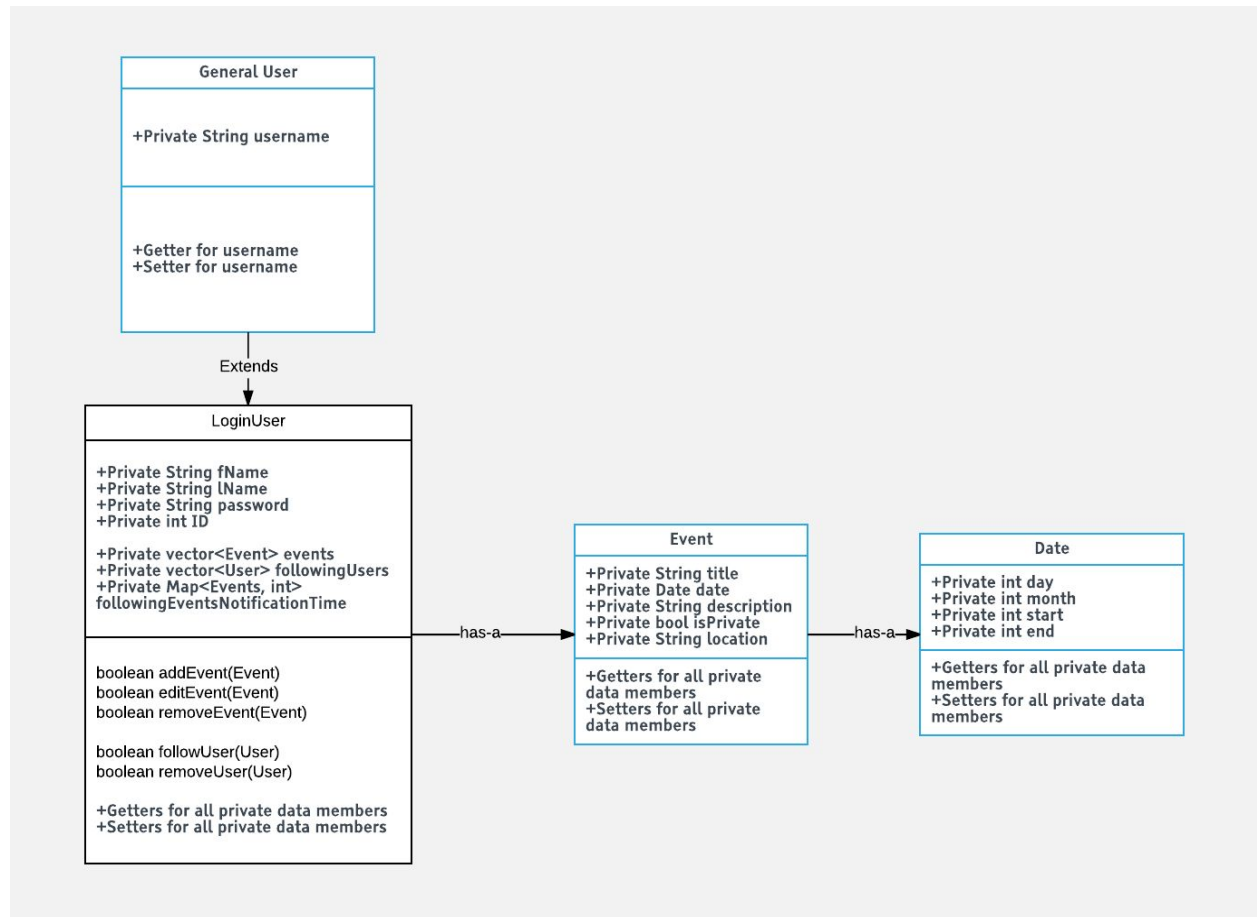


Database



Class Hierarchy (Java Functions Included)





JavaScript Functions

Log-In Page

goToSignUpPage(): When the user clicks the “Sign up” button on the login page, this will redirect the user to the Sign-Up page.

login(): When the user clicks the “Log In” button the information is sent to the Validation servlet. If the information is valid, the user is then shown the Home Page.

loggedIn(): Once the user has successfully logged in, this function will enable all buttons on the application, such as the Add Event and Edit Event buttons.

continueAsGuest(): When the hyperlink “Continue as Guest” is clicked, taking the user to the “Home Page.” Every button on the home page will be disabled except for the search bar and the pertinent search buttons.

Sign-Up Page:

signUp(): Called in the Sign-Up page. When the user has filled out all the required account information and clicks the Create Account button, the information will be sent to the Validation servlet. If the information is valid, then the data will be sent to the “ChangeDatabase” servlet and the user’s information will be added to the database. Otherwise the user will be informed which part of their information was invalid.

Your Page:

logout(): When the user clicks the “Log Out” button, the GUI should return back to the login page.

goToSettings(): When the user click the Settings button, this function should direct the user to the “User Settings” page.

showFollowingUsersEvents(): When the user clicks any of the users in the “Users Following” section, the user will be taken to the clicked user’s information page.

User Settings Page:

updateName(): The function is called if the “Update Full Name” field is not empty and “Submit Change” succeeds. The function makes a call to the ChangeDatabase servlet to send the updated first and last names to the database

updateUserName(): The function is called if the “Update Account Name” field is not empty and “Submit Change” succeeds. The function makes a call to the ChangeDatabase servlet to send the updated username to the database

updateUserPassword(): The function is called if the “Update Password” field is not empty and “Submit Change” succeeds. The function makes a call to the ChangeDatabase servlet to send the updated password to the database

Your Page:

removeEvent(): When the user slides one event to the left, there will be a delete option for that event.

When the user clicks the “Delete” button, *removeEvent()* will be called. This function will make a call to the “ChangeDatabase” servlet and will remove/deactivate that event from the database.

editEventPage(): When the user selects the Pencil icon on the upper right corner of the page. Takes you to the “Edit Event Page”

unfollowEvent(): After following a user, the “Follow” button will become an “Unfollow” button. This will be called when the user clicks the Unfollow button. It will make a call to the “ChangeDatabase” servlet and remove/deactivate the “following event” data in the database.

unfollowUser(): After following a user, the “Follow” button will become an “Unfollow” button. This will be called when the user clicks the Unfollow button. It will make a call to the “ChangeDatabase” servlet and remove/deactivate the “following user” data in the database.

Home Page:

search(): This function will make a call to the Search servlet if there are values in the search input field. It will send the type of search (User or Event), and the servlet will return relevant search results.

goToAddEventPage(): When the + icon button is pressed on the Add Event page, the user is taken to the Add Event Page.

findPeople(): When the user fills out the search input field on the home page and clicks the “Find People” button. The function then makes a call to the “Search” servlet which will generate a list of all events that fit the search term. Once the list is generated, it is then returned and displayed for the user at the bottom of the screen.

findEvent(): When the user fills out the search input field on the home page and clicks the “Find Events” button. The function then makes a call to the “Search” servlet which will generate a list of all events that fit the search term. Once the list is generated, it is then returned and displayed for the user at the bottom of the screen.

followUser(): When the user clicks the “Follow” button beside the search results, the function will make a call to the “ChangeDatabase” servlet and add the user to the Following Users Table with the id of the currently logged-in user.

followEvent(): When the user clicks the “Follow” button beside the search results the function will make a call to the “ChangeDatabase” servlet and add the event to the Following Events Table with the id of the currently logged-in user.

showNearbyEvents(): When the “Nearby” button is clicked, the function will make a call to the GoogleMaps API to populate the Home Page map with all nearby events stored on the database.

showFollowingUsersEvents(): When the “Following” button is clicked, the function will make a call to the GoogleMaps API to populate the Home Page map with all the events of the users the currently logged-in user is following.

Add Event Page:

addEvent(): When the user clicks the “Submit” button, the function makes a call to the “ChangeDatabase” servlet. The servlet then validates the information; if there is any problem with the information, a response is displayed to the user. Otherwise, the servlet adds the event info onto the database.

Edit Event Page:

editEvent(): When the user clicks the “Submit” button, the function makes a call to the “ChangeDatabase” servlet, updating the event’s data fields on the database.

//dont include this

//Add the Feed Page mockup

//update the user page GUI to include a follow/unfollow button

High-level Requirements

10/15/2017

A phone app that will allow users to view the University Park Campus map and to search for places, such as buildings, food locations, bus stations, etc., on and near campus. They will be able to get directions to a destination on campus from their current location. Users can register and create an account by inputting their full name, email, and password.

Users will be able to login by entering their email and password. After a secure authentication, they will have access to more features such as being able to view a list of upcoming or ongoing events, ~~adding events~~, entering a list of their classes for immediate directions to classrooms, and ~~viewing USC tram stops~~ ~~a live status of USC tram and routes~~ using the map's filter menu.

User settings

On the side-bar menu, there is an option to change the user's settings. In the settings menu, the user's name and email will be displayed. There is an option for the user to change their password. Users can add extra profile information such as frequently visited locations and favorite place to get food

Events:

In map's filter menu, events can be toggled between displaying currently ongoing events, upcoming events or both.

Add Classes:

Users can add classes to their account from a list of all classes. After adding the classes, users can see the list of their classes and immediately get directions to their class location.

~~Adding events to schedule:~~

~~When adding a new event onto their schedule, the user must provide the title of the event, the date on which it occurs, it's location, and choose a color to color code events on their schedule.~~

~~If the event is a class, the user is able to select from all the list of classes offered in a given semester, with the location and dates being automatically populated.~~

~~Once a class is added to their schedule, the user will gain access to that class's board where users in the same class will be able to communicate with each other. Users will be able to share information regarding the class and make announcements such as the class being canceled or the class being moved to a different location.~~

USC ~~Shuttle~~ Tram Stops Display:

There will be an option in the filter menu to toggle tram stop locations. ~~or filter color coded routes for visible USC shuttle routes, allowing the user to view only the desired routes.~~

~ Technical Specifications ~

10/15/2017

Hardware Requirements:

- Smartphone (iOS and Android), since the app is cross-platform

Software Requirements:

- Web server
- HTML5
- CSS
- Framework: Ionic
- Google Maps API
- USC Event API
- Database: MySQL

Application Interface:

Login Page (4hrs):

- A login page for username and password for user login and validation.
- A hyperlink to the sign-up page, in the event that the user does not have an account.
- A hyperlink that will take the user to the default homepage, without logging in.

Sign-up Page (3hrs):

- A registration page with three text fields for taking in email, name, and password
- A sign-up button which sends the form information to the back-end for validation.
 - If the information is valid, it will create a new user on the database with the form's information. Display the homepage after creating the user.
 - Otherwise, tell the user which part of the form was invalid.
- A hyperlink to go back to the login page
- A hyperlink to go to the default homepage without logging in or signing up.

Homepage (8hrs):

- Interactive map of campus (generated via the Google Maps API)
- Search box to search for buildings and locations on or near campus
- The map will show the user's location if Location Services is enabled in the phone settings for this app.
- ~~➤ A "Get Directions" button for generating a path from user's current location to the desired destination on or near the campus.~~
- ~~➤ A transparent button named "Filters" that expands the filters menu.~~
- ~~➤ A "Classes" button that expands the Classes menu.~~
- A settings button that expands the settings menu that contains buttons for logging in and logging out, signing up, and user account.
- When the user is logged in, fetch class and campus event data from the database and display it in the interface

- When the user is not logged in, do not fetch any data from the database. Just show map and the search functionality
- A menu button that expands the menu to access settings and account information

Classes Menu (2hrs):

- A button for adding classes from a provided list of classes
- A button for removing an added class
- List of all the classes that have been added by the user
- Each class includes the class name, location, and time
- Each class has a 'get directions' button to immediately provide a path to the location of the class.

Filter Menu (2hrs):

- Show campus events
 - Display upcoming event in the following week.
 - Display currently happening events.
- Show USC tram stop locations
- Show the name of USC buildings
- Show locations to purchase food on campus

Settings Menu (2hrs):

- Logout button if user has logged in
- Login and sign-up button if user has not logged in
- Change their password

Password Change Menu (1 hr)

- Type in the old password.
 - Send the salted password to the database to check if it matches what is stored. If it doesn't match, then display an error
- Type New Password
- Retype the New Password
- Submit Button

Database (8hrs):

- ❖ mySQL tables for:
 - User information
 - Username will be user submitted
 - Passwords will be encrypted (salted)
 - Class table will be saved under user information
 - Class Event information
 - Stores class name, code, semester, and year

- Date/time saved in a nested table
- Saves whether the users wants notifications before the start of the class event, and to select how long before the user receive the notification
- Following Users
 - Primary ID
 - User ID
 - Following User ID
- Following Events
 - Event ID
 - User ID (the user that is following the event)
 - Notification time will be user submitted
- ~~Date/Time table in class info~~
 - ~~This exists because you can't store multiple pieces of information in one cell~~
 - ~~Lists which days and times the class occurs~~