Bar-Pop

Version 1.0 approved

Prepared by Evan Bonsignori

Team: Kyle Bergeron, Keyur Patel, Ayman Bagagas, Eid Alshareef

Supervised by Dr. Su

For CSCI 3321: Software Engineering

December 7th, 2017

Table of Contents

Table of Contents 2

Revision History 2

1. Statement of Product Requirements 3

1.1 Product Need 3

1.2 Document Conventions 3

1.3 Purpose 3

1.4 Intended Audience and Reading Suggestions 3

2. Glossary of Terms 4

3. Functional Requirements Specification 5

3.1 Stakeholders 5

3.2 Actors and Goals 5

3.3 Casual Descriptions 5

4. System Requirements 6

4.1 Functional Requirements 6

4.2 Non-Functional Requirements 7

3.4 Fully-Dressed Descriptions of Use Cases 8

3.5 Use Case Diagram 14

3.6 System Sequence Diagrams 15

6. Domain Analysis 18

6.1 System Domain Model 18

6.2 Database Model 19

7. Interaction and Class Diagrams 20

7.1 Sequence Diagrams 20

7.2 Class Diagrams 23

8. Interface Specification 24

8.1 Mobile Application User Interface 24

8.2 Website User Interface 26

8.3 Hardware Interfaces 26

8.4 Software Interfaces 26

8.5 Communications Interfaces 27

9. System Architecture and Design 28

9.1 Architecture Styles 28

9.2 Persistent Data Storage 28

9.5 Network Protocol 28

9.6 Hardware Requirements 28

10. Appendix 29

10.1 Contribution Breakdown 29

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason for Changes** | **Version** |
| Evan Bonsignori | 11/1/17 | Add Design and Architecture | .3 |
| Evan Bonsignori | 11/30/17 | Combine Design and Architecture with Requirements Document. | .5 |
| Evan Bonsignori | 12/4/17 | Add functional requirements and traceability table. Compact redundant use cases. Add Sequence and class diagrams. | 1.0 |

# 1. Statement of Product Requirements

## Product Need

Downtown Savannah Georgia is a popular location for nightlife establishments. The historic city draws tourists of all ages, SCAD draws college students from all around the world, and the city’s open container laws encourage visitors to spend their night ‘hopping’ between bars. However, there is a problem for tourists and locals alike, and that is the need to identify where to go for a drink or a crowd. Some people prefer bustling crowds while others prefer a small crowd when finding a nightlife venue to attend. Our app, Bar-Pop seeks to guide these venue seekers to the venue that best suits their needs, while providing a platform for venue owners to advertise events and see the number of people in their venues.

## Document Conventions

The term *Venue-Hopper* will include any customer seeking out a nightlight location regardless of if it’s a bar, club, or other venue. Venue-Hopper does not necessarily mean that customers are actively switching between venues, since they could be using the app to locate their first and only venue. The term *Venue-Owner* will refer to anyone on a nightlife venue’s staff that needs to know the number of people in a venue’s location. We will call the number of people on a specific venue’s property: *population*.

## Purpose

Bar-Pop is a mobile application that provides population metrics, event, and location data to customers looking for nightlife venues in Savannah GA. Additionally, Bar-pop seeks to provide a mobile application interface for nightlife venue staff that provides population metrics and the ability to edit event data for their venue.

## Intended Audience and Reading Suggestions

This first portion of the document is written in natural language and should be comprehensible to anyone who has attended nightlife venues and owns a smartphone. After the first few sections, this document will require some moderate technical knowledge in the form of UML diagrams and terminology.

# Glossary of Terms

* **Advertisement** – A source of revenue for the owners of bar-pop. These can include relevant drinking or local attraction advertisements that users are shown either directly or indirectly within the application.
* **Amazon Web Services** (AWS) – A web hosting service provided by Amazon that Bar-pop will use for its remote server.
* **Android OS** – An operating system that runs on mobile devices.
* **Dynamo Database** – A database service provided by Amazon for persistent storage. This will store user, location, event, and advertisement information.
* **Event** – A live show, sale offering, or another special event that a nightlight establishment may hold and want advertised.
* **Frequently Asked Questions** (FAQ) – A section that serves as a tutorial to perceived (or generated from requests over time) user questions
* **Global Positioning System** (GPS) – Technology used to get the location of a user with their phone.
* **Graphical User Interface** (GUI) – The visual elements that a user interacts with for Bar-Pop. This includes the mobile app, logging in with a login screen, and browsing venues from a graphical map interface.
* **Manager** – The highest type of user and perhaps owner. A manager has privileges to do anything that a venue-owner or venue-hopper can do, but also additional privileges for adding/removing venues and managing advertisements.
* **Map** – The default screen (GUI) that the user accesses to interact with Bar-Pop. This is a map whose logic generates nearby venues, based on a person’s current location, and displays them on a navigable map interface.
* **Mobile Application** (App) – The software on the mobile device that users can install, open, or remove from their mobile device. It is the primary method that a customer (venue-hopper) will interact with Bar-Pop.
* **Mobile Device** – The phone hardware that contains an operating system with features that Bar-Pop will implement. This includes a GPS, the dependencies used to generate a GUI, an internet connection, and more.
* **Person Sensor** – The device(s) located at venue entrances and exits that detect when a person passed by them, and in which direction (exit or enter). This device interacts with Bar-Pop providing population data.
* **Population –** The number or proportion (proportion is what a user sees) of people at any venue.
* **Privileges –** The rules that prevent certain types of users from accessing or editing information, while allowing other users to. This pertains to the three levels of account: Manager, Venue-Owner, and Venue-Hopper with decreasing privileges from left to right.
* **Statistics –** Data that reflects real world trends in venues with person sensors. Statistics include, but are not limited to: The venues population over time, the most popular times of a venue on average, and population changes in response to venue events.
* **View –** A screen element of the GUI. Our app contains a login view, a register view, a map view, a list event view, and an options view.
* **Venue –** A nightlife establishment whose population may or may not be being tracked.
* **Venue-Owner –** The owner of a venue. Posts events and monitors his/her venue’s population.
* **Venue-Hopper –** The primary mass customer of Bar-Pop, they are seeking to find a venue to attend and use Bar-Pop to aid in their decision.

# 3. Functional Requirements Specification

## Stakeholders

* Venue-Hopper
* Venue-Owner
* Manager

## Actors and Goals

|  |  |
| --- | --- |
| **Actors** | **Goals** |
| Venue-Hopper | To register or log into an account, and to obtain venue information. |
| Venue-Owner | To obtain information about their venue and to edit information about the venue they own. This includes adding, removing, or modifying events. |
| Manager | To manage accounts and approve user privilege requests. To add and edit venue locations, events, and to obtain population and user statistics about the system. To manage advertisements and view statistics about the app, its events, and ad revenue. |
| Person Sensor | To detect and notify the system when a person has entered or exited a venue. |

## Casual Descriptions

**UC1** – Register: To create an account and request privileges.

**UC2** – Login: To log into a registered account.

**UC3 –** View Map: To populate the map displayed to users with nearby venues.

**UC4 –** View Venue Information: To obtain event and population information of a venue.

**UC5**- Update Population: To get the population of a specific venue.

**UC6 –** Manage Event: To create or edit an event for a specific venue.

**UC7** - Manage Venue: To modify any information about venue or to add or remove a venue.

**UC8** - Approve Privileges: To approve or decline registration privileges requests.

**UC9** - Manage Advertisements: To add, edit, or remove advertisements in the app.

**UC10** - View Statistics: To obtain app usage, ad revenue, and event statistics.

# 4. System Requirements

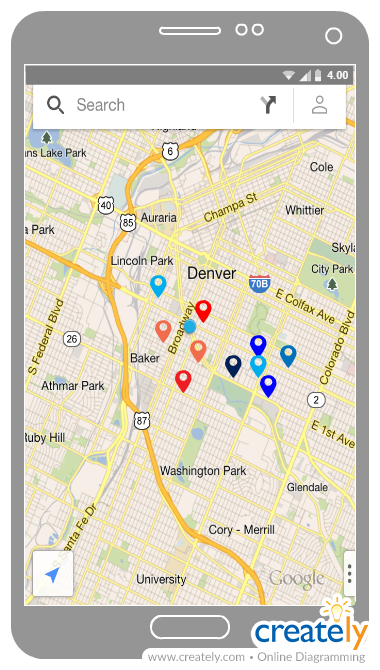
## 4.1 Functional Requirements

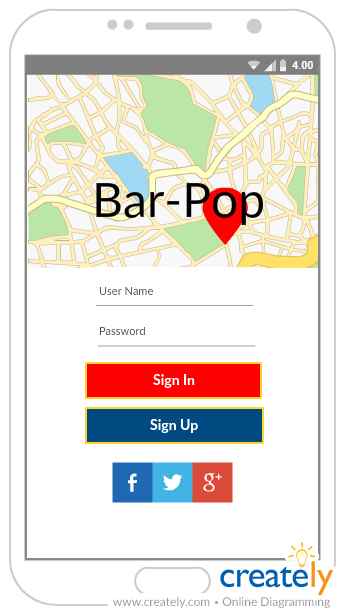
Development priorities are on a scale from 0 to 10 and are abbreviated as **PRI**

|  |  |  |
| --- | --- | --- |
| **Identifier** | **PRI** | **Requirement** |
| REQ1 | 10 | The system shall allow a user to create, log into and log out of an account. |
| REQ2 | 10 | The system shall provide a map GUI for users to view nearby venues and to see details about that venues population from the venues icons. |
| REQ3 | 8 | User accounts should provide different privilege levels and different functionality at each privilege for managers, venue-owners, and venue-hoppers. |
| REQ4 | 7 | The map interface shall allow users to select venues to view events at that venue. |
| REQ5 | 7 | Managers shall be able to create new venues and edit any information about existing venues (including events). |
| REQ6 | 6 | Venue-Owners shall be able to add and edit events at their venues. |
| REQ7 | 3 | Managers shall be able to manage advertisements and choose where they are placed within the app. |
| REQ8 | 2 | Managers and venue-owners will have a separate website GUI for managing venues, events, advertisements, and viewing statistics. |
| REQ9 | 2 | Managers and venue-owners will be able to view statistics about venues including (but not limited to): The venues population over time, the most popular times of a venue on average, and population changes in response to venue events. |

### 4.2 Traceability Table

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | UC1 | UC2 | UC3 | UC4 | UC5 | UC6 | UC7 | UC8 | UC9 | UC10 |
| REQ1 | **X** | **X** |  |  |  |  |  |  |  |  |
| REQ2 |  |  | **X** |  | **X** |  |  |  |  |  |
| REQ3 | **X** |  |  |  |  |  |  | **X** |  |  |
| REQ4 |  |  |  | **X** |  |  |  |  |  |  |
| REQ5 |  |  |  |  |  | **X** | **X** |  |  |  |
| REQ6 |  |  |  |  |  | **X** |  |  |  |  |
| REQ7 |  |  |  |  |  |  |  |  | **X** |  |
| REQ8 |  | **X** |  |  |  | **X** | **X** | **X** | **X** | **X** |
| REQ9 |  |  |  |  |  |  |  |  |  | **X** |

**REQ1 Illustration REQ2 Illustration**



## 4.2 Non-Functional Requirements

### Performance Requirements

Speed requirements should be fast enough to provide users with accurate population information, but not to the speed that the remote server slows down. Implementation will require adjusting the interval that population data is fetched from the hardware to the database until stable speeds are reached.

### Safety Requirements

As a mobile application that will potentially be used when walking around late at night, users should be warry of not looking up from their phone to avoid any harm. There is also the matter of intoxication and underage drinking. However, the app will not have a causal relationship with drinking. Periodic warning popups to look at one’s surroundings and not to drink and drive might be a good caution measure to prevent harm and lawsuits that may follow (though this seems highly unlikely).

### Security Requirements

User email and city of residence location must be secure from malicious attacks. Additionally, population information could be used for nefarious purposes. Thus, exact population numbers will not be displayed for any venue. Additionally, this information must be secured to prevent malicious access.

### Software Quality Attributes

Bar-Pop will be easy to use due to its minimalistic nature of having only one main view (Map View). The software will be free and available for Android in the Google Play Store and for IOS in the App Store. Thus, the app will be available to nearly every venue-hopper since both major operating systems are supported and use isn’t barred by income level.

In its initial stages, the application may not provide accurate and reliable data. However, if dedication is given to maintaining the project over time, then the robustness of the application will increase with updates and further development.

## Fully-Dressed Descriptions of Use Cases

In main success scenarios, odd numbers correspond to a user that is performing the event, and even numbers correspond to the system.

|  |  |  |
| --- | --- | --- |
| **UC1** – Register | | |
| Initiating Actor | User (A Venue-Hopper or Venue-Owner included by extension) |
| Actor’s Goal | To create an account that allow the user to view or do things based on granted privileges |
| Participating Actors | Venue-Hopper, Venue-Owner, and Manager |
| Preconditions | User has an email address, access to the app, and an internet connection. |
| Postconditions | User is registered in system and privilege are pending approval by manager if they were requested |
| **Main Success Scenario** | | |
| **1** | User selects the register option |
| **2** | System requests required user information including privilege requests |
| **3** | User fills out required information and submits the information to the system |
| **4** | System verifies and stores information and sends a confirmation email to user |
| **5** | User confirms their email address |
| **6** | System shows success notification and sends privilege request notification to manager |

|  |  |  |
| --- | --- | --- |
| **UC2** – Login | | |
| Initiating Actor | User (A Venue-Hopper, Venue-Owner included by extension, and a Manager whose account is manually registered) |
| Actor’s Goal | To log into an already registered account |
| Participating Actors | Venue-Hopper, Venue-Owner, and Manager |
| Preconditions | User has access to the app or website, an internet connection, and their user information. |
| Postconditions | User is logged in and able to view or do things based on their privileges |
| **Main Success Scenario** | | |
| **1** | User selects the login option |
| **2** | System requests required user login information |
| **3** | User fills out required information and submits the information to the system |
| **4** | System verifies the login information and sends confirmation message |

|  |  |  |
| --- | --- | --- |
| **UC3** – View Map | | |
| Initiating Actor | User (Venue-Hopper and Venue-Owner included by extension) |
| Actor’s Goal | To view the map of nearby venues |
| Participating Actors | Venue-Hopper, Venue-Owner |
| Preconditions | User has access to app, and an internet and GPS connection. User is logged in. |
| Postconditions | User can navigate map view and select venues to view their information |
| **Main Success Scenario** | | |
| **1** | User choses an option to open the map |
| **2** | System returns the map with venues in a radius of the user’s location |

|  |  |  |
| --- | --- | --- |
| **UC4** – View Venue Information | | |
| Initiating Actor | User (Venue-Hopper or Venue-Owner included by extension) |
| Actor’s Goal | To obtain venue information |
| Participating Actors | Venue-Hopper, Venue-Owner |
| Preconditions | Venue-Hopper has opened the map view and has selected a venue from which they desire information |
| Postconditions | Venue information is provided to the user |
| **Main Success Scenario** | | |
| **1** | User selects a venue on the map |
| **2** | System successfully returns venue information, including the population count and events |

|  |  |  |
| --- | --- | --- |
| **UC5** – Get Population | | |
| Initiating Actor | Person Sensor |
| Actor’s Goal | To detect and notify the system when a person has entered or exited a venue |
| Participating Actors | Person Sensor |
| Preconditions | Person Sensor has a connection to the system |
| Postconditions | The venue population associated with the person sensor is up to date. |
| **Main Success Scenario** | | |
| **1** | A person passes by the person sensor and the person sensor sends the event to the system |
| **2** | System receives event information and updates the appropriate venue. A success message is sent to the sensor. |

|  |  |  |
| --- | --- | --- |
| **UC6** – Manage Event | | |
| Initiating Actor | Venue-Owner or Manager |
| Actor’s Goal | To create an event for a venue or to edit an existing event. Venue is specifically the venue-owner’s venue if venue-owner initiated the scenario. |
| Participating Actors | Venue-Owner or Manager |
| Preconditions | User is logged in, has access to website, and has the privileges to create an event. If managing a preexisting event, then the event must exist. |
| Postconditions | An event is added to the venue and included in that venue’s information or an existing event is updated with modified information. |
| **Main Success Scenario** | | |
| **1** | User selects an add or edit event option for a specific venue |
| **2** | System checks for user privileges and if they are appropriate, requests event information |
| **3** | User fills out event information and submits it to system |
| **4** | System validates information and if it is ok, adds it to the venue. A confirmation message is sent to the user. |

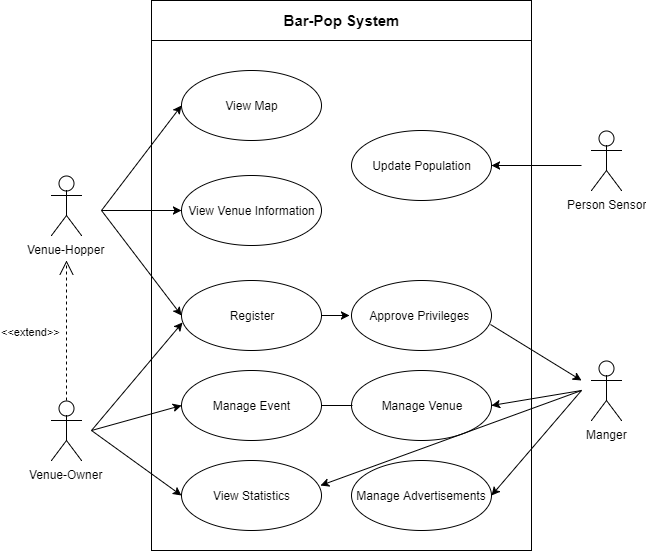
|  |  |  |
| --- | --- | --- |
| **UC7** – Manage Venue | | |
| Initiating Actor | Manager or Venue-Owner |
| Actor’s Goal | To create a new venue (manager only), edit an existing one (venue-owner), or delete venue (either). |
| Participating Actors | Manager or Venue-Owner |
| Preconditions | User is logged in and has access to the website and if editing an existing venue, the venue exists. |
| Postconditions | A new venue is created and can be viewed from the map, an existing venue updates its information, or the venue is removed. |
| **Main Success Scenario** | | |
| **1** | Manager selects the add or edit venue option |
| **2** | System requests new venue information |
| **3** | Manager fills out new venue information and submits it to system |
| **4** | System validates information and if it is ok, adds it to the venue. A confirmation message is sent to the user. |

|  |  |  |
| --- | --- | --- |
| **UC8** – Approve Privileges | | |
| Initiating Actor | Manager |
| Actor’s Goal | To approve or decline user privilege requests |
| Participating Actors | Manager |
| Preconditions | Manager has received a request for user privileges |
| Postconditions | A user is either granted or denied the privileges they requested |
| **Main Success Scenario** | | |
| **1** | Manager logs in and selects option to verify user privileges |
| **2** | System requests manager’s decision |
| **3** | Manager selects decision |
| **4** | System updates user information and returns success notification. |

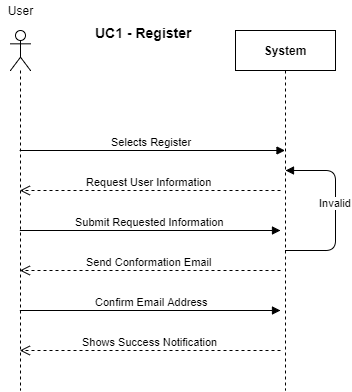
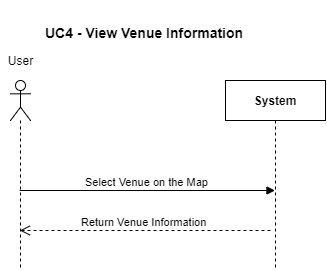
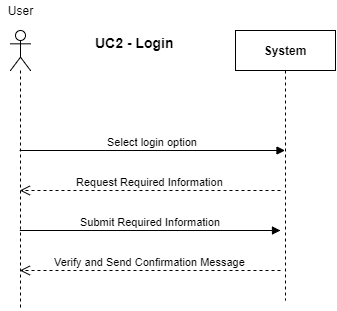
|  |  |  |
| --- | --- | --- |
| **UC9** – Manage Advertisements | | |
| Initiating Actor | Manager |
| Actor’s Goal | To add, remove, or edit advertisements |
| Participating Actors | Manager |
| Preconditions | Manager is logged in and has access to the website. Advertisers want to advertise in the app or an advertising platform provides ads to be used. |
| Postconditions | An advertisement is added, modified, or removed from the app. |
| **Main Success Scenario** | | |
| **1** | Manager selects manage advertisements option |
| **2** | System checks user privileges and if they match that of manager, system returns existing advertisements |
| **3** | Manager selects to add, remove, or modify an advertisement |
| **4** | System requests ad information from user for add or modify ad option. If remove option, system removes the add and sends confirmation message |
| **5** | User adds or modifies the advertisement and submits the information to the system |
| **6** | The system updates the advertisement and returns a success notification |

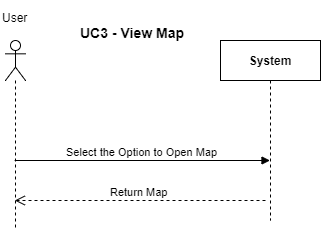
|  |  |  |
| --- | --- | --- |
| **UC10** – View Statistics | | |
| Initiating Actor | Manager |
| Actor’s Goal | To obtain app usage, ad revenue, and event statistics. |
| Participating Actors | Manager |
| Preconditions | Manager is logged in and has access to the website. |
| Postconditions | Manager can view statics about app usage, ad revenue, and events. |
| **Main Success Scenario** | | |
| **1** | Manager selects a view statistics option |
| **2** | System checks user privileges and if they are manager the system successfully returns the appropriate statistics |

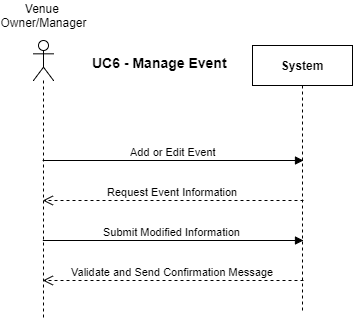
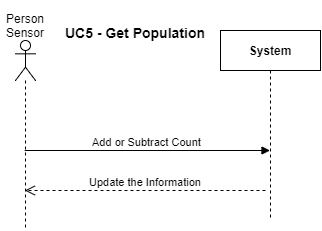
## Use Case Diagram

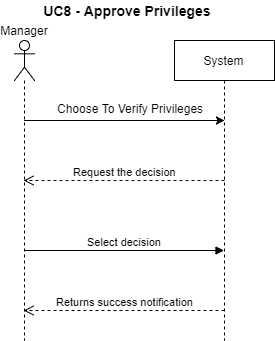
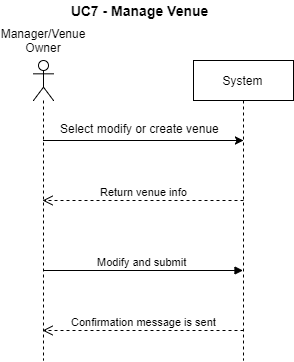


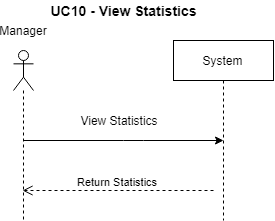
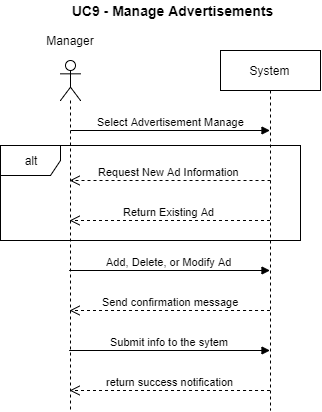
## System Sequence Diagrams











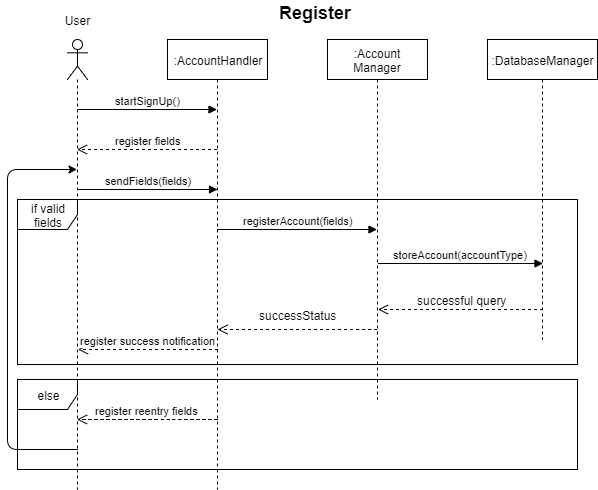
# 6. Domain Analysis

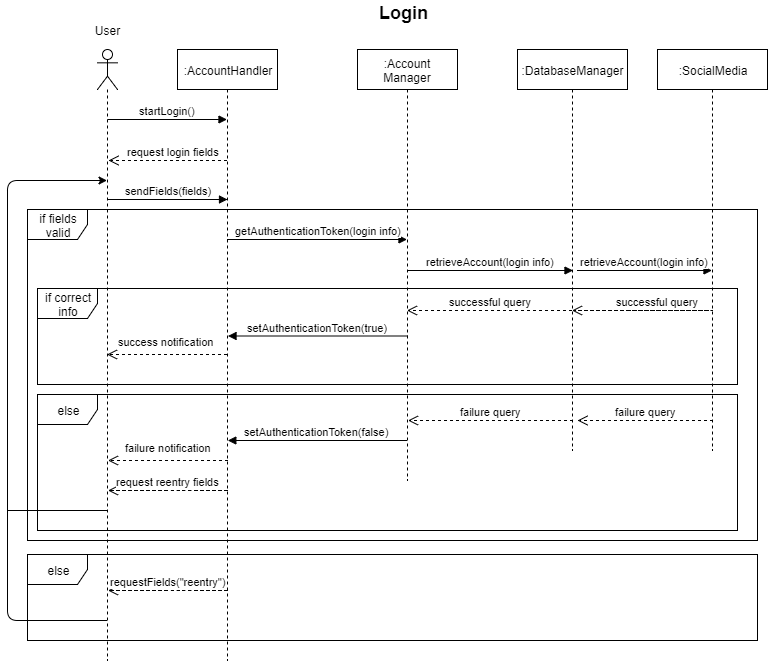
## System Domain Model

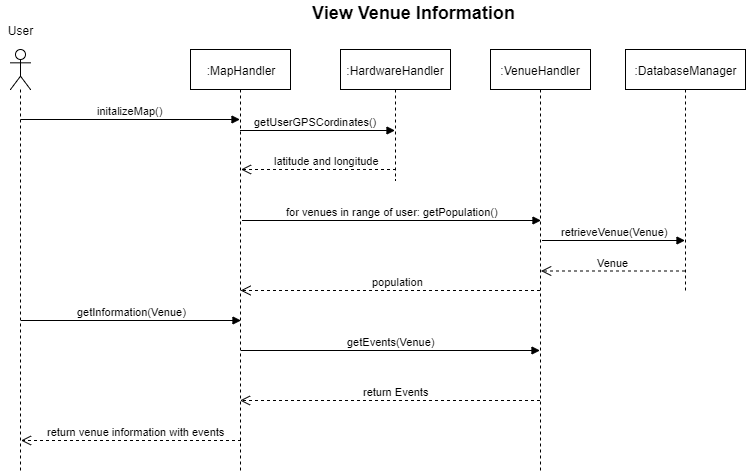
## Database Model

# 7. Interaction and Class Diagrams

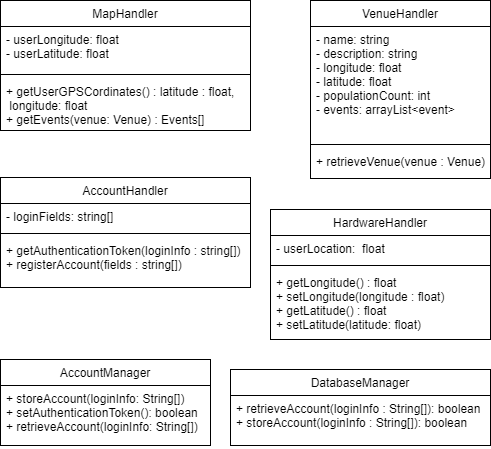
## 7.1 Sequence Diagrams

****

****

****

## 7.2 Class Diagrams

****

# 8. Interface Specification

## Mobile Application User Interface

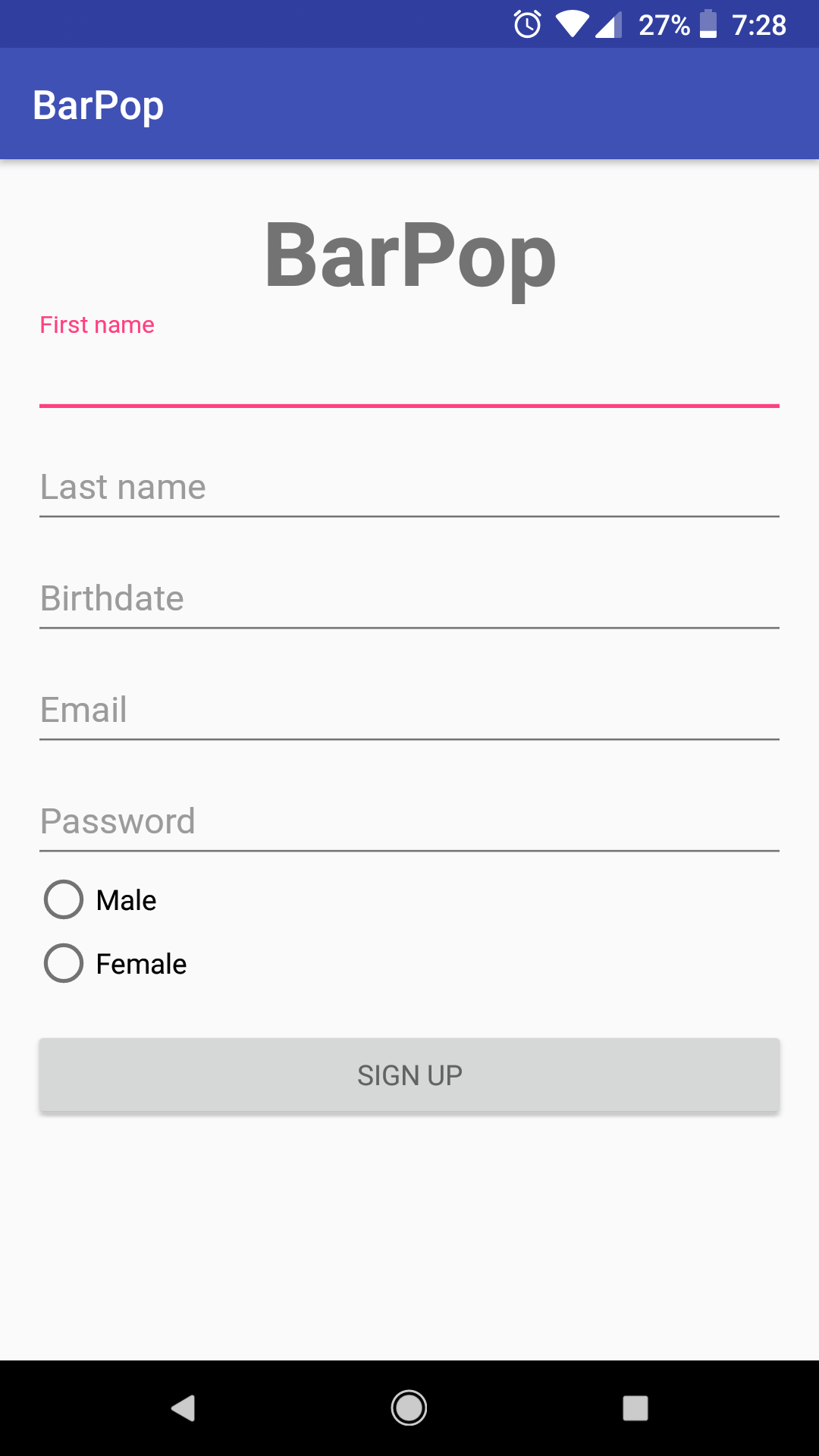


Figure 2 – Registration View

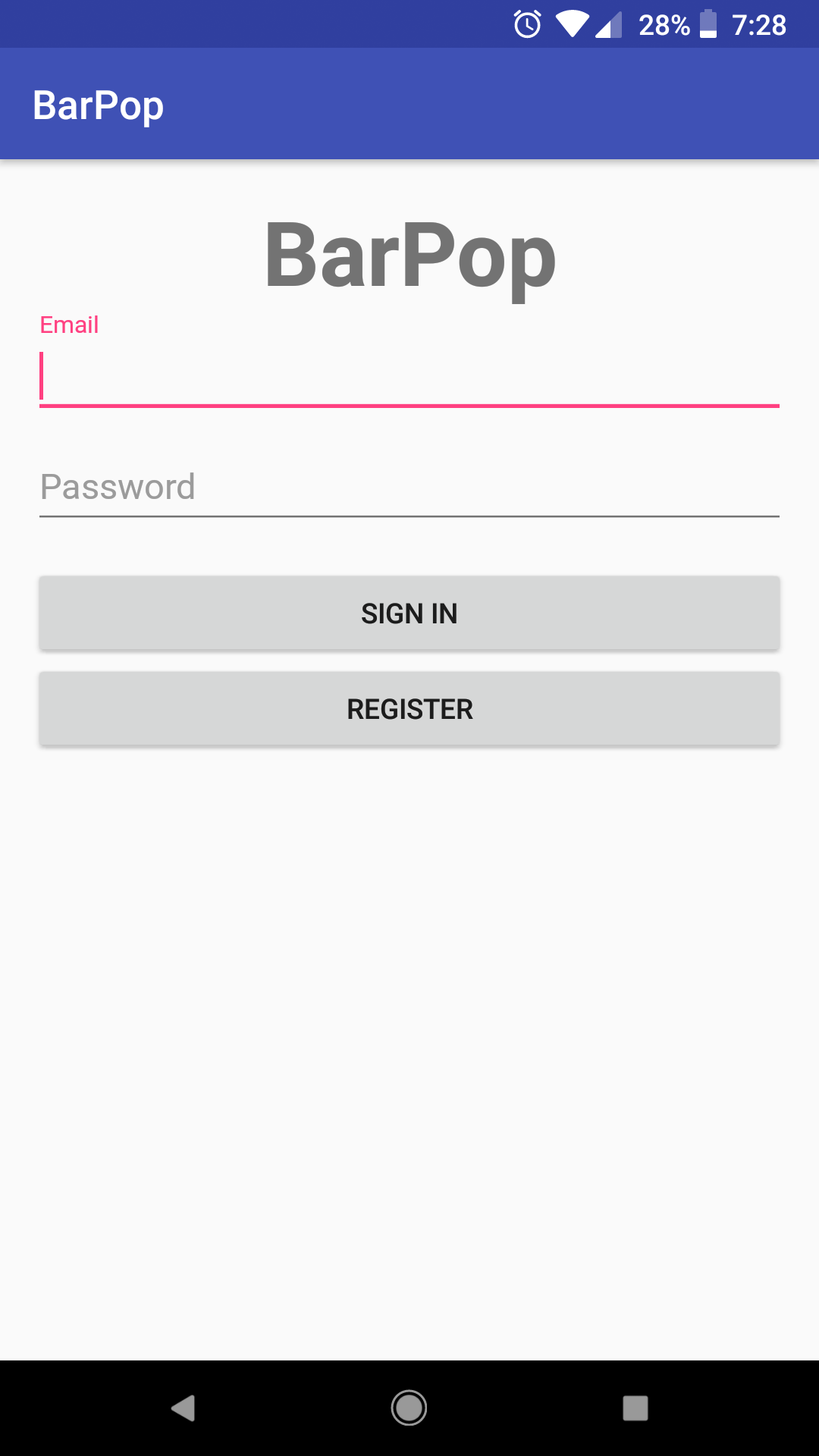


Figure 1 – Login View

1. **First Time Use**

* Login and Registration Views

First-time users will be brought to the Login View (Figure 1). Users can potentially login using Facebook, Google, Twitter, or email and password forms. Under these forms is a “Sign In” button followed by a “Sign Up” button. The Registration View (Figure 2) is opened by pressing the “Register” button. Registration View consists of email, password, and date of birth input forms. There will also be an option to claim a venue (request privileges). Once logged in, users will automatically stay logged in for future application openings, unless specifically choosing to log out in the options view (Figure 4).

1. **Consecutive Application Openings**

* Map View

After completing the Tutorial Overlay, logged in users will first see a map centered on their location (Figure 3), with a radius (size TBD) surrounding them. If any supported venues are within this radius, they will appear as colored pins on the map and may be clicked on. In the top left of the map view will be a small menu button that can be pressed to get to the options view.

* Map Pins

Map Pins will be colored according to population ratio. If a venue is mostly full, then it will be a darker shade of red than a venue that is less full. Less full venues will be a lighter shade of blue than moderately full venues. Yellow is a color used for values in the middle, between blue and red.

* Venue Info View

Upon pressing a pin an overlay will appear, called the Venue Info View (GUI Pending) This view contains population information, and event information.

* Menu View

The menu view is reached from pressing the top left button in the Map View and the is an option to go to profile and log out (Pending). The top of the Menu View shows the name of the user logged and map navigation options.

* Help View

The Help View contains an FAQ section, an option to review the app, and a “Contact Us” button which opens the default mail app linked to our email (email address pending). We will potentially have a Facebook page for which a “Visit Us on Facebook” button will link to. The last option in the Help View is to sign out of the logged in account.

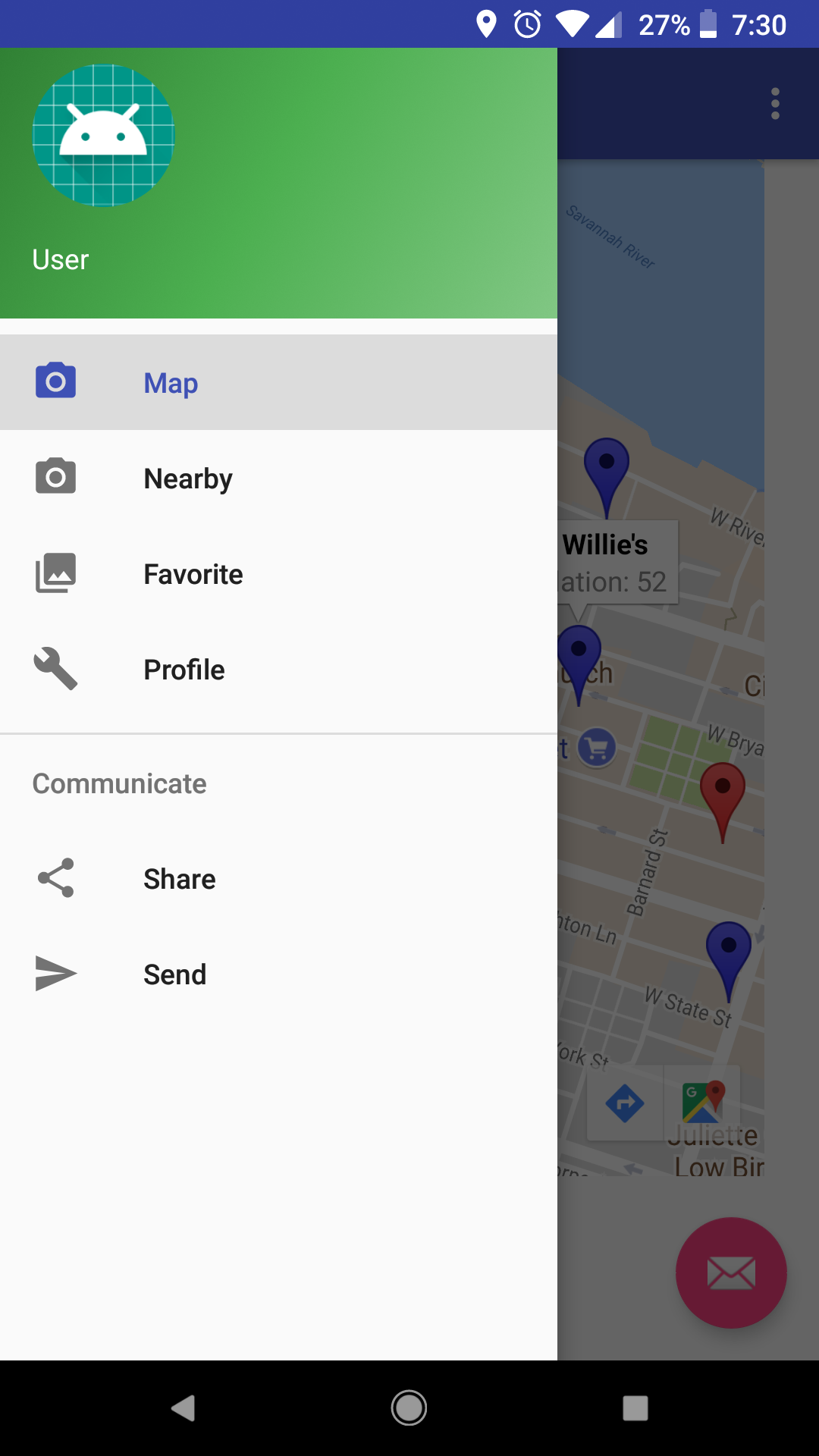


Figure 4 – Menu View

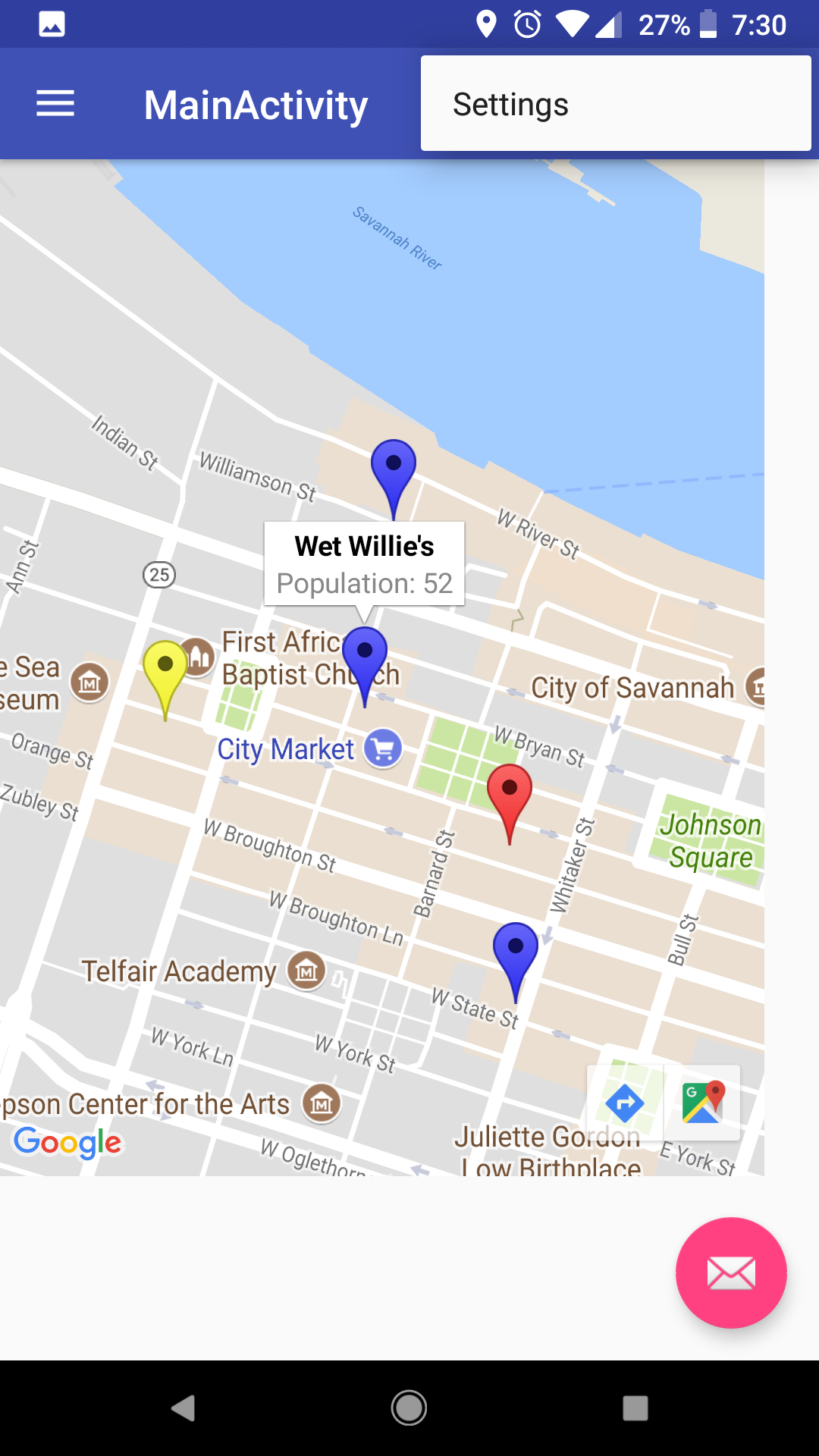


Figure 3 – Map View

## Website User Interface

* Login and Registration Views

Users can potentially login using Facebook, Google, Twitter, or email and password forms. Under these forms is a “Sign In” button followed by a “Sign Up” button. The Registration View is opened by pressing the “Sign Up” button. Registration View consists of email, password, and date of birth input forms. There will also be an option to claim a venue (request privileges). If the user has an account, they can choose the login option to log in.

* User dashboard

Upon logging in the user will be taken to their dashboard. Users are shown their user information and what else is displayed alongside the information depends on the user’s privileges. If the user is the owner of a venue, they will be given an option to manage their venue. If the user is a manager, they can select from any venue to manage, or to create a new one. Additionally, managers will be given an option to manage advertisements.

* Manage Venue

The options to edit venue information and add events are here for venue-owners, and for managers the option to add or delete a venue is here.

* Manage Event

The option to add or edit event information that is tied to a venue. Venue-owners can only access events that are tied to their venue. Managers can access all events.

* Manage Advertisements

The option to add, remove, or edit advertisements is available only to the manager. They can choose where in the views the advertisements are displayed and what advertisements are displayed.

* View Statistics

Here the manager can view a variety of to be determined statics.

## Hardware Interfaces

* **Sensors** - People counting with lasers

The hardware used to gain populations from each venue is a Raspberry Pi attached to two lasers. These lasers detect the direction of the object -hopefully person- passing them. This object should be a person entering or exiting a venue. At an interval that can be sustained by the remote database, the Raspberry Pi will connect to the remote server using a built in WIFI chip and update the database located on that server with its population count. The Raspberry Pi holds the current count in its onboard memory and potentially resets the count to zero after closing hours of a venue. Sensors will potentially allow remote access for maintenance.

* **Phones** – Android OS enabled phones

The hardware required from the phone is that which is required to run the mobile application, in addition to providing GPS location data.

* **Server** - Remote Hosting

The hardware for the remote hosting is in a black-box to the developers, thus we only interact with the server itself, which will be running Ubuntu Server LTS OS. This will be accessed through SSH and web calls.

## Software Interfaces

* Android Developer Studio for Mobile Application
  + Mobile Application uses native APIs for maps and GPS data
  + Mobile Application will potentially fetch data from Facebook, Google, and/or Twitter for logins, events, and reviews
* AWS for hosting an Ubuntu Server LTS OS
* MySQL for database that will be installed and running on the Ubuntu Server
* Raspberry Pi (people sensor) running Python to send SQL queries for updating the MySQL database

## Communications Interfaces

The mobile application, Raspberry Pi, and website will connect to the server containing the MySQL database using TCP/IP protocols.

# 9. System Architecture and Design

## Architecture Styles

Since our application is largely focused on fetching and providing information that is stored on a server in a database, our architecture pattern is most accurately captured by the client-server pattern with aspects of a repository architecture. There is the possibility of thousands of clients that will all access our server, each with their own hardware connecting to and request information from our hardware (server).

On each user’s phone or web-browser, they can access our application views. The models that drive these views are what gets information from the database. In this way, our method of processing that data on the application and feeding the information to the view reflects that of a model-view architecture pattern.

## Persistent Data Storage

Mongo Database, hosted on AWS, will be used as Bar-Pop’s primary persistence storage. In addition to database storage, Bar-Pop will implement caching for user login sessions on their mobile phones to improve usability and prevent users from having to login upon each app reopening.

## 9.5 Network Protocol

Interfacing between the mobile and web applications and Mongo Database involves RESTful web calls to AWS. Thus, HTTPS protocols and RESTful API protocols are required for communication. We will not be implementing any custom protocols.

## 9.6 Hardware Requirements

Android enabled device with no later than three updates behind the most current Android OS. Modern web-browsers that can process JavaScript and HTML5 will be required to access the web based interface that is available to managers and venue-owners.

# 10. Appendix

Link to code, sprint reports, and other documentation can be found on our project’s GitHub:

<https://github.com/CSCI-3321-Sofware-Engineering>

## 10.1 Contribution Breakdown

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sprint 1 |  | 8/27 | 10/3 | 37 | Completed |
| Assess Requirements | Team [All] | 8/27 | 9/7 | 11 | Complete |
| Learn Android Studio | Team [All] | 9/7 | 10/5 | 28 | In Progress |
| Create AndroidStudio Hello World | Team [All] | 9/7 | 10/5 | 28 | In Progress |
| Write Vision Document | Write - [EB], Diagrams [All], Proofread- [All] | 9/29 | 10/3 | 4 | In Progress [70%] |
| Sprint 2 |  | 10/5 | 10/19 | 14 | Completed |
| Create Web Server | [EB][AB] | 10/5 | 10/19 | 14 | In Progress |
| Create Database and Tables for User Registration and Map | [EB][AB][KB] | 10/5 | 10/19 | 14 | Completed |
| App: User Registration | [KB][KP] | 10/5 | 10/19 | 14 | In Progress |
| App : User Login | [KB][KP][AB] | 10/5 | 10/19 | 14 | In Progress |
| Connect Raspberry PI To Database | [AN][AB] | 10/5 | 10/19 | 14 | Not started |
| Sprint 3 |  | 10/19 | 11/2 | 14 | Completed |
| Map View | [AB][KB][KP] | 10/19 | 11/2 | 14 | In Progress |
| Create Web Server | [EB][AB] | 10/19 | 11/2 | 14 | Complete |
| Connect Sensor To Raspberry Pi | [AN][AB] | 10/19 | 11/2 | 14 | Not started |
| App : User Registration | [AB][KB][KP] | 10/19 | 11/2 | 14 | Complete |
| App: User Login | [KB][AB][KP] | 10/19 | 11/2 | 14 | In Progress |
| Design Architecture Document | Team [All] | 10/19 | 11/2 | 14 | Complete |
| Sprint 4 |  | 11/2 | 11/23 | 21 | Completed |
| Map View | [AB][KB][KP] | 11/2 | 11/23 | 21 | Complete |
| Connect Sensor To Raspberry Pi | [AN][AB] | 11/2 | 11/23 | 21 | In Progress |
| App: User Login | [KB][AB][KP] | 11/2 | 11/23 | 21 | Complete |
| App: Advertisement | [KB][AB][KP] | 11/2 | 11/23 | 21 | In Progress |