System Architecture and Design

for

Bar-Pop

Version 1.0 approved

Prepared by Evan Bonsignori

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

CSCI 3321

11/7/2017

Table of Contents

Table of Contents 2

Revision History 2

1-4. Sections in Requirements Specifications Document 3

5. Use Cases 3

5.1 Actors and Goals 3

5.2 Casual Descriptions 3

5.3 Fully-Dressed Descriptions 4

5.4 Use Case Diagram 12

5.5 System Sequence Diagrams 13

6. Class Diagrams and Interface Specification 19

6.1 System Domain Model 19

6.2 Database Model 20

6.3 Mobile Application User Interface 21

6.4 Website User Interface 22

6.5 Hardware Interfaces 23

6.6 Software Interfaces 23

6.7 Communications Interfaces 24

7. System Architecture and Design 24

7.1 Architecture Styles 24

7.2 Subsystems 24

7.3 Subsystems Mapped to Hardware 24

7.4 Persistent Data Storage 24

7.5 Network Protocol 24

7.6 Hardware Requirements 24

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason for Changes** | **Version** |
|  |  |  | 1.0 |

# 1-4. Sections in Requirements Specifications Document

The first three sections for the final document can be found in the requirements document. This document instead focuses on interaction diagrams, class diagrams, and System architecture and Design. The final document will concatenate every section.

# 5. Use Cases

## 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 –** Add Event: To create an event for a specific venue.

**UC7** - Modify Event: To modify information about an event or remove it.

**UC8** - Add Venue: To create a new venue.

**UC9** - Manage Venue: To modify any information about venue or to remove it.

**UC10** - Approve Privileges: To approve or decline registration privileges requests.

**UC11** - Manage Advertisements: To add, edit, or remove advertisements in the app.

**UC12** - View Statistics: To obtain app usage, ad revenue, and event statistics.

## Fully-Dressed Descriptions

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** – Add Event | | |
| Initiating Actor | Venue-Owner or Manager |
| Actor’s Goal | To create an event for a venue. 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 |
| Postconditions | An event is added to the venue and included in that venue’s information |
| **Main Success Scenario** | | |
| **1** | User selects an add 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** – Modify Event | | |
| Initiating Actor | Venue-Owner or Manager |
| Actor’s Goal | To modify the information of an existing event for a venue. 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 modify the event |
| Postconditions | The event is updated with the modified information |
| **Main Success Scenario** | | |
| **1** | User selects an edit event option for a specific event at a specific venue |
| **2** | System checks for user privileges and if they are appropriate, returns event information to user |
| **3** | User modifies event information and submits it to the system |
| **4** | System validates information and if it is ok, updates the event. A confirmation message is sent to the user. |

|  |  |  |
| --- | --- | --- |
| **UC8** – Add Venue | | |
| Initiating Actor | Manager |
| Actor’s Goal | To create a new venue. |
| Participating Actors | Manager |
| Preconditions | Manager is logged in and has access to the website. |
| Postconditions | A new venue is created and can be viewed from the map. |
| **Main Success Scenario** | | |
| **1** | Manager selects add 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. |

|  |  |  |
| --- | --- | --- |
| **UC9** – Mange Venue | | |
| Initiating Actor | Venue-Owner or Manager |
| Actor’s Goal | To modify the information of an existing venue. Venue is specifically the venue-owner’s venue if venue-owner initiated the scenario. If a manager, venue can be deleted. |
| Participating Actors | Venue-Owner or Manager |
| Preconditions | User is logged in, has access to website, and has the privileges to modify the selected venue |
| Postconditions | The Venue is updated with the modified information |
| **Main Success Scenario** | | |
| **1** | User selects a modify venue option for a specific venue |
| **2** | System checks for user privileges and if they are appropriate, returns venue information to user |
| **3** | User modifies venue information and submits it to the system |
| **4** | System validates information and if it is ok, updates the venue. A confirmation message is sent to the user. |

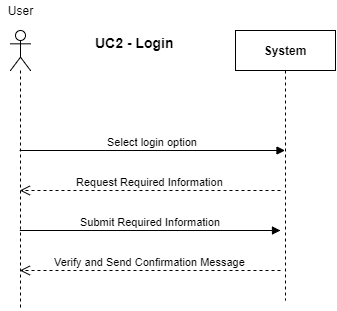
|  |  |  |
| --- | --- | --- |
| **UC10** – Approve Privileges | | |
| Initiating Actor | Manager |
| Actor’s Goal | To approve or decline user privilege requests |
| Participating Actors | Manager |
| Preconditions | Manager has received a request for user privileges |
| Postconditions | A user is either 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. |

|  |  |  |
| --- | --- | --- |
| **UC11** – 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 |

|  |  |  |
| --- | --- | --- |
| **UC12** – 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

## C:\Users\_\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Sequence-UC1.pngSystem Sequence Diagrams

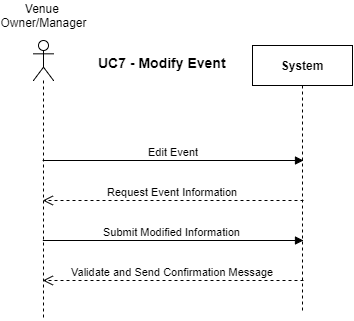
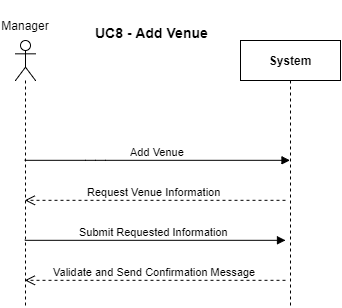


# 

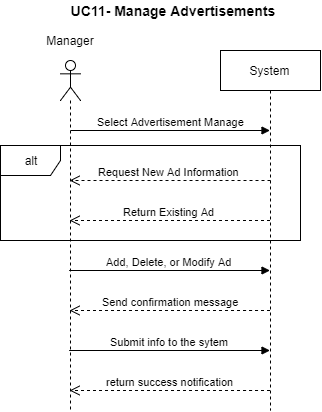
# 

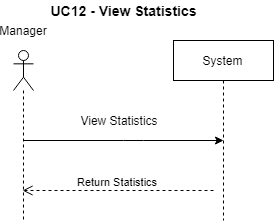
# 

# 



# 





# 6. Class Diagrams and Interface Specification

## System Domain Model

## C:\Users\_\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Database.pngDatabase Model

## Register ViewMap View Mobile Application User Interface

Figure 4 – Register View

Figure 3 – Login View

1. **First Time Use**

* Login and Registration Views

First-time users will be brought to the Login View (Figure 3). 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 4) 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). Once logged in, users will automatically stay logged in for future application openings, unless specifically choosing to log out in the Help View (Figure 5).

1. **Consecutive Application Openings**

* Map View

After completing the Tutorial Overlay, logged in users will first see a map centered on their location (Figure 4), with a radius (size TBD) surrounding them. If any supported venues are within this radius, they will appear as pins on the map and may be clicked on. In the top left of the map view will be a small menu button (Where the search icon is in Figure 4).

* 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. Potential customization options could include user notifications for certain types of locations. More popular locations could be labelled “hot” and less frequented locations as “chill.”

* Venue Info View

Upon pressing a pin an overlay will appear, called the Venue Info View (GUI TBD). 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 (search icon in Figure 4) and at the bottom gives an option to log out and be brought back to the Login View. The top of the Menu View shows the name of the user logged in and a “Help” button (GUI TBD).

* Help View

The Help View (Figure 5), contains FAQ, an option to review the tutorial, and a “Contact Us” button which opens the default mail app linked to our email (email TBD). 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.

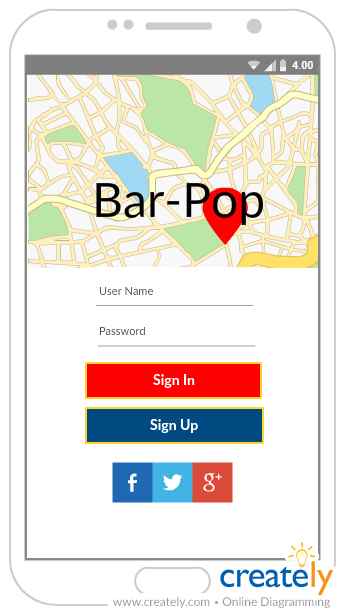
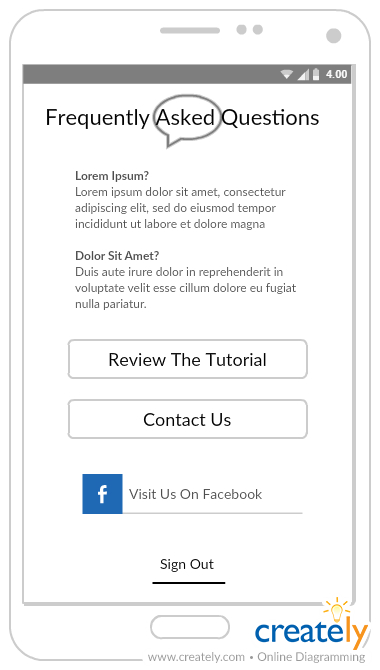


Figure 4 – Map View

Figure 5 – Help 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 (Figure 4) 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 log ins, events, and reviews
* Python Django Framework for website
* AWS for hosting an Ubuntu Server LTS OS
* MySQL for database that will be installed and running on the Ubuntu Server
* Raspberry Pi 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.

# 7. 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. 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.

## Subsystems

This and the following systems are incomplete. As this document is a draft, they are pending completion.

## Subsystems Mapped to Hardware

## Persistent Data Storage

## Network Protocol

## Hardware Requirements