**Software Design Specification:**

**Around Town**

***Version 1.0***

***February 25 , 2019***

**Developers:**

Matthew Hendrick

Brandon Stehling

Chloe Osgood

Misael Aguayo

Sean Stepan

# Table of Contents

[**Table of Contents**](#_pvcq0g9w3ffj) **1**

[**1. Introduction**](#_3oxp2bje21th) **2**

[1.1 Recap](#_8k7kw9wn5u3e) 2

[1.2 Purpose of this Document](#_pnlkz6td4zug) 2

[1.3 Scope](#_w67o0ok9lj2a) 2

[1.4 Context](#_l0ohlpa3vhyq) 2

[1.5 Major Constraints](#_iwd488w0t87t) 2

[**2. UI Design**](#_1249v7u2xe5t) **3**

[2.1 Description](#_f38p15ahrupt) 3

[**3. Testing Issues**](#_1koum1pn994p) **4**

**4. UML Diagrams 4**

4[.1 Use Case Diagram](#_cduwywg0o7lg) 4

4[.2 Sequence Diagrams](#_f6c2luq096ix) 5

4[.2.1 Follow venue/group](#_i05hm0xsatvr) 5

4.2.2 Listed venue 5

4.2.3 Edit event 6

4.2.4 Location 7

4.3 Class Diagram 7

4.3.1 Entity Classes 7

4.3.2 Notification Class 8

4.3.3 User Interface 8

4.3.4 Relationships 9

**5. Low-Fidelity Prototypes 9**

[**6. Implementation plan**](#_fnbf7b76blj7) **13**

[6.1 Test bed](#_yx4huaeollo) 13

[6.2 Programming Languages](#_3c7pnuatqoki) 13

# 1. Introduction

## **1.1 Recap**

Our application, Around Town, is a centralized place to find events happening near the user. The events will be displayed on a map, with the location of the venue. The application will use either the location of the user or a manually enter zip code in order to show the events happening within a radius defaulted to 50 miles. It will allow for users to select the event and find out more information along with reviews pulled from google.

## **1.2 Purpose of this Document**

The purpose of this document is to outline the framework on which our application is built on. This includes information such as the basic UI design of our application as well as the software and tools we will use to build our application. This document will also provide use cases which will flesh out user stories and overall functionality of the application

## **1.3 Scope**

Since there are many venues which do not have the publicity or interaction with their followers to get everyone in doors, our application will provide the solution. The scope of our application is limited to informing users of events happening in their cities. Our application’s purpose is not to sell tickets. This limits our scope and effectively allows us to focus on user experience.

## **1.4 Context**

There are not many applications which are in direct competition with ours. Although there are large services such as StubHub, we do not consider them a competitor as the goal of our application is not to sell tickets. One application which may be a competitor is Facebook. Facebook has a section on its application which informs users of events happening in their town. However, since Facebook is primarily a social media site, they are not able to cater to finding events as effectively as we would.

## **1.5 Major Constraints**

There are two major constraints categories that must be considered. One of them is time. Because this is a semester project, we only have a limited amount of time in which we could develop our product. This limits the functionality that we could implement within this time constraint. A second constraint category deals with dependencies. Our application will be dependent on the Google Maps API as well as MongoDB. If functionality happens to change in either of these dependencies, we’d have to refactor our application.

# 2. UI Design

Around Town will use a simple and easy to use User Interface that will be aesthetically pleasing. We want people to enjoy using our app and an important part of that is the User Interface.

## **2.1 Description**

From the first time the app is opened the user will have to make a major decision about what the app will do and display. This will be based on whether they want to use the app as a normal user, or as the manager of a venue. They will receive a prompt, two buttons, asking them to choose either user or manager. Depending on what they choose they will be directed to either a user or manager sign in page. If they do not have an account they will be asked to create one.

**2.2 User Side UI**

User side user interface, as stated before, will be simple to understand will being aesthetically pleasing. Here is where the user can search for events going on around them or for events in other cities. The home page will display a map and below this map there will be a list of venues and or events in the area that they user can browse by scrolling up and down. When a user selects an event they will be directed to that event’s page and from there the user can view the event’s details and be able to bookmark or star that event and be able to set up notifications for that event. Also, whenever a user selects a venue they will be directed to that venue’s page, here they will be able to view all of the events that are upcoming for this venue. If they like the venue enough they will be able to hit the “follow venue” button and it will add them to that venues notifications list so they can be alerted whenever they post an event. Many of these notifications can be toggled on and off in the settings. Whenever the user enters a new city they will be alerted and the app will send them a notification with all the different events going on in that city (they will be able to turn off this feature in the settings).

**2.3 Manager side UI**

The manager side UI will not have a map integrated with it, this side will be strictly for creating, posting, and managing events that will be/are on the app. Managers will be able to view a calendar of the events they have coming up, as well as a full list of the events they have upcoming. Managers will have the option to edit these by selecting them and hitting the edit button. Creating new events will be very intuitive and easy, managers will be able to do everything from add age restrictions to adding tags for the event so people know what kind if event it will be. Managers will also be able to create a queue of events they want to post, that way they can set a specific time they want their event to be posted.

**2.4 UI Scheme**

Both sides will follow the same button and color scheme. The UI will be very consistent and nothing will seem like it is out of place.

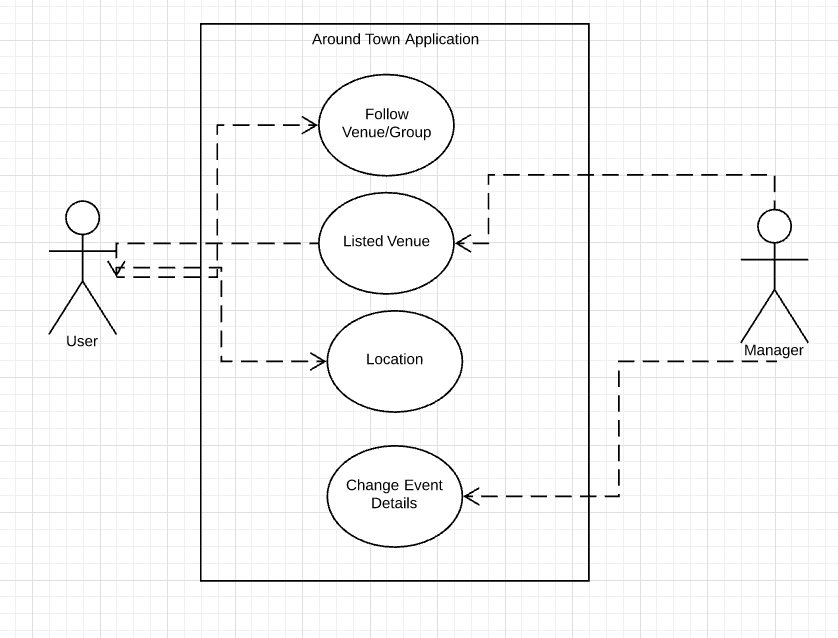
# 3. Testing Issues

Our application will undergo several simulation tests. Some tests may include distributing the application to our target demographic. From user feedback, we can decide to improve or modify our functionality. Another class of test would be venue accounts. Managers of venues such as Mad Hatters can test the application and report their feedback. Based on these tests, we can take a census of working functionality and improvements to be made.

**4. UML Diagrams**

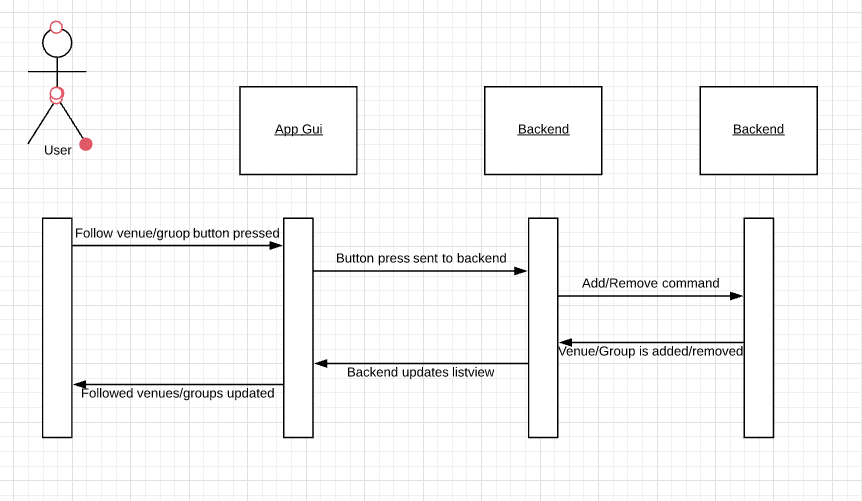
## **4.1 Use Case Diagram**

Users are allowed to follow venues or groups in the application. Both users and managers may interact with listed venues. However, listed venues are read only to users and can be modified by venue manager\s. Users can also interact with location based functionality in the application. Only managers are able to change event details and users are not allowed this functionality.



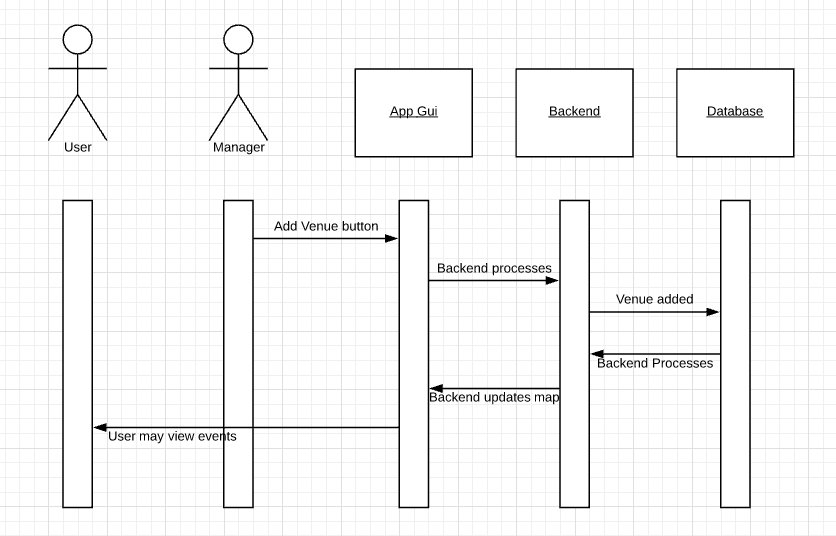
## **4.2 Sequence Diagrams**

### 4.2.1 Follow venue/group



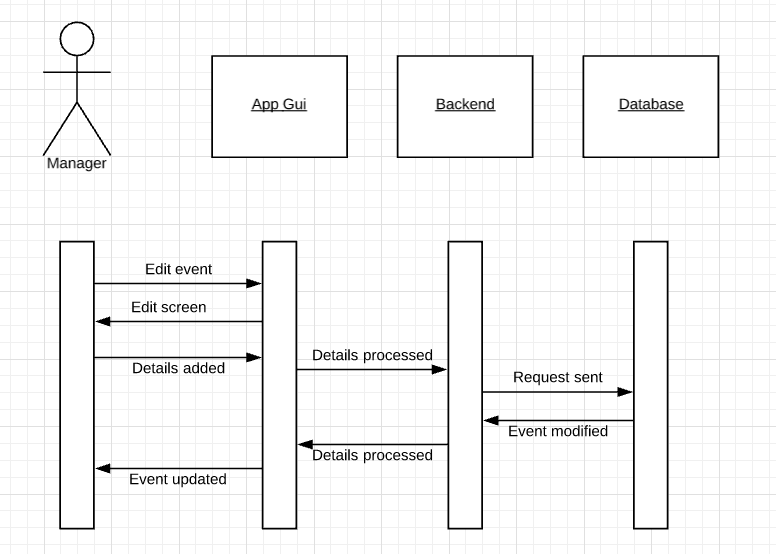
When a user presses the follow venue/group button, the request is sent to the backend logic of the application.The backend will determine whether the user is already following the group/venue or not. If the user is already following the venue, the backend will send a message to the database to remove the venue/group from the followed list. The opposite effect would be achieved otherwise. Once the database has performed the operation, the backend performs the correct modification and updates the followed view on the application GUI.

4.2.2 Listed Venue



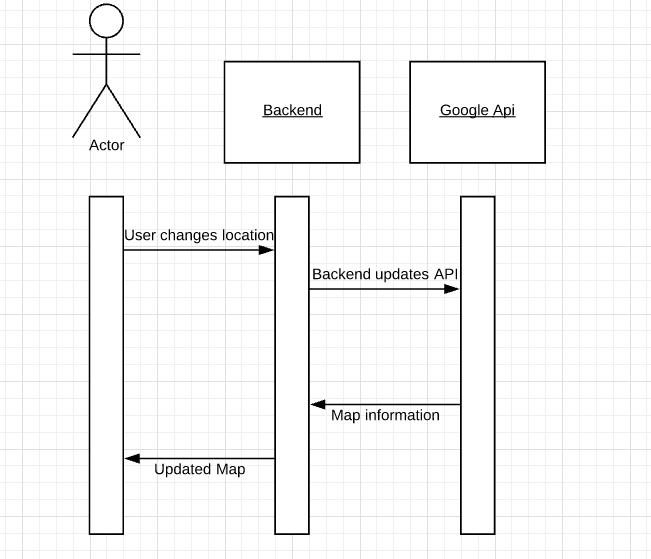
A venue manager is able to list a venue by pressing an add venue button on the application. The action is sent to the application backend which processes the command. This can deal with event and account verification. Once details are verified, the event is stored in the database. After that, the response is set to the backend which updates logic information such as maps. Finally, the backend updates all GUI elements in the application.

4.2.3 Edit event



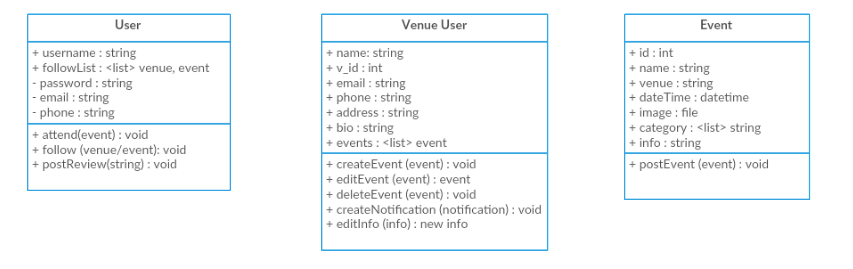
Only a manager account is able to edit event details. This sequence is triggered when a manager account presses the edit event button on the application interface. Once pressed, the GUI prompts the user to add information on which items to edit. Once the details are added, the backend processes the information and requests the database to modify the event details. The database modifies the event details and then the backend processes whatever changes need to be changed in the logic of the application. Finally, the application GUI is notified which allows the manager to see the updated event.

4.2.4 Location



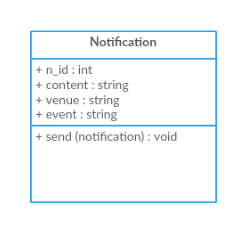
When a user moves visits another city or previews a city through the app functionality, the backend is notified. The backend sends the relevant information and calls the Google Api to update the map. The Google API responds with any relevant map information and the backend processes that information to show information such as events around the city based on locality. Once this data is calculated by the backend, it is displayed on the application’s map.

**4.3 Class Diagram**

4.3.1 Entity Classes:

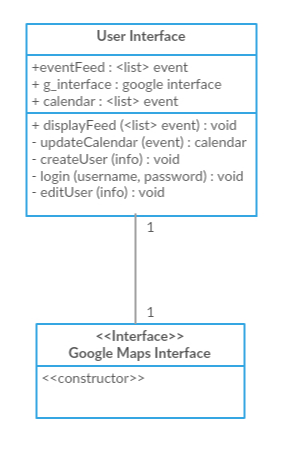
There are three total entity classes that will be used for Around Town. These classes are essential for the application to run correctly. The three classes include User, Venue User, and Event. Both the user and venue user classes will be used to manage the individual user and venue accounts, accordingly. The event class will be used for the actual events that will be posted.

4.3.2 Notification Class:



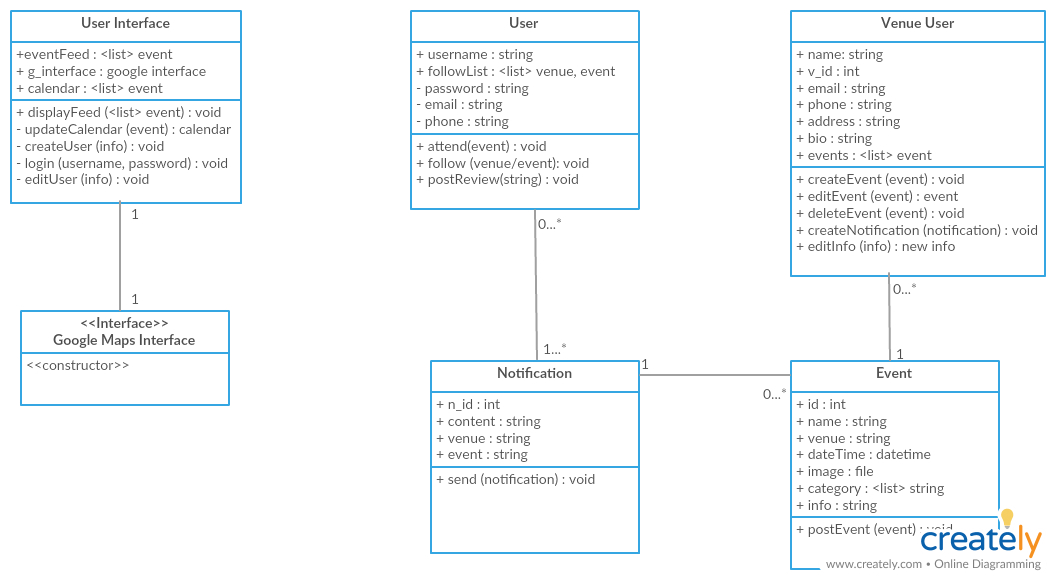
The notification class will be used to create notifications based on the venue users’ specifications. These notifications will be used to notify the user of event and venue information.

4.3.3 User Interface:



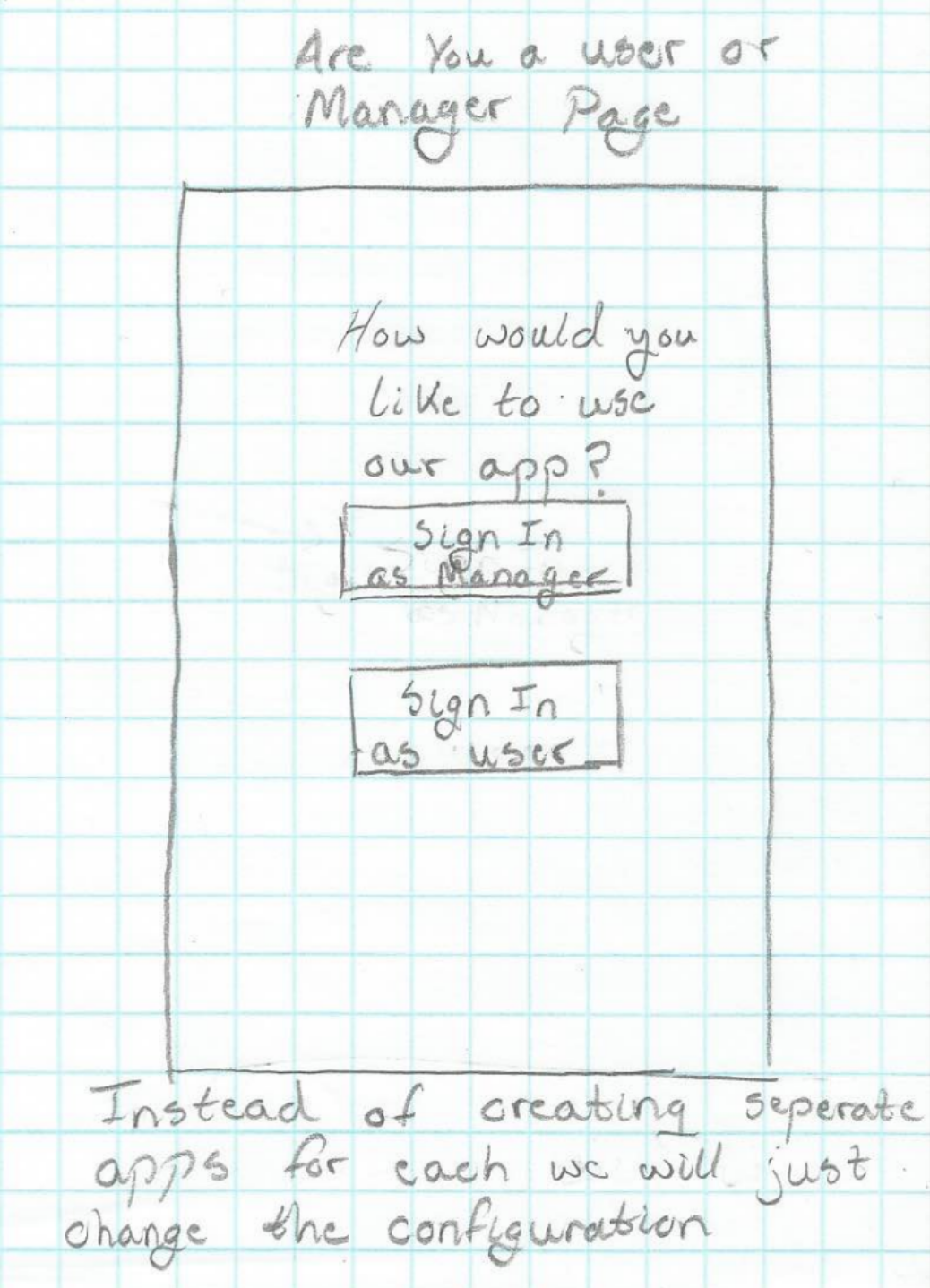
Here are the user interface classes that will be used for the interface that will be viewed by each user when using the application. The user interface will have methods to display a certain users feed and the Google Maps Interface will be an interface class referencing the user interface. This relationship is one-to-one.

4.3.4 Relationships:



Here is the relationship diagram for all classes described above. A Venue User will have 0…\* events that they will post about. Each event will correspond to one venue user and one notification. A User will then have 0…\* notifications that each correspond to a single event.

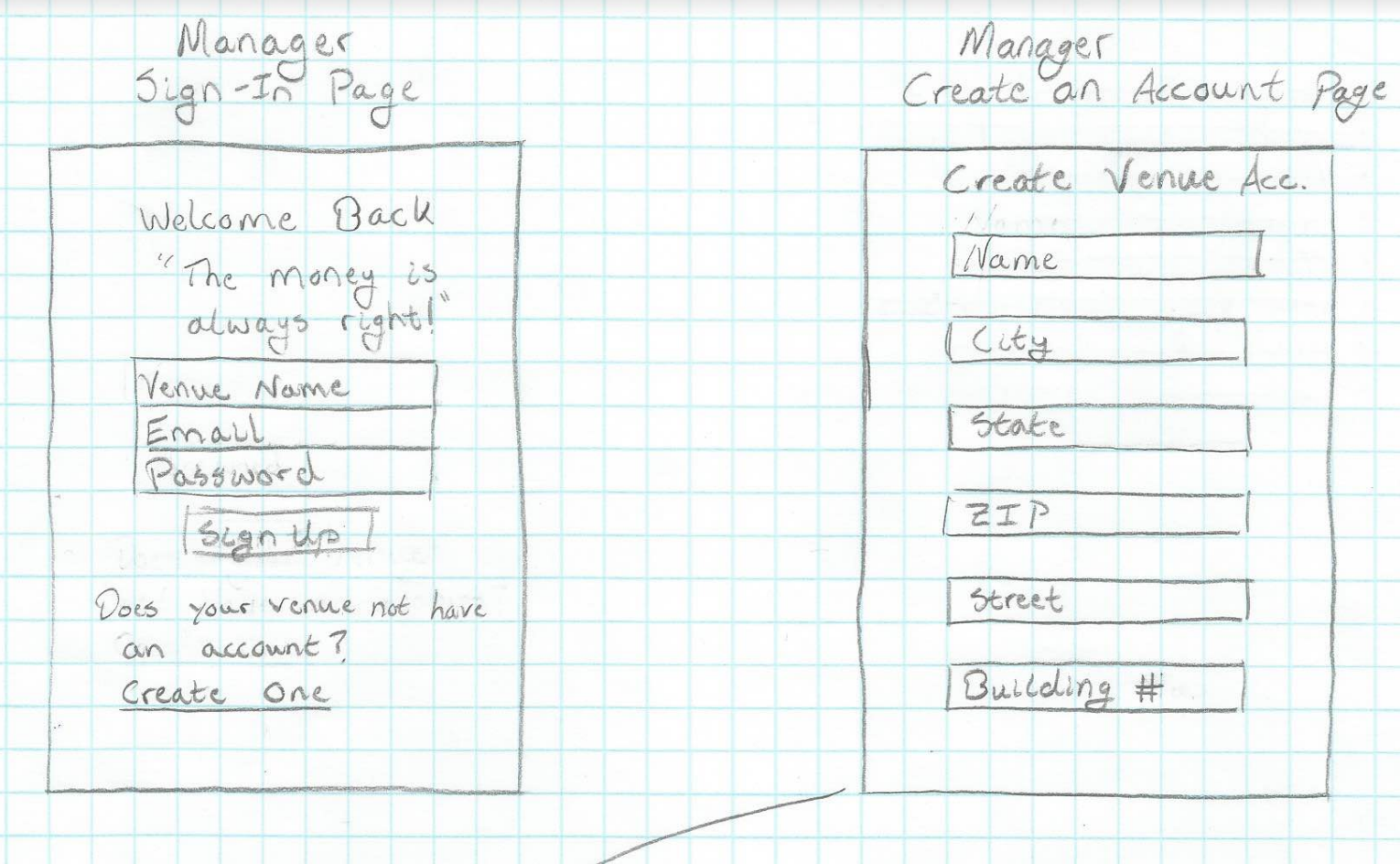
**5. Low Fidelity Prototypes**



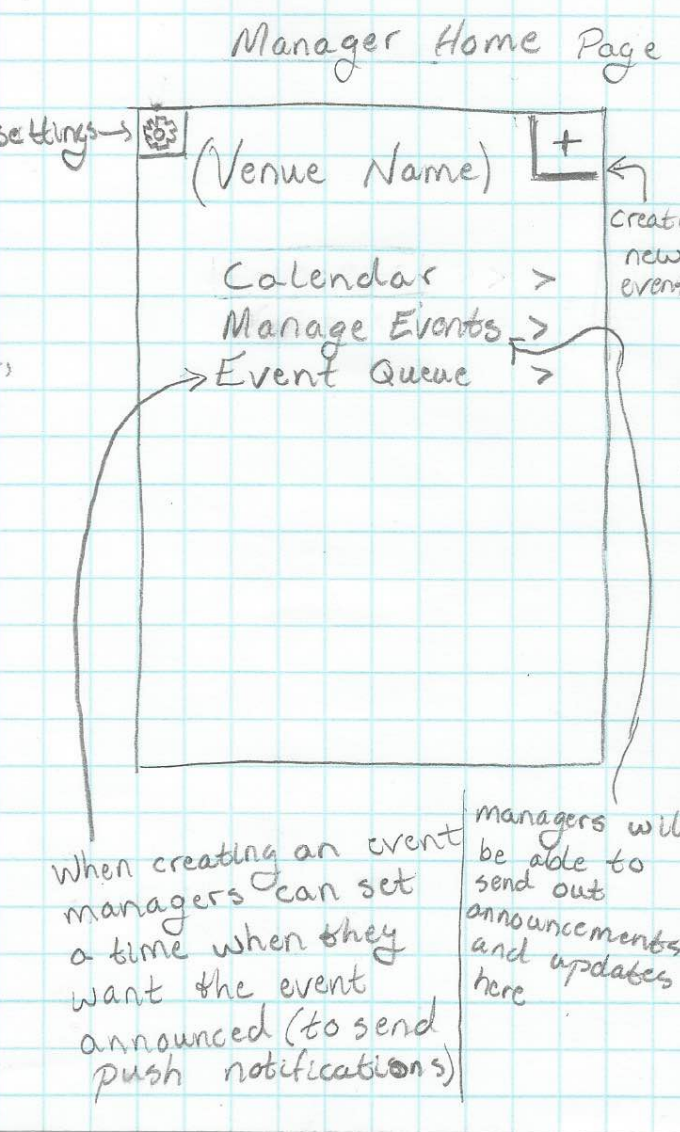
**5.1** Whoever wants to use the app will be asked if they are using it as a normal user or a venue manager.

# 

**5.2** If they choose user they will be directed to the user Sign in page and if they do not have an account they will need to make one and will be directed to the user create account page.



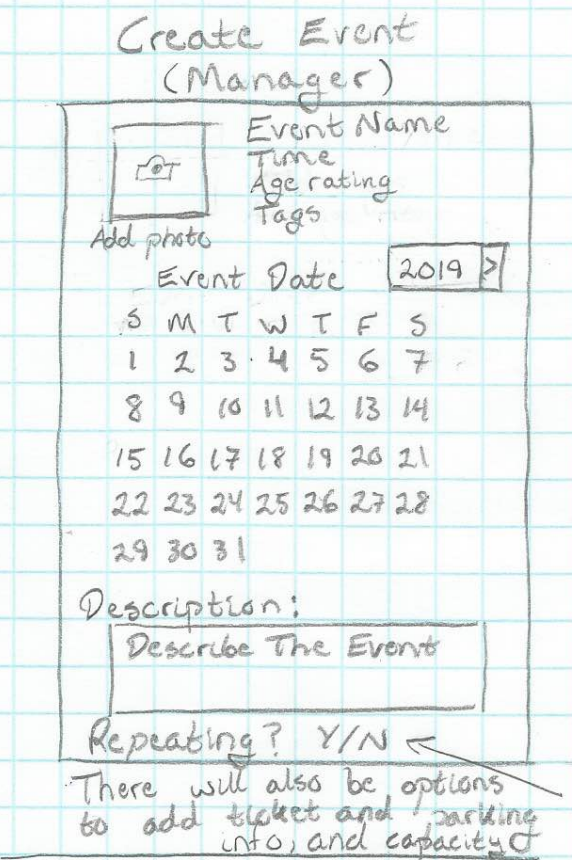
**5.3** If they choose manager they will be asked to sign in through the manager sign in and, again, if they do not have an account they will be asked it create one. However, they will not have access right away, since the establishment will need to be verified by us so that no scam venues are created. We are still work shopping the best way to verify the venues.



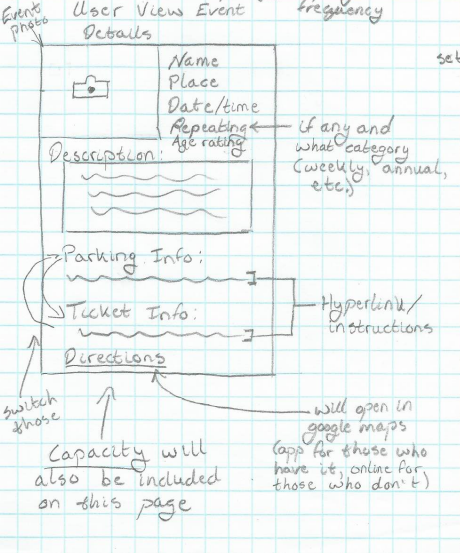


# 5.4 Here are the home pages for the user (left) and manager (right). We did not draw a map because it would have been hard to look at and know what we were trying to convey. We picture it as similar to the map used by google maps.

**5.5** When a user wants to view a venue’s main page with all of their events, it will look something like this.



**5.6** When a manager wants to create an event, this is the screen they will see. Repeating events will be those that are recurring and manager will have the option of making them weekly, monthly, annual, etc.



**5.7** When a user views an event, something like this is what they will see.

# 6. Implementation plan

## **6.1 Test bed**

The test bed we will be using to deploy our application is android studio. Android studio has several tools which allow for certain testing. Some examples that we may use is continuous integration as well as debugging tools provided in the Android SDK. Additionally, we will use a database for our user accounts. We plan on using MongoDB although this may change in the future depending on requirement changes.

## **6.2 Programming Languages**

One of the programming languages which we will be using is Java. This is because Android Studio uses Java. Additionally, we may use JavaScript to interact with our MongoDB database, however this may change as well depending on the interaction between MongoDB and Android Studio