Local Events

Oscar Adrian Campos-Soto

Developer

**Revision History**

|  |  |  |
| --- | --- | --- |
| **Last User to make Changes** | **Date** | **Summary of Changes** |
|  |  |  |
|  |  |  |

[This section is optional. Revision history may also be recorded as part of version control.]

Table of Contents

[Vision Statement 4](#_Toc459581887)

[Requirements 5](#_Toc459581888)

[Step #1: Identify Categories of Users 5](#_Toc459581889)

[Step #2: Create Actor-Goal List 5](#_Toc459581890)

[Step #3: Identify User Stories 6](#_Toc459581891)

[Product Backlog 7](#_Toc459581892)

[[Step #4: Write Use Cases] 8](#_Toc459581893)

[Sprint #1 12](#_Toc459581894)

[Review 12](#_Toc459581895)

[Retrospective 13](#_Toc459581896)

[Design 14](#_Toc459581897)

[Coding Standards 15](#_Toc459581898)

# **Vision Statement**

For Android users that would like to find local events near them. With Local Events they will be able to create, edit, and view local public and private events. Along with being able to maintain those events. Unlike other apps users will have an easier tool to access local events.

Requirements:

## Categories of Users:

**Event-Goer** – Anyone looking or going to an event.

**Event Creators/Admin** – The host of an event or event manager.

## Actor-Goal List:

|  |  |
| --- | --- |
| **Actor** | **Goal** |
| Event-Goer | Find Events in area |
|  | Browse list of available events |
|  | View events information |
|  | RSVP event if needed |
|  | Get Notifications when event is starting |
| Event Creator/Admin | Create an event |
|  | Create events settings |
|  | Edit event |
|  | Edit Settings |

## Identify User Stories:

* As an Event Creator, I want to be able to create an event that people can attend.
  + Estimated Effort: 10 Story Points.
* As an Event-Goer I want to see local events that I can attend.
  + Estimated Effort: 4 Story points.
* As an Event-Admin I want to be able to edit the event I am hosting so that event-goers are up to date on the event.
  + Estimated Effort: 8 Story points.
* As an Event Creator, I want to be able to make private events, so I can invite only certain people.
  + Estimated Effort: 4 Story points.
* As an Event Creator, I want to be able to make public events, so anyone can attend.
  + Estimated Effort: 4 Story points.
* As an Event-Goer I want to be able to request an invitation to an event.
  + Estimated Effort: 8 Story points.
* As an Event-Admin I want to be able to accept event invite request.
  + Estimated Effort: 8 Story Points.

### **Product Backlog**

The collection of stories makes up your project’s product backlog:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Story ID** | **Story** | **Story Points** | **Priority** | **Status** |
| S1 | Allow Event Creator to create an event. | 10 | 1 | Complete  3/18/18 |
| S2 | Allow Event-Goer and creator to view available events. | 4 | 2 | Complete 2/18/18 |
| S3 | Allow Event-Admin to edit the event settings | 4 | 3 | Complete  4/8/2018 |
| S4 | Allow Event Creator to set privacy of event’s. | 8 | 8 |  |
| S5 | Allow Event-Goer to request an invitation to an event. | 8 | 9 |  |
| S6 | Allow Event Admin to accept event invite request. | 8 | 10 |  |
| S7 | App Logo, colors, and Titles | 4 | 7 |  |
| S8 | Database storage of user data | 4 | 5 |  |
| S9 | user verification throughout app | 4 | 6 |  |
| S10 | Change to Firebase for login and Database | 8 | 4 | Completed  4/8/2018 |

# **Sprint #1**

Sprint 1 Backlog

|  |  |  |  |
| --- | --- | --- | --- |
| **Story ID** | **Story / Task** | **Estimated**  **Hours** | **Actual**  **Hours** |
|  | Create Database | 2 |  |
|  | Create user login page. | 4 | 1 |
| S1 | Design main UI and creator UI | 2 | 3 |
|  | Code main UI to creator UI | 3 | 2 |
|  | Test sending to Database | 3 |  |
|  | Code data input to database | 4 |  |
| S2 | Display events in main UI | 4 | 4 |

# **Sprint #2**

Sprint 2 Backlog

|  |  |  |  |
| --- | --- | --- | --- |
| **Story ID** | **Story / Task** | **Estimated**  **Hours** | **Actual**  **Hours** |
| S1 | Create Database | 2 | 4 |
| S1 | Test sending to Database | 1 | 4 |
| S1 | Code data input to database | 4 | 4 |
| S1 | Code user login and storage of login to database | 3 |  |

# **Sprint #3**

Sprint 3 Backlog

|  |  |  |  |
| --- | --- | --- | --- |
| **Story ID** | **Story / Task** | **Estimated**  **Hours** | **Actual**  **Hours** |
| S1 | Code user login and storage of login to database | 8 | 9 |
| S3 | Set up edit protocols | 4 | 3 |

# **Sprint #4**

Sprint 4 Backlog

|  |  |  |  |
| --- | --- | --- | --- |
| **Story ID** | **Story / Task** | **Estimated**  **Hours** | **Actual**  **Hours** |
| S3 | Code edit protocols | 4 | 4 |
|  | Set up API updates in the code | 4 | 2 |
|  | Set up API connection to database | 4 | 2 |
| S10 | Change to Firebase for login and Database | 8 | 8 |

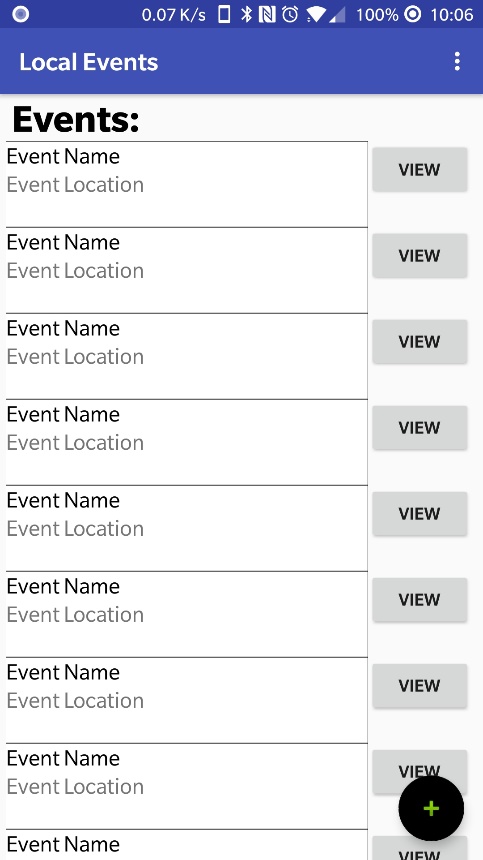
# **Sprint #5**

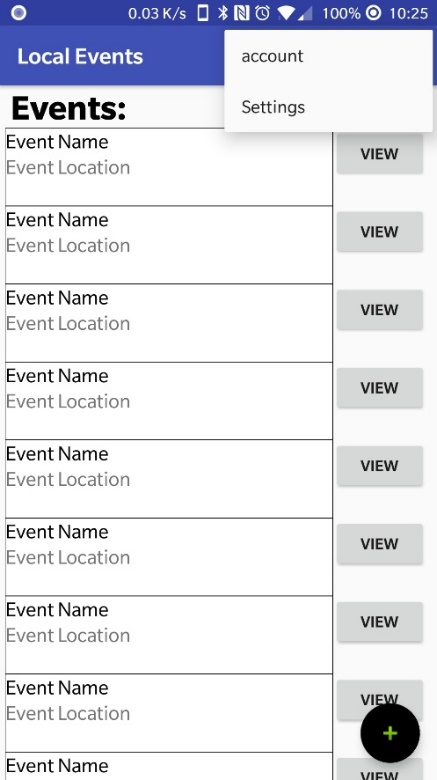
Sprint 5 Backlog

|  |  |  |  |
| --- | --- | --- | --- |
| **Story ID** | **Story / Task** | **Estimated**  **Hours** | **Actual**  **Hours** |
| S7 | App Logo, colors, and Titles | 4 |  |
| S8 | Database storage of user data | 4 |  |

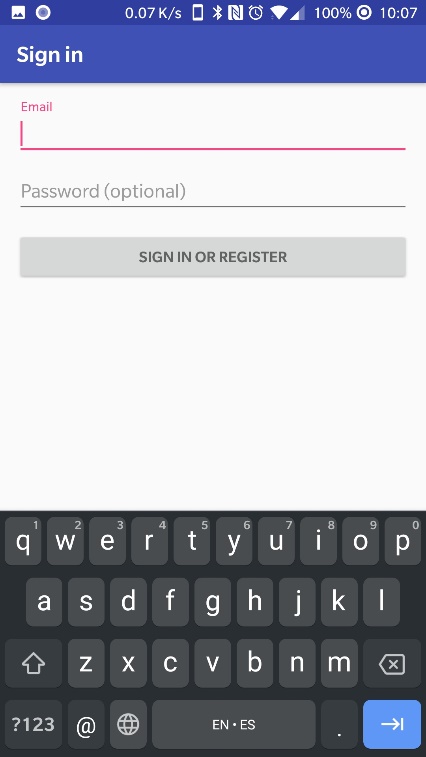
## Review

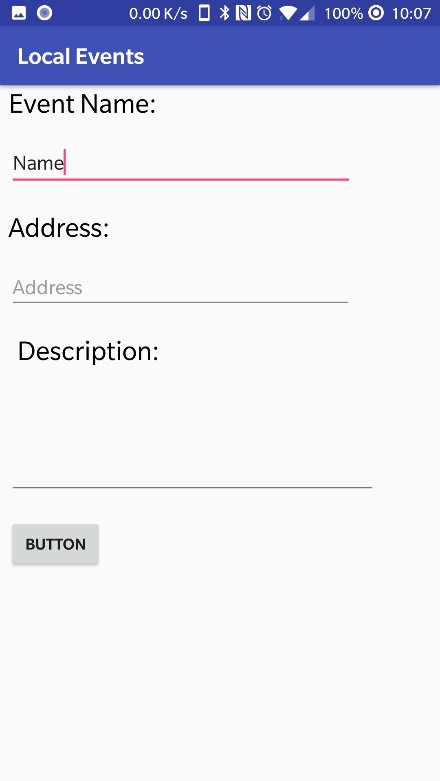
Iteration #1:

Example of the implementation of event-goer and event creator view of available events. Upon clicking the black and green plus sign the user will be redirected to the event creation activity.

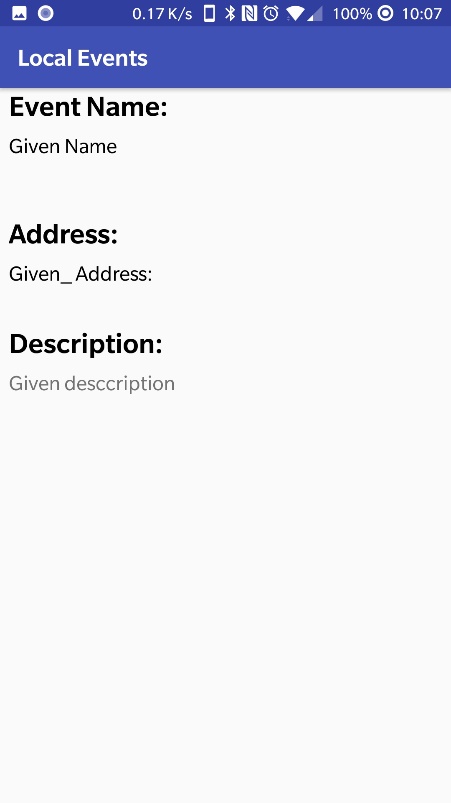
Example of menu options. Where the user can click on account to login.

The login page.

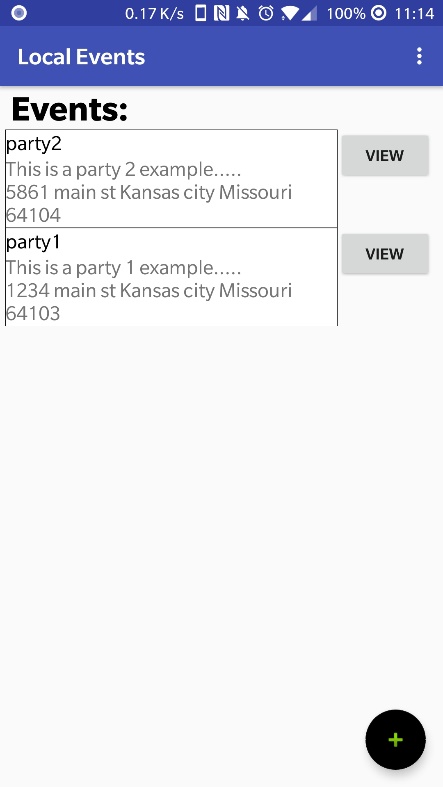




Example of the event creator page.

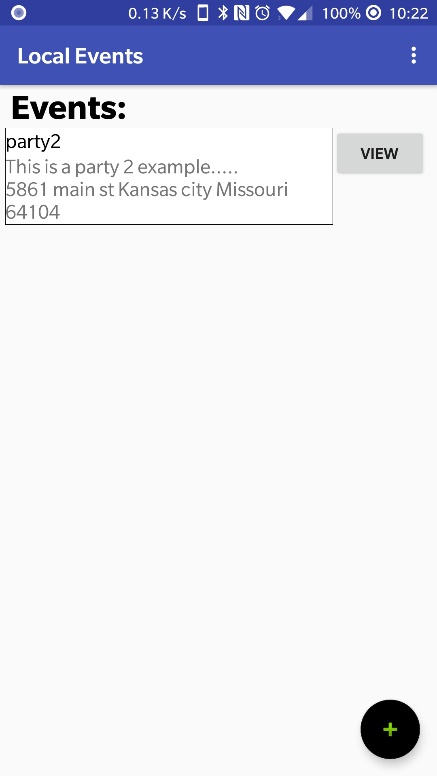
The view page to view the event.

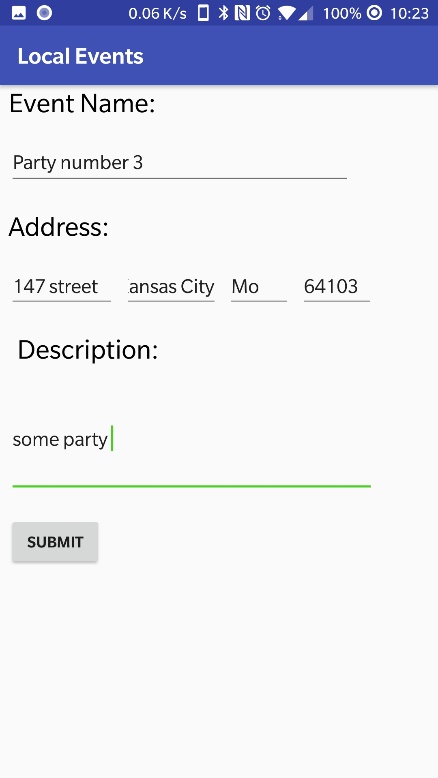
Iteration #2:

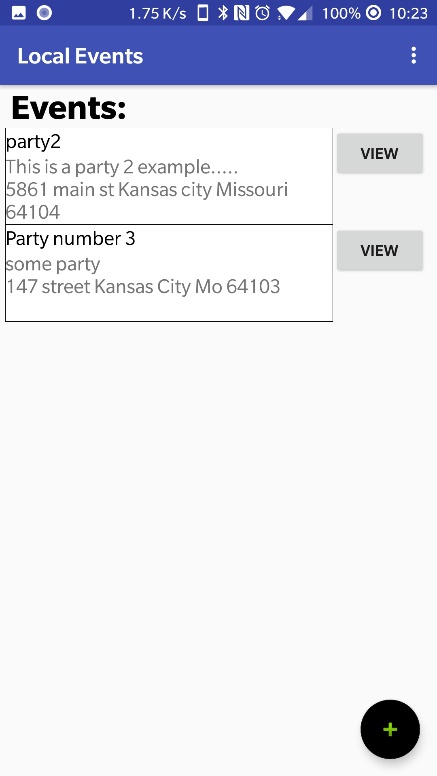


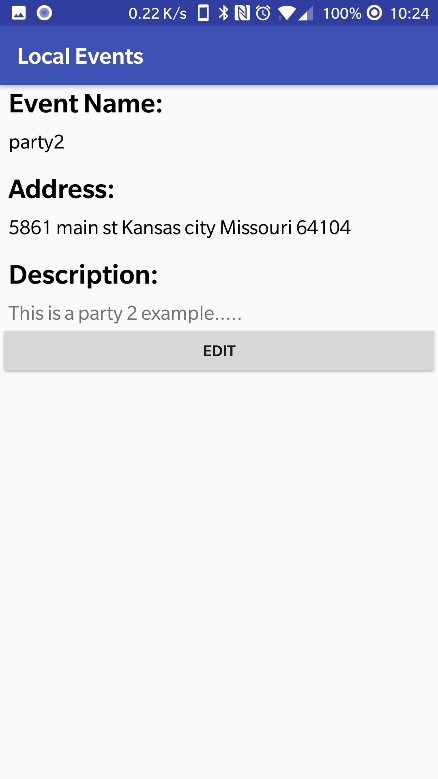
New main activity view. After the page retrieves the data from the database.

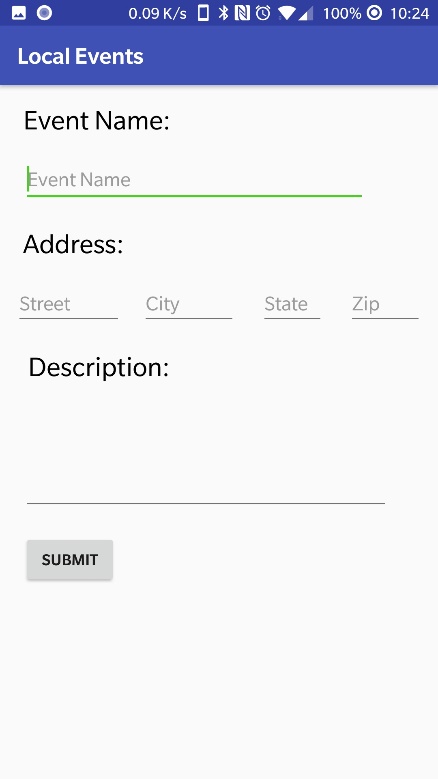
Iteration #3

Main Activity is still the same.

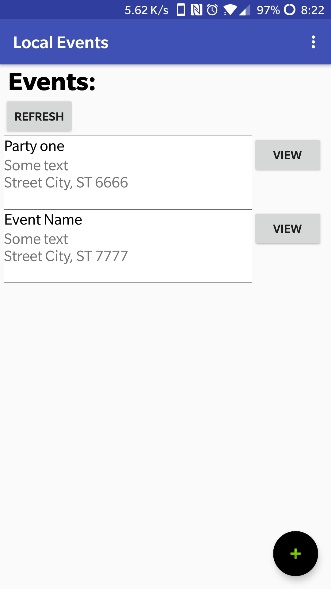
After clicking on the + sign it goes to this page where the user can create an event. As you can see here there is data inputted for the event. Once the user hits submit it will upload the event to the database.

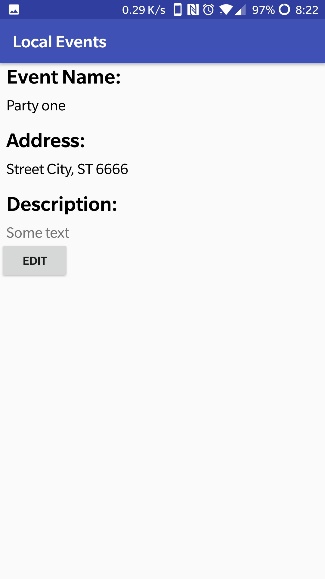
Here you can see how the event is now in the database and displayed in the event list.

Another implementation was the option to view the events by clicking the view button. Here the details are displayed and if the user is the creator/host they are able to edit the event.

 When the user clicks on the edit button they will be sent to this page. Although this is similar to the create event page, it will display the event details for editing in upcoming iterations.

Iteration #4

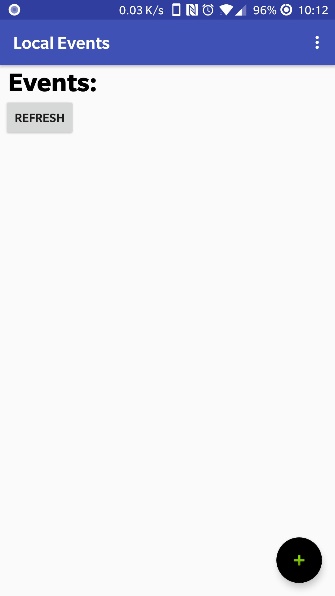
Main Activity is now using Firebase to store and retrieve the events.

Once the user clicks on view they get this screen where they can view more details about the event. The user gets the option to edit the event.

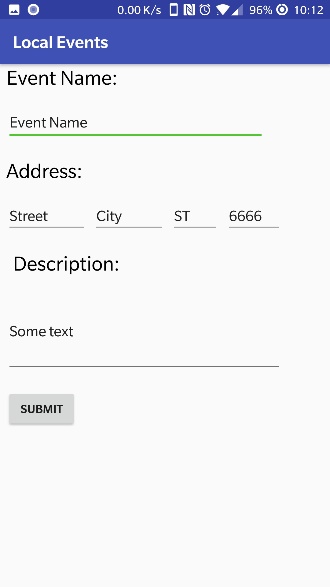
## 

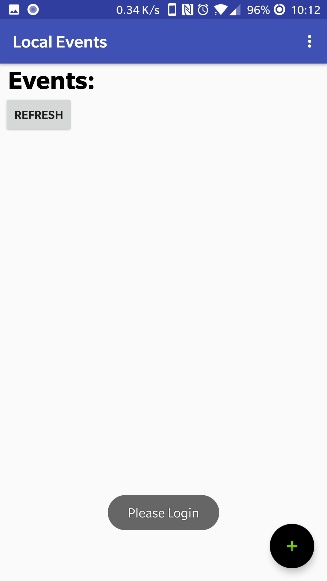
## When hits the edit button they get this. Where they can edit the name, address, and the description of the event.

## Here is the result of the edit above.



Here is an example when the user is not logged in. There are not events shown.

Since the create new option is still available I used an exception to determine if the user can even post to the database.

Since the user cannot post to the database while they aren’t logged in, they get a message asking them to login.

## Retrospective:

Iteration #1:

For this iteration, I planned to complete two stories but only part of one was complete. The reason for this was because of underestimating the first task. Creating a database is normally easy but getting the app to communicate with the database is a little more difficult than I anticipated. Therefore, I had to move on to creating the UI for the event goer and creator to view events. Which went well but I could not implement getting data from the database and displaying it. So, I moved on to creating the login page and the implementation of going from the main page to the login page and the create new event page. Finally, I was able to implement the view page, so the creator or event goer can see more details on the event.

Iteration #2:

For this iteration, I focused on making a working database, since I underestimated it in the last sprint. At first, I attempted to connect to a MySQL server but had no luck finding a good way to do it. So, I ended up just looking up how to make an API that I can connect to and get a JSON string for the app to receive. This took longer than I expected, and I had difficulty connecting to it on my local machine. I decided to just find a website that would allow me to host the server and I decided to use ZEIT’s NOW. It allows me to run the API and let it be accessible through the internet. Once I finished setting up the server I spent the rest of my time figuring out how to send a http GET request from android. Then I had to figure out how to send the request and update the activity data once the request was received. This was particularly difficult because the request had to be done in the background while the page was already displayed. Finally, I proceeded to take the data from the request and add a text view for each and a view button. Including the event information for each event.

Iteration #3

In this iteration, I focused on getting the database API ready for the login information and making sure the app can communicate with the API. Specifically, with HTTP POST and GET. I also created a new activity for the editing of the events and coded the connections between the view event and edit event activities.

Iteration #4

For this iteration I focused on getting the user to be able to edit an already created event. This was quite challenging since I had to update the API and figure out how to post to it from Android. Once I got it working I learned about Firebase and how much easier it was to Authenticate users and have a database in one easy place. During spring break, I had more time to play with Android Studio and decided to add Firebase to the backlog and as a high priority. So, I went ahead and implemented it in the app for getting database information, logging in and signing up users, and posting/updating data to the database. Once I found out about having an exception or assertion I decided to do some error checking with it. Particularly when a user is not logged in and the app needs to make sure of this. Since I have not implemented much error checking I decided to have an exception be thrown and handled with a message to the user asking them to log in.

# **Coding Standards**

Coding standards improve readability. They make it easier to understand code written by others. Good coding standards also improve reliability.