# LEBANON EVENTS
# Mobile Application

A Senior Project Submitted in Partial Fulfillment of the
Requirements for the Bachelor's Degree
in
Computer Science
Department of Computer Science

Faculty of Natural and Applied Sciences
Notre Dame University – Louaize
Zouk Mosbeh, Lebanon

Frederic Eid

Elie Moukhaiber

May 2017

# Abstract

This report describes an android application, designed to inform users about events of any kind, which are held near the user's location. The events are displayed on a map interface, which looks like the Google maps application. Upon clicking on an event marker, the events details will be shown. The application will allow the user to navigate to the event, request a taxi from inside the app, share the event, save it for future reference, reserve a seat at the event, or buy a ticket.

Any user will be able to add an event of their own to the system, and specify whether reservations are allowed, or not. The user can also choose to create tickets for their events, and set a ticket price.

This application aims to widen the knowledge of individuals about existing events nearby, and to group different people with same interests.

**Keywords**: android application, map interface, events locator.

# Table of Contents

# Chapter 1: Business Vision, Glossary, and Risks

## 1.1 Introduction

This project aims to inform individuals of events near their location, which they can access easily via the mobile application. Events can be scaled to fit a single neighborhood, city, or even the whole country. Events can be of different types including, but not limited to, music concerts, academic events, conferences, food festivals, art exhibitions, nightlife events, personal events(birthdays, family gathering), etc... . This project can lead to a more interactive community and to narrow the gap between different cultures, while providing an advertising platform to all kind of events. Among the risks that we might encounter is a poor adoption rate that might reflect negatively on the users of our application, whom might not find a diversity of events to satisfy their tastes.

## 1.2 Definitions, Acronyms, and Abbreviations

### 1.2.1 Business Opportunity

This project provides individuals an easy yet interactive application that works as a gateway to events based on one’s location. A user may easily locate an event nearby using the visual map interface of the application, and access all events details, including the number of attendees, whether Food and alcohol are available, and the event description. A user may also reserve or buy tickets for any event through the application. This will facilitate the process of searching for, and sharing events among friends and family. Users will be encouraged to reserve or buy a ticket because the reservation and ticketing processes are simple, and centralized in one place inside the application. Furthermore, the application will allow the merging of different cultures and mindsets with common objectives, because events will be introduced to a wider audience. Businesses or individuals will have the ability to market their events through the application for added exposure.

An extension of this project may also allow businesses or individuals to pay a fee, so that their events are highlighted and given priority over other events. Commissions may also be collected on tickets bought through the application. Moreover, Private chat with events' organizers, discounted tickets, and discounted taxi fares, are kept exclusive for the paid version of the application.

### 1.2.2 Problem Statement

Table 1.1: Problem Statement

| | |
|---|---|
| The problem solved by the application | Lack of information about existing events. |
| affects | Teenagers and Adults |
| the impact of which is | Missing opportunity of self-development, socialization, and entertainment. |
| a successful solution would be | Grouping all events in one easy to access, central location. |

### 1.2.3 Product Position Statement

Table 1.2: Product Positioning

| | |
|---|---|
| For | Age Category 13-64 |
| Who | Are misinformed about local events |
| The android mobile application | Belongs to the Events and Tourism category |
| That | Locates events of all types in Lebanon, utilizing an easy to use interface. |
| Unlike | Lebtivity.com |
| Our product | Events are classified according to their location.<br>Our product is also a native android mobile application that can be easily installed and accessed on any compatible device. |

## 1.3 Senior Study Objectives

1. Acquire android development skills.
2. Develop a user-friendly interface.
3. Understand how an API (Application Program Interface) is used.
4. Using a Web Service.
5. Creating and accessing Databases through mobile applications.

## 1.4 Approach

Building this application required us to learn new skills, such as android mobile development, SQLite, and Java.

We had to manage our time wisely during the four months of the semester; the majority of our time was dedicated to learning these new skills. Once we felt that our knowledge became sufficient to complete the project, the designing and coding phase began.

## 1.5 Risks

### 1.5.1 Slow adoption rate

Risk Magnitude: 5/5

The adoption rate of the application will be a deciding factor in its success or failure. This application relies on a big user base to be useful and beneficial for all the users.

Impacts: failure of the project

Mitigation Strategy: the application should be heavily advertised to be exposed to as many potential users as we can. Additionally, the application may be linked to other similar services, such as lebtivity.com.

# Chapter 2: Background

## 2.1 User Summary and Competition

### 2.1.1 User Summary

Table 2.1: User Summary

| Name | Description | Responsibilities |
|---|---|---|
| **User** | **The user can access events in the system** | **- Access events description** |
| **Organizer** | **The organizer can create events in the system** | **- Create events** |
| **Admin** | **The admin can modify all components of the system** | **- Access events description**<br>**- Create events**<br>**- Modify existing events**<br>**- Add/remove users** |

### 2.1.2 Alternatives and Competition

Lebtivity.com is our main competitor. It has been around for a long time, and has its own big user base. Lebtivity.com is a web application that serves as a hub for local events.

But Lebanon Events has the upper hand in comparison to Lebtivity.

| | Lebanon Events | Lebtivity |
|---|---|---|
| Native mobile applications | ✓ | |
| Display nearby events | ✓ | |
| Access taxi | ✓ | |
| Buy tickets | ✓ | |

- Lebanon Events is an android Native Mobile application
- Lebanon Events displays events nearby a user's location
- A user can request a taxi from inside the application
- A user can buy tickets for events

All these features are not implemented in Lebtivity.

### 2.2 Ethics

- A user with malicious intents, may create fake events in order to sell imaginary tickets, or just for the fun of tricking people.
- All events must pass by a verification process to check their genuineness, before being displayed on the map. Users creating multiple fake events intentionally will be blocked from the system.
- Every user's credentials are kept private and accessible by the administrators only
- Relaying on Google accounts to sign in to our application since Google account provides a high level of security and encryption for every user

- Ensure adequate testing, debugging and review of software and related documents using android studio
- Identify and report significant issues about an event of social concern to all the clients
- Perform some black box testing techniques: Error guessing, cause effect graph, use case testing.

## 2.3 Impact on Society

Users will have the ability to explore all events of any kinds in their area.

The application will allow the grouping of users with similar interests, and the mixing of people coming from different background.

The application also helps:

- Social outreach (reach lots of people at the same time)
- Small enterprises to assert their presence on the tourism scene
- The authorities to know where events are held to ensure the safety of the people

## 2.4 Other Requirements

### 2.4.1 System Requirements

- Android OS 7.0 or later
- At least 1 GB of RAM

### 2.4.2 Performance Requirements

Once a user launches the application, events should be displayed on the screen in a matter of milliseconds. The Application should respond quickly when a user chooses an event to access its details, creates an event, reserves a place in an event, buys a ticket, or requests a Taxi.

# Chapter 3: Requirements Elicitation

## 3.1 Adopted Technique(s)

Because the project idea was of our own, brain storming was the main technique used to elicit the requirements. Additionally, and since “Lebtivity” is our main competitor, comparing our application features to Lebtivity’s features was another technique. This allowed us to add the missing parts to create a well-rounded event locator application.

## 3.2 Questions

Below is a list of questions established for a complete final project:

- What platform should the application support?
- To whom it must be addressed?
- What interface shall the application use?
- What features shall the application have?
- What would be the objective of the application?
- What kind of events shall the application include?
- Should the events include local and international events?
- Shall the users be authenticated? If yes, how?
- Should the user be able to filter events based on his/her preferences?
- Should the user be able to create events?
- Should the user be able to buy or reserve tickets for an event?
- Should the user be able to see the attendees’ information and personal profiles?
- Should the user be able to write reviews and rate the events?
- Should the application be linked to social media networks?

### 3.3 Stakeholders' General Statement

On the map interface, events shall be indicated. The user is allowed to filter the events based on his preferences. Upon clicking on the event marker, the user will be shown the event details.

Users willing to attend an event may indicate so, and choose to buy or reserve tickets using our application. Users may also add events of their own so that others can access them.

The application should have a friendly interface that displays a map for the user.

# Chapter 4: Enumerated Requirements and Increments

## 4.1 Functional Requirements

Table 4.1: List of Functional Requirements

| ID | Priority | Description |
|---|---|---|
| **REQ1** | **3** | **The application events can be filtered based on preferences** |
| **REQ2** | **4** | **The application shall notify the users of nearby events, if wanted** |
| **REQ3** | **3** | **Users will be able to search for a specific event** |
| **REQ4** | **2** | **Users will be able to save interesting events in order to access them later** |
| **REQ5** | **4** | **Users can reserve or buy tickets in-app** |
| **REQ6** | **3** | **Users can request taxi from and to the event in-app** |
| **REQ7** | **4** | **Users can access the events without printing their tickets, only showing it through their phones** |
| **REQ8** | **1** | **Organizers can create and share their events** |
| **REQ9** | **2** | **Organizers can see a list of attendees in-app** |
| **REQ10** | **1** | **The application should show to the user a map interface, on which events are displayed.** |
| **REQ11** | **1** | **The application should include events from different kind (e.g. Academic events, Entertainment events, Personal events…)** |
| **REQ12** | **1** | **The application should show event details when clicking on the event marker on the map interface.** |
| **REQ13** | **2** | **The application should provide a way to navigate the user to the event.** |

| | | |
|---|---|---|
| **REQ14** | **4** | **The application should provide a way to share events on social media networks.** |

## 4.2 Non-Functional Requirements

Table 4.2: List of Non-Functional Requirements

| ID | Priority | Description |
|---|---|---|
| **REQ15** | **1** | **The application shall have a user friendly interface** |
| **REQ16** | **4** | **The application shouldn't take more than 2 second to return a search result** |
| **REQ17** | **4** | **The application should be available 99.99% of the time** |
| **REQ18** | **1** | **The application should interact with the device GPS** |
| **REQ19** | **1** | **The application should initially support running on android 7.0 (Nougat)** |

## 4.3 Increments

### 4.3.1 Release Backlogs

Table 4.3: Core Features Release Backlog

| Requirement IDs | | | | | |
|---|---|---|---|---|---|
| **REQ8** | **REQ10** | **REQ11** | **REQ12** | **REQ15** | **REQ18** |
| **REQ19** | | | | | |

Table 4.4: Release Backlog 1

| Requirement IDs | | | | | |
|---|---|---|---|---|---|
| **REQ4** | **REQ9** | **REQ13** | | | |

Table 4.5: Release Backlog 2

| Requirement IDs | | | | | |
|---|---|---|---|---|---|
| **REQ1** | **REQ3** | **REQ6** | **REQ2** | **REQ5** | **REQ7** |
| **REQ14** | **REQ16** | | | | |

### 4.3.2 Sprints

Define sprints for the release backlog that has the core features.

Table 4.6: Core Features Release Backlog Sprints

| Requirement IDs | | | | | |
|---|---|---|---|---|---|
| **Sprint 1** | | **Sprint 2** | | **Sprint 3** | |
| **REQ10** | **REQ12** | **REQ8** | **REQ11** | | |
| **REQ18** | **REQ19** | **REQ15** | | | |

# Chapter 5: Functional Requirements Specification

## 5.1 Use Case Diagram

The figure below describes the use case diagram of the application.
Please note that the login process is handled by Google authentication.

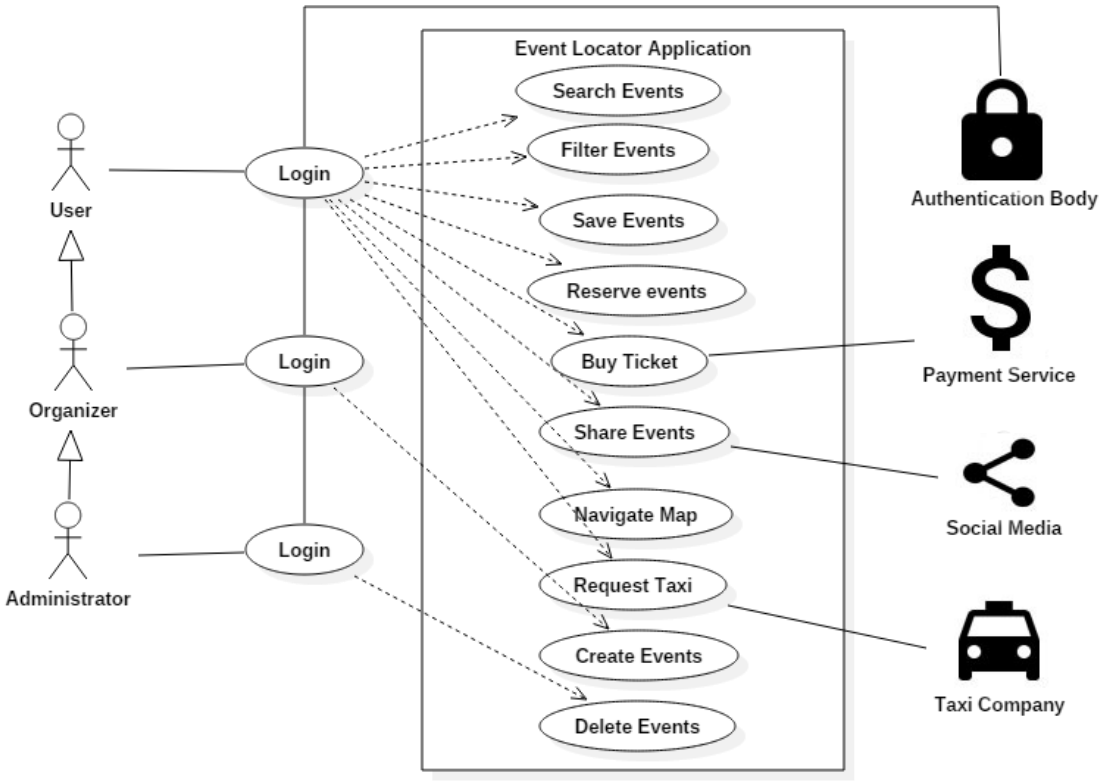



Figure 5.1: Event Locator Application Use Case Diagram

## 5.2 Use Case Specification for “Create Events”

### 5.2.1 Brief Description

The organizer and administrators using our application will be able to create events and add them to the map interface.

### 5.2.2 Actors

- Organizers
- Administrator

### 5.2.3 Basic Flow of Events

The organizers and administrators are able to create events.

1. They will enter the event location, title, description, type, date, event expiry date, and whether or not food and alcohol are available.
2. Then the application will display their event on the map interface.

1.1. If one of the fields is missing, or information is invalid (date specified has already past), the system will prompt the organizer and admin to reenter missing information, or adjust invalid ones.

### 5.2.4 Preconditions

- The organizer or administrator must be logged in to the application

### 5.2.5 Post conditions

- The organizer or administrator will be presented with the map interface, with their new event on it.

## 5.3 Use Case Specification for "Search Events"

### 5.3.1 Brief Description

The user of our application will be able to search events, based on its title, and description.

### 5.3.2 Actors

- User

### 5.3.3 Basic Flow of Events

The user will be able to search for events.

1. The users enter a title or description in the search bar.
2. Results will be displayed as a list of events matching the user requirements.
3. If no results are found, the application will notify the user that no match is found.
4. If the user clicks on a result, the event details page will be displayed to the user.

#### 5.3.4 Preconditions

- The user must be logged in to the application.

#### 5.3.5 Post conditions

- The user will presented with a list of search results.

### 5.4 Use Case Specification for "Filter Events"

#### 5.4.1 Brief Description

The user of our application will be able to filter events based on their type.

#### 5.4.2 Actors

- User

#### 5.4.3 Basic Flow of Events

1. The user will be able to select the types of events that he/she prefers.
2. Subsequently, the map interface will only show events of the types specified by the user.
3. If no events match the type specified, the map interface will show no events.

#### 5.4.4 Preconditions

- The user must be logged in to the system

#### 5.4.5 Post conditions

- The application will filter the events, and post the events with matching types on the map interface.

## 5.5 Use Case Specification for “Delete Events”

### 5.5.1 Brief Description

The administrator or organizer will be able to delete any event created.

### 5.5.2 Actors

- Organizers
- Administrator

### 5.5.3 Basic Flow of Events

1. Once the organizer selects an event that he/she created, he shall be able to delete this event.

2. Once an administrator selects any event, he/she shall be able to delete it.

### 5.5.4 Preconditions

- The organizer or administrator must be logged in to the application

### 5.5.5 Post conditions

- The deleted event will no longer be displayed on the map interface.

## 5.6 Use Case Specification for “Buy Tickets”

### 5.6.1 Brief Description

The user will be able to buy tickets for paid events, which allow the user to buy tickets through the application.

### 5.6.2 Actors

- User
- Payment Service

#### 5.6.3 Basic Flow of Events

The user will be able to buy tickets for the events.

1. When the user presses on the buy ticket button, he/she will be redirected to the payment service portal to complete the payment.
2. If the payment is successful, the application will notify the user that the payment was successful, and will show him a virtual ticket, which he/she can access through the application.
3. If the payment fails, the application will notify the user that the payment failed.

#### 5.6.4 Preconditions

- The user must be logged in to the application
- The payment service must be online

#### 5.6.5 Post conditions

- The user will obtain an in-app virtual ticket allowing him to access the event.

### 5.7 Use Case Specification for "Authentication"

#### 5.7.1 Brief Description

The user shall be authenticated before accessing the application.

#### 5.7.2 Actors

- User
- Authentication Body

#### 5.7.3 Basic Flow of Events

1. The user will have to sign in using his Google account before accessing the application.
2. If the user fail to login, he will be presented with the login screen.

#### 5.7.4 Preconditions

- The user should have a Google account

### 5.7.5 Post conditions

- The user will be presented with the map interface

## 5.8 Use Case Specification for “Save Events”

### 5.8.1 Brief Description

The user will be able to save events to access them later with ease.

### 5.8.2 Actors

- User

### 5.8.3 Basic Flow of Events

1. The user will be able to save events by clicking the save button.
2. Saved events will be added to the user favorites tab.

### 5.8.4 Preconditions

- The user must be logged in to the application
- The user must have chosen an event

### 5.8.5 Post conditions

- The user will be prompted that the event has been saved

## 5.9 Use Case Specification for “Reserve Event”

### 5.9.1 Brief Description

The user will be able to reserve a seat at an event within the application.

### 5.9.2 Actors

- User

### 5.9.3 Basic Flow of Events

1. The user will be able to reserve a seat at an event by clicking the reserve button.
2. The user will be prompted that a seat is reserved.

2.1. If no places are available, the user will not be allowed to click on the reserve button.

### 5.9.4 Preconditions

- The user should be logged in
- The user should have chosen an event

### 5.9.5 Post conditions

- The user will be prompted that his reservation is successful

## 5.10 Use Case Specification for “Map navigation”

### 5.10.1 Brief Description

The user of the application will be able to press a button that will open his default navigation app that will navigate him to the event location.

### 5.10.2 Actors

- user

### 5.10.3 Basic Flow of Events

1. The user by pressing the button will be switched to his default navigation app in order to get directions for a chosen event.
2. In case no default navigation app was found the operating system will prompt him to install a navigation app.

### 5.10.4 Preconditions

- The user must be logged in to the application

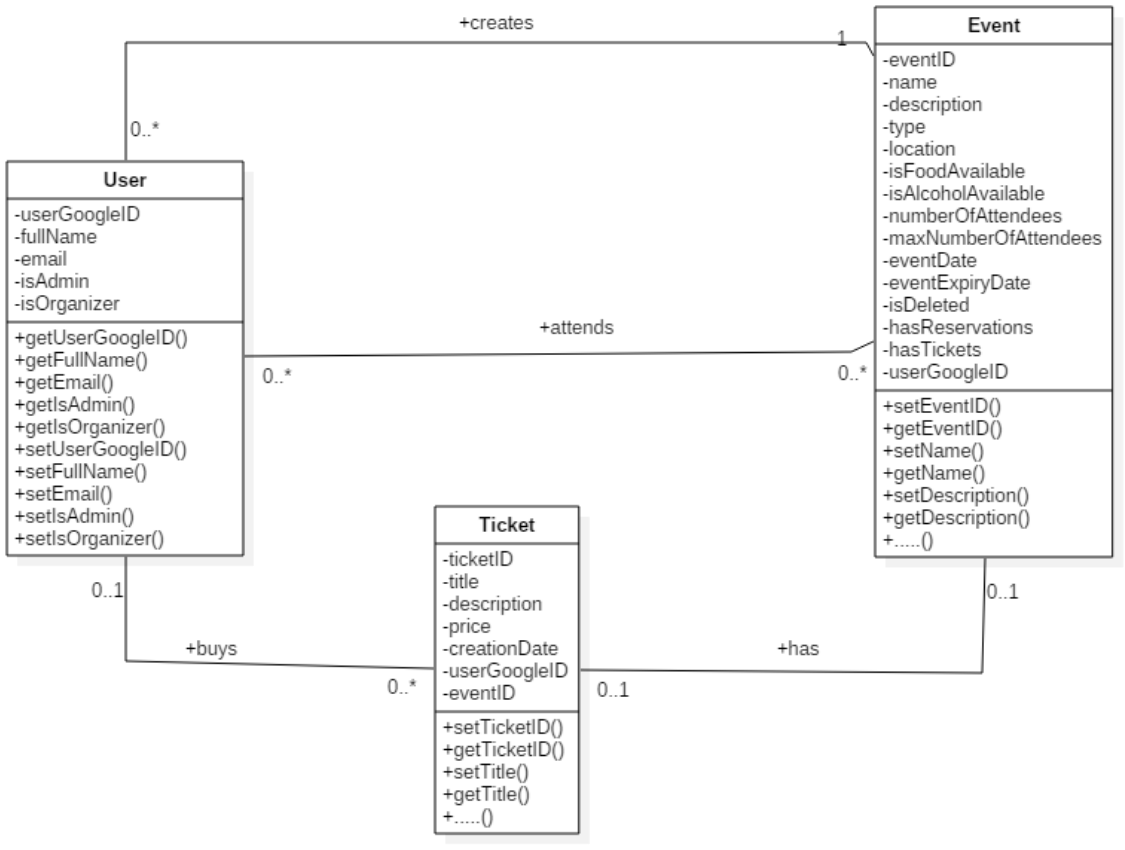
### 5.10.5 Post conditions

The user will access his default navigation app to get directions.

# Chapter 6: Domain Analysis

## 6.1 Domain Model Class Diagram

Below is the domain class diagram for our application representing the classes found with their respective attributes and functions.



### 6.2 Traceability Matrix

The table below shows the use case traceability matrix.

Table 6.1: Use Case Traceability Matrix

| **Domain concept/ Use case** | User | Admin | Organizer | Event |
|---|---|---|---|---|
| Login | ✓ | | | |
| Search | ✓ | | | |
| Create | | ✓ | ✓ | |
| Delete | ✓ | ✓ | ✓ | |
| Buy tickets | ✓ | | | ✓ |
| Authentication | ✓ | ✓ | ✓ | |
| Save | ✓ | | | ✓ |
| Reserve | ✓ | | | ✓ |
| Request taxi | ✓ | | | ✓ |
| Map navigation | ✓ | | | ✓ |
| Share events | ✓ | | | ✓ |
| Filter | ✓ | | | ✓ |

# Chapter 7: Interaction and Data Flow Diagrams

## 7.1 System Sequence Diagrams

Below is a list of several System Sequence Diagrams (SSD) that shows how each type of user interact with the application.

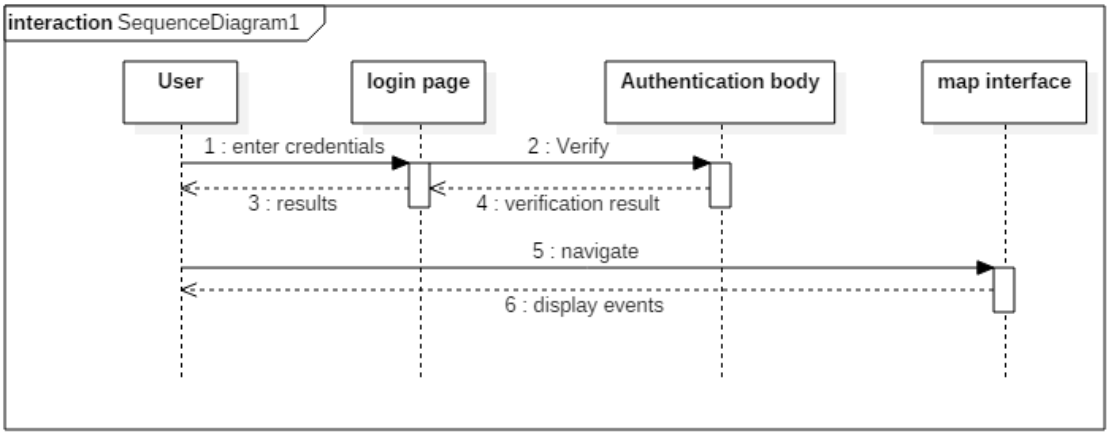



Figure 7.1.1: Basic Sequence Diagram for the Login page

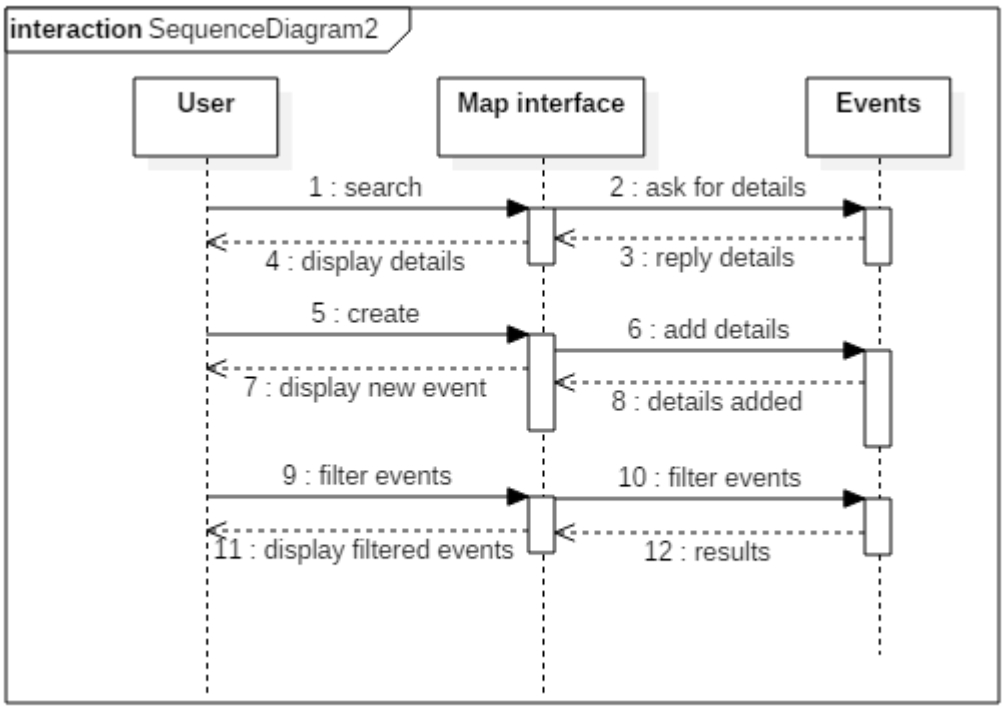



Figure 7.1.2: Basic Sequence Diagram for search create and filter

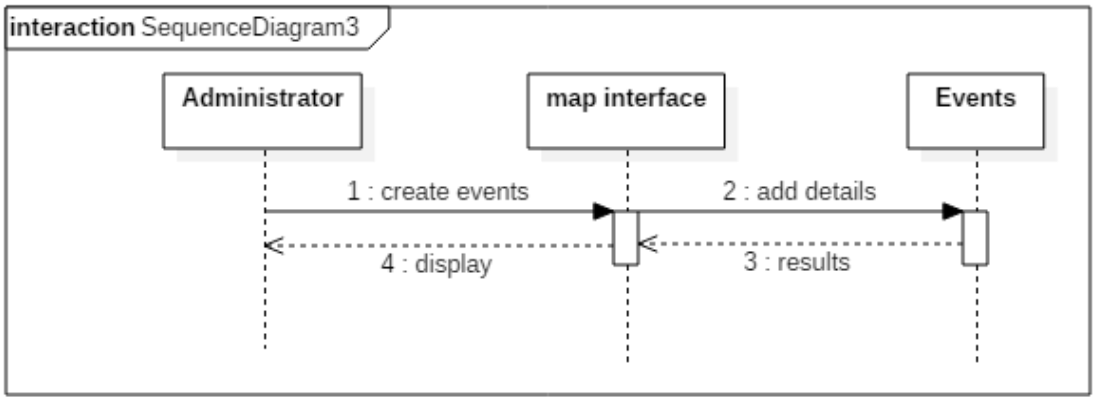



Figure 7.1.3: Basic Sequence Diagram for Admin creating events

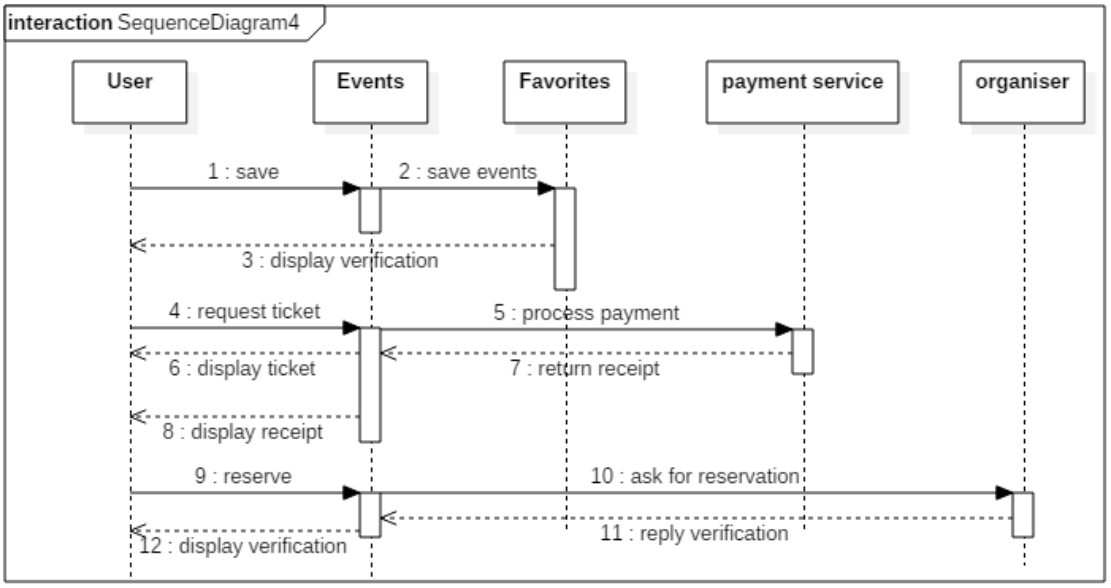



Figure 7.1.4: Basic Sequence Diagram for save events, reserve and buy tickets

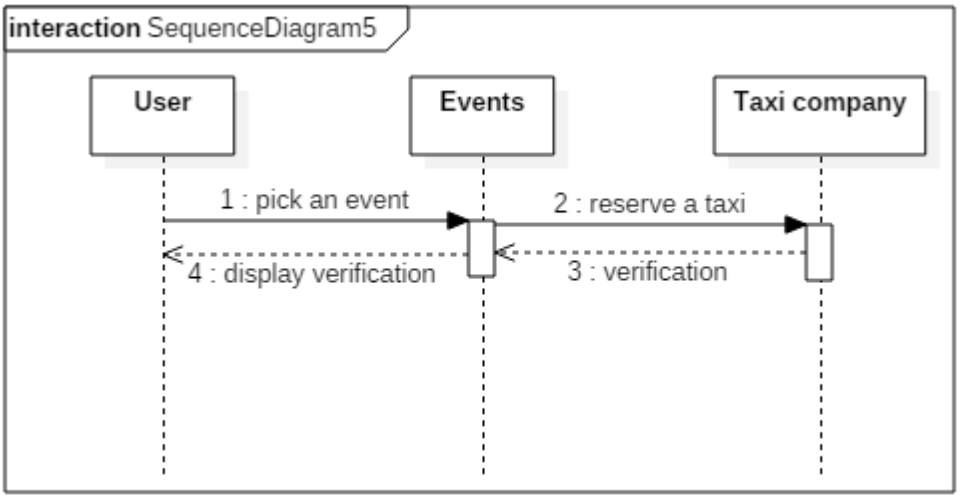



Figure 7.1.5: Basic Sequence Diagram for reserving a taxi for an event

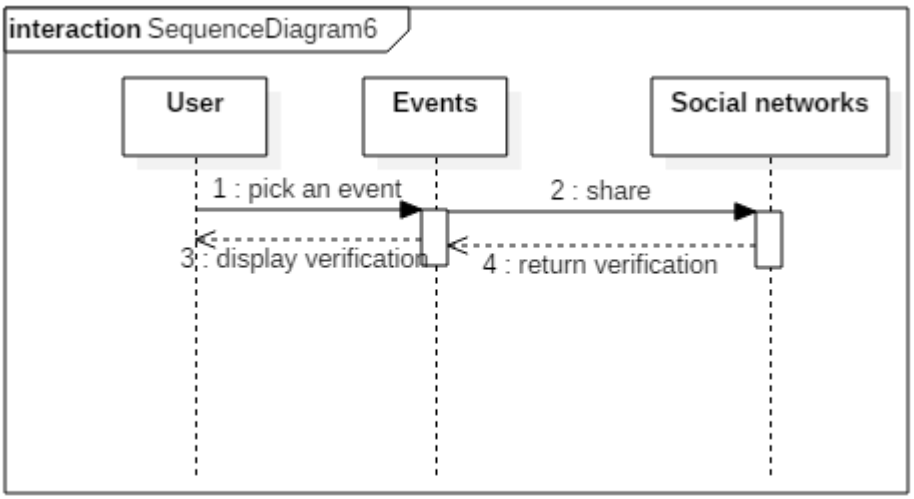
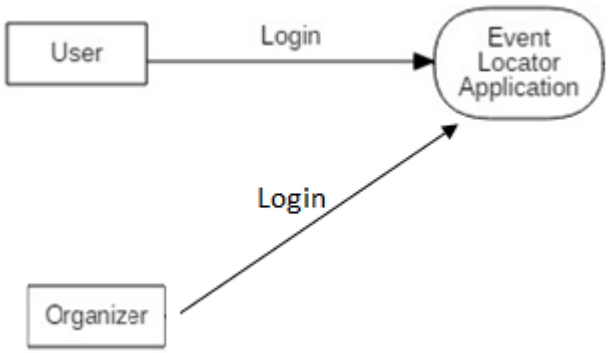



Figure 7.1.6: Basic Sequence Diagram for sharing an event on social network

## 7.2 Data Flow Diagram

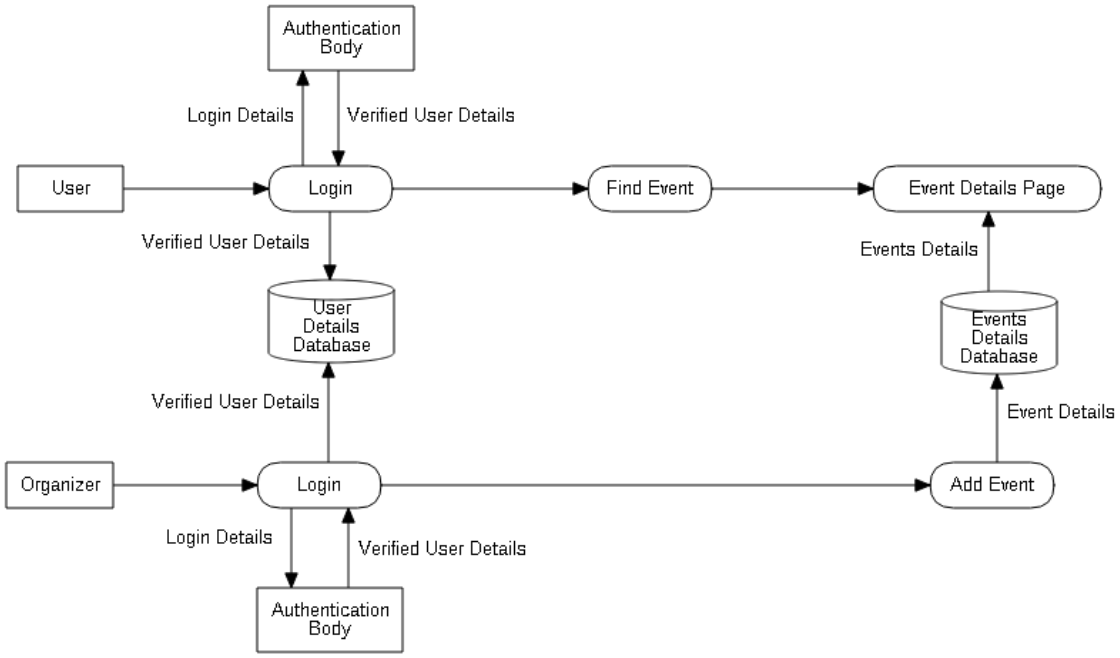
### 7.2.1 Context Diagram (Level 0)

Below is a figure that shows the level 0 of context diagram:




### 7.2.2 Data flow Diagram (Level 1)

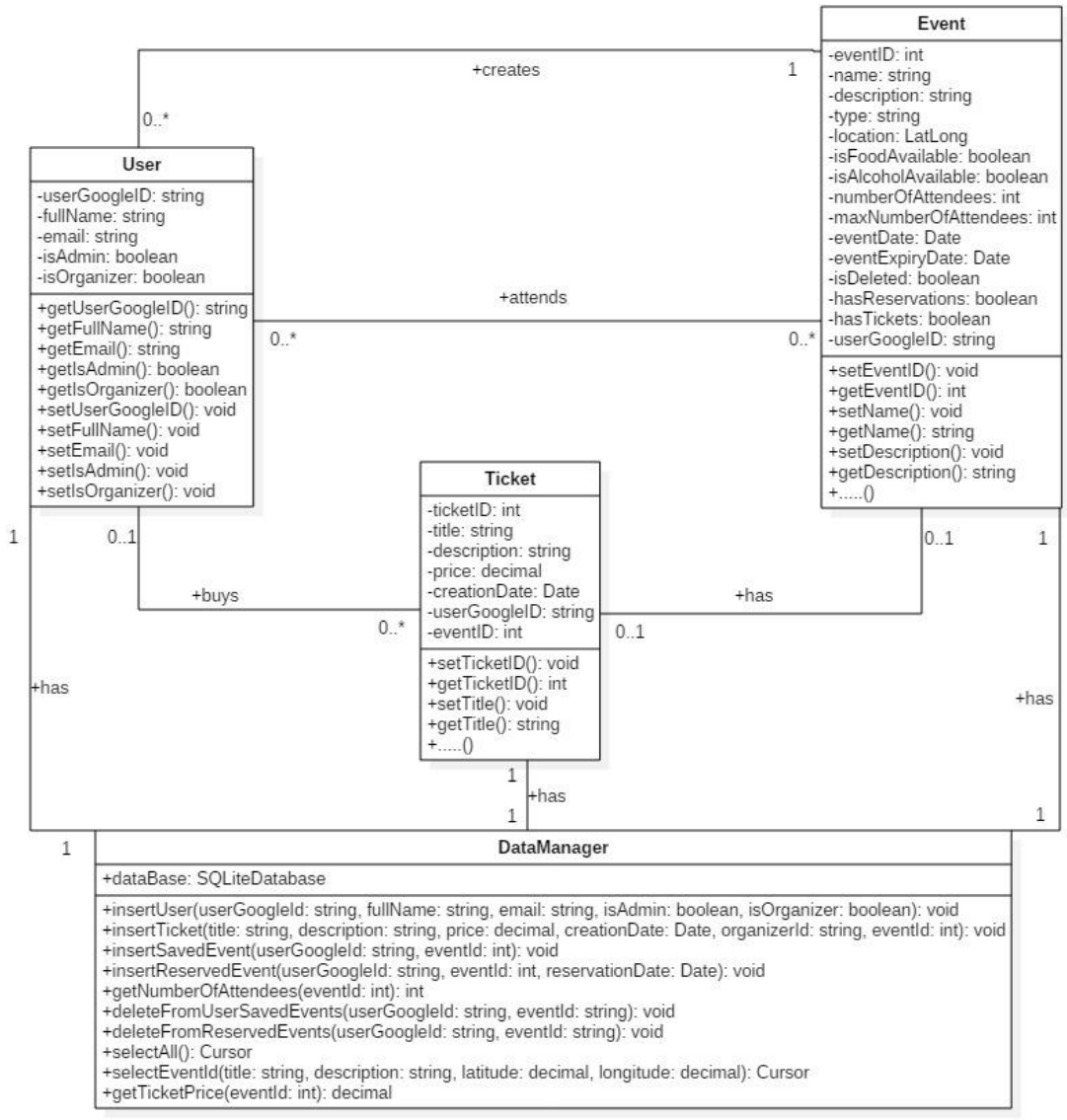
Below is a figure that shows the level 1 data flow diagram:



# Chapter 8: Software Architecture and Design

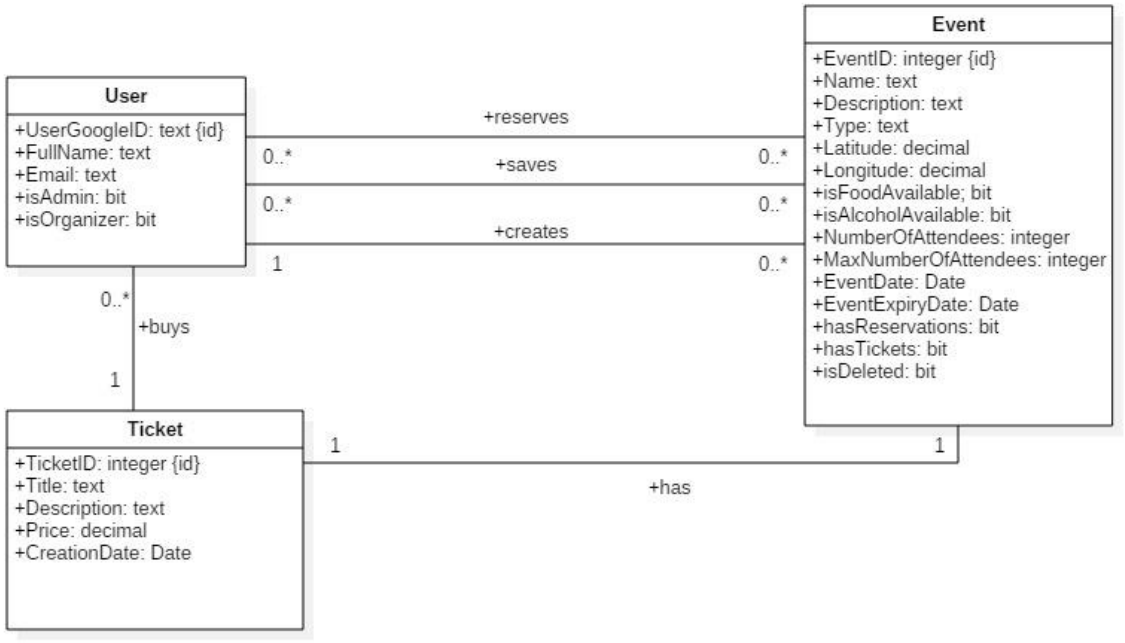
## 8.1 Design Class Diagram

Below is the Design Class Diagram that shows the application's classes with their attributes, data types and respective methods



## 8.2 Database Design Diagram

Below is the Database Diagram of our application that shows available tables in our Database.



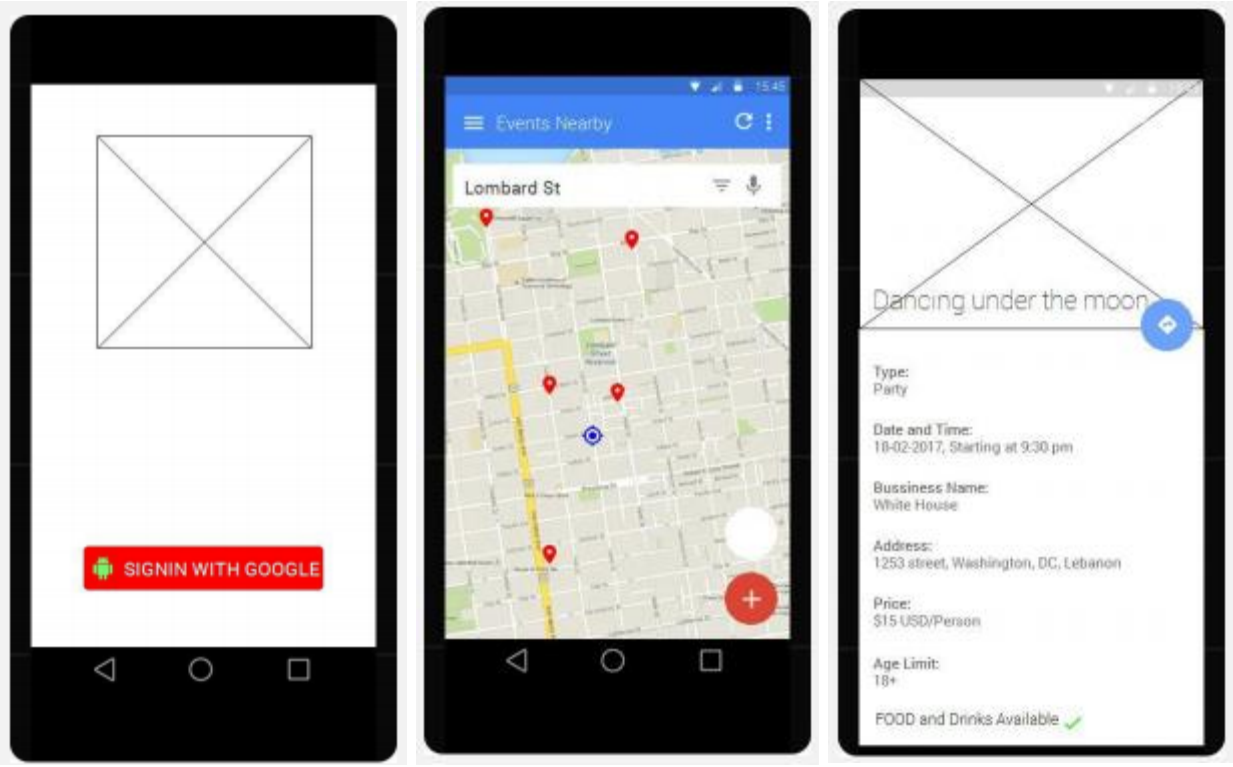
# Chapter 9: User Interface Design

## 9.1 Preliminary UI Sketches

The first screenshot shows the login screen that only allows signing in using a Google account.

The second screenshot shows the main map interface, which will be displayed after the user logs in successfully. The blue indicator shows the users current position, while the red markers indicates events' location on the map. The floating red button in the bottom right corner, will allow the user to add an event to the map.

The third screenshot shows the events details once the user click on a marker. It shows a picture of the event, its Title, type, date, business name or organizer name, address, price, age limit, and whether food and alcohol are available.



## 9.2 Final UI Design

The first image represents the sign in page to our application using Google accouts.

The second image represents the application's menu for available pages.

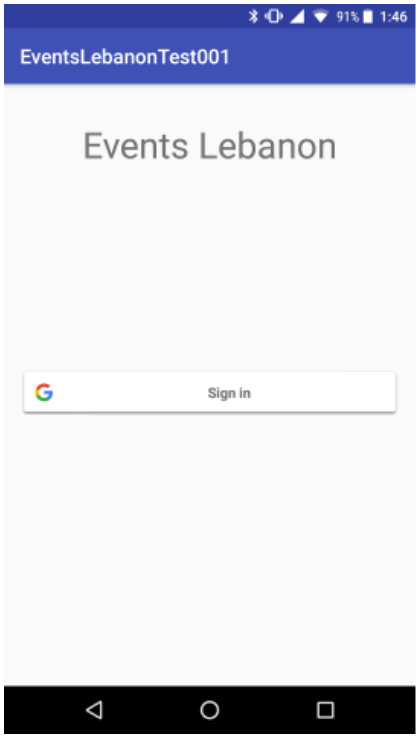



Figure 1

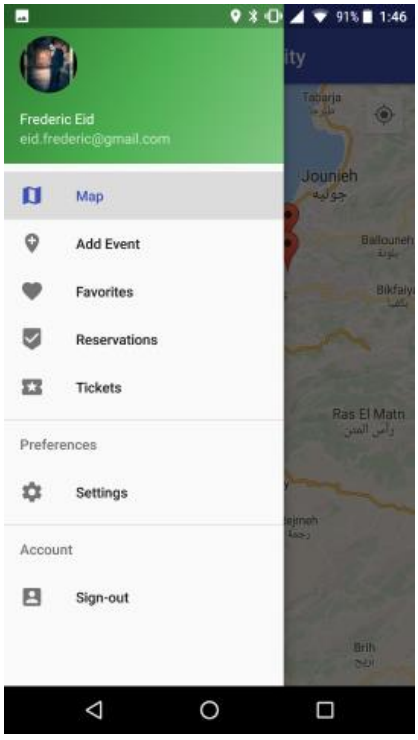



Figure 2

The third image represents the navigation map of the application

The fourth image represents the event name and type clicking on a certain event with addition to the navigation, share and request a taxi buttons

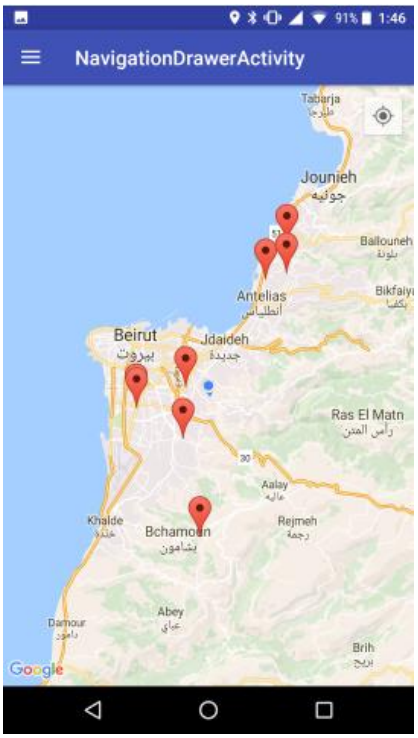



Figure 3

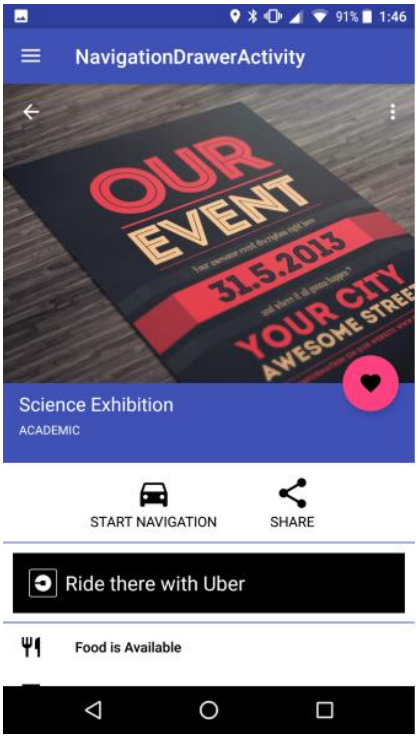



Figure 4

The fifth image represents the details of an event upon clicking on it and scrolling up.

The sixth image represents the add event page that shows the required fields to create an event

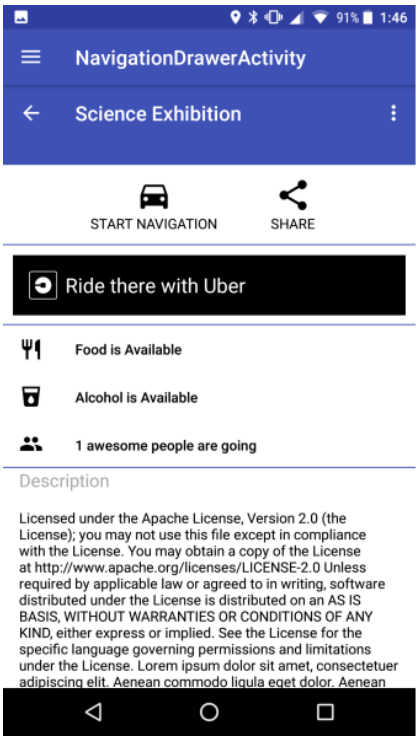



Figure 5

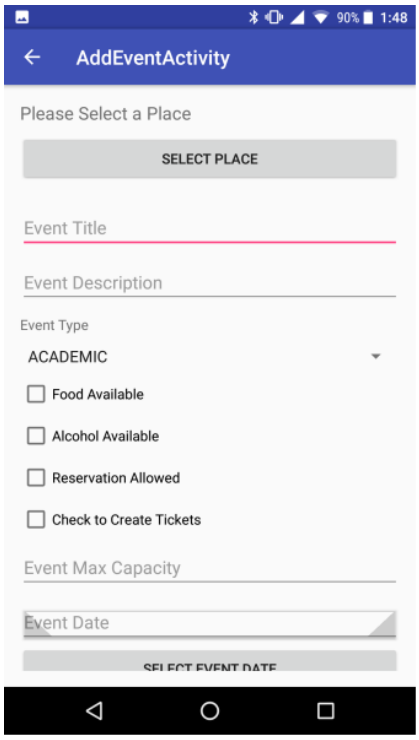



Figure 6

The seventh image represents the saved events page where a user can chose to keep or remove saved events.

The eighth image represents the reserved events page where a user can chose to keep or cancel any reservation.

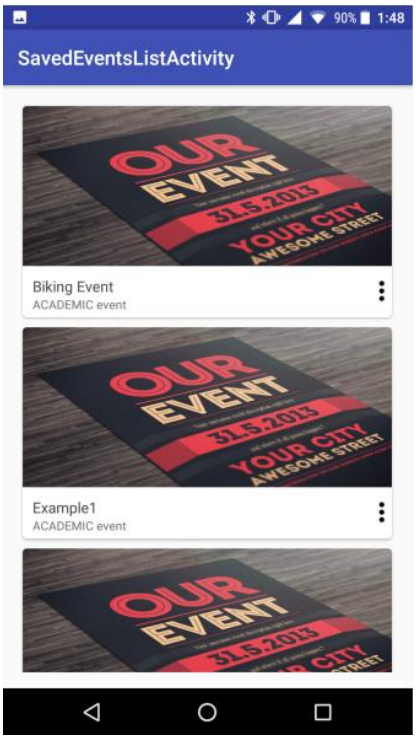



Figure 7

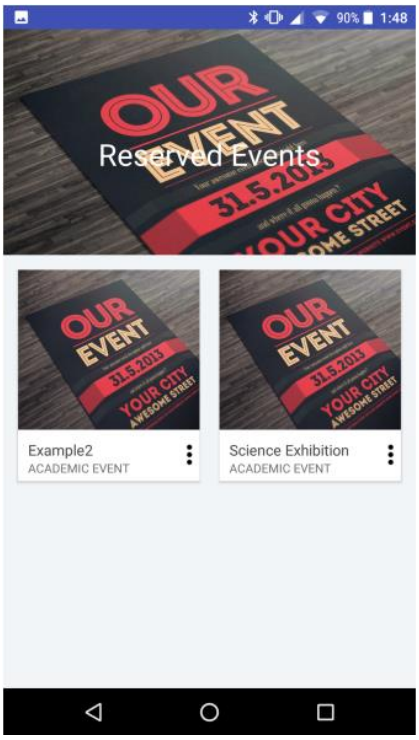



Figure 8

# Chapter 10: Coding

## 10.1 Code Samples

### 10.1.1 Saving Event to Database Code Sample

This algorithm collects all user input from the add event screen, and checks whether the same data has been collected before. If the data wasn't entered before, it is saved in the database, and the user is redirected to the map screen.

```
1.  if (!(dm.CheckIsDataAlreadyInDBorNot(txtEventTitle.getText().toString(), (float) placeL
    atLng.latitude, (float) placeLatLng.longitude)))
2.
3.                  dm.insert(txtEventTitle.getText().toString(), txtEventDescription.getTe
    xt().toString(),
4.                          spinnerEventType.getSelectedItem().toString(), (float) placeLat
    Lng.latitude, (float) placeLatLng.longitude,
5.                          checkBoxIsFoodAvailable.isChecked(), checkBoxIsAlcoholAvailable
    .isChecked(), 0,
6.                          false, Integer.parseInt(txtMaxNumberAttendees.getText().toStrin
    g()),
7.                          txtEventDate.getText().toString(), txtEventExpiryDate.getText()
    .toString(), false,
8.                          checkBoxReservationAllowed.isChecked(), checkBoxCreateTicket.is
    Checked(),
9.                          prefs.getString("personGoogleId", "112106928278476143898"));
10.
11.                 if(checkBoxCreateTicket.isChecked()){
12.
13.                     Cursor c = dm.selectCorrespondingEventId(txtEventTitle.getText().to
    String(), txtEventDescription.getText().toString(),
14.                             (float) placeLatLng.latitude, (float) placeLatLng.longitude
    );
15.
16.                     int eventId = -1;
17.                     while(c.moveToNext()){
18.                         eventId = c.getInt(0);
19.                         Log.i(TAG, "EventID : " + eventId);
20.                     }
21.
22.                     dm.insertTicket(txtEventTitle.getText().toString() + "Ticket", txtE
    ventDescription.getText().toString(),
23.                             Double.parseDouble(txtTicketPrice.getText().toString()),txt
    EventDate.toString(),
24.                             prefs.getString("personGoogleId", "112106928278476143898"),
    eventId );
25.                 }
26.
27.                 // Declare and initialize a new Intent object called myIntent
28.                 Intent myIntent = new Intent(this, NavigationDrawerActivity.class);
29.                 // Switch to the NaviagtionDraweActivity
30.                 startActivity(myIntent);
31.             } else {
32.                 new AlertDialog.Builder(this)
33.                         .setTitle("Duplicate Entry")
34.                         .setMessage("This Event already exist. Please Create a new Even
    t.")
35.                         .setCancelable(true)
36.                         .setPositiveButton("OK", null)
37.                         .show();
38.             }
```

### 10.1.2 Displaying Markers on the map Code Sample

Initially, the algorithm tries to get the user's current location. If successful, the map will zoom in on their current position.

Then, the algorithm will go through the List of events containing all the events in the database, and for each one will place a corresponding marker on the map.

```
1.  googleMap.setOnMarkerClickListener(this);
2.
3.
4.          LocationManager locationManager = (LocationManager) getActivity().getSystemServ
    ice(Context.LOCATION_SERVICE);
5.          Criteria criteria = new Criteria();
6.
7.          Location location = locationManager.getLastKnownLocation(locationManager.getBes
    tProvider(criteria, false));
8.          if (location != null)
9.          {
10.             googleMap.animateCamera(CameraUpdateFactory.newLatLngZoom(new LatLng(locati
    on.getLatitude(), location.getLongitude()), 11));
11.
12.             CameraPosition cameraPosition = new CameraPosition.Builder()
13.                     .target(new LatLng(location.getLatitude(), location.getLongitude())
    )      // Sets the center of the map to location user
14.                     .zoom(11)                   // Sets the zoom
15.                     /*.bearing(90)                // Sets the orientation of the camera
    to east
16.                     .tilt(40)                   // Sets the tilt of the camera to 30 de
    grees*/
17.                     .build();                   // Creates a CameraPosition from the bu
    ilder
18.             googleMap.animateCamera(CameraUpdateFactory.newCameraPosition(cameraPositio
    n));
19.         }
20.
21.         googleMap.setMyLocationEnabled(true);
22.
23.
24.         for (Event event: listOfEventsDB
25.              ) {
26.             Marker myMarker = googleMap.addMarker(new MarkerOptions().title(event.getEv
    entName())
27.                     .position(event.getEventLocation()));
28.
29.             myMarker.setTag(event);
30.
31.             listOfMarker.add(myMarker);
32.         }
```

# Chapter 11: Collaboration and Individual Contribution

## 11.1 Collaboration Approach

In general, collaboration was made face to face; we met after class, at the library or at home. Our first meetings were dedicated to brain storming and researches to come up with an idea that is achievable giving the limited timeframe that we have. Later meetings became more fruitful, it's where the coding starts. The last few meetings were dedicated for testing, trials and finishing the report, and the project poster.

## 11.2 Problems that Occurred

At the beginning, we encountered a major technical problem; The language used for developing android applications was new to us. It took us quite some time to learn it to be able to start coding. We followed a tutorial book called “Android Programming for Beginners”, by “John Horton” that made a great difference in the progress of the project.

## 11.3 Individual Contribution Breakdown

Table 11.1: Individual Contribution

| Task | Frederic Eid | Elie Moukhaiber |
|---|---|---|
| **Brain storming** | **50%** | **50%** |
| **coding** | **70%** | **30%** |
| **report** | **30%** | **70%** |
| **Database** | **50%** | **50%** |

# Chapter 12: Conclusions and Recommendations

## 12.1 Summary of the Main Results

This senior project had a main aim of designing and implementing an android application. The idea of the application can be visualized by a map interface that can locate different kind of events in Lebanon. The software application was written using java, XML for designing the layout, SQLite for the database, and took four month to be completed. The first month was dedicated to come up with the idea and collect necessary information and requirement for the application. The following two months were required to learn java and be able to achieve a well-designed application. Then following month was dedicated to learn more about SQLite databases. Finally, the last month after acquiring enough information, and received enough knowledge, the coding to create the map interface followed by some features such as navigation, event sharing, and requesting a Taxi in-app.

## 12.2 Main Contributions

- First, this project required knowledge in many computer science fields such as database design, object oriented design, and software engineering. Faculty members provided a huge help for us in an effort to completing this project.
- Second, we had to learn android development in addition to "Java" programming language. This process took the most time, taking into consideration we started from scratch. It was a real challenge.
- Finally, in this project we learned how to deal with APIs (Application Program Interface), and how helpful it can be in pushing forward the development process.

## 12.3 Possible Extensions and Future Work

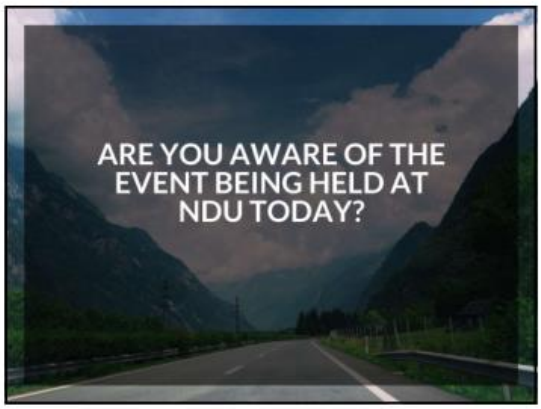
Designing an event locator application is a large project and could evolve in many different ways. The team was only able to implement basic functionality and features, due to time constraints. One of the main missing features, is the database over the network. Since implementing a remote database would require knowledge in PHP, that we lack, the remote database was replaced with a local one.

A possible feature that is planned to be added to the application, is the ability to verify an event before being posted on the map interface. Events will go through manual verification by administrators or moderators, to make sure that the events are real, and the organizers are trusted.

# List of References

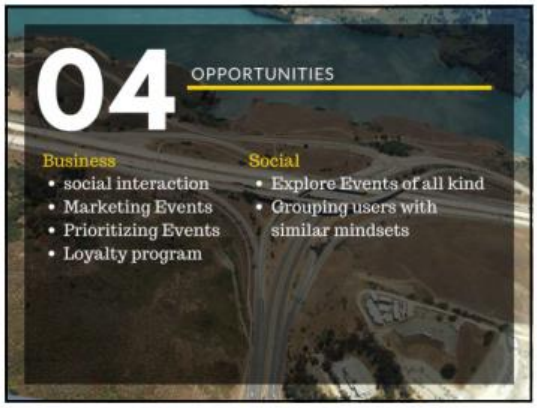
1. Horton, John (2015). *Android Programming for Beginners.* Livery Place, Birmingham: Packt Publishing Ltd.
2. Google developer site – http://developers.google.com/
3. StarUML - http://staruml.io/

# Appendix A: Presentation

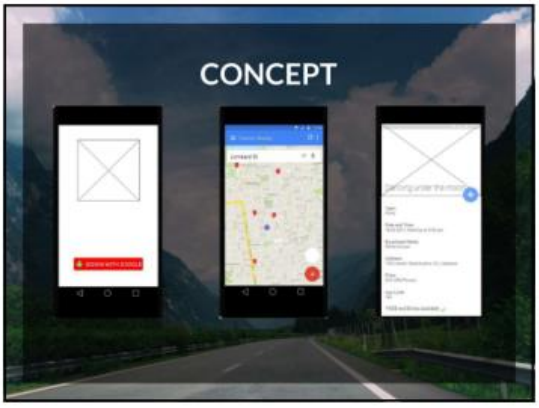



03
GOALS AND OBJECTIVES
• Creating and sharing Events
• Searching Events
• Filtering Events
• View number of attendees
• Bookmark Events
• Reserve or buy tickets
• Request taxi
• Access Events through application
• User-friendly interface

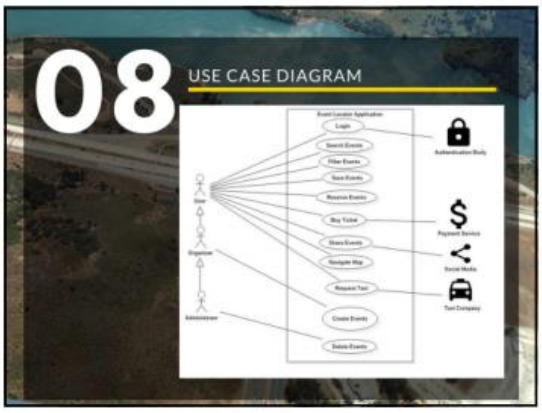

04
OPPORTUNITIES
Business
• social interaction
• Marketing Events
• Prioritizing Events
• Loyalty program
Social
• Explore Events of all kind
• Grouping users with similar mindsets

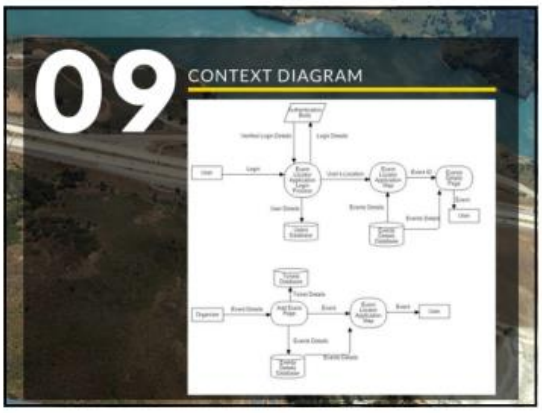

05
RISKS AND COMPETITOR
Risks
• Low adoption rate
• Losing users to competitors
Similar services
• Lebtivity
• GoSawa

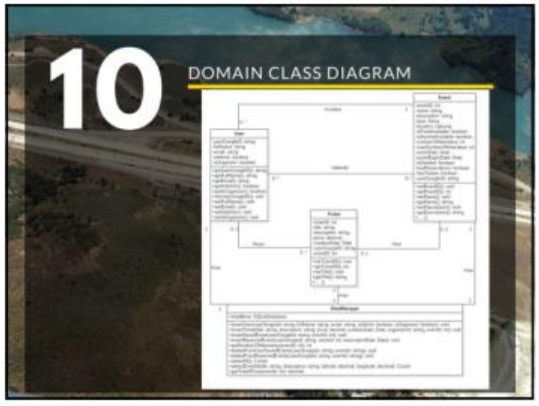

06
ADVANTAGES OVER COMPETITORS
Lebanon Events
Lebtivity
Native mobile Applications
Display nearby Events
Access Taxi
Buy tickets

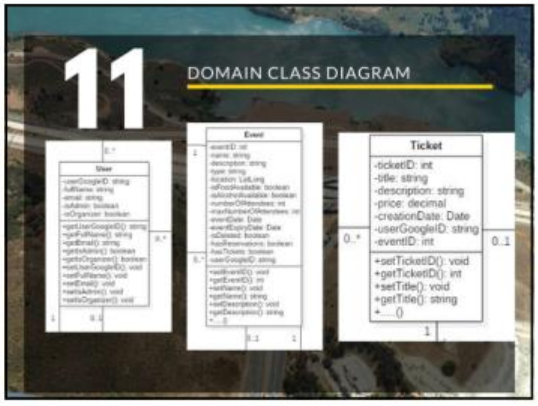

CONCEPT

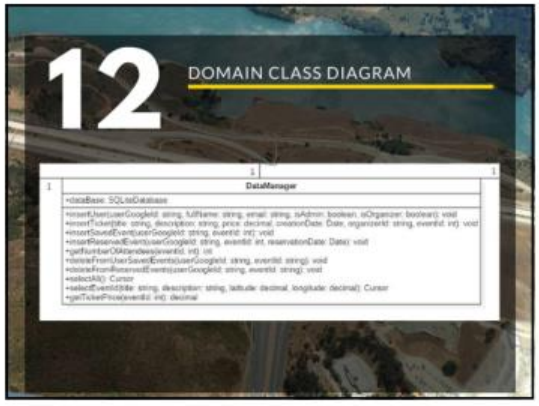

07
DIAGRAMS
• Use case Diagram
• Context diagram (level 1)
• Design Class Diagram
• Database design diagram

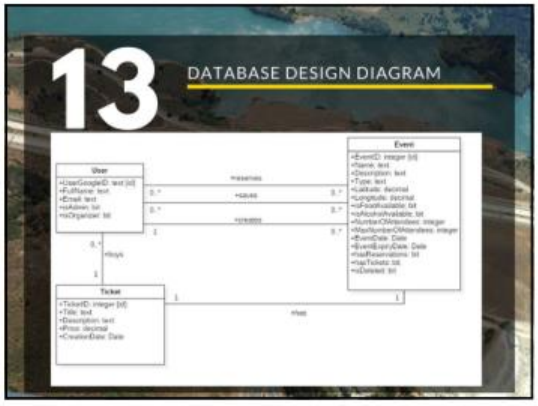

08
USE CASE DIAGRAM


09
CONTEXT DIAGRAM

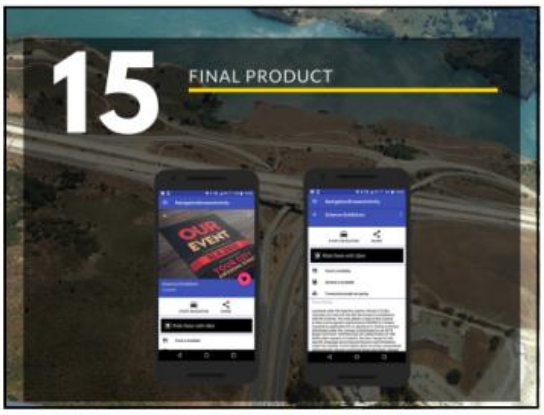

10
DOMAIN CLASS DIAGRAM

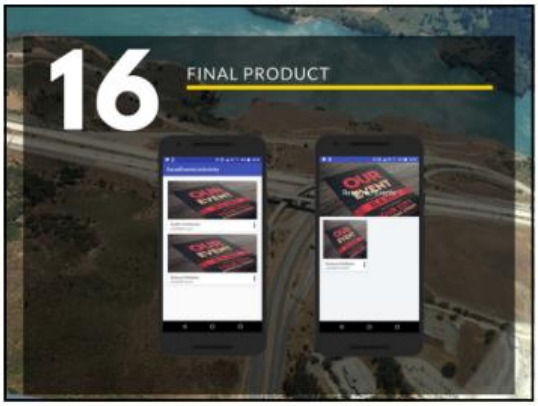

11
DOMAIN CLASS DIAGRAM
Ticket
-ticketID: int
-title: string
-description: string
-price: decimal
-creationDate: Date
-userGoogleID: string
-eventID: int
+setTicketID(): void
+getTicketID(): int
+setTitle(): void
+getTitle(): string
+.....()

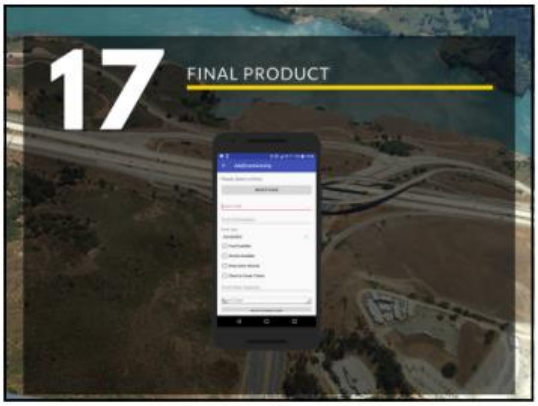

12
DOMAIN CLASS DIAGRAM
DataManager


13
DATABASE DESIGN DIAGRAM


14
FINAL PRODUCT


15
FINAL PRODUCT


16
FINAL PRODUCT

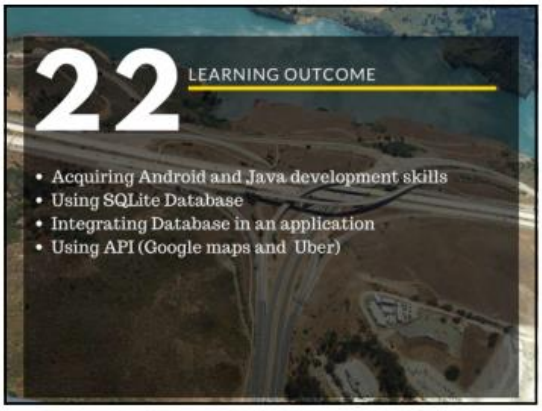

17
FINAL PRODUCT


18
DESIGN COMPARISON

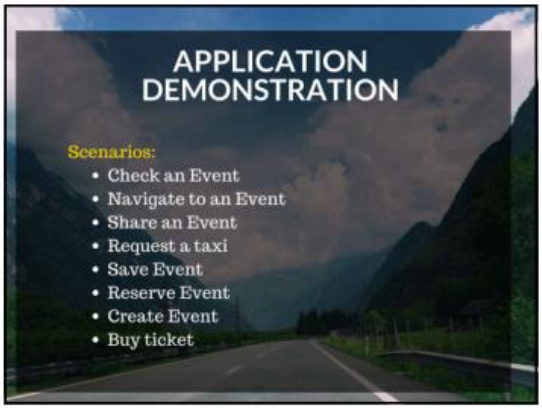

19
DESIGN COMPARISON

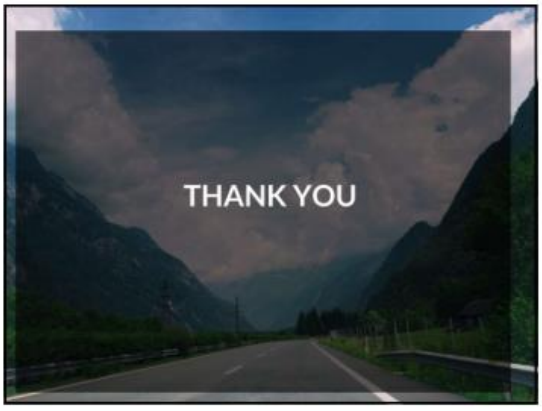

20
DESIGN COMPARISON
OUR EVENT
NOTRE DAME UNIVERSITY


21
FUTURE WORK
• Database moved to remote server
• Ability to edit Events after creation
• Changing the marker icon to correspond to the Event type
• Verifying Events


22
LEARNING OUTCOME
• Acquiring Android and Java development skills
• Using SQLite Database
• Integrating Database in an application
• Using API (Google maps and Uber)


23
REFERENCE
John Horton
PACKT publishing
Android Programming for Beginners


APPLICATION DEMONSTRATION
Scenarios:
• Check an Event
• Navigate to an Event
• Share an Event
• Request a taxi
• Save Event
• Reserve Event
• Create Event
• Buy ticket


THANK YOU