LocoLit: A Real-Time View of Local Attractions

by

Atef Khan

B.E., Muffakham Jah College of Engineering, 2016

A REPORT

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Computer Science
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2018

Approved by:

Major Professor
Dr. William Hsu

Abstract

This report describes the development of a location-based restaurant finding application, with a machine learning bases classification capability for summarizing user rating. The main idea behind the application is to allow the user access rating, images and videos uploaded in real-time before deciding to visit that respective location. The goal of tis implementation is to apply information integration- especially geolocation, video data and text bases rating analysis – to develop a usable, responsive information retrieval and access system for area restaurant and user review. There is a huge tradeoff in terms of information gain and time spent viewing a certain page. It is increasingly becoming important to consolidate information about a place into one tap. A simple online search may provide data about time and location but does not provide the user a visual representation of what is going on. The purpose of this project is to develop an application to reduce this technical gap and to serve the cliental of local businesses and patrons.

The main motivation was to create a minimalistic application that encompassed all the needs a young adult would require making day-to-day decisions. It also increases transparency and drastically saves users the cost of physical presence by providing summative information about local businesses, including georeferenced videos and other users written review.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank my major professor Dr. William Hsu for his encouragement and guidance throughout the project. I would also like to thank my committee members Dr. Torben Amtoft and Dr. Mitchell Neilson for serving in my committee and for imparting me with sufficient knowledge during my course work.

Finally, I would also like to thank the administration and the department of Computer Science for their support and guidance throughout my graduate degree.

Dedication

I would like to dedicate this project to my Mother, Shabana Tasneem and my father Mazher Ali Khan.

Chapter 1 - Introduction

1.1 Overview

This report describes a project on information retrieval systems for location-specific recommendation of businesses such as restaurants and incorporate video retrieval and review summarization - using machine learning for classification - to give tourists and visitors a way to preview the business remotely. To complete this project, interfaces for various social media services (such as Facebook and Instagram) and geoinformatics services (such as Google Maps) were used, as well as a machine learning library (Canopy), to implement a text-based review classification system that performs a rudimentary form of sentiment analysis.

1.2 Use Case

As a college student it is essential to plan the day ahead and prevent procrastinating or wasting time. After successfully managing a hectic college schedule, students often blow off steam after their classes are done but are restricted to relying on a few reviews and images uploaded about a local business. Human nature craves stability and information, a user wants to be informed in detail about a location without having to spend too much time searching for it. Also, the same location traffic may differ over different times, the question arises "is it worth going out today?". With an average attention span of 12 seconds for a millennial, the split-second decision must be made quick about a place.

This was the motivation behind developing this application. The project required a significant amount of work and was a challenge because this was the first iOS application the author had developed. This might be a good platform for students to view local business attractions and to improve transparency.

LocoLit provides a platform for users to capture images and videos in addition to being able to see a grid view of recently uploaded images. The application consists of two databases, one for image and video storage, while the other database is purely text based.

1.1 Related Work

There are two major applications that provide the current services but are two separate entities namely Snapchat and Yelp.

Yelp is also another popular website that allows users to rate and leave comments about a website. In the last few years there have been cases of users being paid to leave positive reviews. This do not allow transparency in the rating process. Since rating are updated along with an image or video upload, the user can make their own judgement if the original user’s comment was justified or not.
In addition to this usually users do not write reviews on a regular basis but do take a lot of images or videos.

The author is not aware of any prior applications that encompasses these same ideas into a simple application in a user-friendly manner. Snapchat, despite being a popular application, does not provide a rating system for locations. The motivation behind this project is to help the community since increased spending in the local area eventually contributes to more spending and job creation. The existing companies have overcrowded their applications with irrelevant sections and ads; this application only provides a good alternative.

Chapter 2 - Requirements Analysis

The goal of this implementation is to apply information integration - especially geolocation, video data, and text-based ratings analysis - to develop a usable, responsive information retrieval and access system for area restaurants and user reviews. A major criterion here was to account for the lack of attention span students have during college, hence this app has a few seconds to make a good first impression. The application must be user friendly, should be robust, secure and must update constantly in real-time. These requirements consisted of asking hypothetical questions to a group of students who have different ethnicity and age groups. It was essential to ensure that the application provided all the necessary information before a user can make an informed decision about visiting a business location such as a bar or a restaurant. A simple Google search may provide the location but does not paint a vivid picture of what is occurring in real-time.

The requirements were implemented in a strict two-week sprint, followed by an additional requirement gathering session, which is essentially following a scrum model of development.

The requirements gathered could be classified into two categories, the functional and non-function requirements.

2.1 Functional Requirements

In software development functional requirements are defined as description of the behavioral aspect of the application. The requirements provide an abstract idea of what the system should do. A functional application which can upload and download content on demand while allowing users in the vicinity to upload real-time updates to ensure authenticity. The user can see the distance from current location to the prospective location and data should be handled to prevent breach of security. The user should be able to see real-time video recordings instantly on a single click, while displaying valuable information like the average cost per person, image of the location, real-time ratings and real-time video updates. The user must be able to clearly distinct one location of a club or restaurant from another. To promote simplicity and application must detect the user location and upload to the video feed of that location without any prompts. To maintain credibility the user will not be allowed to contribute to the live feed if the location is not a recognized location. The application also maintains authentication with extensive logging in the database.

2.2 Non-Functional Requirements

Nonfunctional requirements are defined as the attributes that are used to judge the operational system. These are features hidden from the user but are essential to the running of the application

1. **Cost**: The application is free on the app store and will help promote businesses that are under appreciated in the local area which in turn contributed to more revenue for the county.
2. **Accessibility**: The application is easy to use and operates without any additional requirements than a phone.
3. **Performance**: The response time to on demand requests must be handled without delays. The application takes only takes two seconds for submission time while testing on a regular LTE and WIFI connection.
4. **User interface**: The user has access to all information by clicking a marker
5. **Flexibility**: The application was deigned to support various screen sizes and resolutions.

2.3. iPhone Specifications

This section gives the specifications for iPhone models as of the implementation period in 2018. The experimental model of iPhone was an iPhone 7

2.3.1. User Hardware Specifications

- Device: iPhone / iPod
- Minimum Space Required: 100MB

2.3.2. User Software Specifications

- Operating System: iOS 11.1 or higher.

2.3.3 Hardware Requirements for Developer:

- Device: iPhone/ iPad / iPhone

- Minimum Space Required: 100 MB
- Lightning cable for simulator

2.3.4. Software Requirements for Developer:

- Operating System: OS X
- Tools: X-code, Debugger
- Language: Swift
- Debugger: iPhone Simulator is a part of Xcode which allows running the application on a simulator.

Chapter 3 - Tools, APIs and Dependencies

3.1 Firebase Database

The firebase database[3] is a real-time NOSQL database that stores information in a JSON Format. Firebase operates as the back end of the system and supports API calls to ensure synchronization of data uniformly over the platform. It also provides an extra layer of security by allowing addition of rules such as read or write. It acts like a middle layer which constantly handles requests for data.

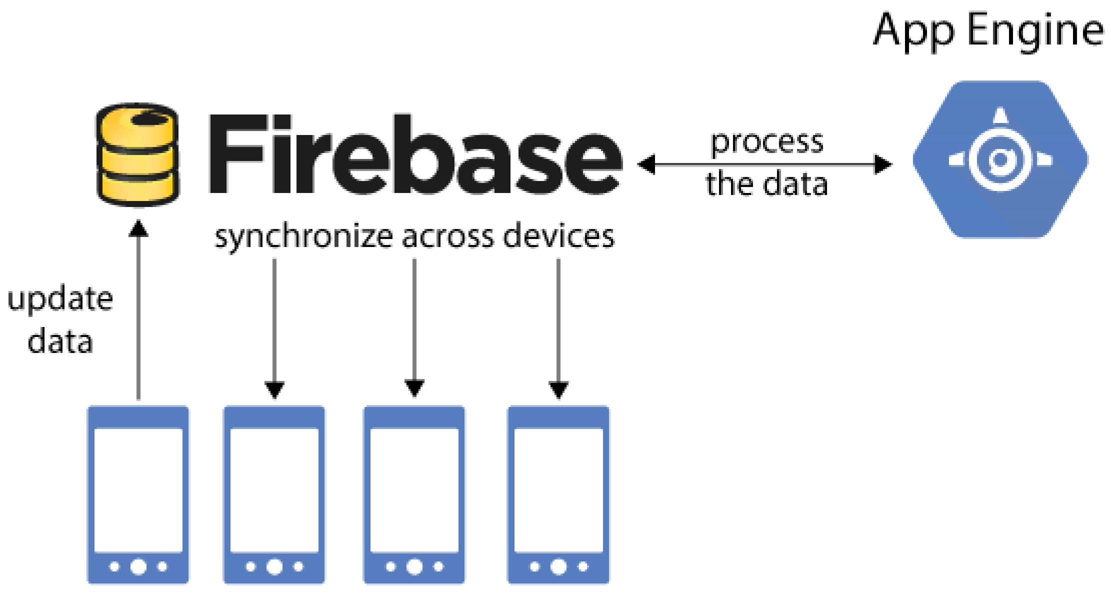


Figure 3-1 Firebase Architecture

3.1.1 Firebase Authentication

Firebase authentication[1] is used to handle login authentication requests. This API requires a user to be authenticated before performing any changes to the database such as upload and downloads. Once a user is authenticated the application personalizes the experience according to the user requirements and handles all authentication once a user is successfully logged in. It provides an extra layer of security since each login and session is logged to ensure data authenticity.

3.1.2 Firebase Storage

Firebase storage[2] provides a secure API to upload and download videos and images. This storage handles transfer of data files and synchronizes all devices. It was essential to ensure that data requests were not made directly to the database for better security. Therefore it first authenticates the user to access the database then the app engine internally sends a following request to the storage. Because there is a strict format checker this prevents an adversary from injecting malicious code to the storage directly.

3.2 Player API

Player API allows local media, remote screening of media over HTTP by providing a customizable player to queue and play the videos one after another. This uses a built-in AVFoundation that coordinates with the Player API to provide a replay loop of the video.

3.3 AVFoundation

AVFoundation[5] is a built-in package provided with swift that allows the user to capture images and movies. Using AVFoundation we can play create and edit Mp4 and Mpeg-4 file formats which is captures by the camera of an iPhone. AVCaptureSession is an object that manages the flow of data from the input device to the capture device. An AVCaptureSession must be setup to detect the inputs used with session such as camera and microphone

3.4 Google Maps API

Google Maps API provides the visual representation of the map and the markers that represent the various locations. The SDK handles access to the Google Maps server and displays the queried result and directions to the location

3.5 Facebook API

In this case Facebook API[4] was used as an additional way to sign up for the application. In other words it facilitates Facebook profile integration. The email id and general information is extracted from the packet returned and is entered into the database.

3.6 Instagram API

Instagram API[5] was also used as an additional signup service as well. Again the application only retrieves relevant data and provides Instagram integration.

Chapter 4 - System Architecture

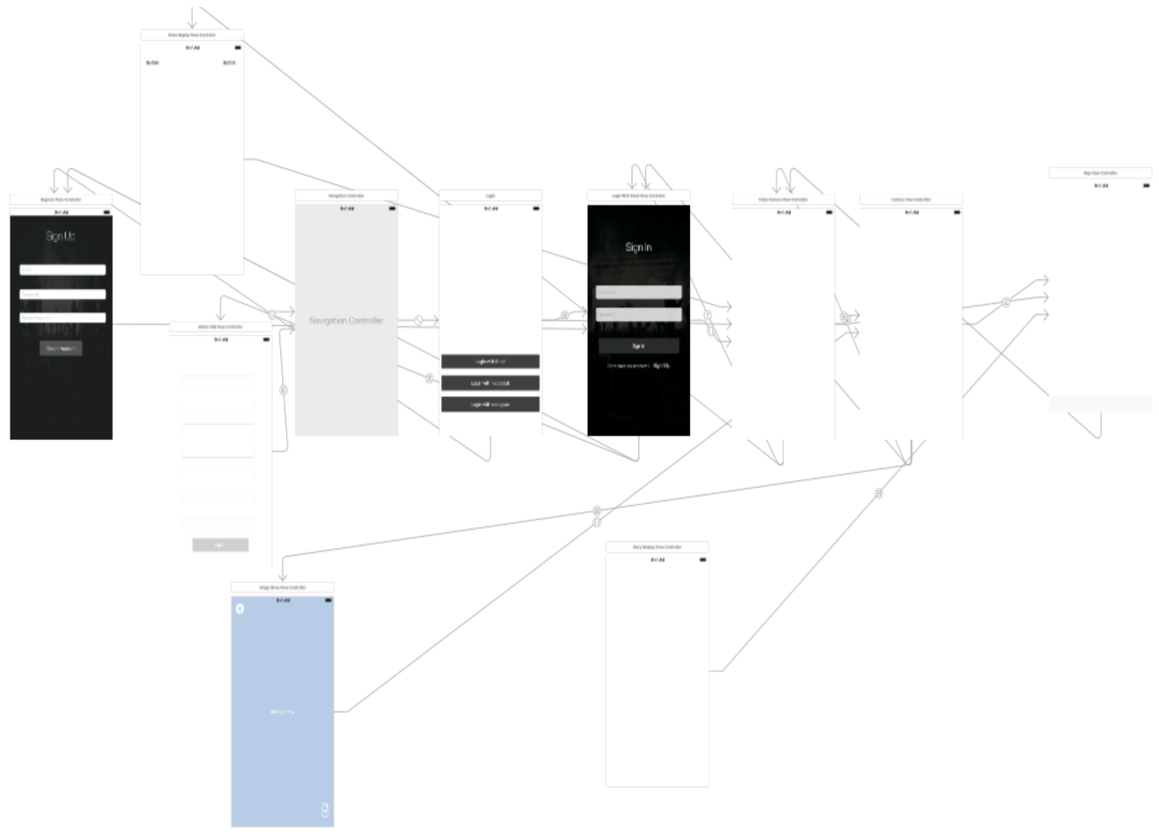


Figure 4 -1 System Architecture

4.1 View Controller Architecture

The architecture of the application consists of multiple views and controllers which can grouped into various architectural layers. The various controllers of the System are mentioned below with each purpose.

1. HomeViewController: The HomeView Controller consists of the initial startup screen that manages the user’s choice of authentication. This consists of signing up by email, facebook or Instagram. The screen also handles a route to signing up as well.

2. LoginWithEmailViewController: The login with email controller allows the user to be authenticated with email and handles validation such as the email being in the correct format

3. VideoCameraViewController: The Video Camera Controller manages recording videos and allows a switch of options between the video and camera

4. The VideoReplayViewController This controller receives captured data from the video camera controller and provides options of uploading or cancelling. This also detects the user location on upload to automatically determine the user location and most probable business name
5. CameraViewController: This controller manages image capture and prepares the image to be transferred to the next view controller
6. ImageViewController: Allows user to make comments relating the location and also provide a rating
7. MapViewController: Interacts with Google API to setup custom markers on the map and display location video feed
8. CustomerMarkerView: Describes the format of a marker
9. RegisterViewController: Allows a user to sign up via email and select a password, here validation is performed in the background to ensure passwords match.
10. FeedViewController: The feed view controller displays the video feed of the marker location

4.2 Application Architecture

The application architecture is divided into four layers:

1. Presentation Layer:

The presentation layer is used to provide user interaction with the various controls of the application. The presentation layer manages all on screen gesture by delegating the control to extensions in the code. The presentation layer consists of various pages that interact with each other using Story board segues. The presentation layer consists of various view controllers which help provide essential information for completing operations.

2. Data Layer:

The data layer provides a link between the API Services and the app engine. Data layer in this context is used to upload, download images or videos from the database.

3. API Layer:

API Layer is useful since it allows more functionality to the application without adding excessive code. The API layer is responsible for making requests and retrieving information.

5. Local Storage:

To avoid data loss the local storage is used to store the captured data before being uploaded to the database. Authenticity is also maintained by storing metadata along with the captured media in local memory of the device to ensure format handling is more robust. The local storage layer communicates with each other constantly passing information between each layer while maintaining authentication and easy interaction.

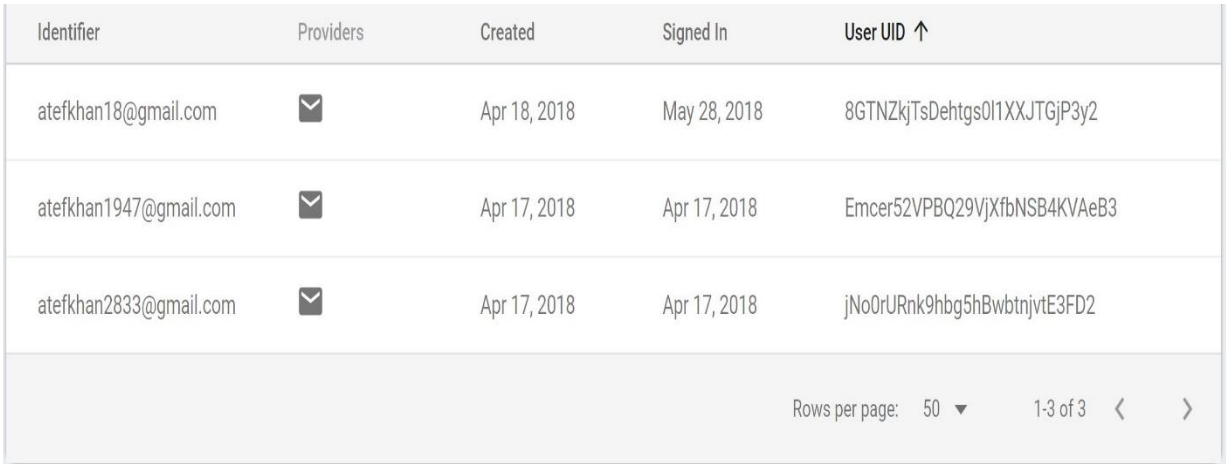
Chapter 5 - System Implementation

The System implementation consisted of four different modules which collectively interact with each other. The modules are given below:

1. Login Module
2. Camera Module
3. Map Module

5.1 Login Module

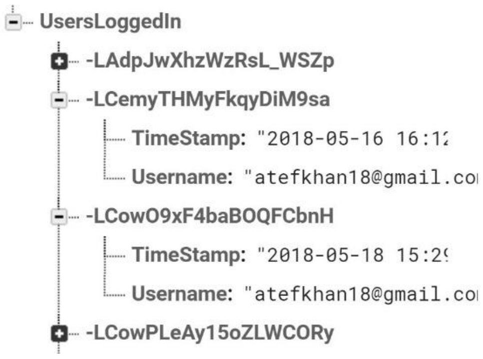
The login module handles user authentication once and maintains the same session unless the user wishes to log out. The Login module also works throughout the application in background for security purposes and to facilitate 'Auth' requests. It receives input from the user and interacts with the Firebase API and returns an authentication response.
Each time a user is logged in it triggers a logger to add an entry to the login database. Each user email is assigned a GUID to ensure a unique identifier for each entry. It also keeps track of important information such as created data and latest sign in.



Identifier	Providers	Created	Signed In	User UID ↑
atefkhan18@gmail.com	✉	Apr 18, 2018	May 28, 2018	8GTNZkjTsDehtgs0l1XXJTGjP3y2
atefkhan1947@gmail.com	✉	Apr 17, 2018	Apr 17, 2018	Emcer52VPBQ29VjXfbNSB4KVAeB3
atefkhan2833@gmail.com	✉	Apr 17, 2018	Apr 17, 2018	jNo0rURnk9hbg5hBwbtnjvtE3FD2

Figure 5 -1 Firebase Email Authentication

To maintain scalability and to prevent data duplication each login is assigned a Unique ID with metadata information such as time stamp and email address used.

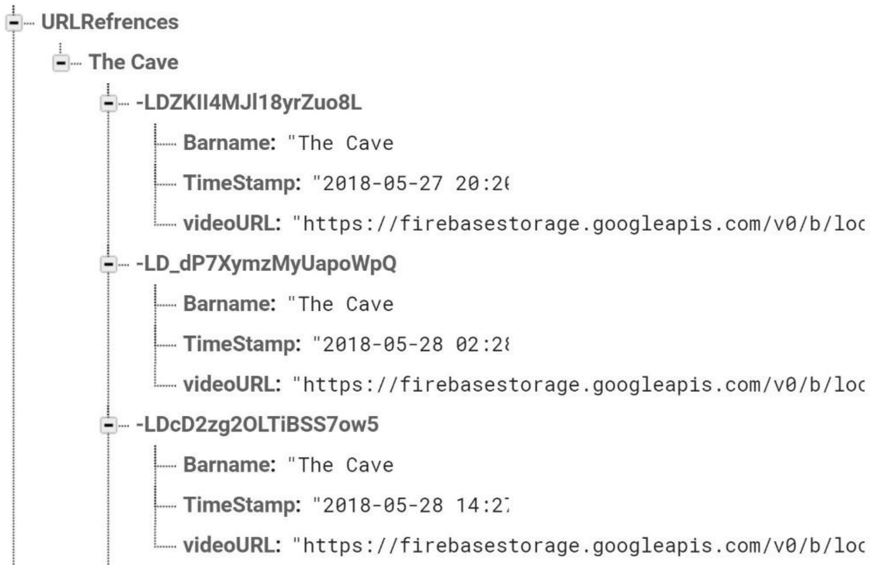


```
UsersLoggedIn
    -LAdpJwXhzWzRsL_WSZp
    -LCemyTHMyFkqyDiM9sa
        TimeStamp: "2018-05-16 16:1
        Username: "atefkhan18@gmail.co
    -LCowO9xF4baBOQFCbnH
        TimeStamp: "2018-05-18 15:2
        Username: "atefkhan18@gmail.co
    -LCowPLeAy15oZLWCORy
```

Figure 5 -2 Tree Structure of Database

5.2 Camera Module

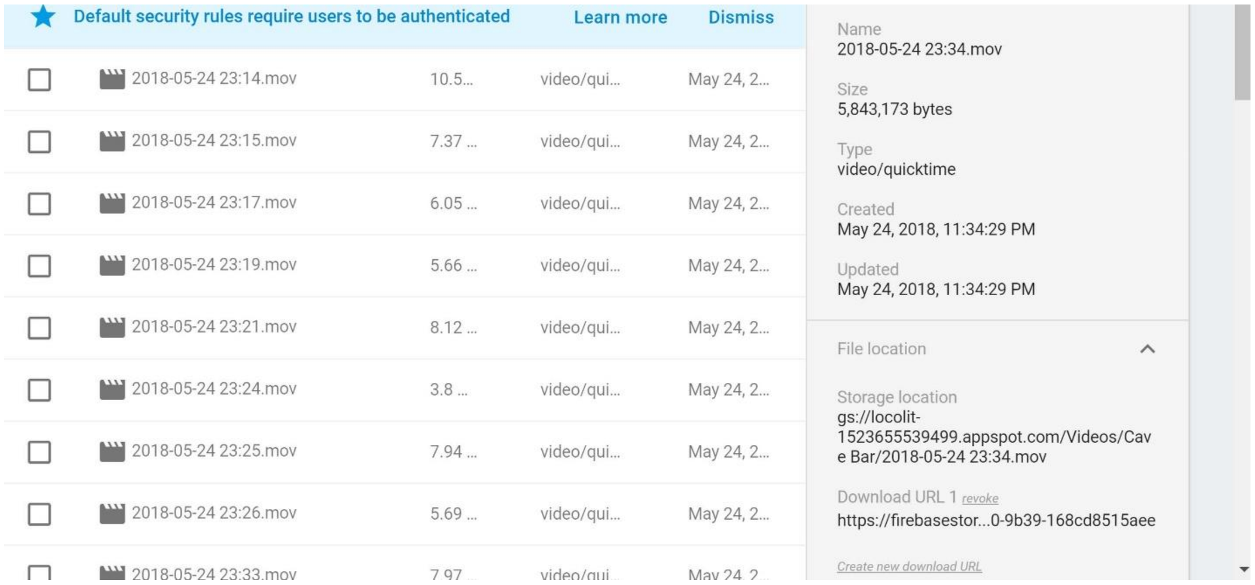
The camera module handles all device image and movie capture. In the back end it automatically detects the user’s location and determines the most probable business location the user is visiting. The user uses the camera module to upload images and videos to the database which is the added to the feed of the individual business locations. The image and video storage is handled by Firebase Storage and Firebase Database API. Each upload consists of a entry in the database consisting of important information such as the club name and a unique link URL which can be used to access the video directly.



```
URLRefrences
    The Cave
        -LDZKIl4MJl18yrZuo8L
            Barname: "The Cave
            TimeStamp: "2018-05-27 20:2
            videoURL: "https://firebasestorage.googleapis.com/v0/b/loc
        -LD_dP7XymzMyUapoWpQ
            Barname: "The Cave
            TimeStamp: "2018-05-28 02:2
            videoURL: "https://firebasestorage.googleapis.com/v0/b/loc
        -LDcD2zg2OLTiBSS7ow5
            Barname: "The Cave
            TimeStamp: "2018-05-28 14:2
            videoURL: "https://firebasestorage.googleapis.com/v0/b/loc
```

Figure 5 -3 Bar Information Database

Since the app engine detects the location automatically it assigns the video to the correct folder with custom written metadata such as latitude, longitude and a full timestamp.



Default security rules require users to be authenticated Learn more Dismiss

☐	2018-05-24 23:14.mov	10.5...	video/qui...	May 24, 2...
☐	2018-05-24 23:15.mov	7.37 ...	video/qui...	May 24, 2...
☐	2018-05-24 23:17.mov	6.05 ...	video/qui...	May 24, 2...
☐	2018-05-24 23:19.mov	5.66 ...	video/qui...	May 24, 2...
☐	2018-05-24 23:21.mov	8.12 ...	video/qui...	May 24, 2...
☐	2018-05-24 23:24.mov	3.8 ...	video/qui...	May 24, 2...
☐	2018-05-24 23:25.mov	7.94 ...	video/qui...	May 24, 2...
☐	2018-05-24 23:26.mov	5.69 ...	video/qui...	May 24, 2...
☐	2018-05-24 23:33.mov	7.97 ...	video/qui...	May 24, 2...

Name
2018-05-24 23:34.mov

Size
5,843,173 bytes

Type
video/quicktime

Created
May 24, 2018, 11:34:29 PM

Updated
May 24, 2018, 11:34:29 PM

File location

Storage location
gs://locolit-1523655539499.appspot.com/Videos/Cave Bar/2018-05-24 23:34.mov

Download URL 1 revoke
https://firebasestor...0-9b39-168cd8515aee

Create new download URL

Figure 5 -4 Firebase Video Storage Structure

5.3 Map Module:

The map utilizes the Google Maps API to display the markers for the locations and show the user's current location so the user gets an idea of the locations nearby. Each location is represented as a marker on the map and displays the video feed for that location with a single touch.

5.4 USE CASE DIAGRAM

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a use-case analysis. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

Use Case Diagram

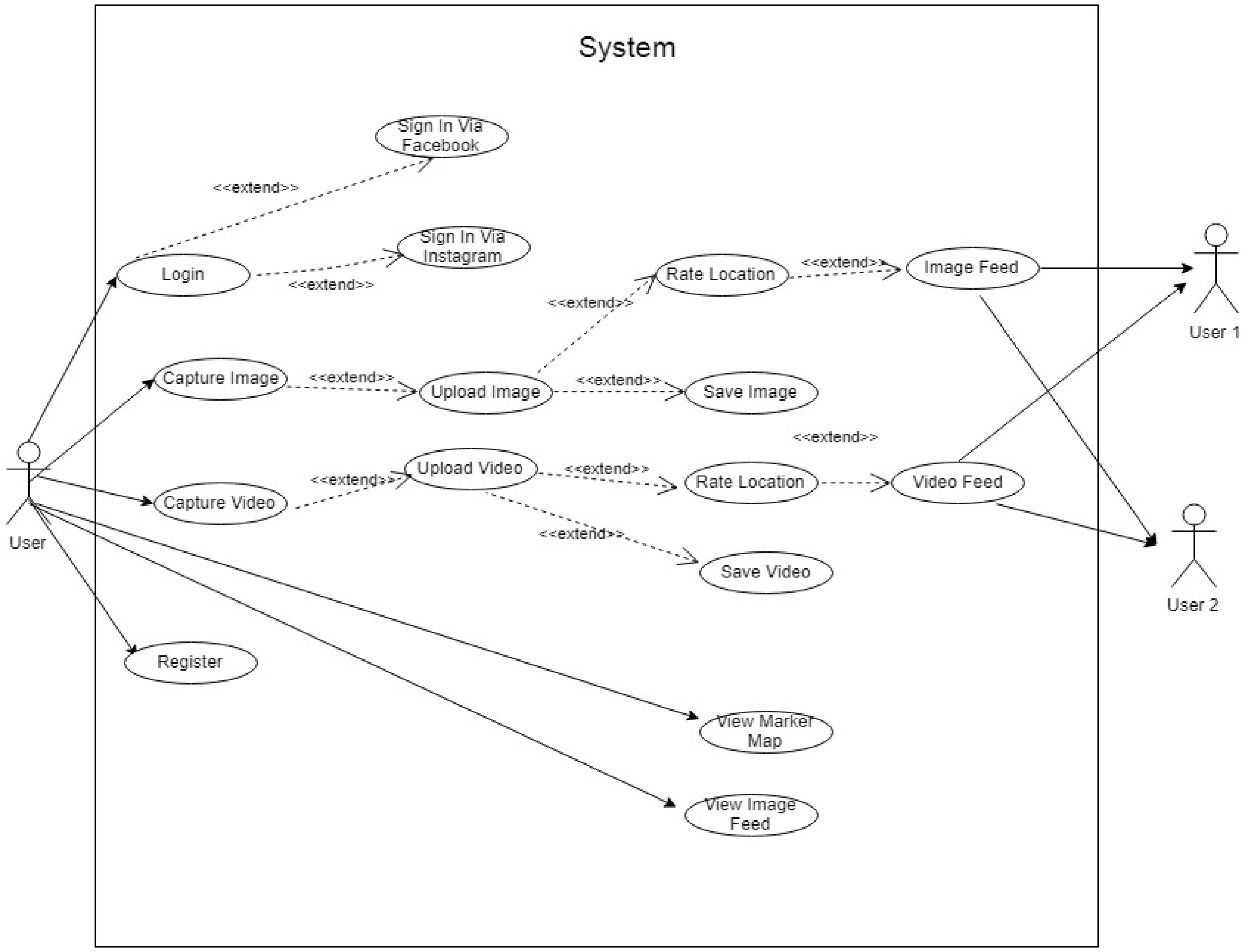


Figure 5 - 5 Use Case Diagram

5.5 CLASS DIAGRAM

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

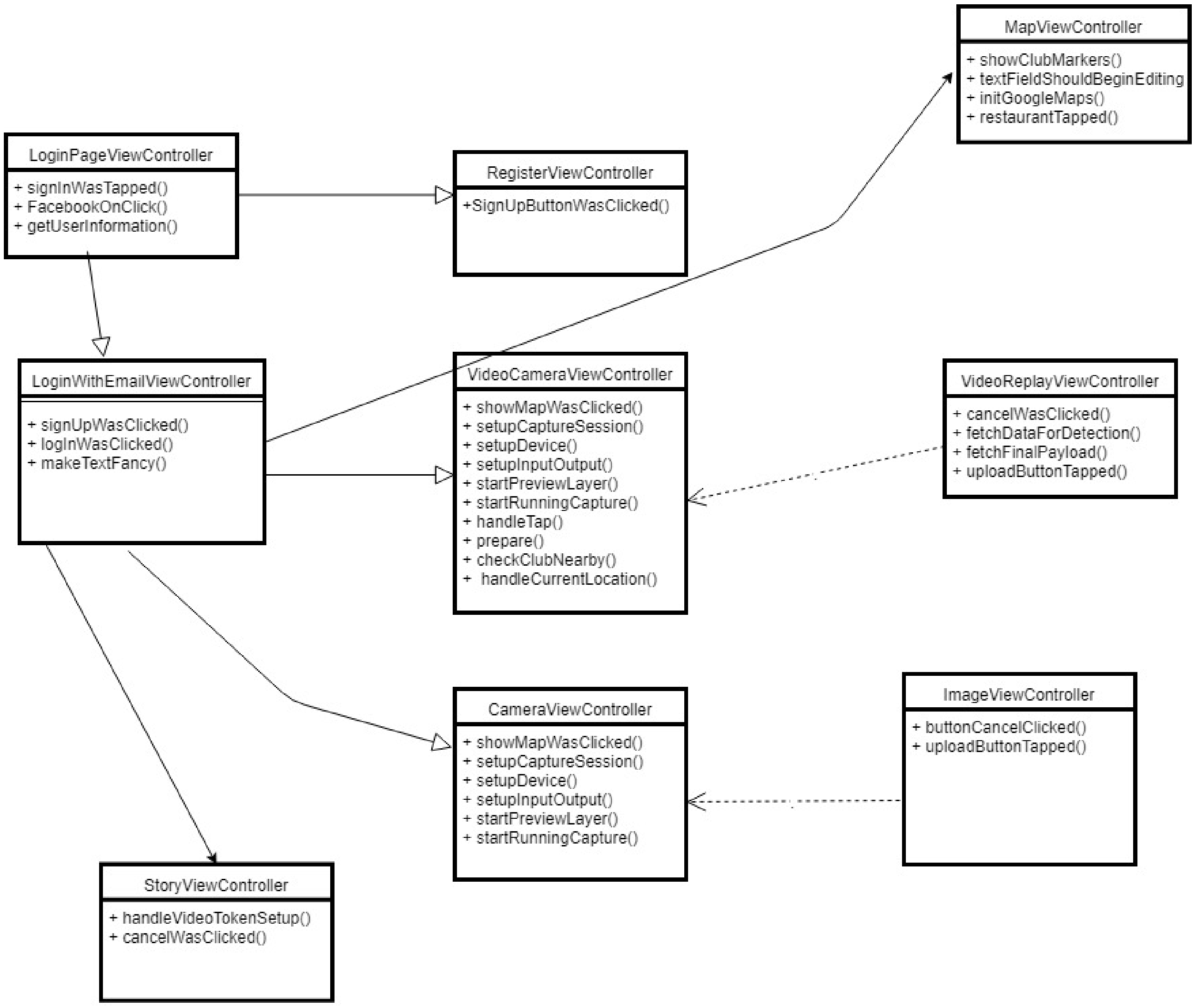


Figure 5 - 6 Class Diagram

Chapter 6 – GUI and Back end

The application can be divided into three segments for this demonstration since each cluster of view controllers are independent of each other, they are

6.1. Login Segment

6.1.1. Login GUI View

On startup the user is presented with options on which method the user would like to log in. The three options are email, Facebook and Instagram which corresponds to their own methods and utilizes their respective API's. Once the user has been logged in a session key is initiated for the user and autologin is setup after the initial log in.

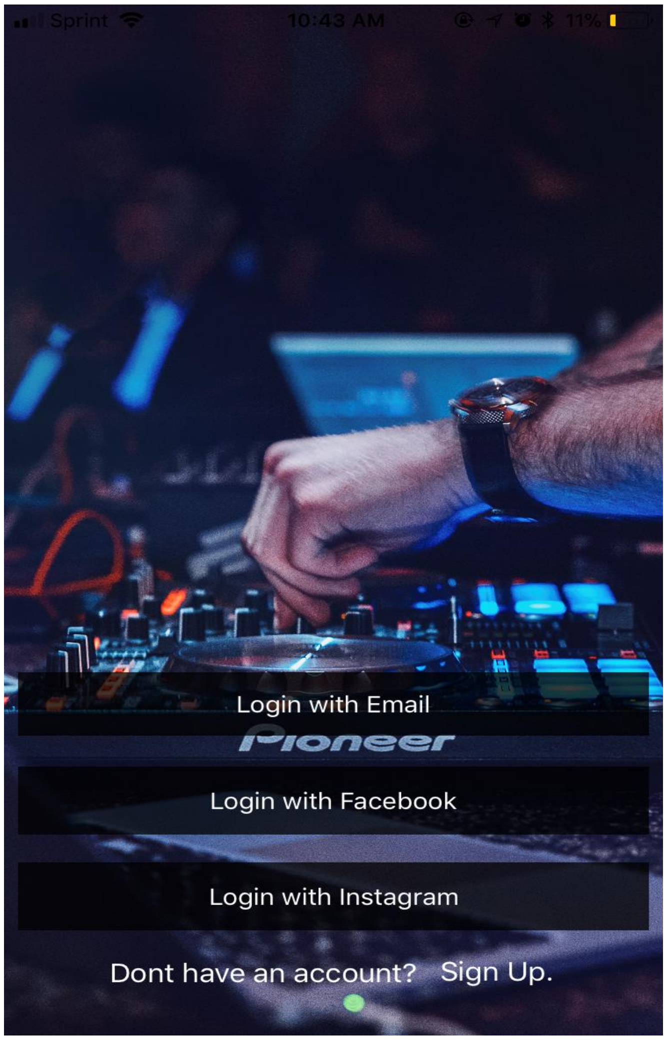


Figure 6 - 1 Login View Controller

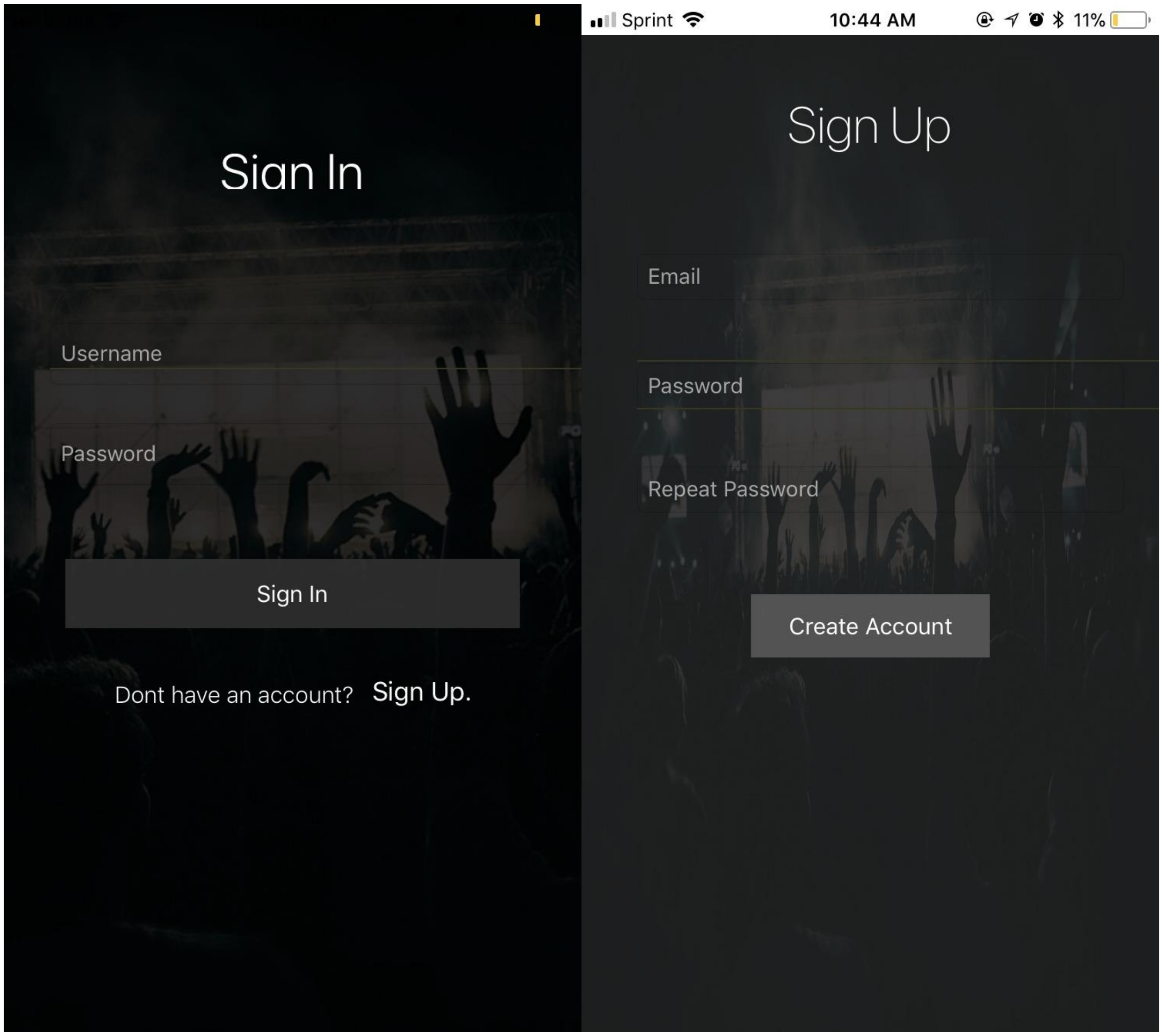


Figure 6 - 2 SignIn View Controller & SignUp View Controller

6.1.2 Login Back End

The login controller in the back end uses Firebase Database API, Facebook API, and Instagram API.

6.1.2.1 Sign in with Email

Tapping on ‘Sign in with Email’ will trigger a segue which transitions to the sign in page. The login and password text fields are extended to the view controller to retrieve the text obtained from the text field which act as parameters for the Firebase Database Authentication API.

The two text fields that store and retrieve information are extended and are declared as follows.

```
 @IBOutlet weak var userEmail: UITextField!
  @IBOutlet weak var userPassword: UITextField!
```

The sign in button triggers a function execution called logInWasClicked which utilizes the firebase authentication to log and authenticate the user.

```
@IBAction func logInWasClicked(_ sender: UIButton) {
    Auth.auth().signIn(withEmail: email, password: password)
        databaseReference?.child("UsersLoggedIn").childByAutoId().setValue(post)
```

6.1.2.2 Sign in with Facebook

Tapping on ‘Sign in with Facebook’ transitions to Facebook login portal, provided the user has a valid account setup, the email address and basic information is extracted from the Facebook packet returned. The swift code snippet is shown below

```
case .success(_,_,_):
  let success = true
  self.getUserInformation{userInfo, error in
    if let error = error { print(error.localizedDescription)}
```

```
                    if let userInfo = userInfo, let id = userInfo["id"] ,let name = userInfo["name"], let
email = userInfo["email"] {
                        self.faceBookRetrieve.text = "ID: \(id), name: \(name), email: \(email)"
                    }
                    if let userInfo = userInfo, let pictureUrl = ((userInfo["picture"] as? [String:
Any])?["data"] as? [String: Any])?["url"] as? String{
                        print(pictureUrl)
```

6.2. Media Capture Segment

The media capture segment encompasses the image camera and movie camera. The cameras can be switch using the camera switch button. After the image or movie is captured a preview of the capture media is displayed before the upload can be done.

6.2.1. Image / Video GUI

The camera was designed similar to that of Snapchat's interface since the target users are of the age group 18-27, despite having an built-in camera, a lot of users still use Snapchat to take images. Thus, the design was minimalistic and all features or available in the background of the video or image capture view.

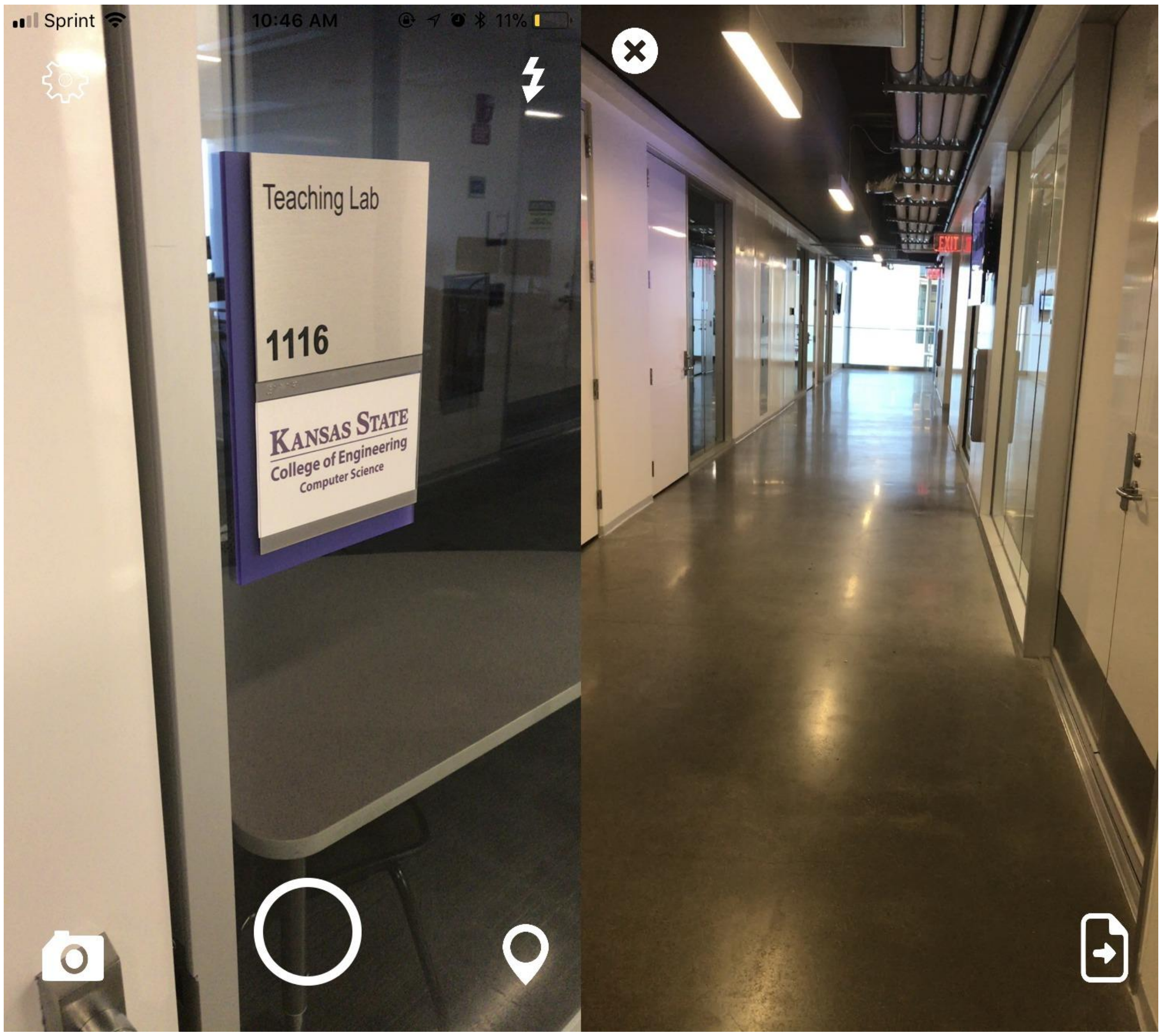


Figure 6 - 3 Video View Controller

6.2.2 Image/ Video Back end

The swift dependencies are AVFoundation and AVPlayer which helps setup and initiate a capture session. The captured video or image is stored to a temporary storage location and the image/video is previewed before an upload is initiated. The entire process is handled by using five methods.

setupCaptureSession()

setupDevice()

setupInputOutput()

startPreviewLayer()

startRunningCapture()

6.3. Feed View Segment GUI

The users can also skip to the Feed segment to view the real-time images and videos that are being uploaded. The videos are played with a full screen view while the images are viewed as grid of images.

6.3.1 Map View GUI

The map shows all the locations nearby which are registered with the application, and display a pop up marker with the Bar or restaurant logo. This marker consists of basic information like entree fee and the current rating. This is handled using the map API methods for Google Maps API.

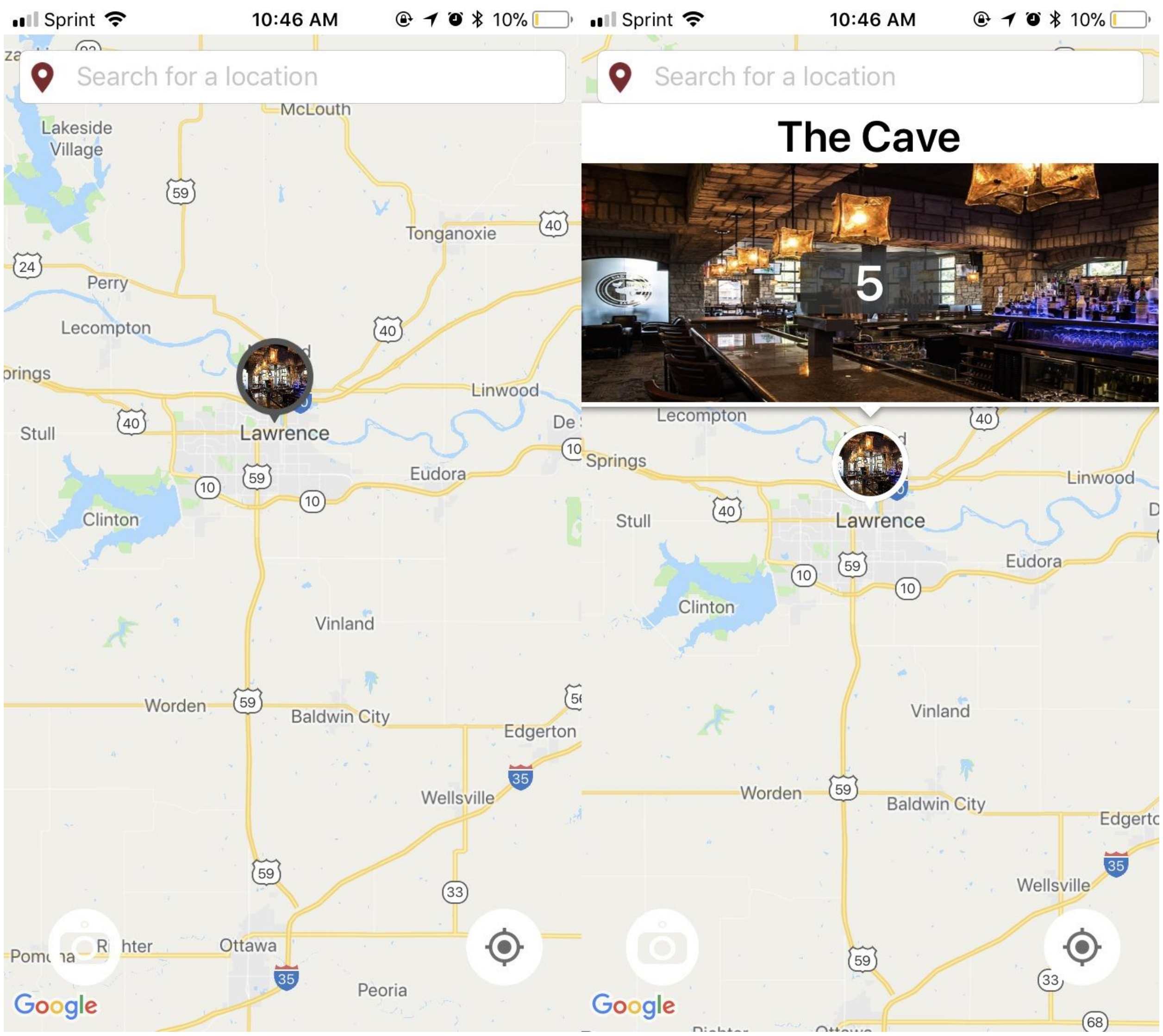


Figure 6 - 4 Map View Controller

6.3.2 Map View Back end

```
    func showClubMarkers(title: String, lat: Double, long: Double, imageURLLink: String){
  let customMarkerTitle = title
  let imageBarURl = imageURLLink
  let customMarker = CustomMarkerView(frame: CGRect(x: 0, y: 0, width:
customMarkerWidth, height: customMarkerHeight), image: #imageLiteral(resourceName:
imageBarURl) , borderColor: UIColor.darkGray, tag: 10, barname: title, imageURL:
imageBarURl)
  marker.iconView=customMarker
marker.position = CLLocationCoordinate2D(latitude: lat, longitude: long)
marker.map = self.googleMapsView        }
```

Chapter 7 - Testing App Performance

Software testing is the process of determining if the system performs the operations it is supposed to. The conditions are evaluated to check whether the required conditions are met and to also analyze the performance of the processes. Testing is an important phase of the development life cycle and is done iteratively throughout the development process. The performance of the application in all conditions must be satisfactory since user data and personal information is at stake. For this application we used unit testing, performance testing, integration testing and compatibility testing.

Software testing is necessary since it is highly desirable to have a robust system which minimizes crashes.

7.1 Compatibility Testing

Performance testing was done with upload and download speeds in mind. All transition between the pages occur smoothly within a second and the submission time of a video is only one second away from being a part of the video feed. Even for larger videos, the performance is still on par with regular videos.

Table 7 - 1 Compatibility Test

S.No.	Test	Device	Result
1	Install and Run	iPhone 6 and above iPad 2 and above	Success
2	Page Segue	iPad 2 and above iPhone 6 and above iPod 5th Generation or Higher	Success
3	Stress Test	iPhone 6 and above iPod 5th Generation or Higher iPad 2 and above	Success

7.2 Unit Testing

Unit testing is a level of software testing which is used to test the individual unites or components of the software. A unit represents the smallest part of the software that can be tested.

Table 7 - 2 Unit Test

S.No.	Test case	Expected Result	Result
1.	Application Startup	Auto log in if the user logged in previously	Passed
2.	Enter wrong email and password	Email address validation alert return to default values	Passed
3.	Background Application	Cache current storyboard	Passed
4.	Launch application from background	Returns to last view controller pushed on the stack	Passed
5.	Initialize video view for recording	Access user permissions and prompt user for audio and video access	Passed
6.	Initialize Image view for capture	Access user permission and prompt user for image access	Passed
7.	Transition between camera and video	Performs segue to VideoViewController	Passed
8.	Transition from video to camera	Performs segue to CameraViewController	Passed
9.	Button Display on Load	Creates custom button and renders subview	Passed
10.	Button Disabled post recording	Post media capture record button disappears	Passed
11.	Capture Image	Transition to imageviewcontroller with image replay	Passed
12.	Ready Image for Upload	Secure connection to database and initiate location detection	Passed

13.	Upload to Database Verification for image	Begin tasks submission and return imageURL object	Passed
14.	Tap Capture Button	Transition to Image View	Passed
15.	Hold record button	Initiate delegation of functions to record media	Passed
16.	Upload to Storage Verification for video	Return video URL from uploaded metadata	Passed
17	Fetch Faacebook Information	Retrieve basic information as a response packet	Passed
18	Initialize Google Maps	Alert prompted to allow location detection	Passed
19	Prepare Markers for Setup	Initiate data fetch to display markers inheriting properties from	Passed
20	Display Markers on Maps	View marker image and distinct markers on Map	Passed
21	Tap on Custom Marker	Performs segue to CustomMarkerView	Passed
22	Display location story by clicking on custom marker	Tapping marker should initiate an instance of AVPlayer	Passed
23	Rate the Location	Uploads media along with rating	Passed
24	Detect User Location	Detects and return the accurate location of the user after prompting to allow access always	Passed
25	Request User Permissions	Check for user permissions in info.plist	Passed
26	Auto Detect Nearby location	Return most probable location from registered clients	Passed
27	Distance calculation according to current location	Correctly calculates distance from current location to nearest club.	Passed
28	Logout of the application	Logs out and disables auto login	Passed

29	Change screen orientation while viewing feed	Video readjusts and fills screen aspect ratio	Passed
30	Check performance of uploads	Submission time less than a second with no data loss or compression	Passed

7.3 Integration Testing

Integration testing combines the software and hardware components together to test how the software utilizes the software. Since this is an iOS application the application must be compatible over various devices of different resolutions. The workflow of the program is carried out using segue therefore it is necessary to verify that the view controllers are interacting with the help of the navigation controller. Since segue result in pushing the last view controller on the top of the stack, the sequence and data may be lost between these views. Integration testing will test for any application crashes.

Table 7 - 3 Integration Test

S. No	**Test Case**	**Expected**	**Result**
1	Change Location depending on movement	The correct location and the probable club nearby is returned and attached to any upload as metadata	Passed
2	Change Map View	The current location is constantly monitored while using the map thus there is a change in visible markers and video feed is fetch automatically	Passed
3	Change Camera View	Change the camera to upload images with comments or upload videos with rating	Passed

Chapter 8 – User Rating Classification

8.1 Classification Task

Canopy[9] is used for the machine learning aspect of this report, here, the classification task is to classify user rating based on application review data. The classifier used in this experiment is Multinomial Naïve Bayes Classifier. During the analysis stage it was found that there is a correlation between rating and rating length. The longer the text length is the higher the rating is. The evaluation metrics used to evaluate the performance of the classifier is F-score, precision and recall[8].

8.1.1 Naïve Bayes Classifier

Naïve Bayes Classifier is a probabilistic classifier based on Naïve Bayes theorem and Naïve Bayes Independence assumption. Despite several other classification algorithms outperform Naïve Bayes Classifier, for this task we consider Naïve Bayes since it performs better in terms of CPU and memory usage. Another advantage is it can train the model with a small training set. The assumption for this experiment is that we assume the features which are used for classification are independent of each other.

8.1.2 Multinomial Naïve Bayes Classifier

This estimates that the conditional probability of a particular word given the relative frequency of term t in documents belonging to a class C. The variation also considers multiple occurrences of a term.

8.1.3 Input

The input here is location reviews and also takes into consideration the fields, cool, funny and useful.

8.1.4 Output

The output is a rating between the range of 1-5. Here one represents a negative rating while 5 represents a positive rating. Since text length seems to have better results when determining a 1 or 5 output, the limit of the experiment is only restricted to 1 or 5 rating. While trying to obtain good classification scores for 3 and 4 rating, the performance was pretty poor with low scores, therefore this classifier can be categorized as sentiment analysis.

8.2 Library Dependencies

- **Pandas:** Used for data analysis and data transformation
- **Numpy:** Scientific library useful for calculations
- **Matplotlib:** Graph and heat Map
- **Nltk corpus:** Data Cleaning and removing punctuation
- **Canopy:** Source Code Editor

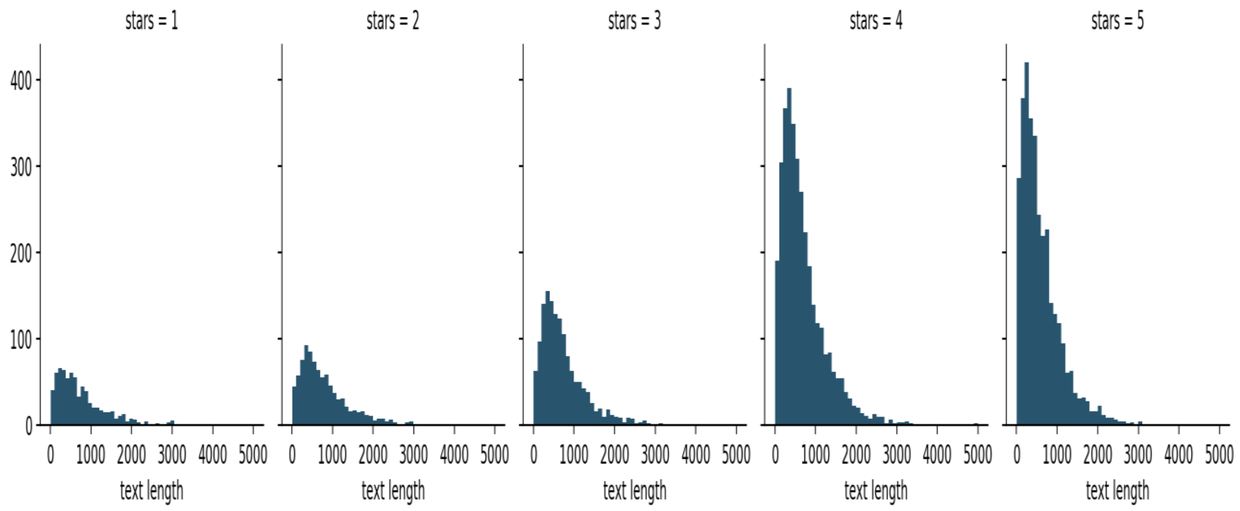


Figure 8 -1 Star and Review Length Correlation

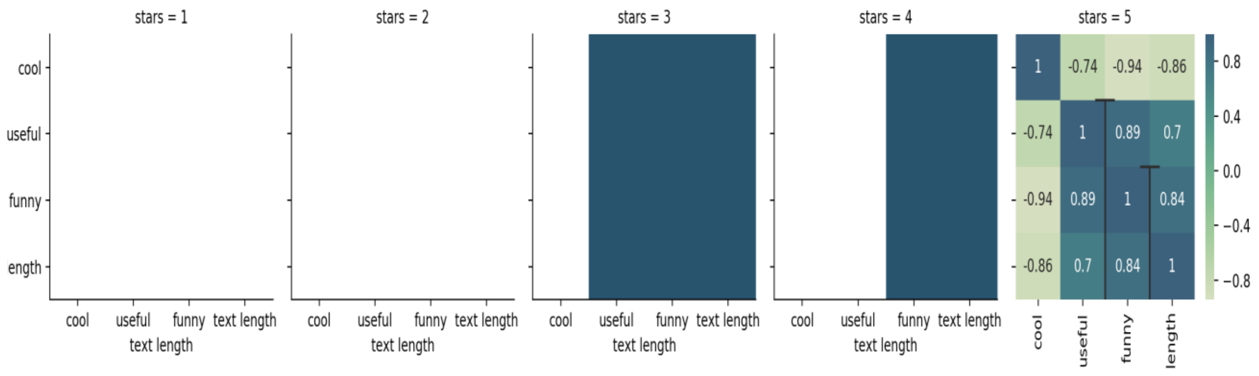


Figure 8 - 2 Feature Heatmap

8.3 Evaluation Results

8.3.1 Precision

Precision Score is the fraction of relevant instances among the retrieved instances, in other words precision is used as a measure of relevance.

$$\text{precision} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|}$$

8.3.2 Recall

Recall is the fraction of the relevant documents that are successfully retrieve

	Precision	**Recall**	**F-1 Score**	**Support**
1	0.87	0.69	0.77	228
5	0.93	0.98	0.95	998
Avg/Total	0.92	0.92	0.92	1226

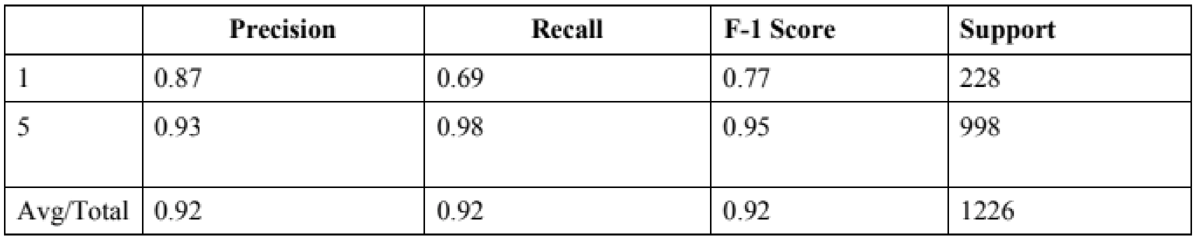
$$\text{recall} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|}$$

8.3.3 F-1 Score

F-1 Score considers both the precision *p* and the recall *r* of the test to compute the score.

$$F_1 = \frac{2}{\frac{1}{\text{recall}} + \frac{1}{\text{precision}}} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}.$$

8.4 Results



	Precision	Recall	F-1 Score	Support
1	0.87	0.69	0.77	228
5	0.93	0.98	0.95	998
Avg/Total	0.92	0.92	0.92	1226

Table 8 - 1 Results

Chapter 9 – Conclusion and Future Work

Mobile devices such as smartphones are diminishing the notion of distance with respect to dining, tourism, and many other such activities. The purpose of this project was to contribute to the community and to make the life of fellow students easier. This project was very challenging since I had no prior experience with swift but the result met the requirements gathered from the original group of students. I learnt the concept of Cocoa pods as well as the basic and fundamentals of iOS programming. In future work I would like to add Facebook integration for sharing posts and images while also adding additional machine learning technique to predict the rating of a location depending on past data.

References

1 Google, *Firebase Documentation*, January 11, 2018

https://firebase.google.com/docs/ios/setup

2 Google, *Firebase Storage Documentation*, January 11, 2018

https://firebase.google.com/docs/storage/ios/start

3 Google, *Firebase Database Documentation*, January 11, 2018

https://firebase.google.com/docs/ios/setup/

4 Facebook, *Facebook Documentation*, February 15, 2018,

https://developers.facebook.com/docs/swift/getting-started/

5 Apple, AVFoundation, *Apple Documentation*, March 15, 2018

https://developer.apple.com/av-foundation/

6 Facebook, Facebook API , Facebook SDK for iOS , January 15, 2018

https://developers.facebook.com/docs/ios/

7 Facebook, Instagram API, Authentication, April 15, 2018

https://www.instagram.com/developer/authentication/

8 Wikipedia, Precision And Recall, Precision And Recall, August 3 2018

https://en.wikipedia.org/wiki/Precision_and_recall

9 Canopy, Documentation, Welcome to Enthought Canopy Documentation , January 4 2018

https://docs.enthought.com/