Victor City Square App

(Project Report)

Group Members

Rabeea Sehar (2286916)

Rafique Nazir (2523462)

Contents

1. Introduction

1.1. Motivation

Travelling is very common in around the world especially in Europe. It is natural when you go to a new place you do not know each and every place of new city especially places like restaurants/bars etc. VictorSquare will not only help tourists to find a restaurant/bar of their own choice but also gives a choice to search restaurant/bar according categories i.e. Breakfast, Lunch or Dinner.

Furthermore, it targets local people as well. VictorSquare allows you to add friends and keep them updated where did you eat (check-in) and which place you liked the most (rating). You can easily choose a place to eat based on its rating and upload pictures to let other people know about that particular place.

VictorSquare is an Application that makes it possible for people to choose a place and socialize at the same time. Now you do not have to go to another platform to check-in or to upload pictures. You can do it all with this App.

1.2. Overview

The objective of this document is to explain the Victor City Square App—an android based application. The document explains all the components of this app in detail. It also discusses the issues faced during the design and development phase of App along with the steps taken to resolve them. This document can serve as a user guide and a technical guide for developers.

1.3. Goals

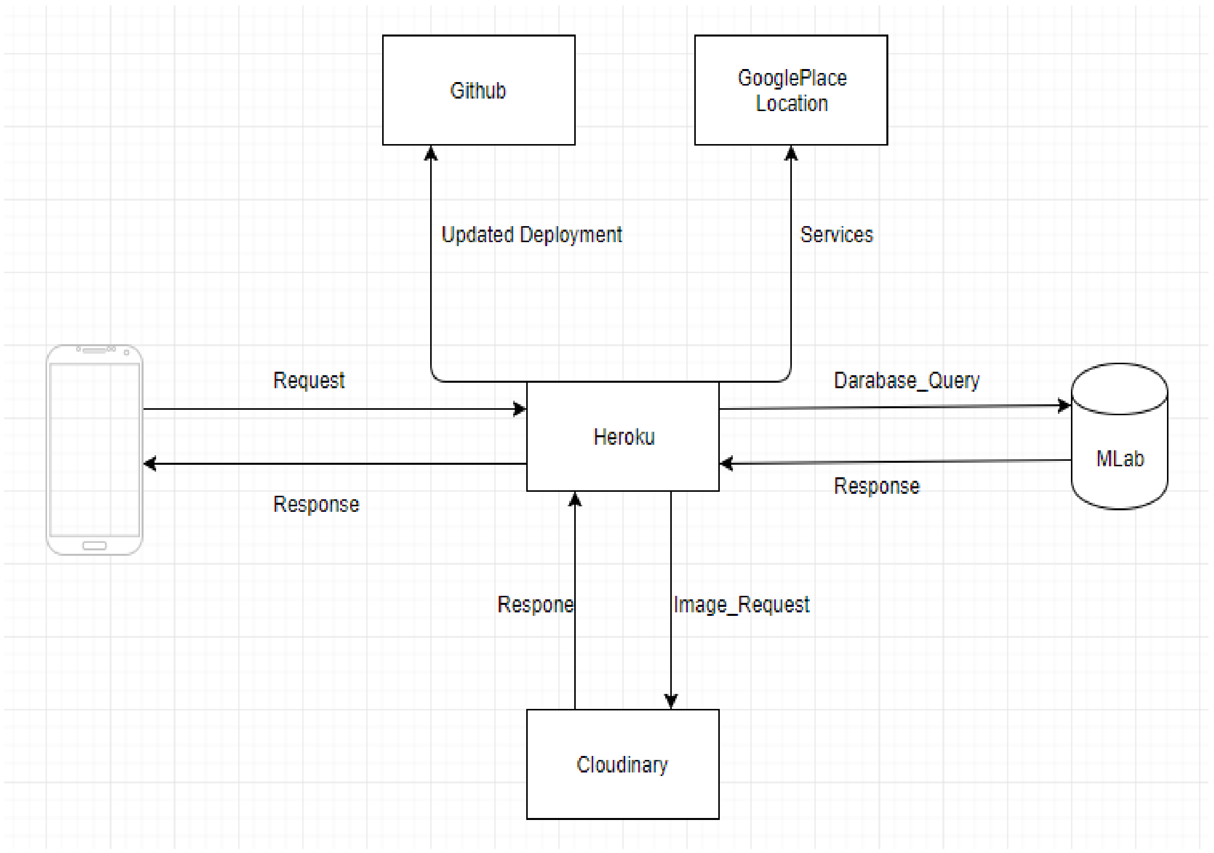
Any **User** of this app can:

- Register to the system by giving relevant information
- Log in to the system and View/Edit personal details. (User Profile Management)
- Delete account from the system
- Search for places by categories (breakfast/Lunch/Dinner)
- Search places using place's name (e.g KFC)
- Add friends by sending friend request
- Upload pictures
- Check-in at bar/restaurant
- Chat with its friends
- Rate the place

2. Architecture

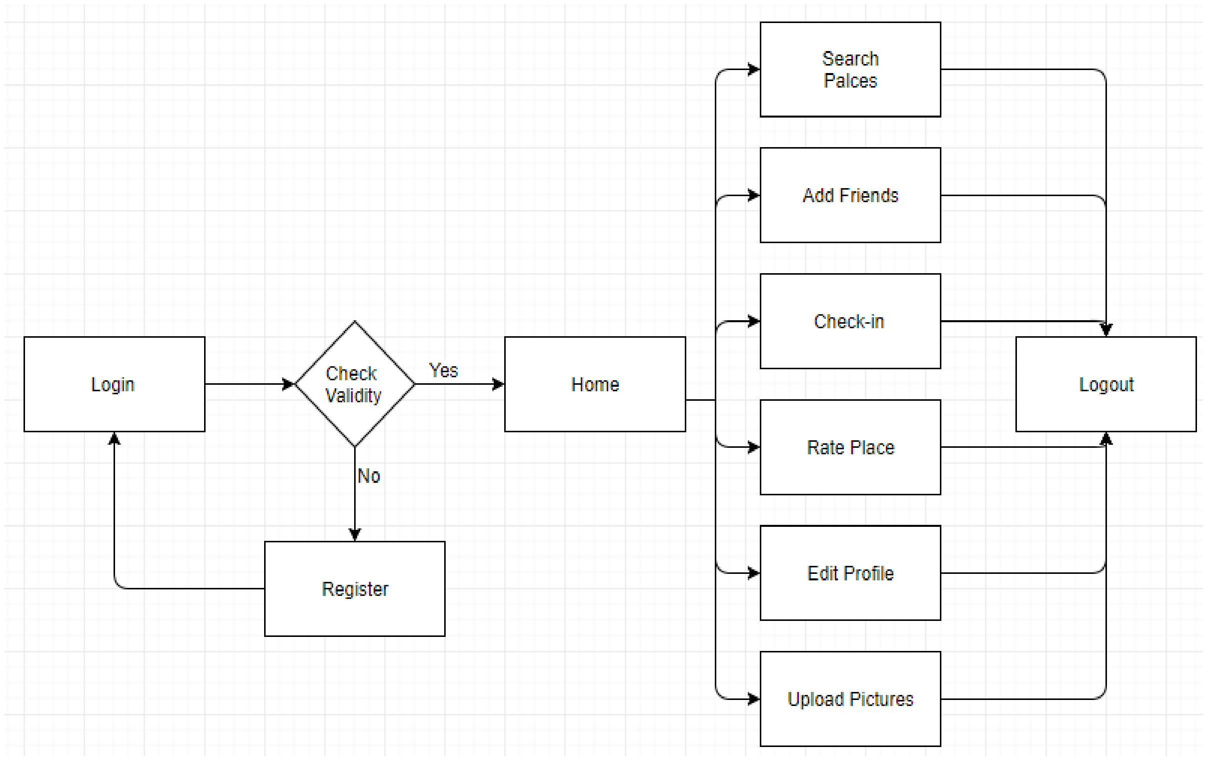
2.1. Overall Architecture

Following is the Overall Architecture Diagram-



2.2. Flow Diagrams

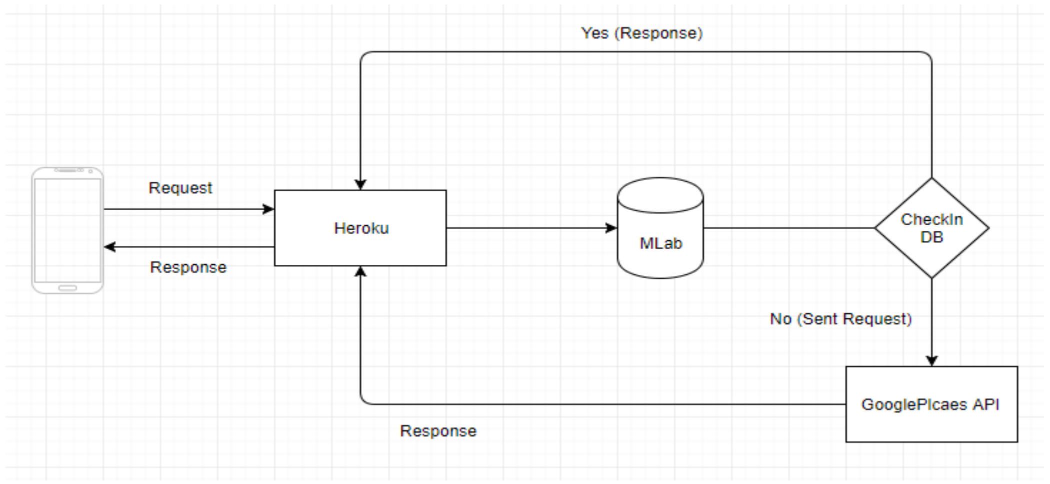
2.2.1. Overall Flow Diagram



- User needs to register in order to get application's services.
- Once user is logged-in followed by the registration process, Home page allows to search place by category and place's name, add friends, check-in, rate place, edit profile and upload pictures.

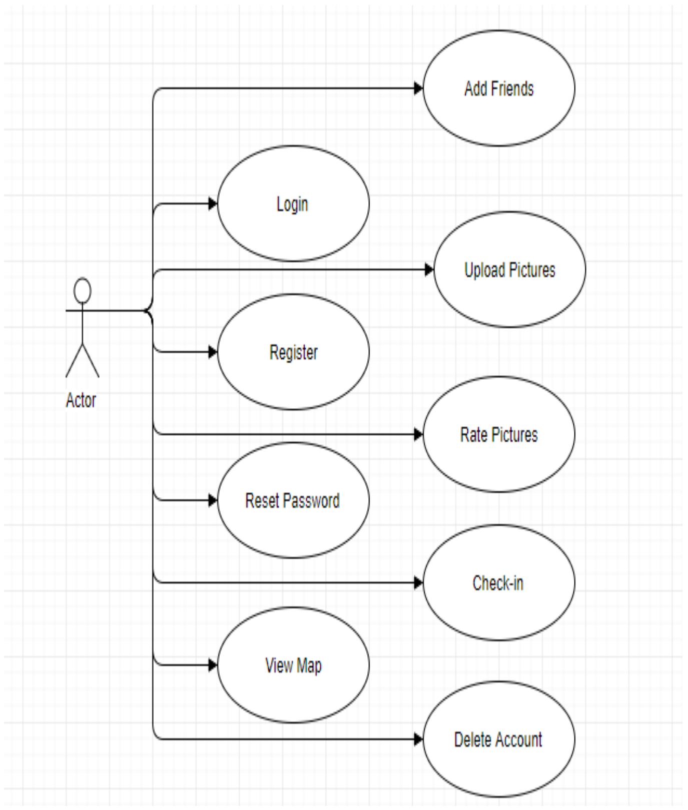
2.2.2. Cache/Data Flow Diagram

Following is the Cache Response Flow Diagram-



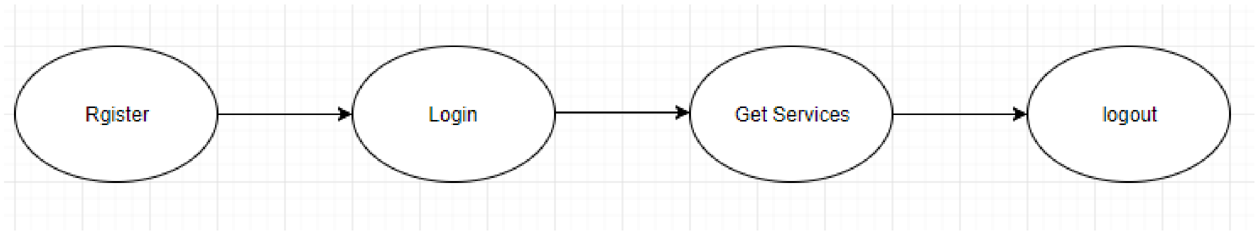
2.3.Use Case Diagram

Following are use cases Diagram-



- User can perform all these tasks mentioned in the use cases.

2.4.State Diagram



The state diagram of App comprises of 4 stages which are as follows –

- **Register Phase:** In this phase, user needs to give his/her information like:

    - Name
    - Email Address
    - Set a password
    - Age (12-110)years
    - Mobile Number
    - City
    - State
- **Login:** In this phase, User get authenticated and processed to the home page.
- **Get Services Phase:** In this phase, User can get all services of App which are as follows:
    - Search Places by category and name
    - Add friends
    - Check-in
    - Rate the place
    - Edit user's profile
    - Upload Pictures
    - Chat with friends
- **Logout Phase:** In this phase, User is logged out.

3. Project Component & Code Structure

3.1. Libraries used

- **heroku:** We used it to deploy our web-services code written in node.js.
- **Appcompat:** It provided the material color themes, widget tinting.
- **Google Play-services:** Provided core functionality like authentication to your Google services.
- **Android Material Design:** Material design is a comprehensive guide for visual, motion, and interaction design across platforms. We used it for new widgets.
- **Vector-drawable:** It is a vector graphic defined in an XML file as a set of points, lines, and curves along with its associated color information. We used it to scale the image without loss of quality.
- **Constraint-layout:** It is the layout that we used.
- **BottomNavigationView:** Represents a standard bottom navigation bar for application.
- **Circleimageview:** Shows an image view surrounded by a circle.
- **Glide:** Since it is a cache library for android we used it for loading images.
- **Cardview:** A FrameLayout with a rounded corner background and shadow.
- **Vertical-Intro:** Vertical intro allows you to integrate material vertical intro to your app.
- **Tastytoast:** Android library which helps developers to get rid of native Android Toast
- **Greendao:** we used it tomap objects to SQLite databases
- **Database-sqlcipher:** It is a plugin to SQLite that provides full database encryption.

- **Customactivityoncrash:** Allows launching a custom activity when your app crashes, instead of showing the hated "Unfortunately, X has crashed.
- **Butterknife:** It is a light weight library to inject views into Android components.

3.2. Front-end

- **Google Maps API:** This library provides an easy way to display a map using the Google Maps JavaScript API. A web page or application displays a Google Street View Image API Panorama using the Google Maps JavaScript API.
- **Google places API:** We used it to return a list of places based on a user's location or search string, return detailed information about a specific place, including user review and supplement data in Google's Places database with data from your application.
- **Google Volley:** It is a library that makes networking for Android apps easier and most importantly, faster. It manages the processing and caching of network requests and it saves developers valuable time from writing the same network call/cache code again and again.
- **Google-gson:** Gson is a Java library that is used to convert Java Objects into their JSON representation. In other words, it is used to convert a JSON string to an equivalent Java object.
- **Picasso:** For downloading images from given URL and URL points to location of images in CDN(Content Delivery Network)

3.3. Services

Node Js

- **Resify:** A Node.js web service framework optimized for building semantically correct RESTful web services ready for production use at scale. Restify optimizes for introspection and performance, and is used in some of the largest Node.js deployments on Earth
- **Googleplace.js:** We used it to return a list of places based on a user's location or search string, return detailed information about a specific place, including user review and supplement data in Google's Places database with data from your application.
- **Passport.js:** We used it for authentication middleware for Node.js.
- **Bcrypt:** Used for password encryption and decryption before saving password in database.
- **Client-session:** Used for session management on client side
- **Cloudinary(CDN):** cloud service for storing images.
- **Googlemaps:** Used to get directions and route to the searched places.
- **Inspect:** Used for debugging node.js code in chrome
- **Mongoose:** Used for persistent storage.
- **Nodemailer:** Used for sending registration email,forgot password email.We are using gmail service for sending and receiving emails.
- **Passport:** Used for session management

- **Passport-local:** Used for session management on server side
- **Request:** Used to make http calls. It supports HTTPS and follows redirects by default.
- **Socket.io:** Used for chat

3.4. Database Schema

User Schema for mongo DB: {
username: {type: String, required: true},
email: {type: String, required: true},
password: {type: String, required: true},
hashKey: {type: String,required: true},
age: {type: String,required: true},
number: {type: String,required: true},
state: {type: String,required: true},
city: {type: String,required: true},
imageURL: {type: String},
verificationCode: {type: String},
friends: [{type:String}],
comments:[{type:String}],
checkIns:[{type:String}],
images:[{type:String}]
}
	Location Schema for MongoDB{
	location: {type: [String]}, // [Long, Lat]
	radius: {type: String},
	keyword: {type: String},
	opennow: {type: String},
	geometry: {type: Object},
	language: {type: String},
	icon: {type: Object},
	id: {type: String},
	name: {type: String},
	opening_hours: {type: Object},
	photos: {type: Object},
	place_id: {type: String},
	price_level: {type: String},
	rating: {type: String},
	reference: {type: String},
	scope: {type: String},
	types: {type: Array},
	vicinity: {type: String}
	}

4. Manual Instructions

This guide will help developer to run the project in your local system, and explains project structure to enable developer to add maintain and contribute to the project.

4.1. Developer's Guide

Following are the steps for a developer to follow to setup the environment for the project

Setting up the Git repository

1. The latest version of the code is present at the repository present at the following link:
   https://scm.informatik.tu-darmstadt.de/projects/iptk-ss2017-team-victor/repository
2. After getting the rights the repository, clone the project code present in the *master* branch.

```
git clone ssh://git@scm.informatik.tu-darmstadt.de/iptk-ss2017/iptk-ss2017-team-victor.git
```

3. Now the code repository is all setup

Setting up the Developer Environment

1. *Download and install Android Studio (2.2.3).*
   - I. Open and run the project in Android Studio which will install all the dependencies via gradle
2. *Setting up Node JS Services on Heroku*
   - I. For initial steps of setup on heroku please follow following link
     https://devcenter.heroku.com/articles/git
   - II. Code is deployed on heroku with pre installed node js.Following is the up and running link
     https://dashboard.heroku.com/apps/team-victor
   - III. For Heroku code needs to be deployed on Github repository.Following is the repository github url :
     https://github.com/victorcitysquare/victorcitysquare.git
   - *IV. To setup and deploy code on heroku*
   - V. Push the updated code in repository, if you have automatic deployment activated from server it will automatically deploy the changes. (**NOTE:** Otherwise login into Heroku Server , navigate to deploy tab and click deploy app.)
   - VI. You can check logs on website itself or you can run following command from command line
     heroku logs --app <AppName> --tail
   - VII. How to restart the app
     heroku restart --app <AppName>

Note: For above commands to work you need to be logged in via command line on local machine

3. *Setting up Mongo database on mLab*
   - I. Signup for an account of mLab and login
     https://mlab.com/
   - II. Create new database with new user and password for user.A link to connect with database would be given. Use that link in config file of your service to connect which should be something like
     "mongodb//<user>:<password>@ds159963.mlab.com:59963/<databaseName>"

III. Setting up cloudinary (Content delivery network for images uploading and downloading)

IV. Signup an account and login

http://cloudinary.com/

V. Cloudinary will give you following details to be used inside node js code

cloud_name: <Number for Your app on cloudinary>

api_key: <key to use cloudinary api>

api_secret: <password for cloudinary api>

After installing all the dependencies your project is ready to be executed.

4.2.User Guide

Following are the steps for the user to get acquainted with the application.

1. Download and install the application apk.
2. Make an account by filling all required information.
3. Now you are all done and ready to use our awesome app.

5. Summary

This lab has been a learning curve for all of us, through this project we get to know different aspects of development using numerous plugins for node.js and android development. We have integrated functionality of searching places and check-in using google maps API and google places API. We have used MongoDB for backend storage where we are storing user profiles, reviews, comments and check-in for places visited. For backend services we are using node.js with plugins like restify for services, bcrypt for encryption and decryption and passport.js for session handling. For android we have used different plugins like butterknife, volley, Picasso for multipurpose. Though this application is complete yet there is always some space available for enhancements which can be our future work.

6. Future Work

In this section, we have highlighted certain features which can be implemented for the future releases to make this app more User friendly. Application can show restaurants/bars on priority which have coupons for Breakfast/Lunch/Dinner for users. In this way, user can easily decide the where to have next meal with friends.

7. References

- https://scm.informatik.tu-darmstadt.de/projects/iptk-ss2017-team-victor/repository
- https://developers.google.com/places/web-service/
- https://mongodb.github.io/node-mongodb-native/
- https://developer.android.com/design/material/index.html
- https://github.com/armcha/Vertical-Intro