Bilkent University
Department of Computer Engineering

Senior Design Project

High Level Design Report

Urbscope

urbscope.com

Group Members

Burak Mandıra
Mustafa Motani
Waqaas Rehmani
Syed Sarjeel Yusuf
Abdullah Wali

Supervisor

Mustafa Özdal

Innovation Expert

Veysi İşler

Jury Members

Selim Aksoy
Mehmet Koyutürk

Table of Contents

1. Introduction

As of 2018, people are working on different industries from transportation, accommodation to engineering, medicine and science. Nonetheless, everyone at some point of their life is a tourist when they go for holiday regardless of whether it is to abroad or domestic. As the technology has been developing, the tourism industry has evolved as well as others. The fast and radical changes due to recent developments has changed people's lives and this, of course, affects how people travel and learn about different holiday locations. Therefore, we propose to develop a mobile application that will serve the people's needs when they are on holiday with the solutions that use recent technologies. Recent developments in technologies regarding human perception, such as augmented reality (AR) and virtual reality (VR), have become very popular and are seen as the future of our lives. They can provide more realistic, descriptive and yet simple solutions to the problems that we have had so far.

When people go to holiday, most of them do not know what kind of places are available to visit or which part they should visit for a specific city. They search for museums, galleries, architectural landmarks, historic monuments etc. on the Internet and try to find information as much as they can before their trip. However, this is not efficient and all the information that has been searched is prone to be forgotten when the trip starts. Therefore, some people prefer doing this information retrieval phase throughout the trip. Nevertheless, within a short time, this can be quite cumbersome because checking everything that surrounds you on the Internet takes a lot of time and finding the decent information can become quite time consuming.

Also, one might not want to spend his/her time by searching and filtering things on the Internet while he/she is visiting some venues that he/she should make the most of them. Most of the time, it is very crucial to ask the right questions to the search engines when one wants to elicit the information. This becomes quite challenging when tourists even do not know the name of the buildings. What is worse can occur when they are unaware of the popular tourist attractions and they skip those attractions as they are just passing by.

1.1. Purpose of the System

Urbscope is to be a mobile application that is aimed at running on cross platform devices. It is aimed at providing tourists and travellers alike travel and tourism services in an augmented reality environment. This augmented reality platform is to be coupled with a basic recommender system that is aimed at making the product more personalised and target the user directly, rather than providing general information. The aim of the product is to put all the touristic information into the palms of the user.

Urbscope incorporates augmented reality by having AR markers indicating points of interest that are associated with tourism. These AR markers will be presented in the two modes of the application. These two modes include the Exploration Mode and the Recognition Mode of the system. The Exploration Mode displays the AR markers in two views. The first view is the Nearby View which allows the user to see all points of interests around him, according to the filters that the user has entered. Similarly the second view, which is the Recommendation View, presents the user will all landmarks that the recommendation engine generates according to the previous travels of the user. The second Mode is the Recognition Mode and is meant to be the initial Mode of the application, and is aimed at using the Cloud Vision API to detect a landmark in the view of the user, via the camera view of the system. Therefore the Recognition Mode the first activity of the application as the user activates the application and hence allows the user to detect the touristic landmark in front of him/her immediately. With the recognition of the landmark, the user is also presented with relevant information of the landmark in front of the user.

Additional features of the map also include a navigation system that allows the user to navigate to the desired point of interest. The recommendation system which is a secondary feature to the AR identification and building recognition is also added to provide personalized services to the user.

Hence, it can be seen that Urbscope aims to solve all the issues that the modern travellers and tourists face when on a thrilling adventure. Landmark detection, personalized recommendations, navigation, and landmark information retrieval are all aimed at revolutionizing the tourism and travel industry.

1.2. Design Goals

We have split our design goals into those for the client side system (the phone application) and the server side system (Node server).

1.2.1. Client Side Goals

Accuracy of Landmark Detection: The application's main purpose is first and foremost the detection of touristic and historical landmarks, and thus our top design goal is to maximize the accuracy of the landmark detection feature. This mainly involves sending accurate information to the landmark detection server, and ensuring that the photo taken by the user is blur free and covers the landmark properly.

Accuracy of AR markers placement: One of the features of the application is having AR markers on landmarks that are detected in the camera view, however, we want to ensure that the application places those markers accurately in order to avoid confusing the user by placing a marker for a

certain landmark in the wrong place. Achievement of this design goal involves accurate mapping of the coordinates from the user's phone camera to the underlying 3D plane in which the 3D objects are placed. Several techniques exist to achieve this goal, some of which rely on a technique called raycasting [1] that projects a ray in the direction that the user is looking at from the camera and checks for objects intersecting with this ray to determine the coordinates at the intersections.

Accuracy of Recommendations: One of the feature of the system is to match similar users and then suggest landmarks of similar categories. One of the major problems of the recommendation is that if all landmarks in the world are considered, the matrix would be too sparse to have any real collaborative filtering done. Thus the dimensions of the matrix are reduced to the categories that the landmarks fall into, as listed by the Foursquare API [2]. However, the reduced dimensions of the model may lead to ineffective collaborative filtering, and hence the algorithm, slope one and gradient descent chosen is aimed at solving the issue.

Responsiveness: The application must be responsive and smooth to use. In order to achieve this we should minimize the use of the main GUI thread and use asynchronous threads whenever possible in order not to block the UI while long processes are running. We will also include indicators when a long running process is being executed (such as detecting the landmark) so that the GUI wouldn't look clunky or unresponsive and to inform the user that such a process is running in the background.

Minimize Bandwidth Usage: The user will almost certainly be on mobile data while using Urbscope, therefore we should ensure that the application isn't bandwidth heavy and doesn't burn through the user's data limit, otherwise users will stop using the application especially since it's a tourism application and the user might be on data roaming mode. Thus, we will ensure that the requests to the server and the responses are minimal in size, and perform compressions where possible while maintaining a high accuracy of landmark detection.

Battery Usage: Similar to mobile data, the phone battery is an important resource for a traveller and thus we should ensure that our application does not drain the phone's battery while running.

1.2.2. Server Side Goals

External API Cost: We want to keep cost of using external APIs under control. Therefore we will try to limit the number of requests that the server will send to external APIs. We can potentially cache some of the results and reuse them when duplicate requests are received from users.

Scalability: We will design the server using scalable data structures and algorithms whenever possible in order to be able to support a high number of users.

Reliability: The server should restart automatically in the case of a failure and report any errors to the maintainers via log files.

1.3. Definitions, Acronyms and Abbreviations

- API - Application Program Interface
- AR - Augmented Reality
- Client - Mobile device that shall communicate with external elements i.e server
- Render - automatic process of generating a photorealistic or non-photorealistic image

- Server - Online centralised resource for training set and mode of communication with APIs
- Slope One - Algorithm for recommendation system
- Spider - an Internet bot that will systematically browse TripAdvisor to collect Training Data
- Training Data - Pre collected data that shall be used to initialise the recommendation engine
- Venues - Locations returned by the FourSquare API

1.4. Overview

Urbscope shall be implemented with a client-server type architecture that shall also incorporate aspects of a three tier architecture. Even though there is a server being used, its main purpose is simply to communicate with the external APIs being utilised, and also to store the trained recommendation engine. Thus, the client's UI shall rely greatly on the response of the server and hence there is a need for the properties of a server-client architecture such as centralised data management in terms of the recommender system.

Moreover, the rise for a three tier architecture comes from the fact that there are three major communicating components which can be visualised in a three tier manner. The front component is the mobile application that shall be the only entry point for the user. The middle layer includes processing components that work on encoding pictures for the Cloud Vision API and the log file system management which is necessary for the recommendation system. The mid tier then communicates with the web server which can be considered the outermost layer of the system.

As it can be seen, there is an absence of a concrete database system. This is because Urbscope requires no such large data storage and processing, as shall be discussed in Section 3.4. This greatly affects the choice the systems architecture.

2. Proposed Software Architecture

2.1. Overview

Implementation is planned on dividing the system into components such that each of these component will be self-standing and thus will be able to implement them on their own, module wise.

2.2. Subsystem Decomposition

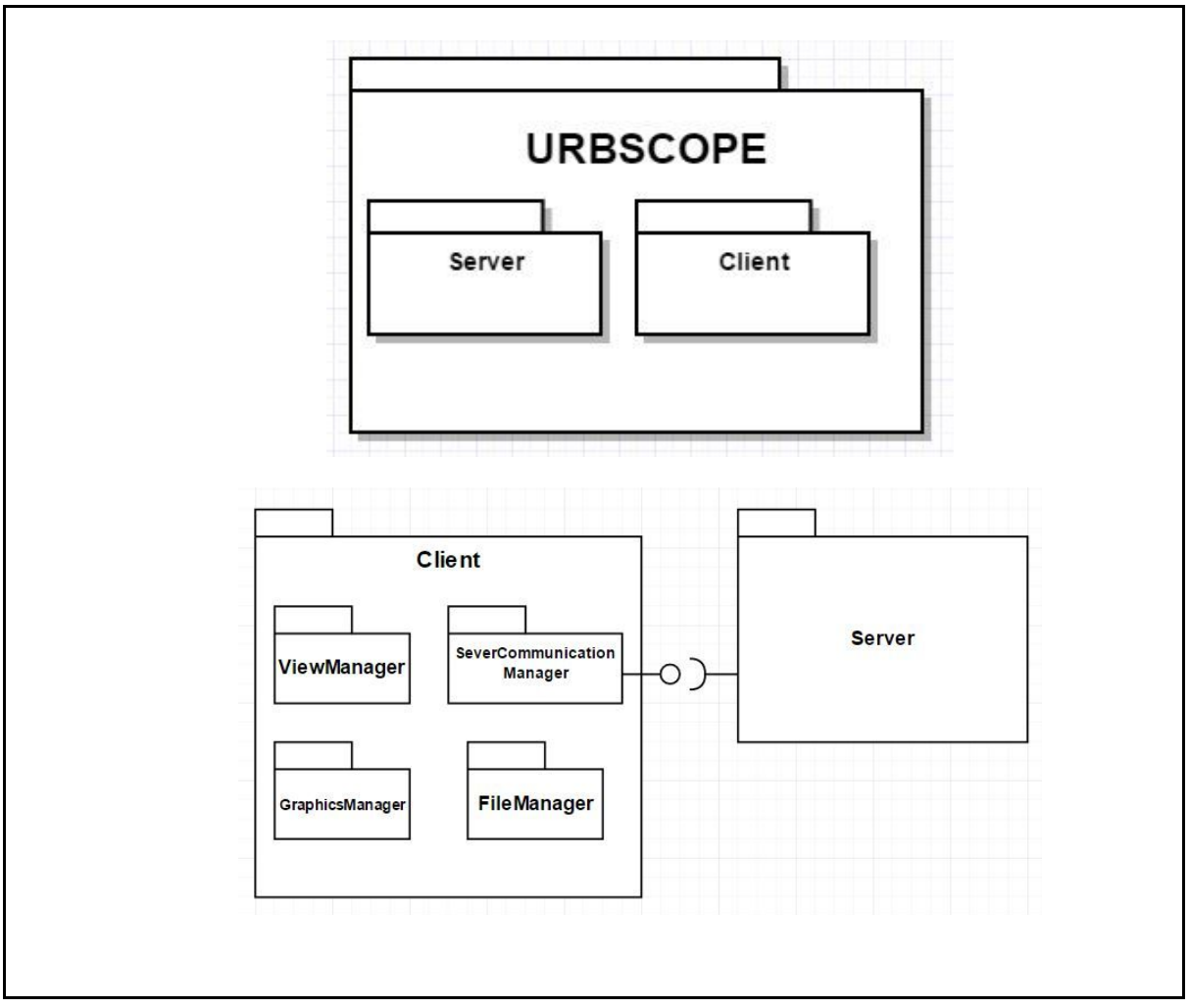

Fig 3.1

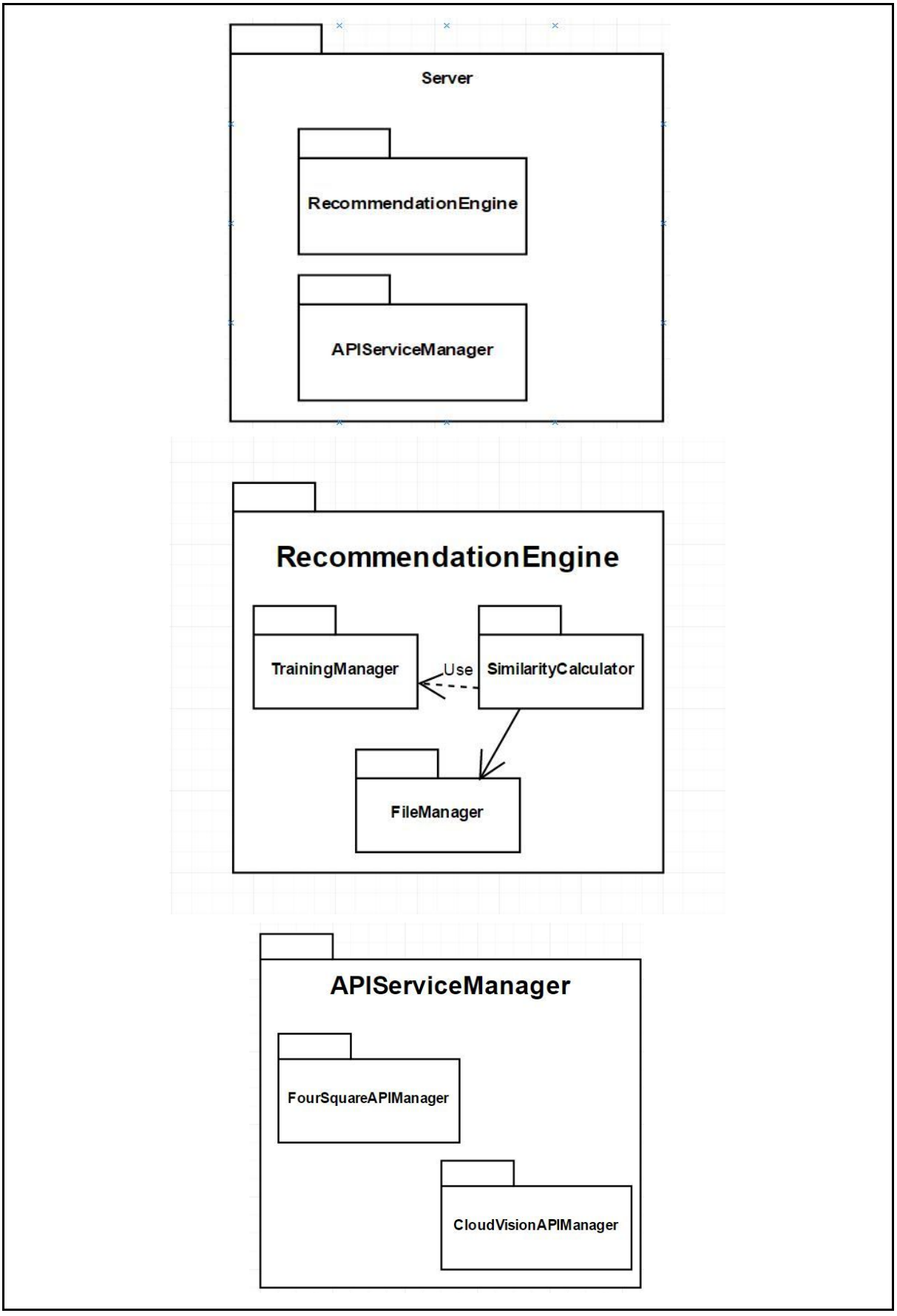
Server
RecommendationEngine
APIServiceManager
RecommendationEngine
TrainingManager
Use
SimilarityCalculator
FileManager
APIServiceManager
FourSquareAPIManager
CloudVisionAPIManager

Fig 3.2

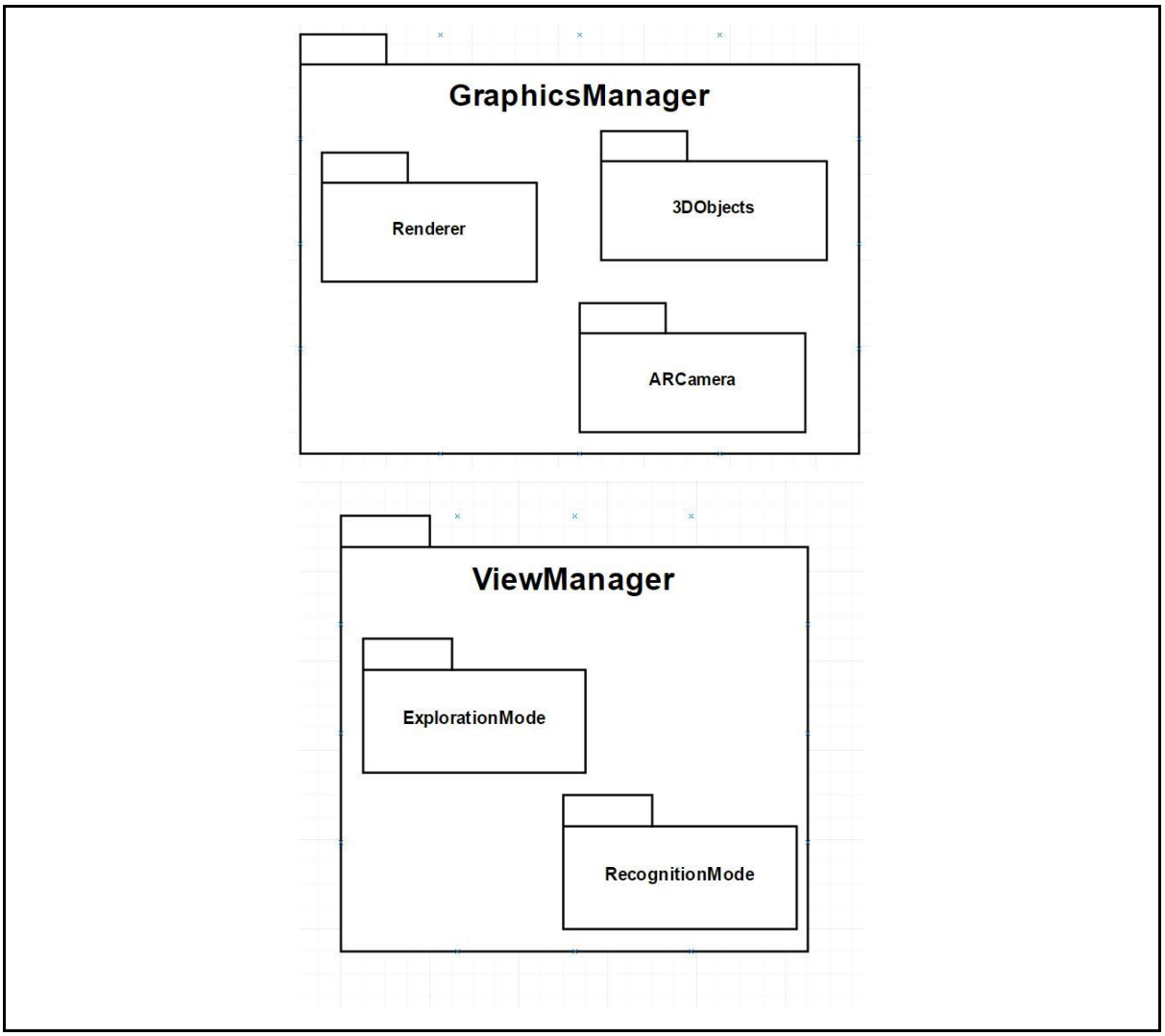

Fig 3.3

As it can be seen from *Fig 3.1* to *Fig 3.3*, the system is viewed in terms of a server client sub-decomposition. By magnifying the server in *Fig 3.2*, we see that the server is responsible for holding the recommendation engine and the APIService Manager. The later of the two components simply contains the Foursquare API and the Cloud Vision API. Recommendations Engine consists of the Training Manager which periodically creates a training matrix. The Similarity Calculator is the basic algorithm that finds similar users and the File Manager is aimed at resolving how to store the training dataset onto the server.

The client subsystem is more elaborate in terms of decomposition. The difference between View Manager and Graphics Manager is that Graphics Manager is solely dedicated to the AR component of the system. This can be seen in *Fig 3.3* in the sub decomposition of Graphics Manager, which is created according to the functioning of the Expo AR. Similarly, View Manager can be seen to have the two modes of the system.

2.3. Hardware/Software Mapping

In this section we will talk about how different components in our system will communicate with each other. Servers will act as intermediaries between client and all the APIs that our app will need. Server will fetch information from FourSquare API in JSON objects which will be unwinded at the server and passed down to client in a compatible format. We are currently deciding on which format to use with possible candidates being XML, Text, HTML and protocol buffers. Protocol buffers provide a way of serializing structured data chunks in a particular fashion. They are much smaller than XML files and it is faster to generate them. Our choice should favor performance and time-efficiency. In case of CloudVision API, client will upload an image to server which in turn will send it to CloudVision API in Base64 encoding and get the results of the identified landscape. Following is pictorial representation of mapping:

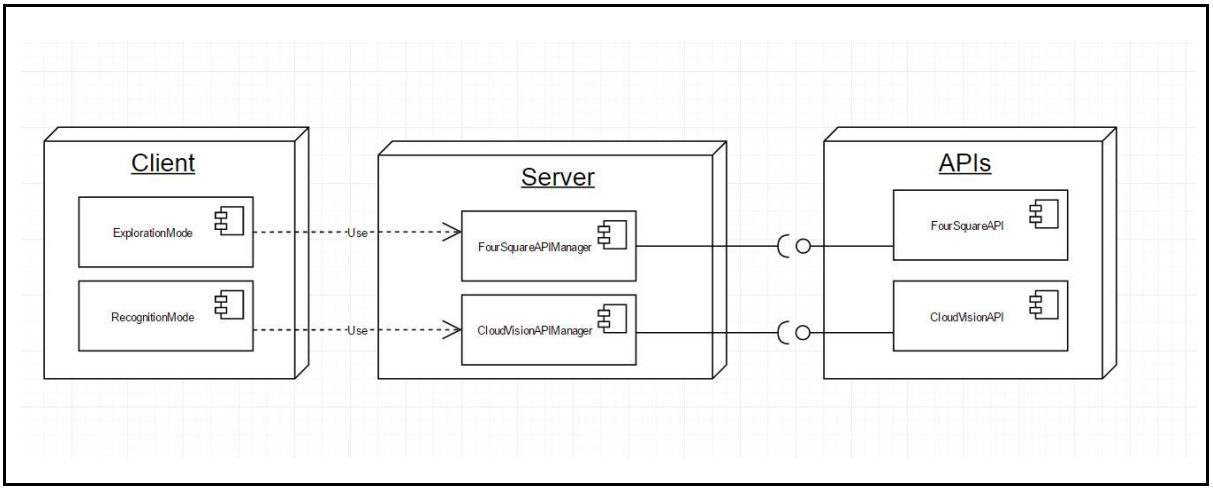

Fig 3.4

2.4. Persistent Data Management

In Urbscope, we have two different kinds of data: internal and external. The following sections describe their meaning and how they are going to be managed in the app.

2.4.1. Internal Data

Internal data refers to the data that will be used in order to train recommendation system. There are two types of data in this category. One is going to be stored in the client-side and the other in the server-side. Client-side data will be stored in the user's devices locally. Therefore, it should not be a huge dataset. Urbscope will store a log file locally in order to keep track of which places the user visited and his/her ratings regarding those places. The content of this log file will be very simple: it will contain several category names for places such as museums, nature & parks, sights & landmarks etc. along with the average rating of the user for each of them. Storing categories in lieu of the actual names of the visited places will significantly decrement the size of the local file. It will also ease recommending places to users since the recommendation system doesn't have to

check thousands of places but it should only consider few category names when to make a new recommendation to a user.

The content of the server-side data will be collected from TripAdvisor [3] with a spider script that is written by us. This script basically will collect the places that several users have visited and their ratings. At the end, we will have places and ratings for several users that the script processed. However, in order to eliminate having a biased database for training, collecting this data should be considered thoroughly.

First of all, the script will not collect places and their ratings randomly. Instead, it will collect fixed number of places from each users' various starred ratings, e.g. 20 places that are 1-starred, 20 places that are 2-starred etc. up to 5-starred places, per user. The reason why it will collect the data in such a manner is to provide balance between preferred and disapproved places. It will hinder having a data set whose significant percentage is ranked positively or vice versa.

Another point to consider is that normalization of the collected data. Normalization is a standard of the slope one algorithm that shall be implemented by the recommender engine to standardize the ratings that different users give. This is because different users have different scales to rate certain venues, and hence the data needs to be normalised to have accurate predictions. This shall yield a trained similarity matrix. The matrix shall be re-trained periodically with new data appended to the dataset.

2.4.2. External Data

The significant amount of data Urbscope utilizes is external data. This data is in essence provided from two API services: Google Vision API and Foursquare API. This data will not be stored or handled by Urbscope. Therefore, it is called external data. Vision API will provide all necessary information about the detected landmark such as longitude, latitude and the name of the landmark when Urbscope is running in Recommendation Mode. Foursquare API will provide longitude, latitude and names of the nearby recommended places when Urbscope is running in Exploration Mode.

2.5. Access Control and Security

Urbscope does not have a designated user system with seperate accounts for each user. Instead, each device is considered as a separate client of the service; hence, there are no elaborate subsystem to regulate the authentication of users on a device. There is, however, a designated separation between the server and client which exists due to the client-server architecture paradigm. The characteristic of separation is thus utilized to ensure there is restricted access to the server, limited to only the admins of Urbscope. Even though the server only holds the training set, it is still considered sensitive data and it should not be accessed by a client device. Hence, the natural barriers of access that arise due to the manner in which the system is to be employed result in no need for extra implementation of the access controls.

Other security concerns that need are acknowledged is the ratings log of the user stores in the individual's device. This cannot be accessed by any external entity without gaining full control of

the user device. Again, the barriers that naturally arise by employment leads to no need of additional implementations of access control.

2.6. Global Software Control

2.6.1. Recommendations Training Set

As mentioned before, ML models will be trained on the server; the initial training is not a function of the system. However, as the user base of the system increases, this training will be done in periodic cycles, mostly whenever the number of new user ratings increases above a threshold.

2.6.2. API services

It can be seen through the description of the system that the functionality relies heavily on external APIs. This translates to the viability of data accessed from these APIs. Such things need to be considered such as new categories being added to the Foursquare venue descriptions, and how the recommendation system will deal with such additions. The system is to continuously check the categories listed in the JSON file returned by the venues, and if a new category is detected, then the list of categories is to be updated on the server. This should thus update the manner in which the log files of the user should be stored.

Another concern is the manner in which the Cloud Vision detects images. Any changes in their protocol would lead to the need of a system update of Urbscope. However, at the moment such a drastic overhaul is not expected.

2.7. Boundary Conditions

2.7.1. Initialization of Main Server

- Main server will be running at all times except during offline maintenance. Moreover, server might not be able to perform desired functionalities if Cloud Vision API or FourSquare API is down.
- When main server goes online, it will establish connection with FourSquare API and Cloud Vision API and wait for user requests.
- When application requests identification procedure, it gets an image from the user and encodes in Base64 before passing it to Cloud Vision API. It retrieves details of the objects from API and transmits information in the organized manner to the user.
- When application requests nearby places to recommend the user, it makes a JSON object with longitudes and latitudes of the user and passes it to FourSquare API which returns all the venues in the desired radius of the coordinates. Then the server filters the venues with desired categories and sends the data to the user.

2.7.2. Initialization of Client

Application running on mobiles will be the client.

- Upon initialization app establishes connection to the server.

- In ‘Exploration Mode’ it sends user’s log file stored on his mobile to get customized recommendations.

2.7.3. Termination of Client

- Users can voluntarily close the app.

2.7.4. Termination of Main Server

Servers will be offline for maintenance e.g. training the updated recommendation matrix.

- Users will not be able to access the functionalities of the app.
- Users will get the notification message.

2.7.5. Failure/Exception Handling

There can be several cases whereby client and servers may fail to perform their functionality.

- User experiences power failure. In such a case connection to main server is dropped.
- User experiences internet connectivity failure. Several problems may arise from such a scenario. If the server is trying to retrieve log file of the user it will attempt for specific number of times before terminating the connection. On the client side user will get error notification stating no internet connection.
- APIs on which main server is dependent may be down. After several attempts of data retrieval server will relay an error message to the user.

3. Subsystem Services

Urbscope can be broadly divided into two main subsystems as mentioned below:

- Server-side Components: It mainly acts as an intermediary between client side and desired APIs. It also does number crunching to determine recommendations for each user.
    - API Manager: It handles all the communication with both Cloud Vision and FourSquare APIs. It retrieves the query from the client side and communicates with the correct API accordingly.
    - Recommendation Engine: Server obtains the log file from the client side and uses it as a training data to ascertain preferences of that and thereby recommend places that fall under those preferences. It obtains those places from FourSquare API.
- Client-side Components:
    - User-Interface: Its basic functionality is to provide graphical user-interface and allowing user to switch between Exploration Mode and Recognition Mode.
    - File Manager: It holds the log files of places that user has visited in the past.
    - Graphics Manager: It is responsible for rendering image, camera interface and AR pointer on the locations.

4. Glossary

AR Marker: This is basically a bubble on the screen. However, this bubble is on the AR layer and it is stationary with respect to users. Users have to turn towards the AR marker in order to see it on the screen.

Exploration Mode: One of the two modes that Urbscope has. This mode is used when users want to see the nearby or recommended landmarks' AR markers on the screen.

Nearby Switch: One of the two options that the switch which is shown at the top of the screen has when Urbscope is in Exploration Mode. It is activated when nearby landmarks are sought to be explored by the users.

Recognition Mode: This is the default mode of Urbscope. It is used for recognising the landmarks that user points out via the mobile phone's camera.

Recommendation Switch: One of the two options that the switch which is shown at the top of the screen has when Urbscope is in Exploration Mode. It is activated when user wants to see recommended landmarks for him/hem.

Expo AR: Expo is a free and open source toolchain built around React Native to develop cross-platform native iOS and Android projects using JavaScript and React. Expo AR is a functionality of it that allows developers to code AR applications.

Base64 Encoding: Base64 is a group of similar binary-to-text encoding schemes that represent binary data in an ASCII string format by translating it into a radix-64 representation.

JSON: It is a language-independent data format that uses human-readable text to transmit data objects which consist of attribute-value pairs and arrays. It is a very common data format used for browser–server communication. It is originally derived from JavaScript and stands for JavaScript Object Notation.

5. References

[1] J. Polvi, T. Taketomi, G. Yamamoto, A. Dey, C. Sandor, and H. Kato, “SlidAR: A 3D positioning method for SLAM-based handheld augmented reality,” *Computers & Graphics*, vol. 55, pp. 33–43, Nov. 2015.

[2] Foursquare, “Get Venue Categories,” *Foursquare Developer*. [Online]. Available: https://developer.foursquare.com/docs/api/venues/categories.

[3] *Scraping Hotel Reviews from Tripadvisor.com | Octoparse, Free Web Scraping*. [Online]. Available: https://www.octoparse.com/tutorial/scraping-hotel-reviews-from-tripadvisorcom/.