Bilkent University
Department of Computer Engineering

Final Report

Urbscope

urbscope.com

Group Members

Burak Mandıra
Mustafa Motani
Waqaas Rahmani
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.

2. Added Features

2.1. Landmark Compass

This was added to reduce the importance that AR-Markers held in the Urbscope application. The feature involves a 3-D pointer implemented in Web GL which is points in the direction the user must traverse to reach the selected destination. This involves the pointer pointing in the direction of the the way that user must go, and then changes accordingly depending on the next part of the road. The pointer direction is directed according to Google Map polygons that are generated, creating a somewhat corridor from the user's current location to the destined landmark. Moreover, if no landmark is not selected, then the pointer points North.

2.2. Sponsor Facilities

The sponsor facility allows the Urbscope to integrate advertisements into the application in a manner that does not disturb the user's travel experience. In exploratory mode, there are two type of markers that are displayed on the exploratory map. These include the normal marker, indicating a landmark, and a golden colored marker that indicates a sponsored landmark. These landmarks can be any entity, including shops, restaurants, car rental services or even schools. This not only integrates advertisements and sponsors into the app in an innovative manner that does not disrupt the user, but also supports local businesses and small enterprises. Moreover, it allows for targeted advertisements, without any overstepping of privacy laws. No user data is needed in displaying the sponsor markers, and instead the user shall simply stumble upon them on the exploratory map.

2.3. Category-Landmark Prediction

The recommender system was initially predicting only top categories that the user would be interested in. However it was decided to enhance the recommender system, and utilize the full power of latent factoring, it was decided that the system shall also provide personal landmark recommendation predictions. This was an obstacle in the earlier development stages of Urbscopoe as there was a lack of data and also the computation needed to ensure that all the recommended sites lie within the specified vicinity of the user.

The addition of landmark prediction initially entailed checking each landmark predicted, out of a database of more than 10,000 points of interest, whether the predicted landmark is in the same city or not. Therefore, it was decided that a comprehensive database be built of ratings, where user ratings would be divided depending on the cities that the landmarks lie. For example, all ratings of landmarks in Bangkok would be in the Bangkok file, all ratings pertaining to Ankara would be in another file, and so forth. Hence all theses files would be trained and separate predictions

generated. However, calculating predictions for all users, for all the cities, would be unnecessarily adding computation load on the server. Why should predictions of a user in Europe be made for landmarks of a city in South East Asia when the user neve intends to visit the Malay Archipelago. Hence a user's shall be added to the city data matrix only when the user actually visits the city. That means Urbscope checks which city the user currently is in, and adds the user to the city file accordingly. However, this presents the cold start problem of the user, as he travels to each city. To overcome the cold start problem, the top categories of the user are predicted initially, and not the landmarks. Urbscope predicts specific landmarks to the user only when the user actually rates a landmark, which entails checking which city the landmark is in and then updating the landmark in the relevant city file.

However, additional problems here too stem up, which were tackled successfully. For example, it is initially impossible to account for all the points of interest that this beauteous world offers in manifold. Hence, once the user rates a landmark that is not in the data file, Urbscope updates the relevant city file with a new column of the landmark. Another hurdle that was manifest was the possibility of the entire city not existing in the data files. Hence once this is the case, Urbscope creates a data file for the city, no matter how small or remote the city is, and populates it with initial landmarks, along with the single user who visited the landmarks. If the user is the only one in the file, then the recommendations that are provided again revert back to category prediction.

3. Final Architecture and Design

The final architecture involves a client-server based system, where the user/individual mobile app is the distinguished user, and the Urbscope server handles all requests. The user makes several requests to the Urbscope server, with API calls. Moreover, the user also makes calls to other APIs which include the following:

- Google Cloud Vision
- Foursquare
- Geo Coder
- Google Maps
- Google Knowledge Graph API

The major issue that thus arises is latency due to varying connection rates between the API calls. Thus the need for efficient API calls is needed which thus reduces the latency of response.

3.1. Subsystem Decomposition

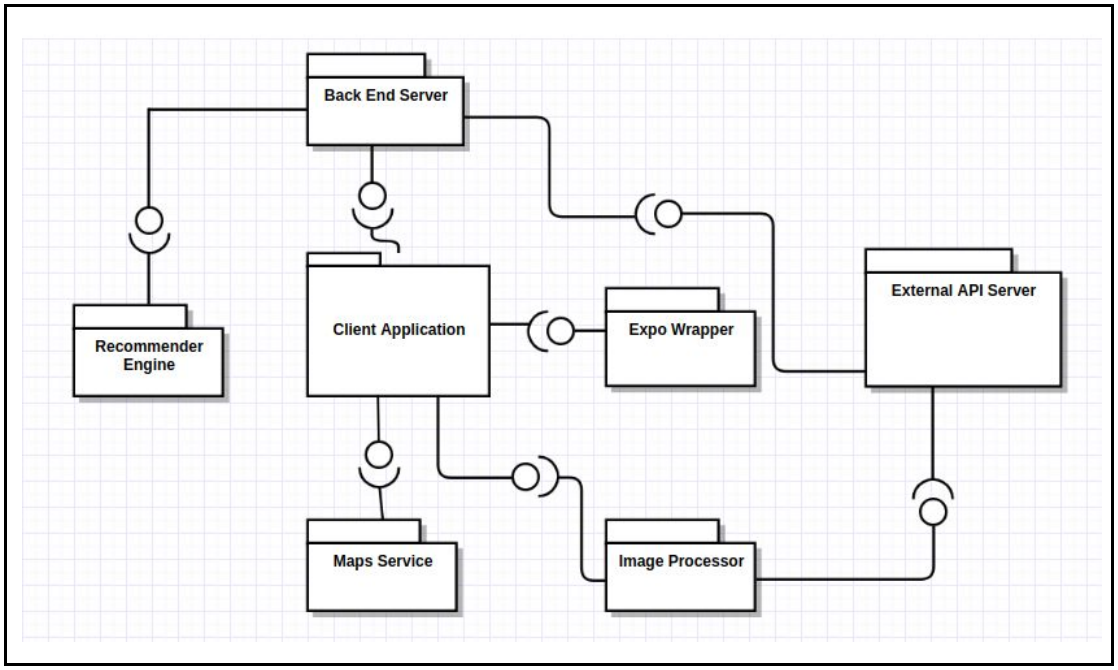

Fig 2.2

3.1.1. Client Application

The Urbscope application on the user's phone shall be the central component of all components in the decomposition. It is built using React Native and makes use of Expo libraries, especially for the UI. This shall be the entry point of the user to access all other components in the decomposition, while using the application.

The urbscope client is connected to the Urbscope server (Backend Server) to access the Recommender engine and procure nearby landmarks. Similarly, the Application component is also connected to the Maps Service component to build the map polygons. The Application is also connected to the External API service module which handles API calls to all external APIs, along with the Image Processor module which relies on the Application for image data.

3.1.2. Recommender Engine

The Recommendation Engine component handles all issues regarding item prediction through latent factoring. There are two modes of the recommender engine. One mode predicts the top categories that the user may be interested in, and the other mode predicts the top specific landmarks that the user may be interested in. The Engine not only predicts ratings that user may give, but also decides when to provide category predictions as compared to landmark predictions, and vice-versa. This is because, as any recommender system, there is an issue of the cold start problem, and the Urbscope system has successfully managed to mitigate the cold start problem.

The engine also trains all the rating CSVs that Urbscope stores and also saves these files in the Back End Server. Therefore, the component is also connected to the Server.

3.1.3. Back End Server

Back End server hosts the Urbscope API. It allows the Application to interact with the Recommender system as well as procure nearby landmarks. It is written as a node.js application that runs on a linux server. Moreover, the Server shall periodically call the training module of the Recommender Engine to generate trained matrices. These matrices shall then be used to predict the top categories or landmarks to the user.

The Server also handles scripts to the Foursquare API where it procures landmark details and returns it in a JSON format. Hence the Urbscope API, hosted by the Server is the most used component with the Application.

3.1.4. Expo Wrapper

The Expo Wrapper is build around React Native that allows the React based Application to use native components such as Audio, Video and WebGL. The UI of the application relies heavily on

the Expo Wrapper component, and would actually be considered as a sub -component of the Application in later illustrations.

3.1.5. External API Service

The External API service allows the application to manage server calls with all the external party APIs. This involves the all the APIs listed in *Section 2*, and is also used by the Server when redirecting Urbscope API calls to external APIs.

3.1.6. Image Processor

The Image Processor module is responsible for preparing the picture data that is sent to the Cloud Vision API as part of a request. This picture is first . The returned response is then processed for use by the UI system and Knowledge Graph API. This involves retrieving the mid code which is essential for the performance of the Knowledge Graph API. The sequence thus is as seen in *Fig 2.2*, which illustrates the component.

3.1.7. Maps Service

The Maps service not only handles the map pointers to be displayed but also creates area polygons that have two pertinent features:

1. Check if the user has entered the perimeter of the landmark
2. Define the path to be taken by the user, which is later translated into the positioning of the directional arrow.

The Maps Service component is an essential part of the system, as it provides all navigational services, related to the user's latitude and longitude. Hence allowing the Application to display positions relative to the user, and allowing better direction management.

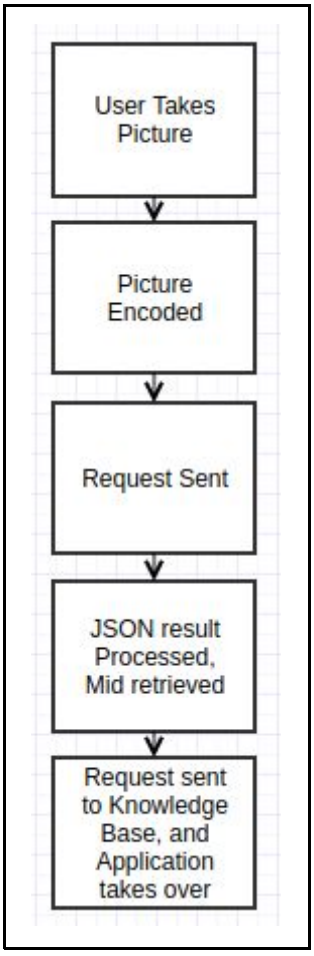

Fig 2.2

3.2.Hardware/Software Mapping

3.2.1. Data Model View

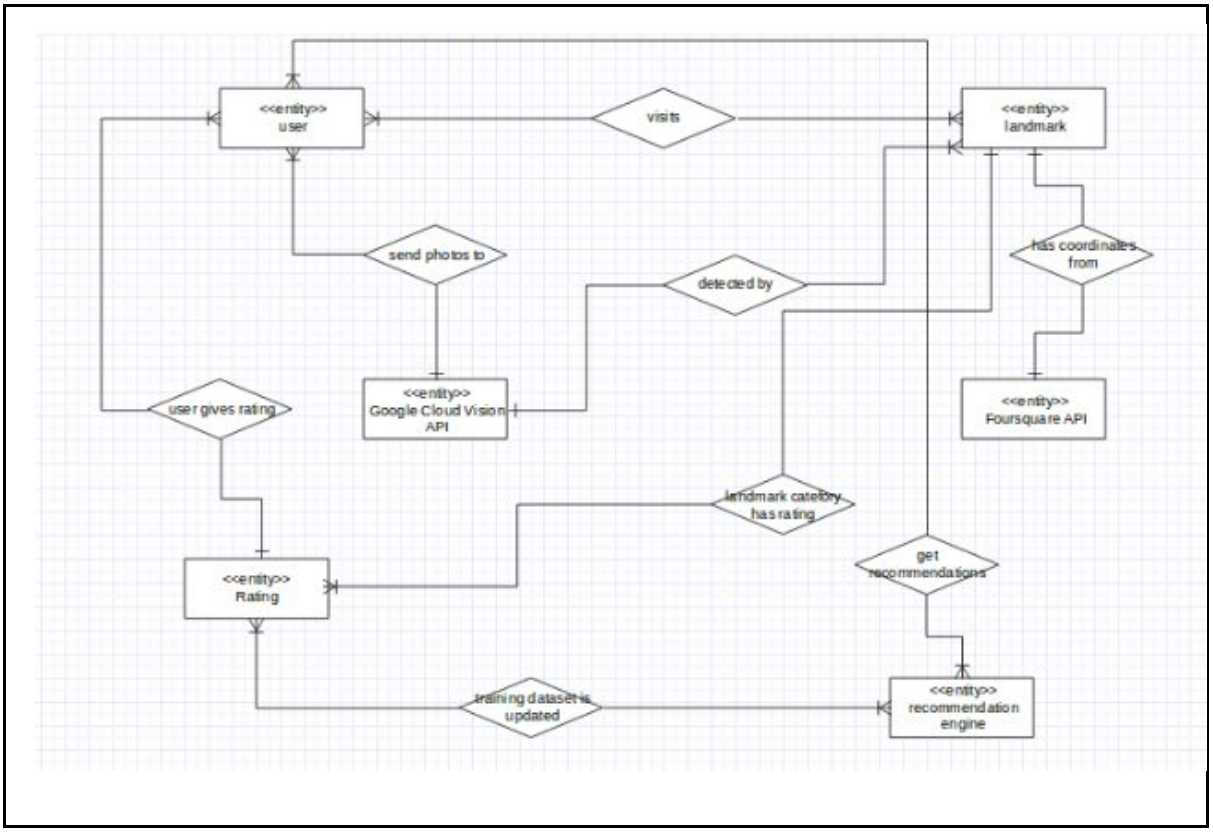

Fig 2.2

The data model illustrates the flow of data and which components utilize which data. This can be seen in *Fig 2.2*. A major part of the diagram illustrates how the recommendation engine utilizes ratings provided by the user to train the recommendation engine and then provide recommendations based upon similarity. Similarly the other major data flow illustrated is how the image detection works to identify buildings.

3.2.2. Client Server Connector

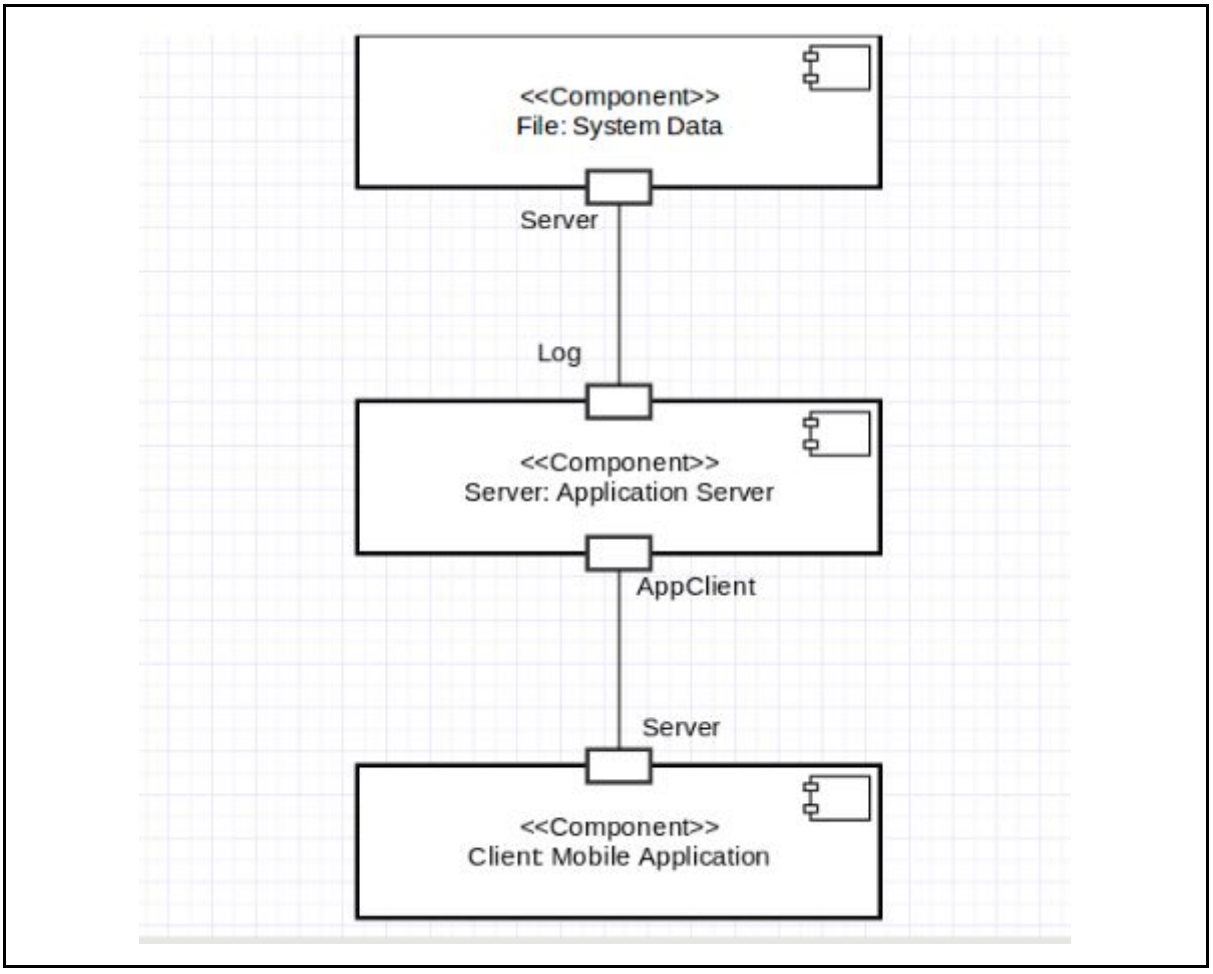

Fig 2.3

The system can be viewed in terms of client server as the system cannot function at all without the server. This is because all API data utilized by the system has to be made through the server and moreover the recommendation engine resides on the online server for all client to access. This is depicted in *Fig 2.3*.

3.2.3. Deployment View

The online application server will be hosted on a Win system using Node.JS for implementation. There are two external APIs that application server will be connected with. This shall include both the Cloud Vision API and the Foursquare API. The system shall communicate with these APIs using API calls and requests. The Urbscope,apk artifact represents the application that shall be deployed by the mobile device, whereas the Urbscope.js represents the server operation file implemented in Node.js.

The diagram in *Fig 2.4* is a overview abstraction of that in *Fig 2.5*. It can be seen that the os of the mobile device is not specified as the application is being created in react and is aimed as a hybrid application.

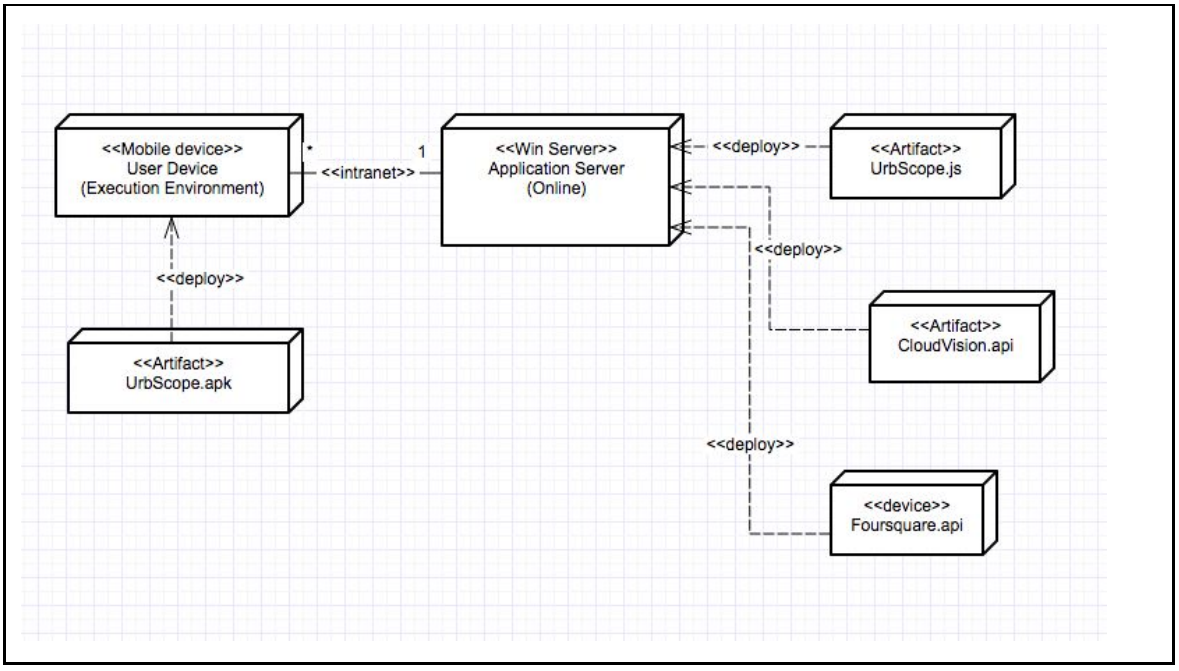

Fig 2.4

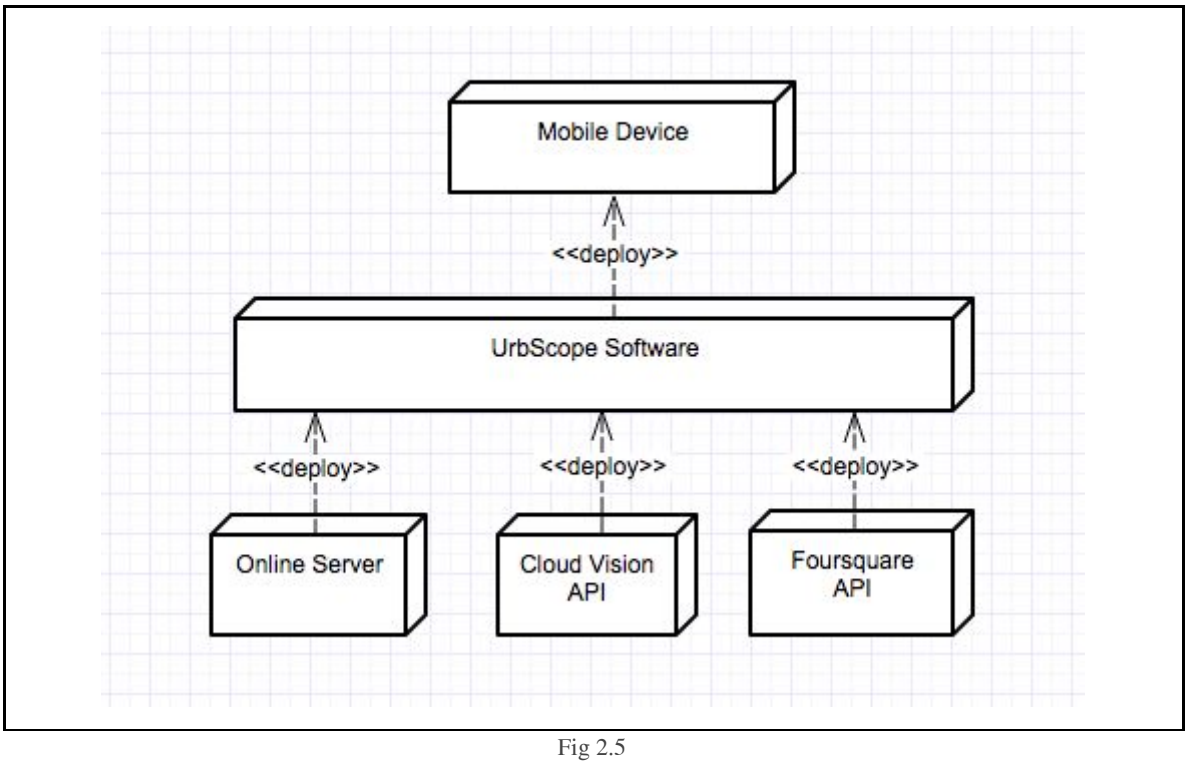

Fig 2.5

4. Algorithms

4.1. Latent Variable Model

In the latent factor model, given the ratings matrix we try to predict the missing entries in the matrix by factorising the ratings matrix $R_{\text{user x landmark}}$ as the dot product of two matrices $Q_{\text{factor x landmark}}$ and $P_{\text{user x factor}}$ where factor is the number of latent factors (dimensionality) that we need to define. Our aim is to find these matrices P and Q such that we minimize the error rate between our prediction and the true ratings. We used with baseline method because it tends to give better results.

In this algorithm we do not assume that the rating of a user for a landmark is based only on the dot product of the respective latent vector but we take into consideration also the landmark and the user bias as they have been described before. Thus, the predicted rating now becomes as seen in *Eqn 3.1*, with the biases incorporated.

$$\hat{r_{xi}} = \mu + b_x + b_i + p_x \cdot q_i$$

Eqn 3.1

The objective function now becomes while incorporating the baselines this is *Eqn 3.2*:

$$\min_{P,Q} \sum_{(i,x)\in R} (r_{xi} - p_x \cdot q_i)^2 + [\lambda_1 \sum_x ||p_x||^2 + \lambda_2 \sum_i ||q_i||^2 + \lambda_3 \sum_x ||b_x||^2 + \lambda_4 \sum_i ||b_i||^2]$$

Eqn 3.2

We apply the stochastic gradient descent algorithm and the algorithm with the baseline incorporation becomes as seen in the set of equations in *Eqn 3.3*:

For each rating r_{xi}:

$$\begin{aligned} \epsilon_i &= r_{xi} - p_x \cdot q_i \\ q_i &\leftarrow q_i + \mu_1(\epsilon_{xi} p_x - \lambda_2 q_i) \\ p_x &\leftarrow p_x + \mu_2(\epsilon_{xi} q_i - \lambda_1 p_x) \\ b_i &\leftarrow b_i + \mu_3(\epsilon_{xi} - \lambda_3 b_i) \\ b_x &\leftarrow b_x + \mu_4(\epsilon_{xi} q - \lambda_4 b_x) \end{aligned}$$

Eqn 3.3

5. Tools, Technologies and Resources Utilized

5.1. React Native

React Native is a development platform made by developers from Facebook that lets you build complete mobile app with hybrid functionality using just javascript. We used React Native for to develop the entire framework of our app including GUI, back-end computations, and server calls. It also supports external API calls which are critical for our application which further eases our work. React native is very responsive to the change in the state in the programs making our app work smoothly. It works in conjunction with Expo[1].

5.2. Expo

Expo is a free and open source toolchain built around React Native to help build native IOS and Android projects using javascript. Expo provides some very useful libraries like Camera and push notifications which are imperative for our application. Also it was used during entire development phase to render our app on the mobile phones[2].

5.3. Node.js

Nodejs is an open-source, cross-platform javascript run-time environment that executes javascript code for the server side. We used Node.js to implement our server side making javascript as dominant programming language in our application. Calls to APIs were implemented in the server.

5.4. FourSquare API

Foursquare is a local search-and-discovery service mobile app which provides search results for its users [3]. The app provides personalized recommendations of places to go to near a user's current location based on users' "previous browsing history, purchases, or check-in history". FourSquare API is used to get the data to display nearby landmarks of the user and also to generate dataset for recommendation engine.

5.5. Cloud Vision

Cloud Vision API allows developers to easily integrate vision detection features within applications, including image labeling, face and landmark detection, optical character recognition (OCR), and tagging of explicit content [4]. In our app we used cloud vision API to detect the landmarks whose image is passed via user’s camera. It returns landmark’s name.

5.6. Knowledge Graph API

It is an API developed at Google to fetch the data about the landmarks. We retrieve image and basic information about the detected landmark[5].

5.7. Geo Coder

Geocoding is the process of converting addresses (like "1600 Amphitheatre Parkway, Mountain View, CA") into geographic coordinates (like latitude 37.423021 and longitude -122.083739), which you can use to place markers on a map, or position the map[6]. Geocoder API is used to get user's coordinates to determine in which city user is so that relevant landmarks can be recommended

5.8. Google Maps

We use Google Maps to get navigation directions from user's current location to the selected nearby location[7].

5.9. Web GL

WebGL is an extension of OpenGL ES (ES for embedded systems). OpenGL ES is supported by mobile applications. WebGL supports building of mobile-based web applications and it also uses JavaScript, which perfectly integrates with our design since it is already in javascript[9]. We use THREE js, which is a library built upon WebGL.

5.9.1. Extrude geometry

Extrude geometry defines shape and extends it on z-axis making it 3D. MeshLambertMaterial, MeshPhongMaterial and StandardMeshMaterial are three available materials in THREE js.

5.9.2. Standard Mesh Material

It uses phone shading. This approach differs from older approaches in that instead of using approximations for the way in which light interacts with a surface, a physically correct model is used. The idea is that, instead of tweaking materials to look good under specific lighting, a material can be created that will react 'correctly' under all lighting scenarios.□□ In practice this gives a more accurate and realistic looking result than the MeshLambertMaterial or MeshPhongMaterial, at the cost of being more computationally expensive.□□ Shading is calculated in the same way as for the MeshPhongMaterial, using a Phong shading model. This calculates shading per pixel (i.e. in the fragment shader, AKA pixel shader) which gives more accurate results than the Gouraud model used by MeshLambertMaterial, at the cost of some performance.

6. Impact of Engineering Solution

6.1. Educationally

Urbscope provides uits traveller user's with a magnitude of information, compacted into a single mobile device in a legible and comprehensible manner. The idea Urbscope had was to revolutionize how foreigners absorb the culture of unknown traditions and societies miles away from their own. Moreover, the information provided by Urbscope in regards to the landmarks is aimed at battling the misinformation spread by illegal tourist guides and unauthorized individuals who attempt to provide guided tours to foreign tourists. The information that Urbscope retrieves is verified, and particularly relevant to the landmark. Hence information regarding historical landmarks and natural wonders is more attainable and is is presented in an even more entertaining manner. This shall affect knowledge and education for all demographics.

6.2. Societally

Urbscope shall revolutionize the way the tourism industry operates. With the amount of information that Urbscope provides, society shall become more aware of the history and culture of far-away cultures. Urbscope shall impact the tourism industry the way Uber has affected the transportation industry. Urbscope will be intended to be the pocket tourist guide, and has the potential of becoming a household name.

Moreover, the landmarks data collected for the landmark prediction recommender system, is a rare entity. There are large amounts of datasets online, ranging from Zulu Sign Language to Dolphin communities of the shore of New Zealand. However, there is no data which is about tourists and their ratings of landmarks and points of interest. Moreover, the two large applications that travellers usually refer to are Tripadvisor and Foursquare, and both of these are not specifically interested in touristic activities as much as they are interested in hotels and restaurants respectively. Hence Urbscope aims to fill the void that these two giants have surprisingly left open. This could mainly be due to the difficulties that the two giants may have faced in procuring landmarks and their ratings. There is also not a clear definition between what constitutes as landmarks and point of interests for tourists, and what are simply places of interest in the domain of entertainment. Urbscope is one of the few projects that attempts to draw that clear defining line. Not all restaurants and entities are touristic landmarks, and this manages to target tourists and travellers specifically.

6.3. Economically

Urbscope aims at revolutionizing the how the tourism industry operates, without disrupting the conveniences of the customer. The project has the potential to ease travel and touristic entertainment/exploration of the more than 1.2 billion tourists and travellers, and this is just the number of international travellers. Urbscope shall also create ripples in domestic markets as Urbscope shall also be used by national traverlors. For example, tourists from Bengal, India, have no idea about South or North India, and hence they can use it for travel. Moreover, the sponsorship feature shall allow a multitude of businesses to get their establishment featured on the app, and this shall generate revenue for all parties.

The only drawback is the negative effect on the local small time tour guides. However, the losses are neglectable, considering the overall financial gains in an economy and the benefits of Urbscope are clear.

6.4. Environmentally

Urbscope, once reaching its full potential shall have the same environmental costs that companies such as Facebook may have. The service is primary software, and can be imagined, it entails very few cost. At the moment, the application makes use of a single server. The environmental drawbacks are negligible considering the mass produced impacts the environment incurs, especially after Donald Trump has put “America First” by withdrawing from the Paris Climate Accord.

7. Related Contemporary Issues

7.1. Touristic Sites Data

As mentioned in *Section 6.2* there is surprisingly a lack of data available online regarding ratings for touristic landmarks and points of interest. By accumulating this data, Urbscope hopes to release the data for open source usage, in an attempt to further the cause of intelligent tourism and travel. Collecting this data was not a simple issue, as it involved a lot of factors, and we a Urbscope hope that the data that has been procured shall be used well.

7.2. Tourism in an Economy

Tourism is one of the most dynamic and fastest developing sectors in Turkey. According to travel agencies TUI AG and Thomas Cook, 11 of the 100 best hotels of the world are located in Turkey. In 2005, there were 24,124,501 visitors to the country, who contributed $18.2 billion to Turkey's revenues, with an average expenditure of $679 per tourist. In 2008, the number of visitors rose to 30,929,192, who contributed $21.9 billion to Turkey's revenues. For 2011, the World Tourism Organisation (UNWTO) reported 34,654,000 arrivals and US$25 billion in receipts for Turkey. According to the World Travel & Tourism Council, in 2012 travel and tourism made a total contribution of 10.9% to Turkish GDP and supported 8.3% of all jobs in the country[9]. Over the years, Turkey has emerged as a popular tourist destination for many Europeans, competing with Greece, Italy and Spain. Resorts in provinces such as Antalya and Muğla (which are located on the Turkish Riviera) have become very popular among tourists. Urbscope hopes to contribute positively to the growth of Tourism not only in turkey but the world.

7.3. Cold Start in Recommendation Systems

Recommending items that have rarely/never been viewed by users is a bottleneck for collaborative filtering based recommendation algorithms. To alleviate this problem, item content representation has been used as auxiliary information for learning latent factor representations. Matrix Factorization based collaborative filtering approaches are widely used in modern recommender systems. One of the disruptive issues of this approach is when the collaborative information is very sparse and in many cases not available. In the case of Urbscope, when considering the extremely large number of touristic points of interest, in the millions of cities across the world, sparsity is no longer an anomaly to be considered, but a norm to be accepted[10]. The manner in which Urbscope aims to solve the cold start problem is quite intuitive and straightforward. Whenever, the problem of extreme sparsity, very cold

start scenario, persists, the recommender system provides recommendations according to categories. Urbscope identifies 19 categories, two of which have sub categories. The recommendation system also stores the mean ratings given per category. Cold start problems in the recommender systems cannot be avoided, but by predicting over only 19 item sets, sparsity is reduced dramatically to almost zero.

3. References

1. “React Native · A framework for building native apps using React,” *React Native Blog ATOM*. [Online]. Available: https://facebook.github.io/react-native/.

2. “Expo,” *Expo*. [Online]. Available: https://expo.io/.

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

4. “Detecting Landmarks | Cloud Vision API Documentation | Google Cloud,” *Google*. [Online]. Available: https://cloud.google.com/vision/docs/detecting-landmarks.

5. “Knowledge Graphs,” *Google*. [Online]. Available: https://developers.google.com/knowledge-graph/.

6. “GeoCode,” *Google*. [Online]. Available: https://developers.google.com/knowledge-graph/.

7. “Google Maps,” *Google*. [Online]. Available: https://developers.google.com/maps/documentation/javascript/tutorial..

8. “The WebGL API: 2D and 3D graphics for the web,” *MDN Web Docs*. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API.

9. "UNWTO Tourism Highlights 2013 Edition," *UNWTO*. Archived 2013-11-27.

10. Sujoy Roy and Sharath Chandra Guntuku. 2016. Latent Factor Representations for Cold-Start Video Recommendation. In Proceedings of the 10th ACM Conference on Recommender Systems (RecSys '16). ACM, New York, NY, USA, 99-106. DOI: https://doi.org/10.1145/2959100.2959172