Bilkent University
Department of Computer Engineering

Senior Design Project

Low Level Design 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.

1.1. Object Design Trade-offs

1.1.1. Ease-of-Use vs Complexity and Functionality

Urbscope is aimed specifically at the tourist industry and hence the usability of the Urbscope must compliment the demographic being targeted. However, the functionalities implemented in the development of Urbsciope are quite novel in nature. The challenge thus lies in presenting these novel features such as Augmented Reality and item recommendation, in an easy-to-use manner, facilitated through the implementation of the UI. A good UI and UX is imperative for Urbscope to take-off on a commercial level and hence even though the features of Urbscope have excessive bounds, they must be retained within the constraints of comprehendable UI.

The conflict between functionality and ease-of-use is illustrated in the manner the markers are to be presented to the user. The Foursquare API returns large amounts of information, but not all that information is to be displayed at once. The name of the landmarks returned is to be displayed at the get go. In fact none of the information that is returned is to be displayed on the AR markers and only when the AR marker is clicked is the information to pop up in a dialog component in the React Native interface. Furthermore, the information that is to be displayed once the Google Vision API is called to detect a landmark is also procured from the Foursquare API. This results in the need for Asynchronous calls to facilitate the functionality of retrieving information of the landmark the user is in front of.

Another issue within this domain arises due to the recommendation system. Since the application is to be made as simplistic as possible to use, the need of a user based system is not incorporated into Urbscope. This allows the user to use the application without the hassle of logging into accounts. However this convesley increases the complexity of the application in storing the log files and the manner the recommendation system is to function. A log file is to be stored to save a history of the user's previous places visited places and this log file is then to be used to retrain the recommender engine on the server and also to make predictions of the specific user. This thus solves the issue of making specific user based predictions without having the user to actually make an account.

1.1.2. Space and Time vs Performance

As can be seen there are several APIs that Urbscope is communicating with, along with the online server on which the recommender engine resides. This causes the issue of latency that may occur as information from the respective APIs are being procured. As a result the performance of the application can be hampered even though he information is correct. Hence algorithms used to process the returned data are developed in a manner to optimize the complexity. For example the

recommender engine uses the Slope One algorithm from the Scikit Surprise library[1]. This is because the execution time of the algorithm was less when compared with the other algorithms in the set provided by Surprise, along with having a substantial RMSE value to be used as the prediction method for Urbsacope. Hence the time required to process the prediction is considerably lower and is aimed at reducing the response time from the server.

Similarly images that are being uploaded to the Google Cloud Vision API are reduced in size and resolution to facilitate a faster upload to the Cloud Vision API. The compression of the images thus reduces the size of the images that are to be uploaded to the server, However, care must also be taken to ensure that the resolution of the image is not distorted to the extent that the image cannot be detected.

1.1.3. Scalability vs Consistency and Cost

Due to the need for several API requests being made to obtain the landmarks for and the the image detection information, scalability concerns arise. At the moment Urbscope avails of the free services that the APIs mentoinded provide. However, as the number of requests increase due to the increased number of users, then higher rates of the APIs will have to be purchased.

Similarly the server being used for the recommendation engine will not be able to process a large load as it is hosted on Digitalocean[2] and is a basic server without any load balancing algorithms.

1.1.4. Security vs Availability of Data

User privacy must be ensured and hence measures need to be implemented to prevent unauthorised access of sensitive data.. Data are needed to be categorized based on its sensitivity that is in accord with the amount of harm that could be done if unintended access is granted. However, in regards to Urbscope, there is very little if no sensitive data that is stored. This is expected as most of the landmarks data that is obtained is retrieved through various APIs and hence the information retrieved is already authorized by well established corporations. As a result the availability of landmark information is secured.

However as previously mentioned the availability of information is also restricted by the quota of API calls that are allocated to the free package that is availed. Nevertheless the information that can be considered as sensitive is the places visited by the user, stored in their log files. These files however are stored in the user’s own mobile phone and therefore the user's device already have pre implemented systems that ensure that these log files cannot be gained by unauthorized parties. The log files are then used to iteratively train the system to improve the recommender system.

1.2. Interface Documentation Guidelines

The class interface hierarchy of the documentation will be provided in the following format:

Class name	The name of the class
Description	A brief description of the class.
Package	The package, which the class is a part of.
Attributes	<attribute> : <object_type>
Operations	<function> : <return_type>

1.3. Engineering Standards

IEEE standards have been used throughout previous reports, this report, and all other reports to follow. Similarly the reports host UML style notations for visualization of Urbscope.

1.4. Definitions, Acronyms, Abbreviations

- IEEE: Institute of Electrical and Electronics Engineers
- IEEE-SA: IEEE Standards Association
- EAB: IEEE Educational Activities Board
- UML: Unified Modeling Language
- GUI: Graphical User Interface
- HTML: Hypertext Markup Language
- JS: Javascript
- JSON: Javascript Object Notation
- REST: Representational State Transfer
- Node.js: A server-side Javascript environment.
- React.js: An open-source JavaScript library for building user interfaces by Facebook.
- ML: Machine learning
- EXPO: Open source toolchain built around React Native
- AR: Augmented Reality
- API: Application Programing Interface
- npm: Node Package Manager a.k.a Nonsense Poetry Manager
- ViroMedia: An AR library.

2. Packages

2.1. Package Overviews

2.1.1. Use View Packages

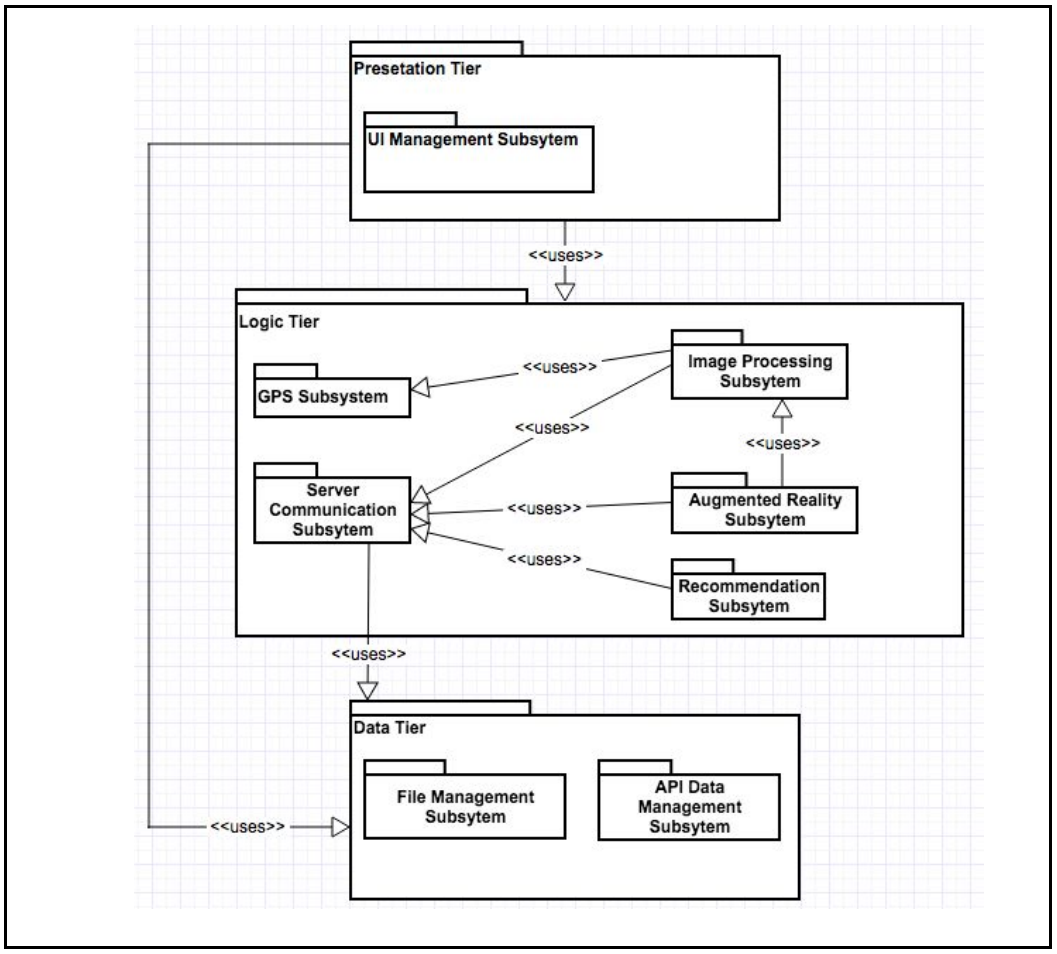

Fig 2.1

The uses view illustrates the manner in which the sub decompositions communicate with one another. Hence from *Fig 2.1* it can be seen that the Data tier is used by all other subsystems. Within the Logic Tier itcan be seen that all subsystems go through the Server Communication Sub-System to communicate with the Data tier. This is because all the data that is utilized by the system is mainly stored on the online server. Moreover, the Augmented Reality System makes use of the Image Processing system and the GPS system to calculate the correct locations to place the Augmented Reality Markers.

2.1.2. Deployment Packages

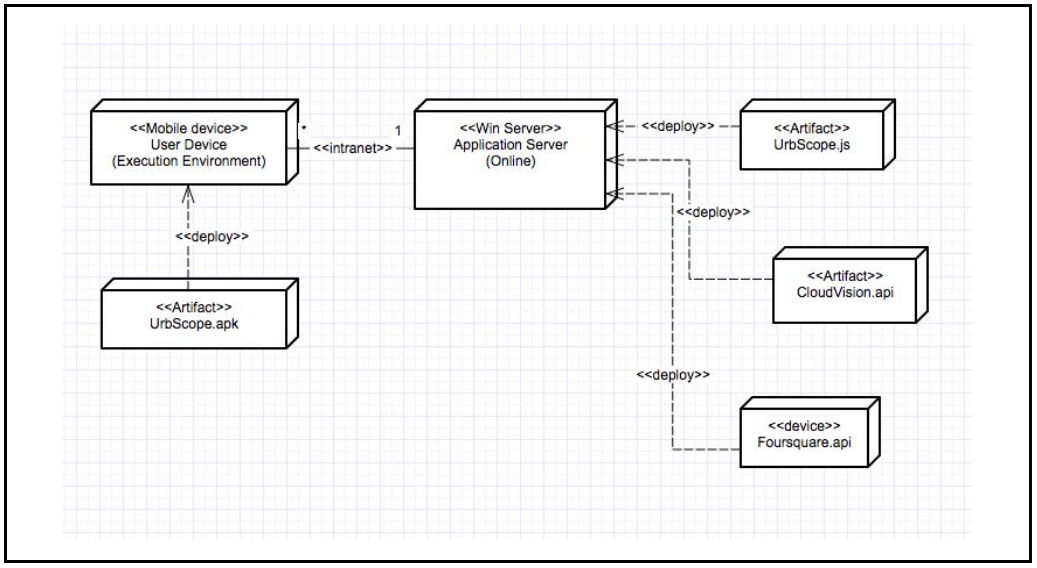

Fig 2.2

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 2.3 is a overview abstraction of that in Fig 2.2. 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.

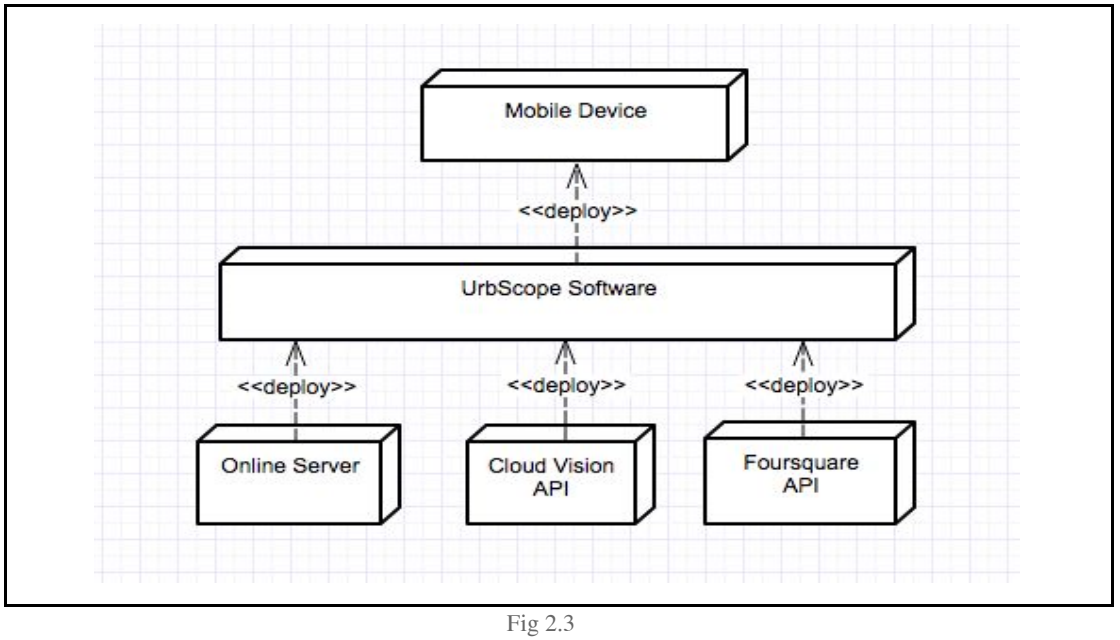

Fig 2.3

3. Class Interface

The application has distinct domains of development as have been illustrated in this and in previous reports. These distinctions arise due to the various functionalities that Urbscope offers and thus facilitates a modular development. The distinct sections are as below:

- Augmented Reality
- Image Detection
- Landmark Retrieval
- Recommendation Engine
- React Native Mobile Application Development

These various components thus have their classes defined to support the required functionalities. Classes are implemented in a myriad of languages and frameworks and are leter integrated using tools such as the python shell and npm modules.

3.1. Augmented Reality

For augmented reality we will use Viromedia AR library to build and render AR objects. Viromedia is built on top of React Native which we are already using in our implementation.

ARSceneNavigator
Handles changes between different AR scenes and passes props to the scenes.
React Package
- InitialARScene: ARScene - ARSceneProps: PropsModule
+ AddScene(ARScene)

ARScene
React Component with ARScene as the top element. Contains AR objects as children. When an AR model is added to the scene, the positioning will be determined using _onArHitTestResults().
ReactReact Package
- ARVisible: boolean - ARObjects: 3D Object Module Files - ARModels: ViroNode (From ViroMedia Library)
+ AddModeltoScene() + _onArHitTestResults(position, forward, results)

GeoLocationService
Determines the geographical location of the user and then maps the location to X and Z values in the Augmented Reality coordinates
ReactReact Package
- Longitude: Double - Latitude: Double - X: int - Z: int
+ getCurrentPosition(callback, error, config) + mapToXZ(callback, error)

3.2. Image Detection

Image detection will be located on Urbscope's server and the user will send requests to the server to obtain information of the landmark that he/she wants to be recognised. Basically, the server works as a middleware layer between the Vision API and client side. In the client side, there exists some helper functions to filter the information returned from the API.

ImageUpload
It takes an image of the landmark that needs to be recognised via POST request and returns the information about the landmark to the client via an JSON response.
Image Processing Subsystem Package
- client: VisionObject - port: int - sslOptions: dictionary

- sharp: SharpObject
+ upload(HttpRequest request, HttpResponse response, String key): Promise + postDetectLandmark(Callback upload, HttpRequest request, HttpResponse response): JSONObject

CloudVisionReact
Displays the information returned from Vision API to the user.
React Package
- vision: VisionObject - req: VisionRequest
+ detectLandmarks(String base64image, JSONObject response): JSONObject

HelperService
Filters the related returned information from Vision API.
React Package
+ fixDetectedLandmarks(JSONObject response): JSONObject

3.3. Landmark Retrieval

Landmark retrieval occurs by communicating with the Foursquare API and retrieving all landmarks that meet the query and radial parameters set by the user. The response is a JSON object that is then parsed to obtain only relevant information. This information consists of open timings, address and other pertinent information. The landmarks retrieved are then utilised by the AR component to generate AR markers.

LandmarkRetrieval
Takes user current cordinates, along with set filters and queries to retun a list of nearby and pertinent landmarks.
React Package
- Coordinates: Coordinates - JSONFilters: JSON
+sendRequest(JSON parameters): JSON +retrieveLandmarks(): JSON + parseResult(JSON resutlt): List

3.4. Recommendation Engine

The Recommendation Engine will be located on Urbscope's server and the user will send requests to the server to obtain the predicted result of a landmark for a specific user. Similarly the user will also have to send the log file to the server for the log file to be integrated into the already present training dataset. The engine is then supposed to be retrained using the updated training to allow for changing trends and improved accuracy.

TrainingService
Trains the recommendation engine by using training data to obtain a trained model
Server Package
- TrainingData: TrainingData - TrainedModel: TrainedModel
+readData(File ratings): int[][] +TrainModel(): int[][] + serviceModel(): File

PredictionService
Retrieves the user id and landmark id from client and calculates related prediction of landmarks to be returned
Server Package
-TrainingModel: TrainingModel - TrainingData: TraiiningData
+calulatePrediction(int userId): int + serviceLandmarks(): List

UpdateTraining
Retrieves the log file of a user and adds it to the training file which exists on the server and calls the TrainingService
Server Package
- LogFile: LogFile - TrainingData: TrainingData
+updateFile(File userLog): File + serviceData(): File

3.5. React Native Application Development

This is the skeleton of our application. It encapsulates all UI designs of our application with camera, modals, navigation etc. By default user will be directed to the “Detection mode” screen at the start of the application. This section contains transitions, animations background colors, image projection and everything that user sees in the app.

ChangeModeSwitch
Toggles between “Exploration Mode” and “Recognition Mode”
React Package
- currentScreen: props - isSelectedDetection: color - isSelectedExploration: color
+ goToDetection(): StateUpdate (Detection Mode) + goToExploration(): StateUpdate (Exploration Mode)

DetectionMode
Renders “Detection Mode” screen. Opens a modal box when user requests for additional information for the detected landmarks.
React Package
- hasCameraPermissions: boolean - errorMessage: String - location: String - modalVisible: boolean - navigation: props
+ detectLandmark: LocationInformation + closeModal: UpdateState + openModal: UpdateState

ExplorationModeSwitch
Retrieves nearby landmarks or landmarks recommended for the user based on the user’s selection.
React Package
- isSelectedNearbyLocations: boolean - isSelectedRecommendations: boolean
+ goToRecommendations: StateUpdate + goToNearbyLocations: StateUpdate

LandmarkDetailsModal
Displays information of detected landmarks.
React Package
- visible: props

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. N. Hug, “Slope One,” *Surprise Library*. [Online]. Available: http://surprise.readthedocs.io/en/stable/slope_one.html. [Accessed: 10-Feb-2018].
2. “Cloud Computing, Simplicity at Scale,” *DigitalOcean*. [Online]. Available: https://www.digitalocean.com/. [Accessed: 13-Feb-2018].