Bilkent University
Department of Computer Engineering

Senior Design Project

High-Level Design Report

Project Name: Nomad

Group Members: Gülnihal Muslu
Berk Ataç
Can Ozan Kaş
Ali Kemal Özkan
Tarık Emin Kaplan

Supervisor: Uğur Güdükbay

Jury Members: H. Altay Güvenir, Özcan Öztürk

Innovation Expert: Veysi İşler

High-Level Design Report
December 31, 2019
This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS491/2.

Table of Contents

1. Introduction

Nowadays, travelling is an easy and usual activity for people[1]. Everybody wants to explore places that they have never been. However, there are a lot of buildings and places or activities to do in a country/city. Therefore, people have to choose the best of them according to their interests because they usually have a limited time to see a city. Deciding which place is worth seeing or which food is worth eating can be tricky in most times due to people usually preferring to explore cities that they have never seen before. There are some applications and websites for that, but they have never been enough for people that are such travel lovers.

To solve that problem, we thought of a project -which name is Nomad- where people can give trip advice to other people. Nomad means a member of a group of people who move from one place to another instead of living in the same place all the time. Our project idea comes from there. In this project, a person can create a route in a city or country and s/he can give advice on that trip. Users will have a chance to select a route from various list of routes that are created by other users and they can follow that path step by step. Since routes can be seen before and they can be scored, interpreted; people can choose their route easily and more close to their dream trip. With this project travels will become easier and more fun.

This report contains analysis of our project with three subtitles: current system, proposed system and other analysis elements. All system models and user interface included.

1.1. Purpose of the System

Nomad is a mobile application which aims to help travellers explore new places by following a "for travellers, by travellers" approach, in which people can create routes, and others can search a route according to their preference of what type of journey they would like to go on and where they want to explore. Users can add stops to the routes they create on specific locations that they deem worthy of attention, which act as "checkpoints" for other users who follow that route. Using augmented reality, users that create routes can also add tokens to their routes for the route users to find, those who obtain such tokens can get achievements and unlockables.

1.2. Design Goals

Extensibility:

- Nomad should allow adding new functionalities, features, components and so forth.
- The system should be kept up to date.

Availability:

- Nomad will be available 24/7 for the users.
- It will open for every android user who use android 8 or higher.

System:

- Nomad will be an Android application.
- Application will require an Internet Connection to work.
- A smooth, minimalistic and fast user interface will be designed.

Performance:

- Nomad should not take more than 5 seconds to log in.
- Application should not take more than 15 seconds to get ready after app icon is clicked.

Scalability:

- Nomad will support large number of users.

Security:

- Nomad needs to secure user information from any possible threats.

Reliability:

- Nomad needs to be stable and crash free.

Usability:

- Application should be easy to use for all user types because anyone who can use a mobile phone is possible user of this application. Thus, we need to provide user friendly and simple user interface.
- Application should clearly explain what it is aim and why user should use this app.

1.3. Overview

Nomad is a mobile application, first will be published in Play Store will be coded in Javascript. It will be addressing to people who want to travel and tour companies. With this application, users can create trip routes, which will be held in a database, as traveling by using GPS in their cell phones. While users are creating a route trip, they can place an object with augmented reality, like a coin or a token, when they arrived where they want to go. These objects are checkpoints for other users who will use routes created by other users.

The trip routes created by users are designed to follow a certain path, which means users are able to see where to go first, then second and then third. This way, they will not struggle finding the way as mostly, people travel to places which they have not been there before and Nomad will show the way to the checkpoints. While users are taking a trip created by other users, they can collect objects and tokens, which were put by the creator of that trip, using their phones' camera. Tokens can be used to create new routes. If user follow more routes, s/he will be able to create more routes. Once, a user completes a trip route, will be able to vote the trip out of 10. Thus, other choose can choose a route according to its rating. Also, all user will have

points that increased by travelling distance. Thus, trip routes and trip creators have their own ratings and points that people can see them.

Nomad enables users to give advice to other users as well as take advice from other users. This provides users a better trip experience as every user can have different interests. To illustrate, if a user wants to take a nature trip and all the high rated places are historical places, s/he can still find a trip that s/he can love from the trip route list of the place s/he visit.

2. Current Software Architecture

Most close application that close to ours is Trip Advisor. TripAdvisor is an American travel and restaurant website company that shows hotel and restaurant reviews, accommodation bookings and other travel-related content. It also includes interactive travel forums. It is wider than our application. It has hotels, restaurants, rental cars, flights, stuff to do. etc., basically all in. They offer places to go in a city. However our app focused on trips and routes. Not only places but also routes of others are important in Nomad. If you don't like someone's route, you can create your own route. Our app does not offer places to go, it offer routes to follow. Routes can involve anything such as a street, a museum, a restaurant, a statue etc., does not have limits.

3. Proposed Software Architecture

3.1. Overview

Since Nomad will be a mobile application, we have used MVC (Model-View-Controller) architecture. Subsystem decomposition of our application will be given in the following section. In addition to that, it will be explained detailed in subsystem services section. In the section 3.3 hardware/software mapping of Nomad is explained clearly. For the data, accessibility and security details are highlighted in the "Access Control and Security" part. Finally, in the last two sections, global software control and boundary conditions, are taken up.

3.2. Subsystem Decomposition

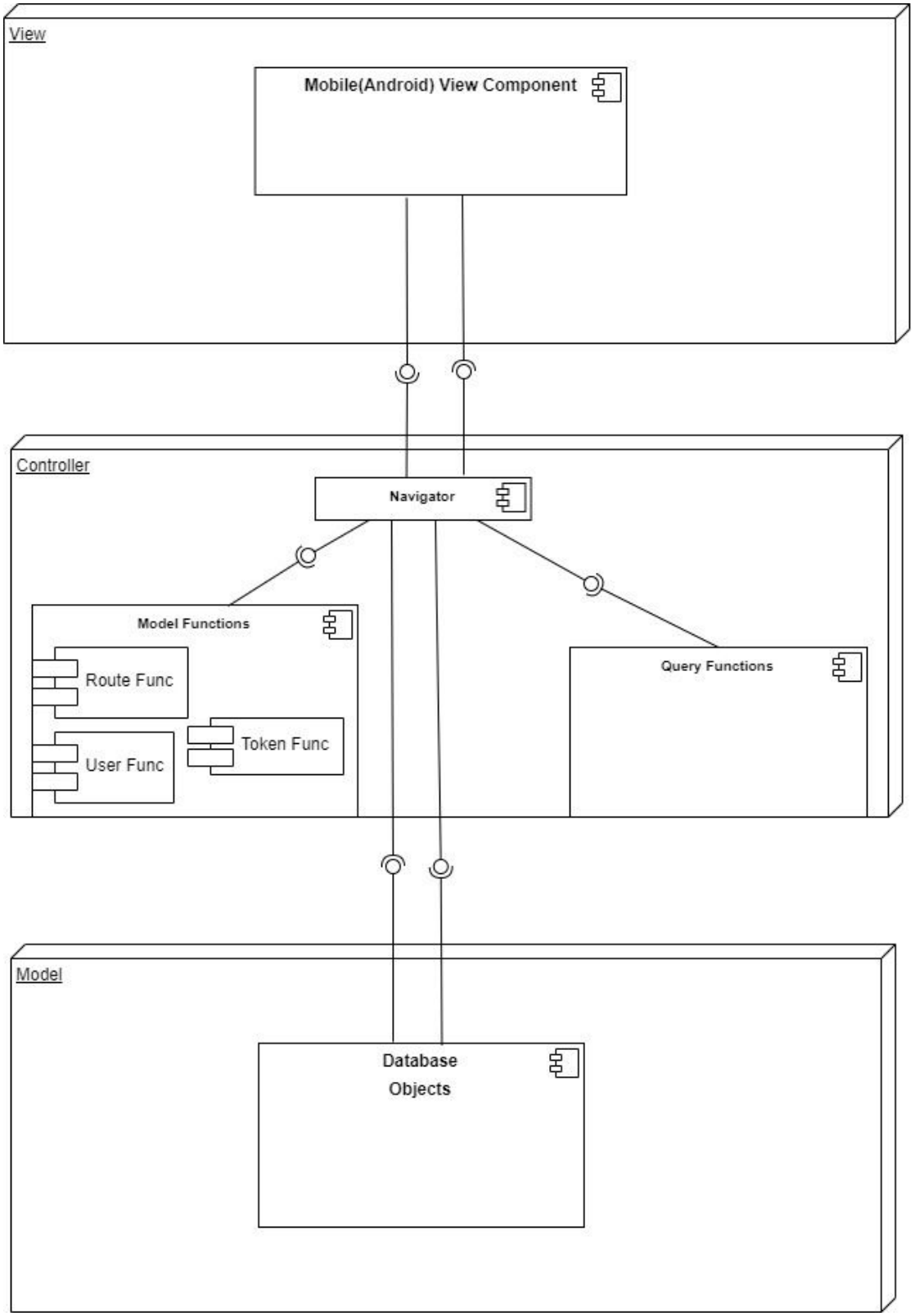

Figure 1: Component Diagram

- The chosen architectural pattern is Model-View-Controller (MVC). This pattern separates and classifies the software to three main components that are UI elements to be visualized, data to behold and requested, and control elements that will pull/push requests to other components in order to ensure the data flow. The MVC pattern provides sufficient and genuine features and it is in a harmony with the design decisions made by our team.
- In Controller, Model functions serve as a beholder of the application's database entries. It allows queries run by the controller package to access the related user data.
- Navigator serves as a controller for UI components.
- Query Functions are database functions, insert, delete etc.
- Our application Nomad will run on Android devices, in View, we have a mobile view component for our client(android).
- In Model, Database objects are information(data) we hold in our database, which is provided by our Controller and viewed by our View component.

3.3. Hardware/Software Mapping

Nomad will initially run on android based platforms. It will use camera of the smart-phone. Account information of the users will be stored on a server with MySQL database management system. Thus, it requires internet connection. Client-server architecture will be used for this purpose. Client-side will be communicating with the end-users via Nomad UI. It will also handle location detection with GPS. Data management operations will be handled in the server-side hosting the application.

Client-side needs to communicate with the database server to get/set the data. This communication will be provided by HTTP.

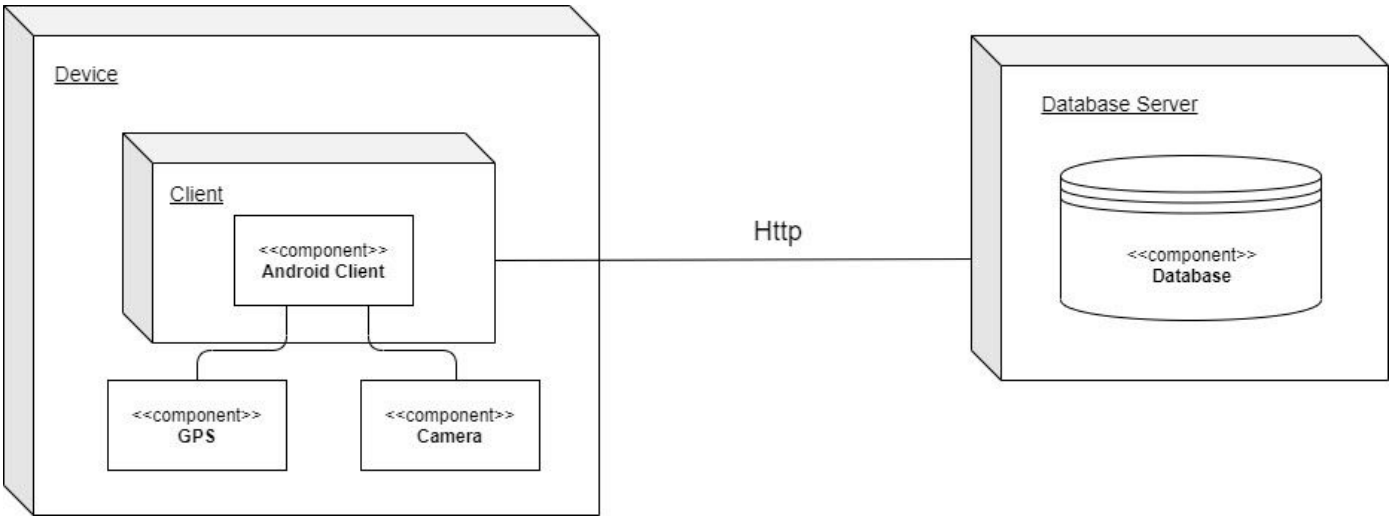

Figure 2: Hardware/Software Diagram

3.4. Persistent Data Management

Nomad will store various user information such as the achievements, routes that has been used and liked, profile and account information. For this purpose, there will be a sufficiently large database. Since the database needs to store the routes as coordinates, there will be a huge amount of data. In order a user to follow a route s/he needs to get the coordinates from the database. This process without any fast searching techniques will take a lot of time. Therefore, methods like indexing and hashing will be used to get the required data.

In order to save the routes created by users and not published yet, the device needs to have enough storage as these routes will be kept in the device until it is published. The application will continuously check If there is enough space in the device to keep the route. When the storage is full, the route creation will automatically stop and current process will be saved.

3.5. Access Control and Security

Application will have login options related with the following social media services Facebook, Google, and Twitter, we will store the public data we are allowed to read from these social media services (like their name and email) and we will store them in our database, and to get these data, we will ask for our users' consent. Of course, we can't even access private data such as their social media account passwords, so storing them in the database is out of the equation.

After logging in with a social media option, for future use, users may also create a password so they won't have to re-enter their preferred social media credentials every time they log in. This password will be encrypted before it's stored to our database by using the Base64 library [2][3].

We will also store other user generated data such as route data, liked routes, comments related to a route in our database, some of these information (e.g. liked routes) will be publicly shown by default so we don't have to take important measures to store them, but some of the data (e.g. public vs private routes) may have limited publicity, but unlike passwords, we won't be needing to encrypt them, just assigning a flag to denote their publicity is enough since we will have full control over these user generated data within our application, and we will use this data accordingly.

Another security measure is having our application host and database on different servers, in the event that one of these servers' security is compromised, we can at least lock down the other one and secure it. We may not be able to afford two different servers so there is a possibility that this security measure may not be taken.

An additional benefit of limiting the login options to social media services is that the spam of accounts we would have normally gotten with a simple email and password sign up system is basically redirected to these

social media services, which are of course more capable than us to deal with such problems.

Some user data (i.e. personal information provided by the user / public data that was retrieved from user's social media account) may be changed by the owner of that data. A good example for this is when the user logs in with any social media login option, we get the email related with that social media account, but after that, the user may change the email registered in our system. Moreover, we need to ensure that our application complies with KVKK laws[4], as such, we will not share any of our data with other third party applications.

3.6. Global Software Control

The general flow depends on user actions. In very broad terms, our program waits for user requests, then passes these requests to our logical layer and what to do with the user input is calculated and the corresponding results are gathered and displayed back to the user and program returns to its waiting state until a new request is given, so the cycle continues.

To explain it in detail, since we will use the MVC design pattern, user is exposed to our view layer, and we wait for the user to submit an input request there; after an input is supplied, the controller layer identifies the request and supplies it to its corresponding handler in the model layer, where the necessary logical and data operations about the request are performed; after that, the result produced is given back to the controller, which identifies the correct view to supply it to and does so.

In our analysis report, we have mentioned about our functional requirements and the sequence diagrams and classes we will use corresponding to these requirements, so we will of course not explain all the event flows here again, but let's look into detail of a main functionality (i.e. route creation) from a flow of actions and MVC perspective. While the user is

creating a route, the CreateRoutePanel screen is displayed to the user and it waits for the user's next move. When they move, the navigator supplies necessary coordination variables to RouteFunctions class, then a connection to the database is established and the correct tables to insert the previously mentioned coordination variables (and potentially other variables related to it) are identified, then these variables are inserted there. The RouteFunctions will successfully return to the navigator with a produced result, and the navigator in turn will give this result to the CreateRoutePanel where the view may be updated accordingly.

3.7. Boundary Conditions

3.7.1. Initialization

Our application first needs to be downloaded from the Google Play Store. Users need at least one of Facebook, Google or Twitter accounts to be able to login to our application for the first time and be registered to our system. Users will need to be logged in all the time to use the application.

3.7.2. Termination

When the app is put in background use (without termination) users won't be immediately logged out since they may want to return to the application shortly (unless the users have manually logged out before putting the app to background use). When application is removed from background use it is terminated.

3.7.3. Failure

The app needs to be connected to the internet throughout all its use, and a disconnection may cause significant errors to occur (e.g. not being able to authenticate the user, not being able to save a created route), as such, any progress made may be lost in the event of a disconnection. External libraries or APIs used in the application may create some unexpected issues too, which may also have significant impact on user experience. Apart from these,

error prone situations such as invalid inputs will be handled by showing the user appropriate messages.

4. Subsystem Services

4.1. View

UI elements are stored under View component.

SignUpView: Where user signs up to our system.

LoginView: This class is for the View where user logs in to our system. Either by our system or with google, facebook, twitter accounts.

MainPageView: Where user what to do. They can either create or follow a route or choose one of the Top 10 routes listed here, they can see their token count.

RateRouteView: Where user rates a route, writes a comment and gets token rewards.

FollowRouteView: Where user sees the route he/she is following and tracks progress. They can leave a route early (leaveRoute()) if they wish. CheckPoint controls are done in this View (CPControl()).

ChosenRouteView: Where user sees detailed information
about a route they choose.

SearchView: Where user searches routes to follow. This
View contains two more Views which are FilterView and RouteView.

FilterView: When user wants to filter search results.

RouteView: Search results (routes).

CreateRouteView: Where user creates a route. User can leave markers and code tracks and shows users movements here.

CreateRouteDoneView: Where user enters detailed info about route selects checkpoints and make his/her route public.

FollowedRoutesView: where user routes they followed.

FavoritesView: Where user sees their favorite routes.

CreatetRoutesView: where user sees their created routes.

ProfileView: Where user sees their user information, edit it, goes to followed or created routes Views or goes to favorites View.
PlaceTokenView: Where user access their mobile phone camera and choses a token to place it on screen.
CollectTokenView: Where user access their mobile phone camera and collects a token from screen.
Search: Search functions.

4.2. Controller

4.2.1.Navigator

SignUpController: This is for creating a sign-up page when it is required.

LoginController: This is for creating a login page when it is required.

RateRouteController: When a user finishes his/her route, the route will be rated by the user. This function will provide the rate route page after completing a route.

Follow Route Controller: When a user wants to follow a route for collecting tokens, this page will be created according to route s/he wants to follow.

MainPageController: Navigator will create this page after the user logs in.

ChosenRouteController: This page will be created when the user wants to see details of a route. It will be displayed after any possible selection point on other pages.

CollectTokensController: When the user wants to collect a token along the route, this function will create the panel. The panel will include the token in augmented reality.

PlaceTokenController: After the user finishes creating his/her route, some tokens must be placed before publishing. This function will create the panel that shows the path.

CreateRoutesController: When the user wants to create a route and clicks the button, this function will display the create routes panel.

FavoritesController: When the user wants to see his/her favorite routes, This function will create the favorites panel.

ProfileController: When the user wants to see his/her profile, this function will create the page.

FollowedRoutesController: This function creates followed routes panel when the user wants to see.

CreateRouteController: This function will create the create route panel.

CreateRouteDone Panel: After creating the route, this function will create the page for details of the route.

4.2.2.Model Functions

Our project consists of three main functions. They are route functionality, token functionality, and user functionality. These functionalities contain functions for checking, regulations, calculations.

4.2.2.1. Route Functions

DistancePercent: This function will do checks for token balanced routes.

IncreaseCount: This function is to keep track of the routes that the user has followed.

Draw: This function takes the sample points of the route and draws it on the screen.

CreateRoute: This function is used when the user wants to create a route. It can be for publishing or personal use.

ExitRoute: If a user wants to exit from a route before finishing it, s/he can exit with that functionality.

4.2.2.2. User Functions

AddToFavorite: This function enables the user to save his/her favorite routes.

WriteComment: When the user finishes following a route, a comment is required from him/her. This functionality is important for our project because other users may want to use the same route.

RateRoute: This function will take the user rating for other users to see.

EditProfileImage: User can change his/her profile image with that functionality. The image is kept in the user's device.

EditAboutSec: This function is for editing the user's information.

ScoreCalculation: Our score calculation also affected by the distance user traveled. This function calculated that score based on the distance.

GetAchievements: This function is for getting the user achievements based on user score.

4.2.2.3. Token Functions

UpdateTokenCount: Users are able to publish a route based on their token number that is taken from the routes. This function is for updating it.

CalculationPrize: Since the users can leave from following a route early, they can get partial points from a route.

PlaceToken: When a user wants to put a token while creating his/her route that will be published, this function will do that.

TakeToken: This function will be used when the user looks a token to be collected from his/her phone.

4.2.3.Query Functions

These functions will be used when the navigator needs to access the database information. The corresponding query functions will be in this subsystem.

4.3. Model

This includes the objects of the system. They are token, route and user. Since the navigator does its operations with objects, we should get those objects' information and create them in our system to make it efficient.

5. New Knowledge Acquired and Learning Strategies Used

We have to decide carefully which framework and SDKs we should use as Nomad is a web based AR application. For this purpose, we will be learning about frameworks React Native and Titanium and Google's ARCore SDK.

React Native is an open source mobile application framework and it is widely used since it supports cross platform development and has an active community which creates many libraries for various uses. Therefore, there are a lot of resources and tutorials for new starters to learn. It also requires high level of JavaScript knowledge. Since we will be working on a web based application, we will also use JavaScript. During the learning process, we will

be reading the tutorial guide of React Native's official site[5] and other well known useful sites like tutorialspoint[6] to see more examples. Appcelerator Titanium is also one of the frameworks that is widely used. It also is an open source application development platform. It allows us to create both mobile and desktop applications using web technologies[7]. We will be using its official site to watch tutorials[8][9]. We will do exercises with easy practises and decide which framework is more suitable for our project.

In order to include augmented reality to our project, we will be using Google's ARCore. It allows motion tracking, detecting the surfaces and estimating the lighting conditions of the environment. Since the application allows users to place and collect objects with AR in anywhere around the world, ARCore will be the SDK that will provide us the most convenience for the project. We will be reading Google's ARCore documents[10] and searching other sites like HeartBeat[11] that gives basic examples to practise first. We will be also in contact with our supervisor and innovation expert, who have experience with AR, to consult when it is necessary.

6. References

[1] J. Fund, “Travel Is So Much Better Than It Was,” National Review, 02-Jan-2017. [Online]. Available: www.nationalreview.com/2017/01/international-travel-today-much-easier-cheaper-it-was/. [Accessed: 12-Oct-2019].

[2] Github repository of “js-base64: Base64 Implementation for Javascript”. https://github.com/dankogai/js-base64. [Accessed: 27.12.2019]

[3] npm (node packager module for Javascript) statistics for “js-base64” library. https://www.npmjs.com/package/js-base64. [Accessed: 27.12.2019]

[4] KVKK – Kişisel Verileri Koruma Kurumu. https://www.kvkk.gov.tr. [Accessed: 27.12.2019]

[5] Facebook, “React-Native,” [Online]. Available: facebook.github.io/react-native/docs/javascript-environment.html [Accessed: 10-Nov-2019].

[6] Tutorialspoint, [Online]. Available:]https://www.tutorialspoint.com/react_native/index.htm [Accessed: 10-Nov-2019].

[7] Value Coders, “Mobile Frameworks to Know in 2019,” [Online]. Available: https://www.valuecoders.com/blog/technology-and-apps/top-javascript-frameworks-for-mobile-app-development/ [Accessed: 10-Nov-2019].

[8] Appcelerator, [Online]. Avaible: tutorials.axway.com/search/Titanium [Accessed: 10-Nov-2019].

[9] Appcelerator, [Online]. Avaible: https://www.appcelerator.com/Titanium/ [Accessed: 10-Nov-2019].

[10] Google Developers, “Developer Guides”, [Online]. Available: developers.google.com/ar/develop/java/quickstart [Accessed: 10-Nov-2019].

[11] HeartBeat, “Build your first Android AR app with ARCore and Sceneform in 5 minutes.” [Online]. Available: heartbeat.fritz.ai/build-you-first-android-ar-app-with-arcore-and-sceneform-in-5-minutes-af02dc56efd6 [Accessed: 10-Nov-2019].