1. **Web API**

We have created the web service from using a domain from an online PHP IDE and a service provider, Cloud9. We had to use some languages that we do not know yet and the progress done made us understand about the programming concepts and design patterns better in many ways.

* 1. **MySQL**

We have chosen MySQL as our main database system because MySQL still holds 70% of the whole database businesses all over the world. Even though some big Tech Firms still use MSSQL for the reliability issues, MySQL is easier in notations and cross-platform usage.

First, we have created our table “Users” and gave it some attributes, such as, username, password, email, age, gender and validation. Then we moved on to the search cases for our login and sign up activities. In this stage, finding use cases were easy but the hard part was implementing these queries in a hybrid language of PHP and SQL.

Then, we have created our PHP file in Cloud9. This file is where all the “post” calls will be hold and where all the SQL operations will be performed. Learning PHP after some high level languages was not easy either. As it is so straightforward, it took a little while for us to start implementing PHP.

As we managed to create this PHP file, we synchronized the data in our MySQL server and PHP file by using Redbean add-on, only created for Cloud9 IDE. Redbean.php only connects a MySQL database with Cloud9 by creating an object for an “innoDB” and the only thing we had to do was entering root adress, password, user and port number of the database.

* 1. **PHP File**

In the PHP file, index.php, we have created three post methods. One of them is for login stage, another one is for signup stage and the final one is for searching buses from a different database.

Login-post method takes a JSON Object from Android and gathers the data for username and password as “username” and “password” respectively. Then serializes them for the database and sends the message to database via Redbean object. Takes the database’s response as integer 1 if the user exists, 2 if username is correct but password is wrong and 3 if both the information are wrong. The response integer-2 is used for security control. Then this method returns the value of response by serializing it into a JSON object.

Signup-post method takes a JSON object from Android and interprets it accordingly as {username, password, age, gender, email}. This method first acts like login-post method and checks if there are any usernames same as the chosen one and if there is no such username, the entries of user is loaded into database in the same way as login-post method. Only difference is that this time, the response from database is a Boolean expression. True if user signed up correctly, false if there is an error about sign-up process.

Final method, checkbus-post method is not fully implemented yet because of the coordination issues with the Bilkent University Transportation Unit. However, there will be a series of attributes to be checked; mainly, GPS data (coordinates) will be the attributes.

1. **Android**

This part is where it all comes together and creates an application and a user interface easy enough for everyone to use. Android part took a lot of time for us to implement because of compability issues, which will be addressed to in the next section.

**2.1) Utils Package**

This section of application is only about different operations and features about network issues. There are 6 classes and an interface in this package. The task of each class is listed below:

**General Class:** Stands as the most general class, at the top of the hierarchy and the reason we did this is that we create the main network object here and then we can use it in any other class without re-declaration or without the need of transferring content from object to object in each activity.

**HttpClientHelper:** This class only declares the context of the REST control object and gives some minor support for other classes.

**UtilsGeneral:** There is only one method in this class so far, but we plan on implementing general fix methods in here rather than changing whole implementation of a method or a class for minor network incompabilities.

**UtilsGson:** This class creates a GSON object for various web searches like system date time corrections and traffic announcements.

**UtilsRetrofit:** This class is one of the most important of all. This class holds 2 methods in.

**- OkHttpClient createClient (Context context):** This method creates a client with using OkHttpClient object and makes the proper bindings with proper protocols according to the host it connects and then deals with connection, time-out issues and network operations.

**- Retrofit createRetrofitAdapter(Context context, String baseURL):** This method takes an OkHttpClient and takes a base url as argument, then builds a connection from Android side to the client side which is already built in the previous method.

**NetworkInterceptorHelper:** This class extends Interceptor class which is the control class for network connectivity. These methods throws an exception is the device is not connected to the internet.

**RestControllable:** This class is where the base urls of the post methods are held. There are currently 2 in use urls but there will be 3 when the implementation is completed.

**2.2) Models Package**

This package holds 2 classes so far, LoginValue and User. This package will hold Bus, Admin, Stop and Location too when finished.

User class holds 5 attributes; username, password, age, gender and email. We have discussed about the validation attribute for this class to define the usability state of the registered user, then we decided to do the validations manually for now. We have created the validation Boolean in the database though. This class is important for exposing the attributes to JSON object. There are serialized names of the attributes in this class.

LoginValue class holds the model for the response of the Login-post method from PHP file. This model is a basic class with a constructor and one accessor for the only value it has, an integer.

**2.3) Activities Package**

There are 2 in use activities in this package right now. LoginActivity and SignupActivity. These classes are where the View part and the Model part comes together and gets manipulated by Controller part. This is where **MVC design pattern** is completed.

**LoginActivity:** This activity controls the the graphical interface that is created in the XML form. There are id’s for example components and this class takes the inputs from XML file and gives commands on other utility classes to form JSON object of username and password. Then sends it to PHP via REST control objects. On positive response, this class directs the Android Manifest to the next class on the list, SignupActivity.

**SignupActivity:** This activity controls the graphical interface created in XML form. Every attribute in xml is connected to the SignupClass written in Java. This activity enables the user to sign up to use our app. Activity also checks entered values and gives attribute error messages if they don’t meet the requirements like password should be at least 8 characters or an invalid address is entered. However, since only Bilkenters will use the app there is also the special requirement that the email address entered is of Bilkent, meaning that it has to have @bilkent.edu.tr in it. If all user inputs are correct, the user can click on signup and the attributes are created to a JSON object and sent to PHP code that connects to the external server via the REST control objects which then directs to LoginActivity for the login. However, since server connecting is not complete at the moment the program crashes once we click Sign Up.

1. **Problems Encountered**

Our main problem in this project so far is learning the Android development logic and process. Although the language is Java, as we all know Java, other constraints are so different. The declarations, activity system, Gradle Scripting and Android Manifest management are all new to us and we work hard to understand all these concepts in such limited time.

On the other hand, Android development showed us a new style of implementation and planing with this course. We have to plan far ahead, consider our extra packages, add-ons, widgets, gadgets and so on, because there is this limitation called Android Version. If we want to make this application for everyone, we have to use many old features for compability with each phone which are supporting old versions of Android. But if we want to make this application as pretty and as compact as possible, we have to trade our user pool with latest versions of Android.

These trade-offs and planings are teaching us more and more about Client-Programmer relationships each day and this knowledge surely will be usefull in our future projects.