CS 319 - Object-Oriented Software Engineering  
Analysis Report  
  
  
Where is My Ring  
Group 1-C  
Hamza Akbar



Emre Dikeneli  
Bahadır Durmaz  
Ali Soyaslan

1. **Introduction**

“Where’s My Ring” is an Android application we are designing for Bilkent University students. Its primary purpose is to tell students in real-time where each Bilkent University’s bus is at the time. This information given to the student using the app will be in the form of a map that will show the location of each bus, and the time it will take a particular bus to reach the student’s location. We chose this app because of late in Bilkent University there have been discrepancies in the buses’ stated arrival times and their actual arrival times due to several factors. By making this app, we hope to alleviate the problem, if not completely eradicate.

1. **Overview**

The app “Where’s My Ring” will make use of three technologies: one, Android library on Android Studio IDE; two, PHP for server-side scripting on CLOUD9 IDE; three, MySQL for database management. We plan to use a remote server, but if that’s not possible, we will establish the server on our own computer, which will serve as both the server and the client.

“Where’s My Ring” will be a simple app; it will show the location of all Bilkent University’s buses on a map, and if a student wants to know the arrival time of a bus, then the app will ask the student’s current location and where the student wants to go in the form of 5 options: “Nizamiye”; “Tunus”; “Sihiyye”; “East Campus”; “Main Campus.” After choosing one of the 5 options, the app will tell how much time it will take until the bus to come to the student’s location.

For tracking Bilkent University’s buses, we need real-time GPS data of the buses. Towards this end, we spoke with Bilkent University Transportation Department and they agreed to provide us with the data; however, later they told us that they do not have the data because the bus drivers don’t turn their GPS devices on, and so due to this problem, we will simulate the data on our own so that simulated data can be fed into our app. Furthermore, we will map the simulated data onto a map through Android’s GPS API. The app will only be available to Bilkent University students for using, as the app will only accept Bilkent University’s webmail addresses during signup; any other email addresses will be rejected.

* 1. **Authentication**

Authentication for this app should be accurate because Bilkent University does not permit people other than its own students and staff to use the services. This process will require students to sign-up with their official Bilkent University webmail ids, and if a user uses an id other than the webmail id, it will be rejected. On a successful signup with a webmail id, the user will get an authentication email confirming the signup.

* 1. **General Map Display**

This function will display city map with colored pins, which represent buses, showing the real-time locations of the buses. Users, whose time schedule is flexible, can use this option to schedule their travels (i.e. to city, to A.Ş.T.İ). If the user touches on any of the colored pins on the map, a dialog box or sliding panel will appear and show the specifications of the bus, driver and remaining time to next stop and final destination.

* 1. **Select Bus**

This function is for students with tight schedules, and particularly for exchange students who can’t navigate the city. On the main page of this function, there will be a few fields to fill, such as, “From where, to where.” Then the application will choose the best bus for your trip and show the alternatives for the trip.

* 1. **Bus Types**

There are 2 types of Bilkent university buses available for transportation. On-campus (ring) buses are buses that only operate within the campus, usually commuting students between Main Campus and East Campus. Off-campus buses, called SMD and TMD, commute students between different locations in the city and Bilkent University campus.

* + 1. **On Campus**

On-campus buses will probably have a lower usage than off campus ones. This is so because the maximum waiting time for these buses is usually 4-6 minutes. This function is good for catching classes in breaks. This function could also be used as an alert mechanism for the rings in the feature.

* + 1. **Off Campus**

Off-campus buses are most often used by students for commuting between the city downtown and Bilkent University. These buses are of two types: SMD and TMD. SMD buses travel to and from Sihiye Merkez and TMD buses travel to and from Tunus. Most of the interaction made by students in our application will be with these buses’ representation on the map.

1. **Requirement Specifications**
   1. **Functional Requirements**
      1. **Login**

The login information is gathered from a database that we are going to build with MySQL. Before the login phase, there should be a signup phase for the new users. The access to our database is protected by standard HTTP and the tools we will use will give us additional functions for data protection. So, our database will be secure enough against penetration or injections.

* + 1. **Signup**

During signup, users will be authenticated, and this will allow only Bilkent University students to use our app. The authentication will be done automatically by our app as only Bilkent University webmail id’s will be accepted during signup, and since only registered students of Bilkent University get a webmail id, only students will be able to sign-up.

* + 1. **Map Activity**

This activity (means page in Android platform) will show the user the map, and buses on it which are represented by colored pins. The map will be the system map of Google Maps and the pins on it are supported by the system map itself. The bus data will be simulated by us and put on the map with a locate function. Users will be able to see all the buses on this activity and plan their trips. The update speed of GPS data changes in time so the application will send a ping over a fixed period of time, check the changes in data and then update the map accordingly.

* + 1. **Select Bus Activity**

The selection of buses is easy enough for everyone to use. Users only have to select “from” and “to” of the bus specifications. The application will show the user the bus best suitable for the journey and display the alternatives. Furthermore, it will tell the User how long it will take before the bus will reach the User’s location. User’s selection will filter the buses on the map and then check the GPS data for time issues.

* 1. **Non-functional Requirements**
     1. **Application Performance**

These days mobile application performance has the highest priority, and so we will implement our application as optimized as we can. We are going to use different tools and packages that free the memory off some unnecessary data. We will avoid instant generation of data and store them in a temporary cache so that our application does not affect the refreshment and usage of other background applications.

* + 1. **User Friendly Interface**

We want make our application’s user interface simple enough so that it is easily navigable by all users. For this we will position all graphical icons appropriately on the map, make all graphical icons large enough to be interacted with by a finger bud, and we will use scroll pages at a minimum.

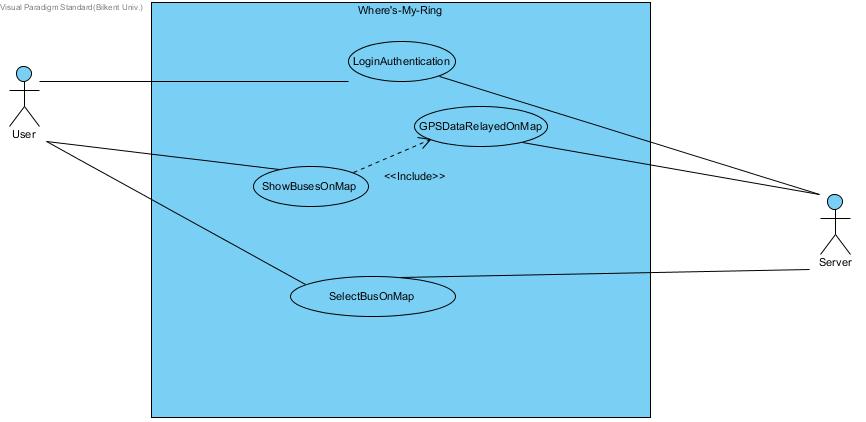
* + 1. **Modifiability**

As new versions of Android are frequently released with new features, our application will be modifiable and patchable in order to accommodate new functionality.

**3.2.4 Extensibility**

Our application will be extensible so that extra application features can be added to match changing user needs.

1. **System Model**
   1. **Use Case Diagram**



Use Case Model of Where’s My Ring

**4.2. Textual Description of Use Cases**

**4.2.1. Login Authentication**

Use Case Name: Login Authentication

Participating Actors: User, Server

Flow of Events:

1. The User inputs his Bilkent University webmail ID and password.

2. The Server checks whether the login information is correct.

3. The User successfully logs in on “Where’s My Ring” if the information is correct. If the information is wrong, the User is denied access.

4. On a successful login, the User is taken to the main screen of the app.

Entry Condition: The User has opened the app on his Android device.

Exit Condition: The User has successfully logged in.

**4.2.2. Show Buses On Map**

Use case name: Show Buses On Map

Participating Actors: User, Server

Flow of Events:

1. The User chooses the “Show Rings on Map” option from the app’s main screen.

2. The Server retrieves the GPS data and relays it onto the map.

3. The app displays all the buses on the map at the time.

4. The User views the interactive map with buses displayed.

Entry condition: The user has reached the main screen of the app and chosen the “Show Rings On Map” option.

Exit Condition: All buses operating at that time have appeared on the map.

Quality Requirements: The use case “ShowBusesOnMap” **includes** the use case “GPSDataRelayedOnMap” at all times, as the GPS data allows the representation of buses in real time.

**4.2.3. Select Bus On Map**

Use case name: SelectBusOnMap

Participating Actors: User, Server

Flow of Events:

1. The User chooses the “Where’s My Ring” option from the app’s main screen.

2. The app asks the User his current location by presenting on a new screen 5 options: Nizamiye, Sihiye, Tunus, Main Campus, and East Campus.

3. The User chooses one of the 5 options.

4. The app asks the User where he wants to go by presenting on a new screen 5 options: Nizamiye, Sihiye, Tunus, Main Campus, and East Campus.

5. The Server retrieves the corresponding GPS data.

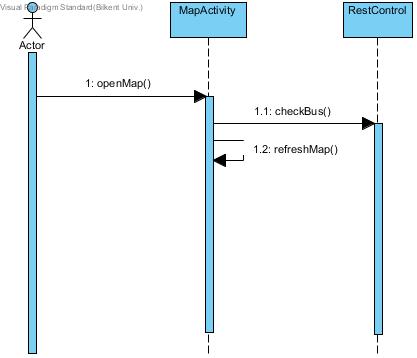
6. The app tells how much time until a bus will reach the User’s location.

Entry Condition: The User has reached the main screen of the app and chosen the “Where’s My Ring” option.

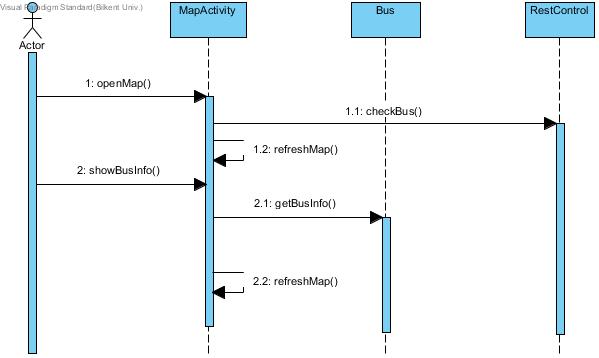
Exit Condition: The application has successfully displayed the time of the bus according to User’s inputs.

**4.3. Sequence Diagrams**

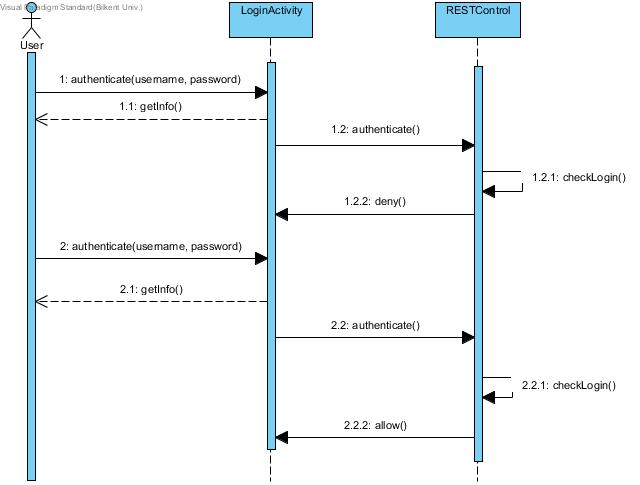
**4.3.1 Show Busses On Map**   
Following sequence Diagram illustrates the scenario explained below:  
Scenario:   User Alice requests to see all the Rings’ whereabouts on map. To do that Alice clicks on “Show Rings on Map” button from SelectBusActivity. Then SelectBusActivity calls showMap() to trigger OpenMapActivity. After that OpenMapActivity, utilizing the RESTControl interface, calls checkBus() and checkGPS(). checkGPS() fetches the locations of the busses and checkBus() fetches the data of the busses (license plate, driver, where its heading…), the data is returned to SelectBusActivity. From that info, SelectBusActivity class updates the Busses. Then busses appear in the OpenMapActivity using the Google Maps plugin. If the user clicks on a certain bus then showBusInfo() is called and the bus info appears.



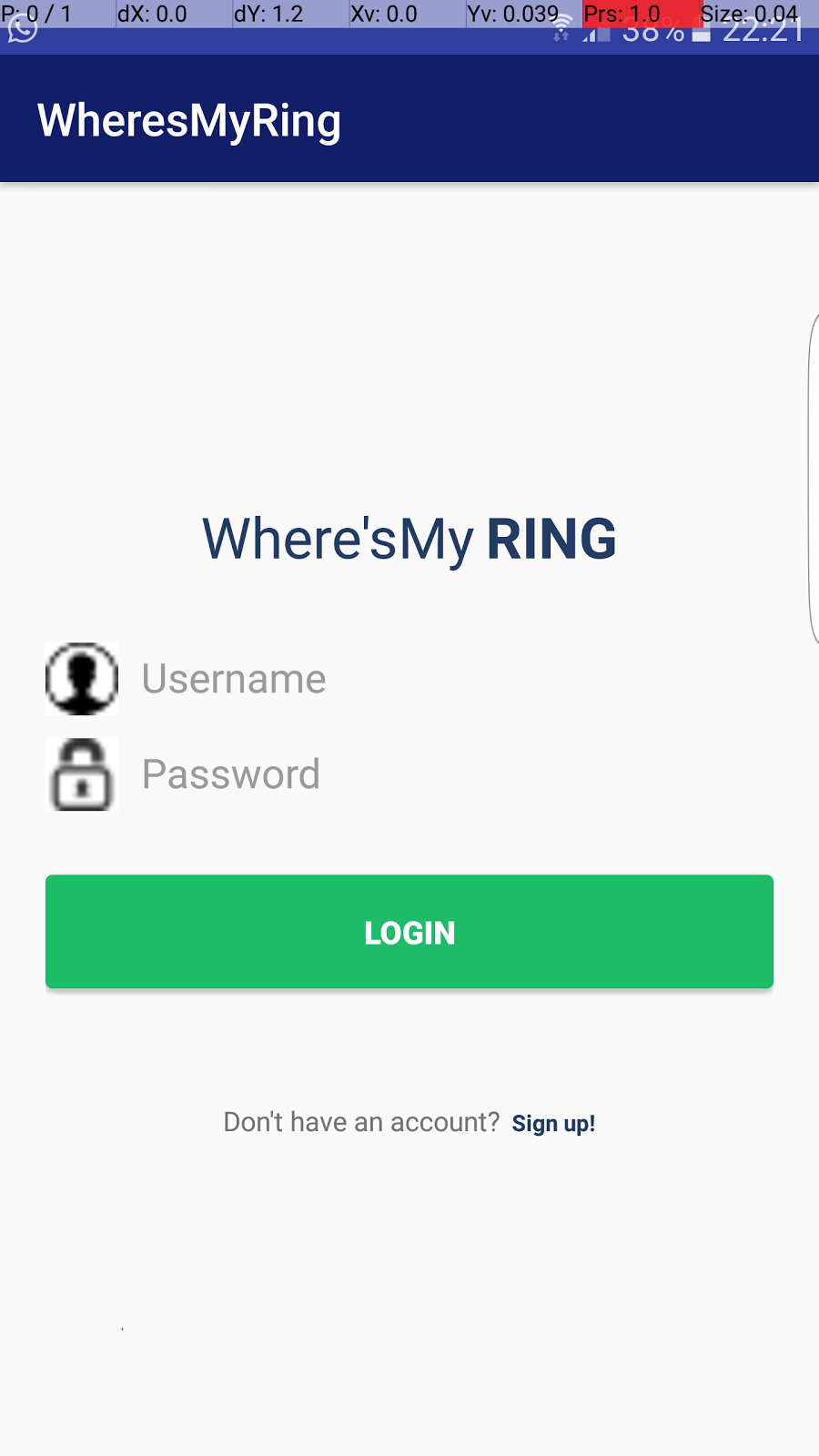
**4.3.2 Tell When Nearest Bus Will Come**  
Following sequence Diagram illustrates the scenario explained below:  
Scenario:   User Alice requests to see when the next she wants to board Ring will come to her destination. Alice clicks on “Where is My Ring” button from SelectBusActivity. Then a new window in opens SelectBusActivity and she clicks on the location she is, out of five options (Main Campus, Tunus, Nizamiye, Sıhhiye, East Campus). What she selects is initialized as the variable “from”. Then she clicks on where she is heading out of the same options and that is initialized as the variable “to”. Then checkBus(), in accordance with the variables to and from, fetches the location of the nearest Ring and calculates when it will arrive to the bus stop from it’s speed. Lastly, SelectBusActivity displays when the bus will arrive to Alice’s destination.



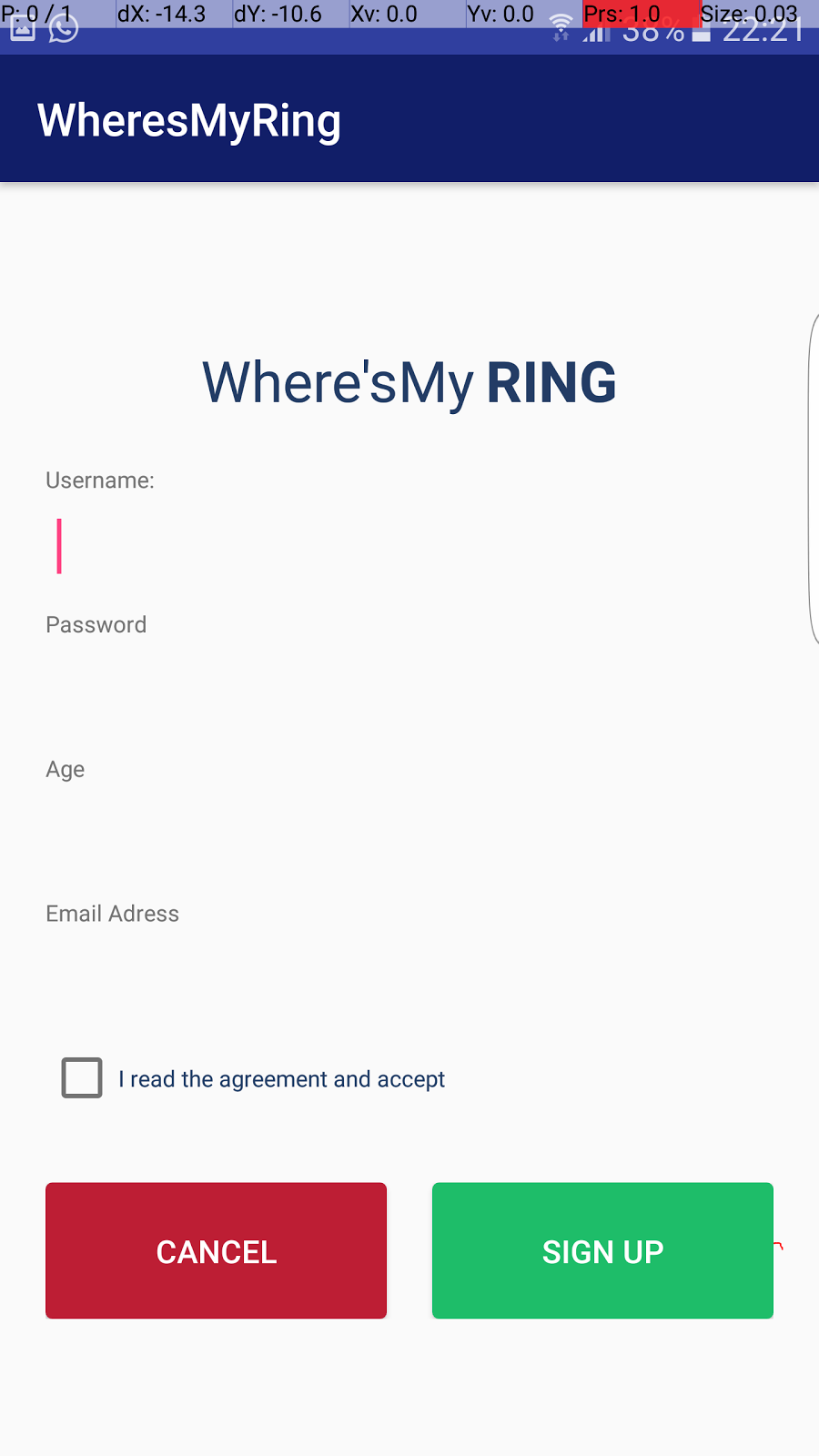
**4.3.3 Login**   
Following sequence Diagram illustrates the scenario explained below:  
Scenario:   The login information is gathered from a database that we are going to form with MySQL. Before the login phase, there should be signup phase for the new users. The access to our database is protected by standard http and the tools we use also gives us some extra functions for protection of data. So, our database will be secure enough against penetration or injections.



**4.4. User Interface**



Login Page: 1 time activity in which user has to login to verify his/her credentials. After which the login procedure will not be repeated and the user will be directed to Map Page at the start. If the user does not have an account set they will click the Sign Up text to be directed to the signup page.

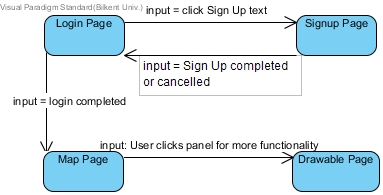


Signup page: One time sign up to make sure user is from Bilkent. The user will be authenticated as a Bilkenter from the email in which if it doesn't belong to the @bilkent.edu.tr domain it will not be accepted. Once signup is completed a verification email will be send and sign up will be completed. If the user clicks sign up with sufficient info like accept the contract, good password, etc. then the user will be directed to the login page with a toast (short notification in Android) saying that sign up was successful. If they click cancel or back they will return to the login page with a toast notifying that the signup procedure has failed.

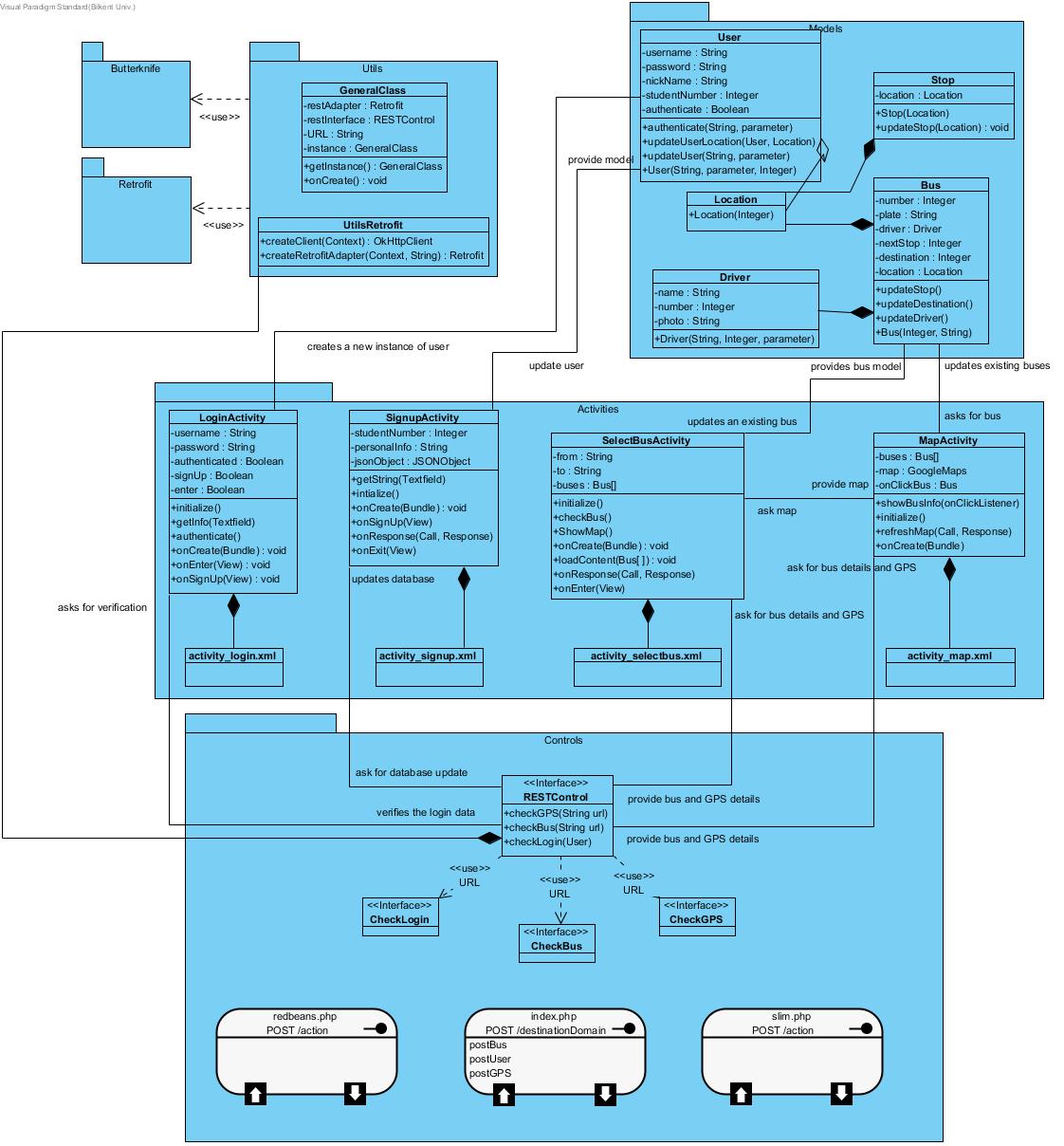


Map Page: This page show all rings according to their whereabouts and the stops on a Google Maps client. After the signup and login procedures this will be the main page user will interract. Though not  yet implemented when finished all the moving busses of Bilkent will be displayed on the map with the user being able to click on each of them to see info like ETA, license plate, where it’s heading. User will be able see their location by clicking on the position button. The 3 parallel lines on the left uppermost corner when clicked will open a drawable window with options such as settings and a wizard that calculates the best route according to their destination (we already know the user’s whereabouts from GPS).

State Machine Diagram:



**5. Object and Class Model**



Class and Object Model of Where’s My Ring

The object model of Where’s My Ring is illustrated above. In Where’s My Ring implementation, there are 12 classes in total.

We are using Model View Controller design pattern, so the 4 classes listed below are Model classes.

**Bus** class represents and manages the buses in Where’s My Ring. The Bus class contains all the information about the bus. The Bus class has a composition relationship with the Location and Driver classes.

**Location** class represents the GPS coordinates of the bus. It manages the location of each bus on the map in real time.

**Driver** class represents and manages the drivers of the buses. It has name, number, and photo attributes.

**User** class represents and manages the users of the app. It has an aggregation relationship with Location class.

**Stop** class represents and manages the stops on the map. It has a composition relationship with Location class, since a stop always has a location.

The 4 classes listed below are View classes. It is worth mentioning that all these 4 classes are Android platform classes. Each of these classes is composed of its relevant XML class.

**LoginActivity** class represents the Login page of the app. It has an association relationship with the User class for the function of logging-in and authenticating user login.

**SignupActivity** class represents and manages the Signup page of the app. This class manages the signup process of users and has an association relationship with User class.

**SelectBusActivity** class represents and manages the real-time operation of buses in the app. This activity is crucial in our app as it shows the interactive map in our app and provides interactivity with the bus icons on the map. This class has an association relationship with the Bus and OpenMapActivity class.

**MapActivity** class represents and manages the map in the app. It has an association relationship with the Bus class and SelectBusActivity.

Then we have a Controller class which is mentioned below.

**RESTControl** class is quite important as it updates the database, verifies login data, and provides bus and GPS details. In addition, this class uses CheckLogin, CheckBus, CheckGPS interfaces.

Lastly, we have 2 utility classes called **GeneralClass** and **UtilsRetroFit.**

**Important Decisions in Overall Analysis**

* We chose Model View Controller because this keeps the interdependence among our classes to a minimum and distinguished, as in Android the Activity classes have XML files associated along with the code. And since we are using PHP in our app also for database interaction, Model View Controller best served our purpose. Also this MVC design pattern adds well-defined hierarchy to our classes. Most importantly, Android Applications are by default in MVC design pattern, and choosing any other pattern would have deeply complicated the application development.
* Bilkent University Transportation Department initially told us that it had bus data, but when we approached them for the data for our app’s implementation, they changed their position and told us that they do not have the data. Due to this problem, we decided that we’ll simulate GPS data ourselves.
* We wanted to add more functionality to our app, but that would have complicated and defeated the primary purpose of our app, which is facilitating user’s commute in Bilkent University buses, so we kept functionality only related to the usage of buses. This also led our UI design to be simple.
* Lastly, we wanted to make our app’s UI visually appealing, but we decided that its visuals should be as close to the STARS app and online application that all Bilkent University students use.
* We chose to include a sliding menu panel on the map for choosing buses instead of an extra page. This will make our app faster in terms of runtime, and less complex in terms of design.

## 

## Conclusion

In this report , we have explained what our application does in the form of its overiew, its functionality in the form of functional and non-functional requirements, explained it’s underlying design in the form of object model/class diagram, shown its dynamic working in the form of dynamic models.

For requirements specification, we endeavored to include all the functionality that could ease student’s commute in buses. Since requirements specification forms the backbone of project development, we spent as much time on it as possible to really capture it completely and catch any inconsistencies and bugs right in the beginning.

The requirements specification naturally led to Use Cases, and so we addressed all possible use cases that capture the intended functionality of our application. Then we dealt with different parts of complexity of our application with Sequence Diagrams, State Machine Diagram and Class Diagrams.

In summary, we have spent enough time on perfecting our analysis, and we hope to implement our application exactly as we have defined in the analysis report.