Marauder’s Map

Iteration – 3

Final

Group 5

CS 6359

Section: 003

November 30, 2014

Project Members

Aishwarya Viswatmula (axv141930)
Bharat M Bhavsar (bmb140330)
Gurleen Chawla (gxc131530)
Jay Tarun Chheda (jtc140230)
Vishal Kankanala (vxk143830)

Acknowledgment

We are extremely grateful to Prof. Mehra Nouroz Borazjany for giving us the opportunity to do this project and also for her guidance.

We would also like to thank Mr. Harish Babu Arunachalam for his constant help and feedback.

We are thankful to the Computer Science department for providing us with all the required help and support needed for the project.

Once again, our heartfelt thanks to UTD for providing us with this opportunity and experience.

Abstract

UTD Marauder’s Map is a leaf from the Harry Potter series written by JK Rowling. This is an android application, which has a map of UTD that can only be accessed by UTD students and staff members with a Net ID used for the login. Every person using the application can be located on the map and their details can be viewed. Various important locations in UTD also can be sited on the map. Our motive for this project is to connect people and help them explore places on the campus. This project has helped us to understand the principles of Object Oriented Analysis and Design, which we have effectively implemented in our application. We have tried to re-create the magical world into a real phenomenon with the assistance of the technology available and hope it is beneficial and excites the UTD folks to use it.

Table of Contents

1. Executive Summary

1.1 Project Overview

The Harry Potter series by JK Rowling has been very inspiring in many ways. This project idea is a leaf from this book. The Marauders Map shows all the people in the university and where they are exactly located. A similar map for UTD is something we thought would be very interesting to do. The UTD Marauder's Map is a mobile application that will be platform independent. Through this application we can trace the location of our friends, professors and everybody else in UTD, who uses this application, on the map. Here we will be using mobile GPS to track the exact location of a person, and project it on the map using their names or icons.

1.2 Problem Statement

In the university, many students have difficulty during their initial stay. We wanted to minimize some of their difficulties through this project, which are listed below:

- ➢ To help students connect with each other via the details of each person on the map, like UTD NetID and social network id's.
- ➢ To locate friends on the map.
- ➢ To locate various places in UTD on the map.
- ➢ To get office details (like office hours, contact details, location) to various offices.

1.3 Purpose and Scope of this Specification

To help find different locations and to get connected to people by means of UTD email ID, Facebook® or LinkedIn® and to find the current location of other users in UTD.

In Scope:

This document addresses requirement specification of the project Marauder's Map for Fall-2014, Semester – 1 Group 5, Section: 003 of CS6359. Mobile Application with following functions:

- Map of UTD .
- Details of different offices, centers, etc.
- Location tracker with details of users to get connected.
- Application will be working for Android OS device only.

Out of Scope:

- No Non-UTD member can use the app as entry criteria will be to have UTD NetID.
- No elevation details will be provided for floors above 1st floor.
- Application will not work on Non-Android device.

1.4 Vision Statement

For project teams providing information technology solutions, the UTD Marauder’s Map is developed to include tools, techniques, guidelines, and instructions that enables:

- Tracking the location of a person inside the campus.
- Users to find places throughout the campus.

The purpose of this project is to create something new and innovative that gives us a real experience of something that is perceived to be magical. The UTD Marauder’s Map is a fun, creative application that will catch the eye of every UTD student.

2. Product/Service Description

This is an android application that the user can install and use on their android device. The android device on which the application is running should be GPS enabled. User should enable the GPS location settings and allow the application to track its location. Also, user should be UTD member, may it be student, faculty or staff, who has an active UTD NetID.

2.1 Product Context

This application is dependent on location sharing where you share your location with others and in turn you are able to see the location of other users using this application. Google Map® shows our location on the map but it does not have the functionality to check for other user's location. Also, UTD has an application that shows the location details of different buildings and areas which is static and not user friendly.

This application interacts with Google Maps® and integrates its feedback with internal application details to produce the output for the user.

2.2 User Characteristics

The application is accessible to students and professors who have the following attributes:

- Name (First Name, Last Name)
- Active NetID

2.3 Assumptions

Following are the assumptions for the smooth operation of the application:

- User should have an Android operated device.
- Device should be GPS enabled.

2.4 Constraints

- Application is expected to run on Android and not below 4.0 (Ice-cream sandwich)
- Application needs to have device ID and location information available.

2.5 Dependencies

This application requires continuous interaction with the server and updation of current location using GPS. Therefore internet connection is a must to run this application, Wi-Fi or mobile data connection. Also, to get the exact location, location settings need to be changed in mobile settings.

3. Requirements

- UTD has staff and students coming from different cultures and backgrounds. Sometimes it is difficult to start interaction with a stranger coming from a different place and culture especially when one is an international student. How to get connected? To solve this problem an application should be there to help unknown people to get connected on campus.
- Also, being new to UTD, we don't know exactly the different offices and their operational timing.
- How to find someone by not calling or messaging when we want to get connected by having their current location information? This application should provide the location details of other person.
- User should be able to use and understand the application easily. Minimum required input is active NetID as this application is for UTD only. Output should be current location of the user and location of other users who are currently using this application.
- While signing up for the application, the user is required to input details such as Name, NetID, Facebook® profile link or LinkedIn® profile link.
- NetID is mandatory while Facebook® and LinkedIn® profile links are optional for displaying.
- The location details such as office location and timings will be available at any time to the user whereas user details of other users will be available only when they are using the application.

Requirement 1:

User should be able to find all locations in UTD

Requirement 2:

Details of all locations should be available when it is looked up on the map

Requirement 3:

User should be able to get in contact with other unknown users through this app by the means of UTD NetID, Facebook® or LinkedIn®

Priority Definitions

The following definitions are intended to be used as a guideline to prioritize requirements.

1. Priority 1 – The requirement is a "must have" as outlined by policy/law

1.1 User should be able to see his current location on Map

1.2 User should be able to find different places on campus through the app

1.3 User should be able to find current location of his friends, professors etc. through the app

1.4 User should be able to get connected to other users by means of NetID, Facebook® or LinkedIn®

2. Priority 2 – The requirement is needed for improved processing and fulfillment of the requirement will create immediate benefits

2.1 Login should be made user friendly.

2.2 Map details need to be added after the main functionality is achieved.

3.1 Functional Requirements

The application allows us to do the following:

1. <u>Creation of Users:</u>

- Any user who wants to use the application must register in the system first. The application will offer a web interface to enter the following things:
    - First Name
    - Last Name
    - UTD Email ID
    - Phone No.
    - Facebook profile link
    - LinkedIn profile link
- Then the user receives a mail to his UTD mail account which has a passcode of 6 digits (Passcode is randomly generated by the ASPX page).
- The user needs to enter the passcode on the web interface. Upon validation of passcode, the user details are recorded in the database.
- The device ID of the user is also recorded. So, the user need not login (sign in) every time he wants to use the application.
- Only if the user logs out (signs out) of the application, he has to login (sign in) again.

2. <u>User Login and Validation:</u>

Note: Only if the user logs out (signs out) of the application since the last time he used it, then he needs to login (sign in).

For login (sign in):

The user must enter his Net ID. Then the device ID of the user is compared with the device ID captured while registration. If the device ID matches, the user is validated and an email confirmation is sent to him.

3. <u>Sending GPS co-ordinates to a server using mobile data communication:</u>

- The GPS device (mobile) gets the co-ordinates (position) of the user and sends them to the server. The position must contain:
    - Net ID
    - Latitude
    - Longitude
    - Altitude
    - Accuracy
    - Timestamp.
- Then the user's position is updated in the database under his Net ID.
- If the Net ID of the user is not present in the system, it gives an error and the position is not recorded.

4. <u>Follow a user in real time mode on the map using GPS device:</u>

The position of the user is refreshed periodically using a "refresh time" parameter (refresh is done for every 1 sec) and is updated in the database. Therefore, a person can be tracked on the UTD Campus map.

5. <u>Track other users in real time mode on the campus map:</u>

Once a user updates his position to the system, he should be able to track other users on the map in the real time mode (To update his position, the user must have GPS enabled). If the other users don't have GPS enabled, their previously updated position is shown.

6. <u>Locate places on UTD Campus map:</u>

A detailed map of the UTD campus needs to be developed for the user to search and locate all the places. For important places and offices, the details (location, contact details, working hours, etc.) are shown for user's convenience.

3.2 Non-Functional Requirements:

These are the quality attributes and they do not affect the behavior of the system.
Following non-functional requirements will be there in our project:

1. Reliability

If there is an error we should have the ability to figure it out and reset immediately.

2. Accuracy

Accuracy of the position is very important as with accuracy the exactness of the position varies.

3. Maintainability

In order to make sure that code can be fixed and reused easily, we should separate the whole system into several independent functions. Then new functionalities can easily be added.

4. Portability

Avoid using classes and features that makes the application non-portable.

5. Reusability

Using it has relationships, Inheritance, Aggregation to accommodate reusability.

6. Performance

As position is updated for every 1 second, we need high performance of the system.

7. Application Compatibility

Compatibility has to be maintained so that upgrades and replacements are done easily.

3.3 Software/ Hardware Requirements

- We are creating a **hybrid application** that is platform independent. Hybrid apps, like native apps, run on the device, and are written with web technologies (HTML5, CSS and JavaScript). Hybrid apps run inside a native container, and leverage the device's browser engine (but not the browser) to render the HTML and process the JavaScript locally. The hybrid application framework that we are using is called **PhoneGap.**
- We are integrating **Google Maps** for the map interface.

  Software Requirements:
  - Operating System: Windows
  - Technology: .NET
  - IDE: Visual Studio 2013
  - Database: SQL Server 2012 Express
  - Server: utdmaraudersmap.somee.com
  - Android OS
  - Java

  Hardware Requirements:
  - Processor: Pentium architectural
  - RAM: 1 GB

4. Analysis

4.1 Domain Model

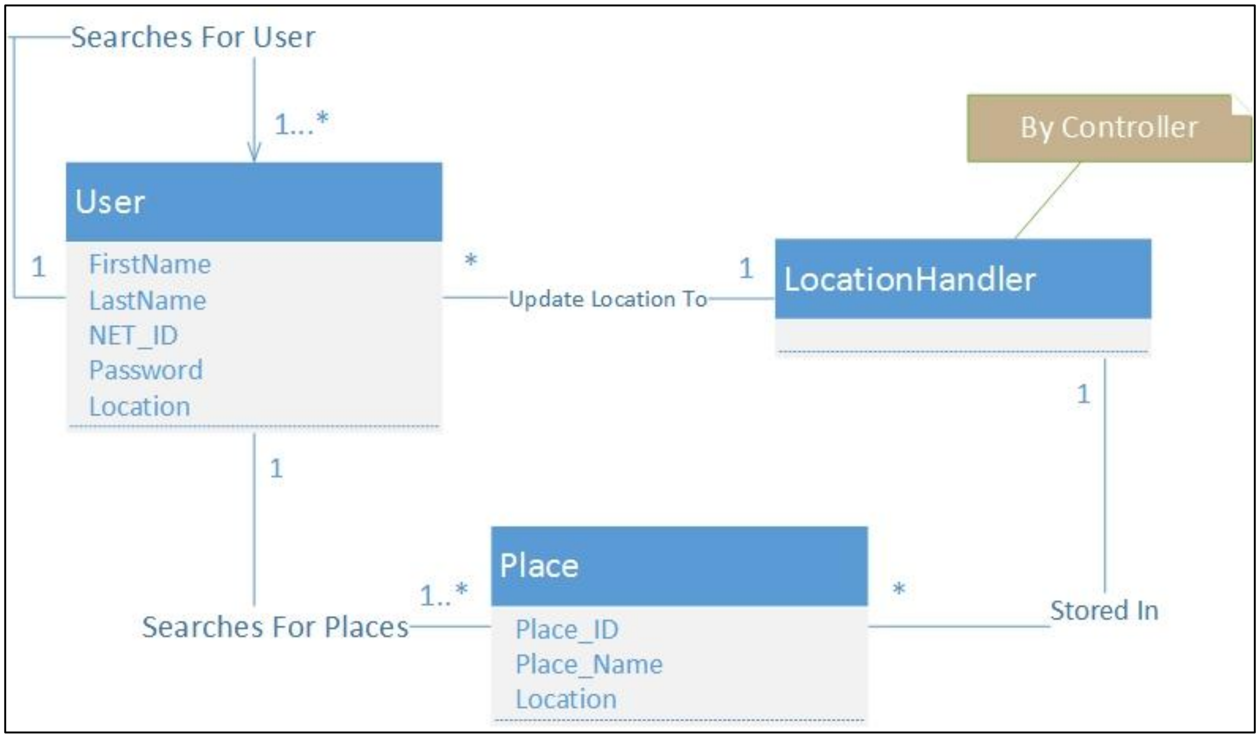


Figure 1: Domain Model Diagram

<u>Domain model text:</u>

'User' contains all information of all saved users in database. 'Place' is domain for all places, gives all details for stored places. 'Location Handler' handles both users' and places' location and it is responsible for updating locations for both domains. Cardinality is mentioned in diagram within those domains.

<u>GRASP Usage:</u>

All system calls related to location retrieval or modification are handled by Location handler and it controls input to database through DAL by creating different objects according to the system calls it receives.

4.2 Use Case Diagram

Core Functionalities:

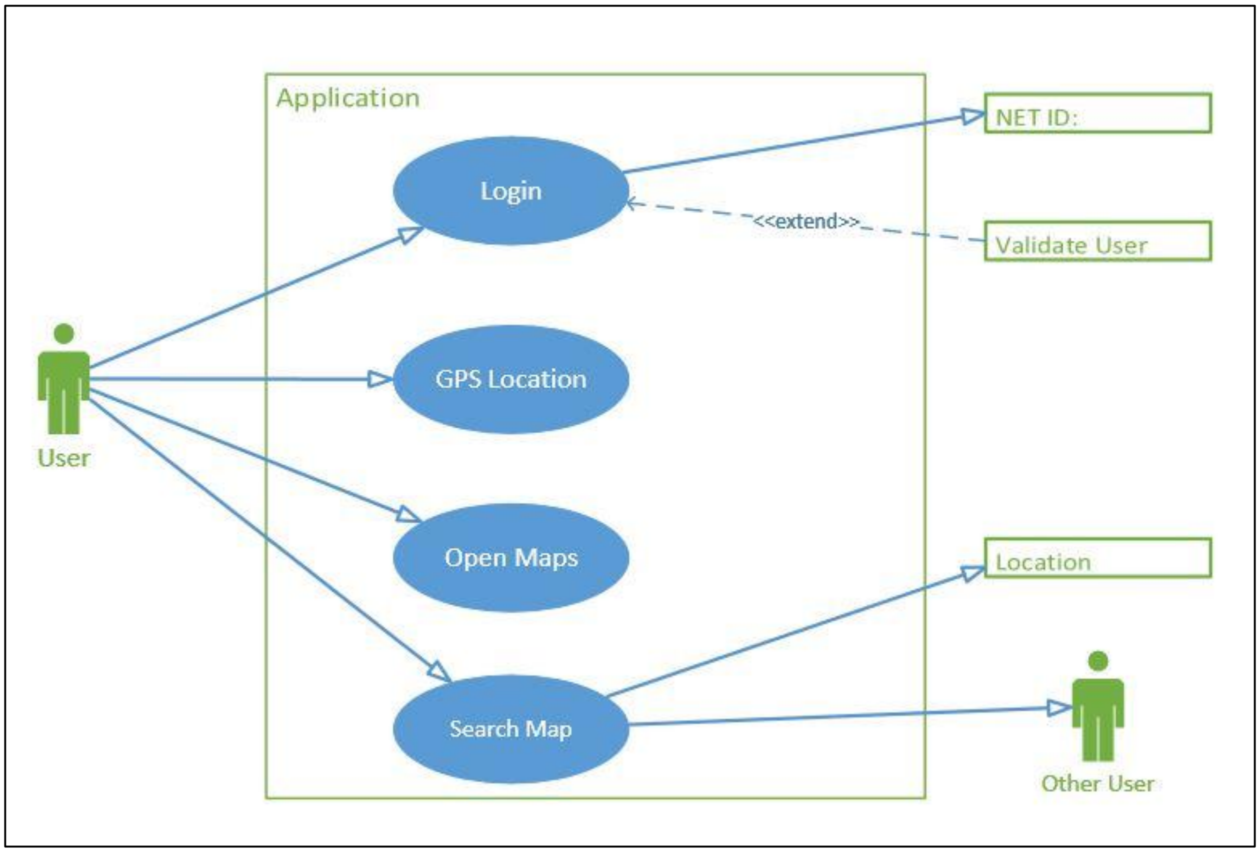


Figure 2: Use case for Core functionalities

Marauder’s Map application is having 4 core functionalities as mentioned in above diagram. User is the primary actor and database is supporting actor. The application is responsible for all core functionalities where it is interacting with database for validation of user and his location and also location of other users and places. These functionalities are discussed in more detailed further.

4.2.1 Login & Registration

4.2.1.1 Use Case Diagram 1

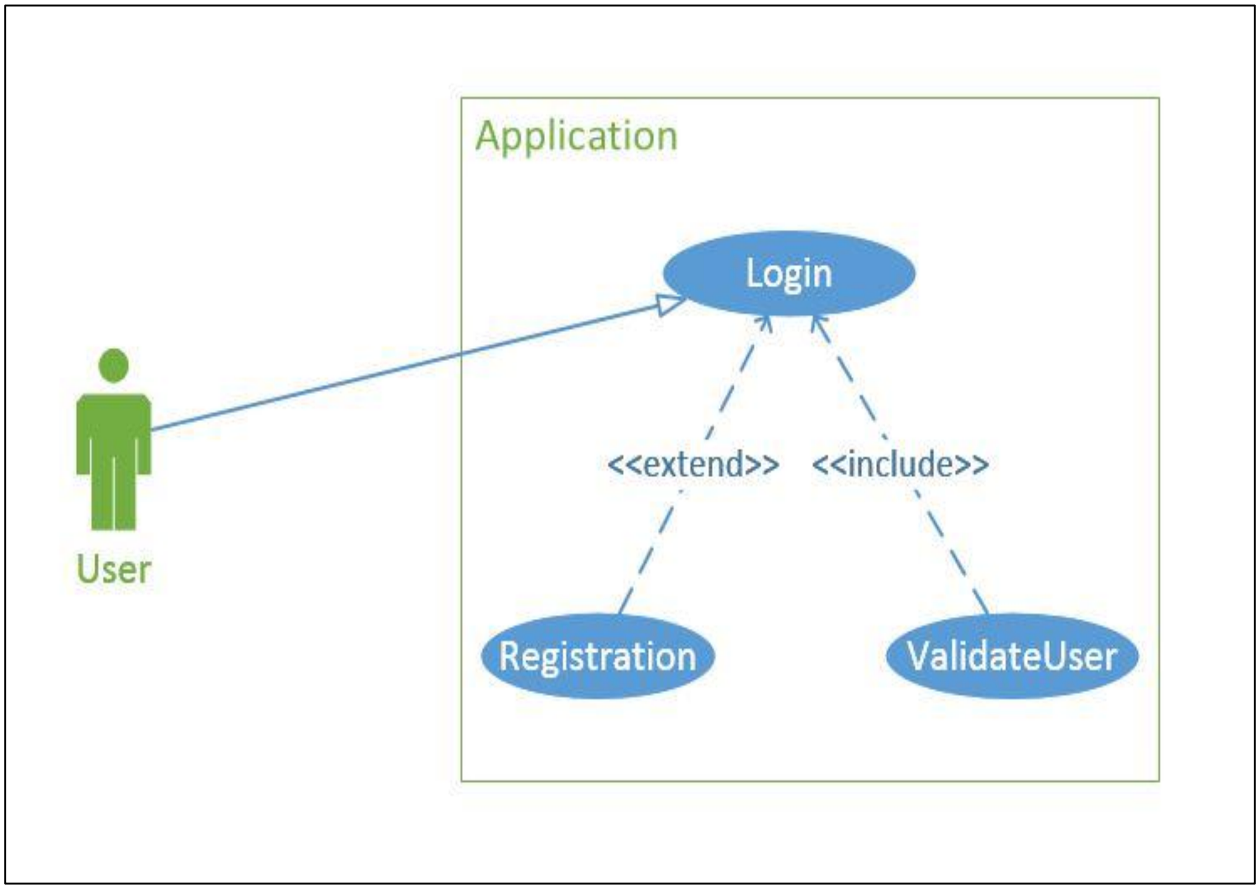


Figure 3: Use case diagram for Login & Registration

4.2.1.2 Fully Dressed Use Case 1

Use Case Name: Login & Registration Use Case

Scope: Marauder's Map

Level: User-goal

Actors: User (Primary Actor), Database (Supporting Actor)

Stakeholders: User: Registers for the application (or) Logs into the application

Preconditions:

- There is an active internet connection on the smartphone and the User is trying to access the application.
- The user has a valid Net ID.

Main Success Flow:

- The use case begins with user trying to register into the application (if accessing the application for the first time) (or) trying to login to the application (if the user is already registered).
- Use Case: Login and Registration into the application.
- **For Registration:** The user is given a web interface to enter the following things:
    - First Name
    - Last Name
    - UTD Email ID
    - Phone Number
    - Facebook profile link
    - LinkedIn profile link
- After entering the information, user receives a mail to his UTD mail account which has a passcode of 6 digits (Passcode is randomly generated by the ASPX page).
- The user enters the passcode on the web interface. The passcode is validated and the user is registered (recorded) in the database.
- Note: The device ID of the user is captured. So, the user need not login (sign up) every time he wants to use the application.
- **For Login:** The user must enter his Net ID. Then the device ID of the user here is compared with the device ID captured previously while registration. If the device ID matches, the user is validated and the email confirmation is sent to him.

Alternate flows (details left out):

- User not registered
- Invalid User
- No active internet connection

Post-Conditions (Success Guarantee):

- Successful Completion: The user is able to register (or) login to the application.
- Failure Condition: The user is not registered with the database or user is not able to login to the application.
- A UserPropObj was created.
- UserPropObj was associated with the account and authenticated via details (NetID) entered during registration

Special Requirements:

Only if the user logs out (signs out) of the application the last time he has used it, then he has to login (sign up).

Actors	Roles	Goal
User	Primary	To share location with other users and view location of other users.
Database	Supporting	To hold the authentication credentials and manage user details.

4.2.1.3 Sequence Diagram 1

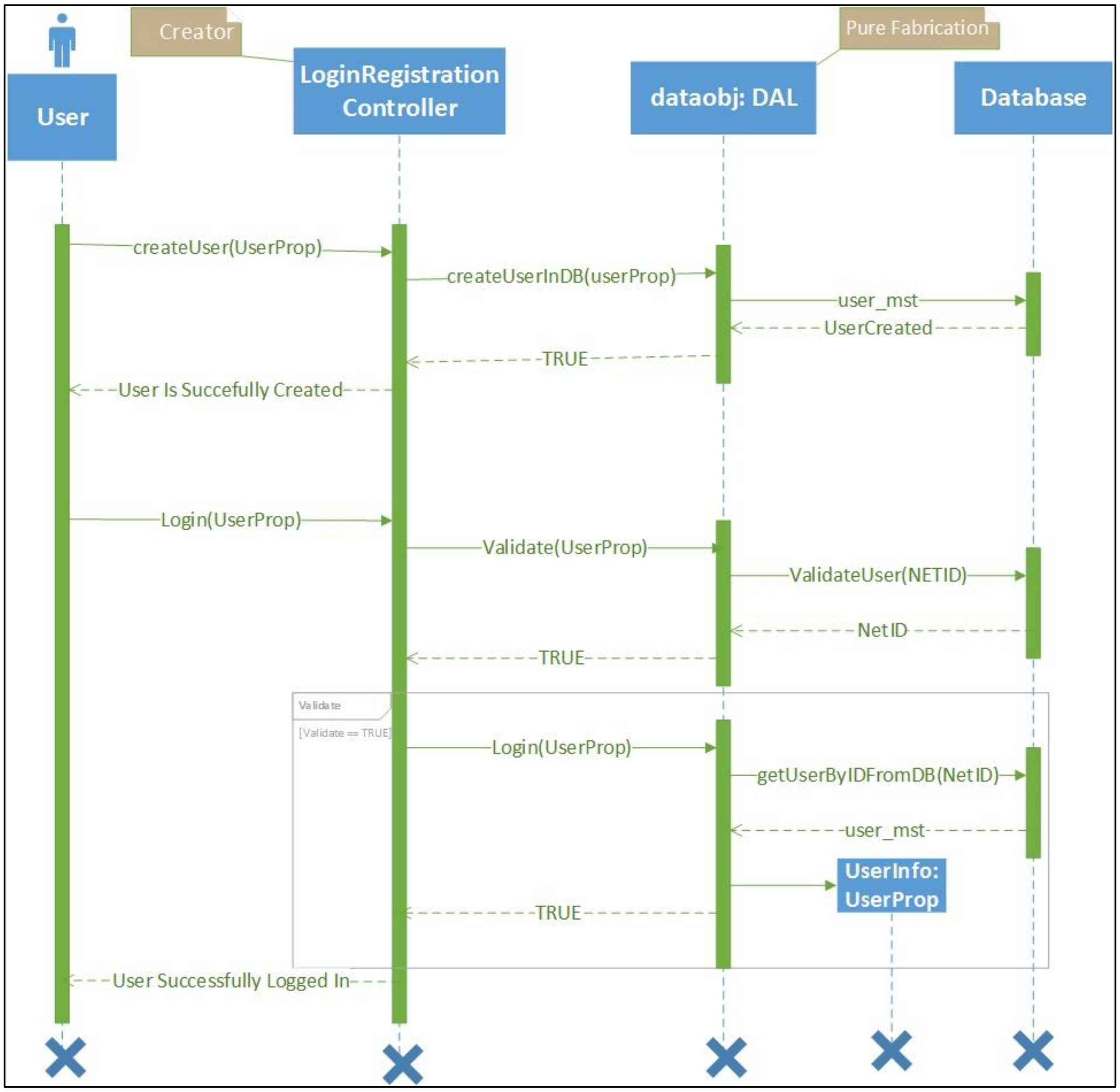


Figure 4: Sequence diagram for Login & Registration

<u>GRASP Usage:</u>

LoginRegistrationController:
This controller is responsible for creating the user object when user registers first time or logins into the system. Therefore it is identified as 'Creator'.

DAL:
As we are assigning responsibility to this intermediate object DAL to mediate all transactions between database and BLL so that they are not directly coupled, we identify this object as performing 'Pure Fabrication'. DAL performs Pure Fabrication for each transaction between BLL and database, hence marked under 'Pure Fabrication' in each SSD.

4.2.1.4 Operation Contract 1

- Operation: createUser(UserProp); login(UserProp)
- Cross References:
    - Use Cases : Login & Registration
- Preconditions:
    - The user has a valid netID
- Post-conditions:
    - A user instance (UserProp) was created.
    - UserProp was associated with the account and authenticated via details (NetID) entered during registration.

4.2.2 Update User Location

4.2.2.1 Use Case Diagram 2

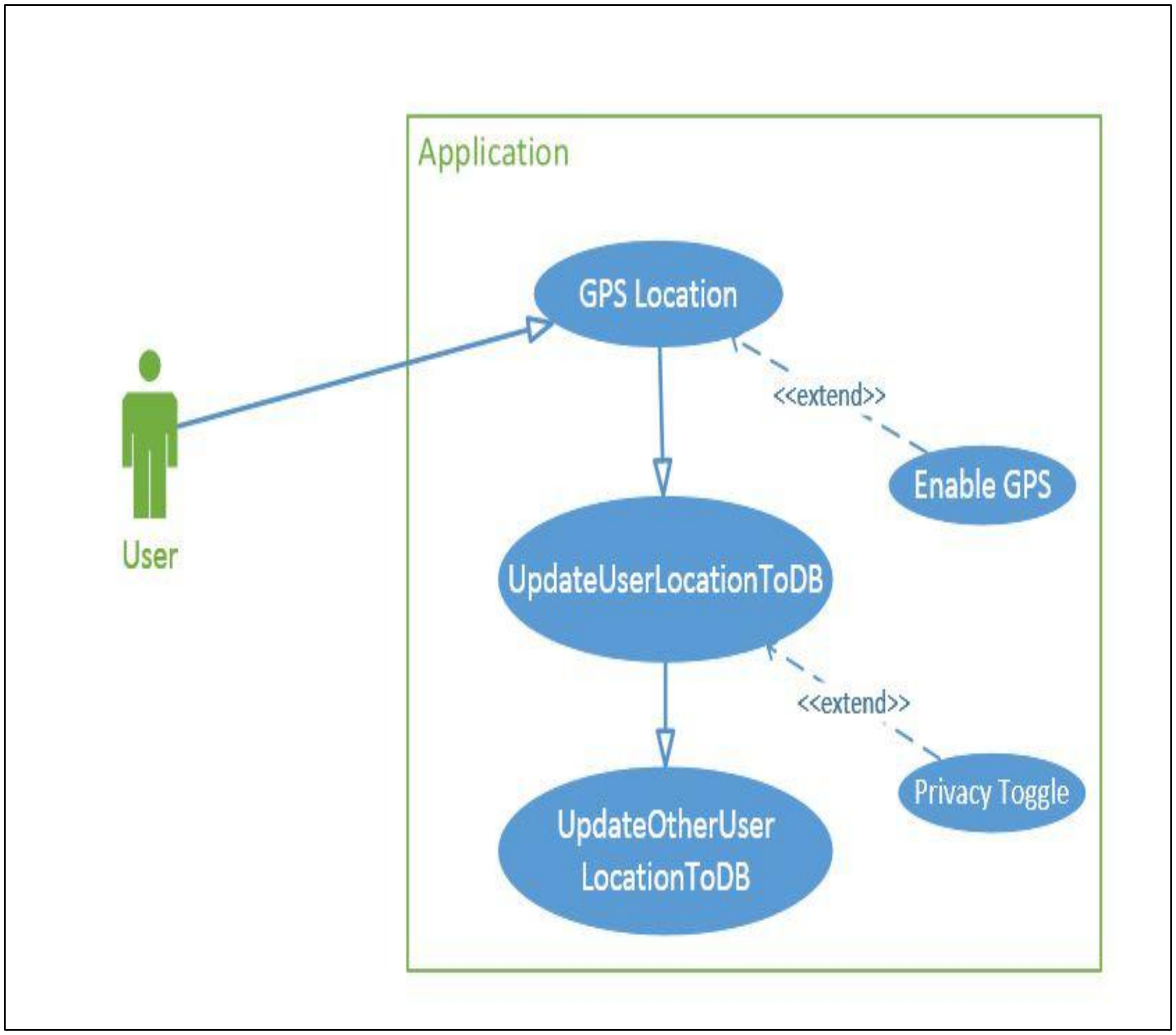


Figure 5: Use case diagram of Update user location

4.2.2.2 Fully Dressed Use Case 2

Use Case Name: Update User Location Use Case

Scope: Marauder's Map

Level: User-goal

Actors: User (Primary Actor), Database (Supporting Actor)

Stakeholders: User: Updates his location and gets the location of other users

Preconditions:

- There is an active internet connection on the smartphone and the location services are enabled and privacy settings are OFF.
- The user is logged in and validated by the application.

Main Success Flow:

- ➢ The use case begins with user begins when user logs in to the app.
- ➢ Use Case: Update User to the application.
- ➢ The GPS device (mobile) gets the co-ordinates (position) of the user and sends them to the server. The position must contain:
    - Net ID
    - Latitude
    - Longitude
    - Altitude
    - Accuracy
    - Timestamp.
- ➢ Then the user's position is updated in the database under his Net ID.

Alternate flows (details left out):

- User not registered- Net ID of the user is not present in the system, it gives an error and the position is not recorded
- Invalid User
- No active internet connection
- The System fails to update the record due to internal failure (database error)
- Location Services are not enabled.
- Privacy Settings are ON.

Post-Conditions (Success Guarantee):

- Successful Completion: The user is able update his position to the application.
- Failure Condition: The user's location is not updated with the database.
- The User must send UpdateLocation (UserDynamicLocationProp) to the User Controller which in turn must send getUser (Userprop) to Data Access Layer.
- The Data Access Layer gets the details of user's position from the database (user_mst table) and must send details to the Data Access Layer which in turn must send them the UserProp Object to User Controller. Then the position of the user is updated.
- After updating his position, user will get the location of other users.

Actors	Roles	Goal
User	Primary	To share location with other users and view location of other users.
Database	Supporting	To hold the authentication credentials and manage user details.

4.2.2.3 Sequence Diagram 2

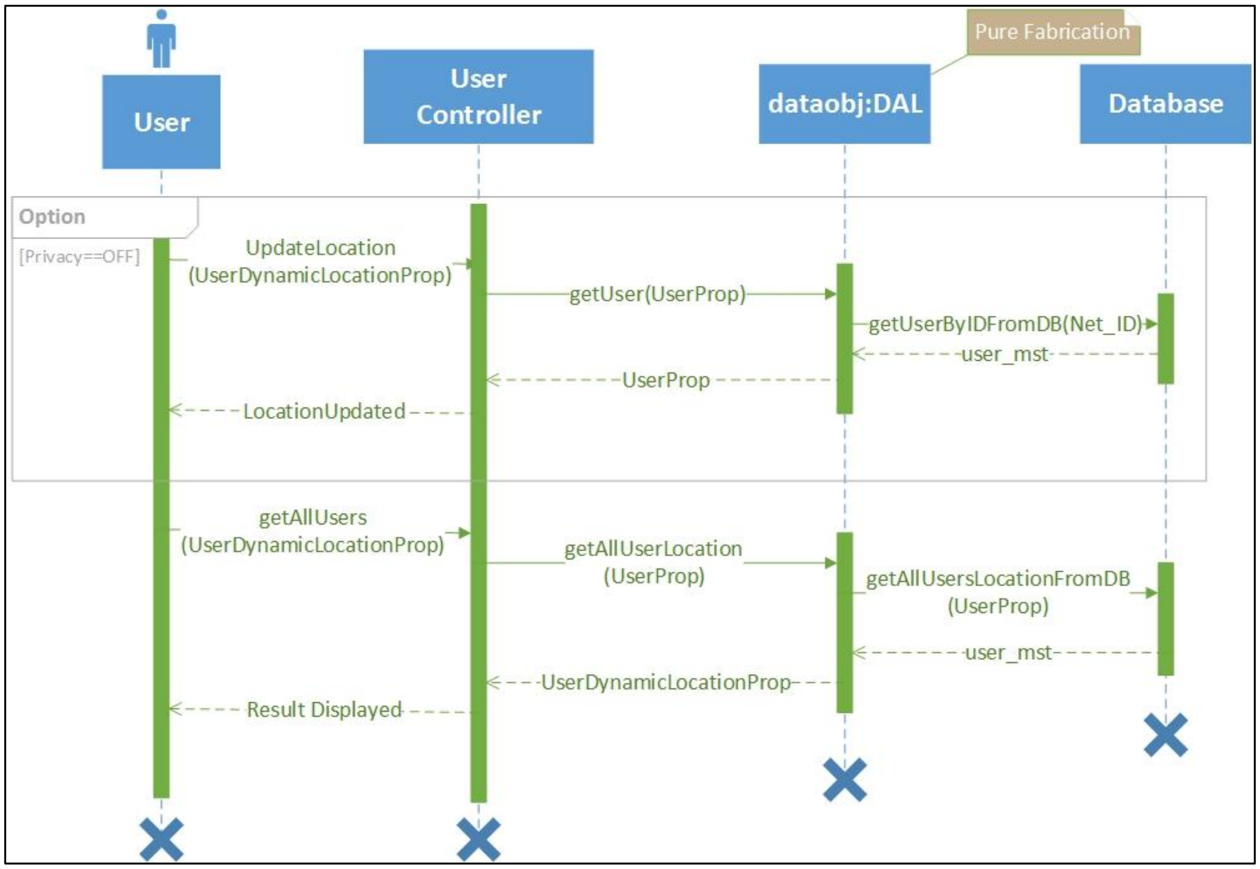


Figure 6: Sequence diagram for Updating user location

4.2.2.4 Operation Contract 2

- Operation: UpdateLocation(UserDyanamicLocationProp)
- Cross References:
    - Use Cases : Update User Location
- Preconditions:
    - The user is logged in and validated by the application.
- Post-conditions:
    - The UpdateLocation(UserDyanamicLocationProp)l instance was updated on the map.
    - UserDyanamicLocationProp was associated with retrieving other users loaction.
    - The user location is updated on the map and other users location is retrieved and updated on the map too.

4.2.3 Get Other Users' Details

4.2.3.1 Use Case Diagram 3

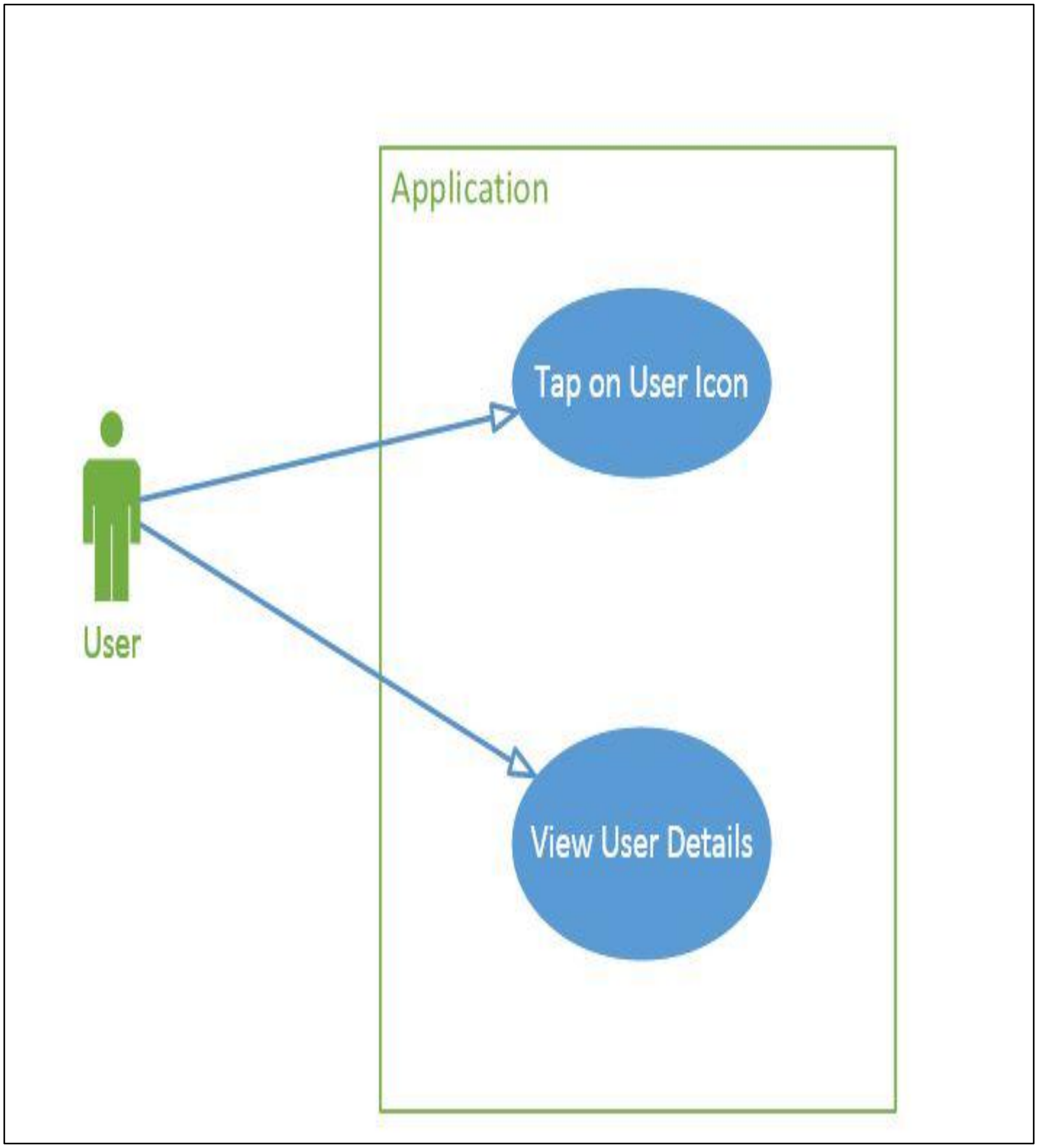


Figure 7: Use case diagram for Get other User details

4.2.3.2 Fully Dressed Use Case 3

Use Case Name: Get other user details

Scope: Marauder’s Map

Level: User-goal

Actors: User (Primary), Database (Supporting)

Stakeholders: User: Gets the other users details.

Preconditions: The user is logged in to the application.

Main Success Flow:

- The use case begins when user logs in to the app.
- Use Case: Get other user details.
- As the user logs into the application, a map opens up to him.
- The user taps on another users’ icon and the other user details are displayed.
- The application displays the other users’ details in another tab.

Alternate flows (details left out):
Invalid User – User is not logged into the application.

Post-Conditions (Success Guarantee):

- Successful Completion: The user is able to view the details of the other user.
- Failure Condition: The user is not logged in the application to view the details other user.

Special Requirements:
The application must have an appealing interface to display the user details.

Actors	Roles	Goal
User	Primary	To view details of other user.
Database	Supporting	To hold the authentication credentials and manage user details to display details of the other user.

4.2.3.3 Sequence Diagram 3

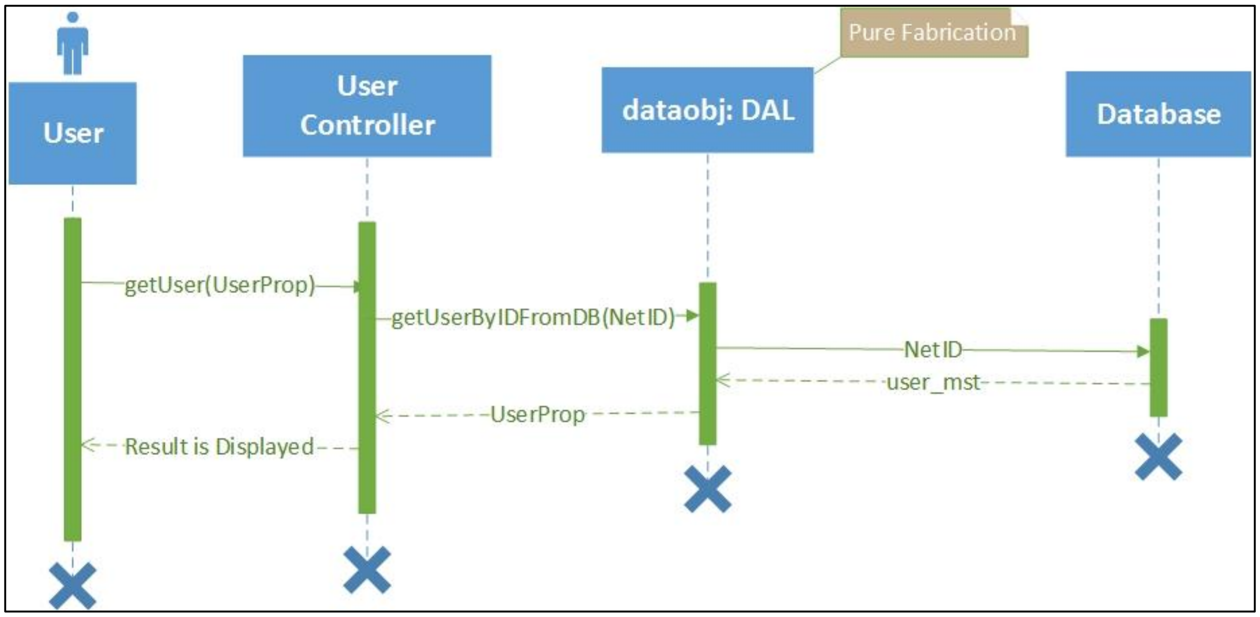


Figure 8: Sequence diagram for Get other User details

4.2.3.4 Operation Contract 3

- Operation: getUser(UserProp)
- Cross References:
    - Use Cases : Get list of Users and Locations
- Preconditions:
    - The user is logged in and validated by the application.
- Post-conditions:
    - The getUser(UserProp) instance was updated on the map.
    - UserProp was associated with getting the list of users.
    - The list of users is displayed on the map.

4.2.4 Search for other users and different locations

4.2.4.1 Use Case Diagram 4

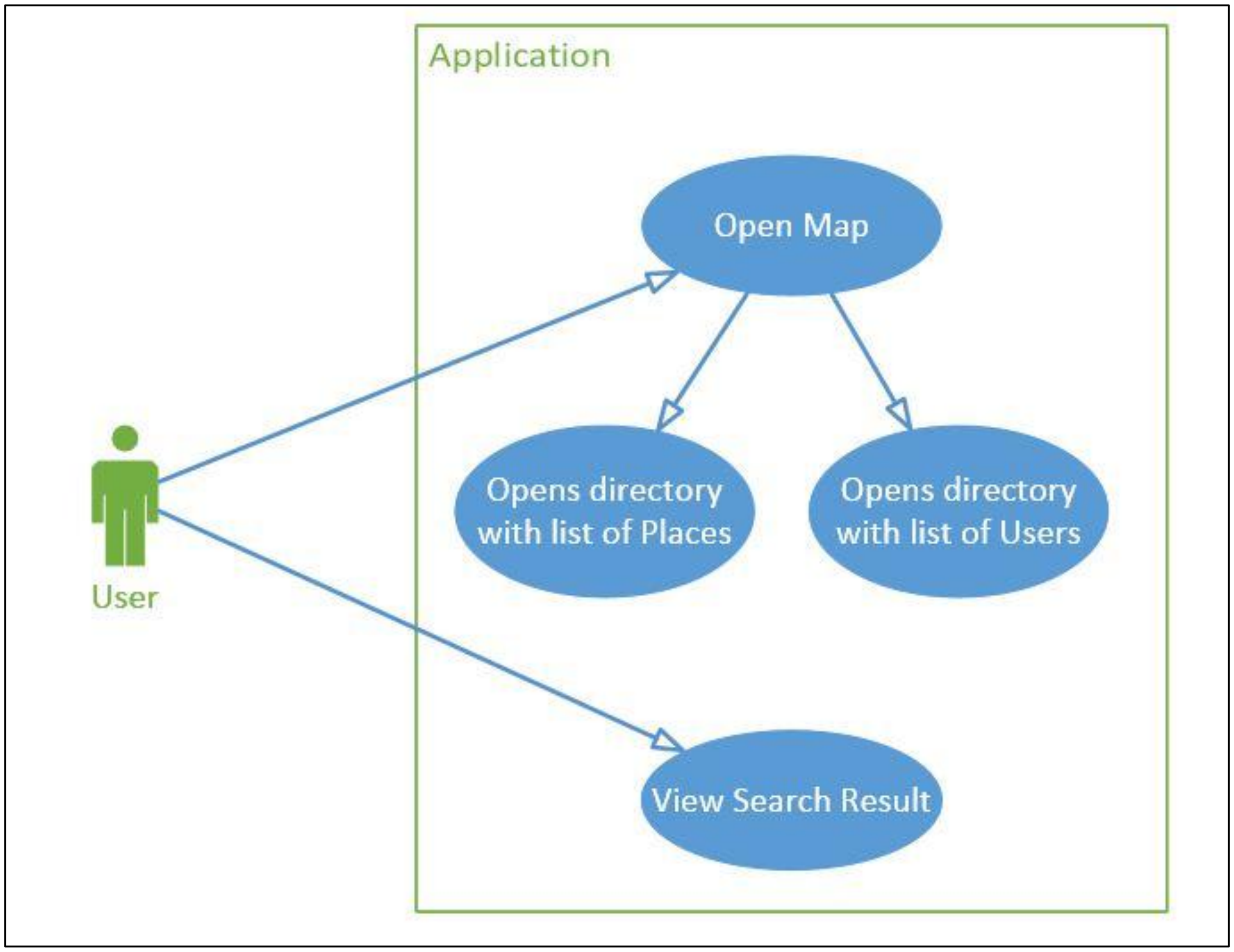

Application
Open Map
Opens directory with list of Places
Opens directory with list of Users
User
View Search Result

Figure 9: Use case diagram for search

4.2.4.2 Fully Dressed Use Case 4

Use Case Name: Search for other users and locations

Scope: Marauder's Map

Level: User-goal

Actors: User (Primary), Database (Supporting)

Stakeholders: User: Searches for other users and locations

Preconditions:

- There is an active internet connection on the smartphone and location services are enabled to be able to search for registered users and locations in the campus.
- The user is logged in and validated by the application.

Main Success Flow:

- The use case begins when user logs in to the app.
- Use Case: Search for other users and locations.
- The user can use the search bar to search for other registered users and places in campus.
- User enters the name or Net ID of a user he wants to locate or the name or Place ID of any place he wishes to see inside the campus.
- The User selects the location or friend and some corresponding details are displayed on the screen.

Alternate flows (details left out):

- No active internet connection
- Location services disabled
- Searched user is not registered
- Place does not exist in the database

Post-Conditions (Success Guarantee):

- Successful Completion: The user is able to locate the place or person he was looking for.
- Failure Condition: The searched user is not registered or the searched place does not exist in the database.
- The getUserByIDFromDB(text) (or) getUserByNameFromDB(text) (or) getPlaceByNameFromDB(text) (or) getPlaceByIDFromDB(text) instance will be updated on the map.
- These instances are associated with searching for users or locations.
- The user or location details which was searched for is displayed.

Special Requirements:

The application must have an appealing interface to display the searched user or place in the application.

Actors	Roles	Goal
User	Primary	To search for a user or place inside campus
Database	Supporting	To hold the authentication credentials and manage user details to display list of searched users and places.

4.2.4.3 Sequence Diagram 4

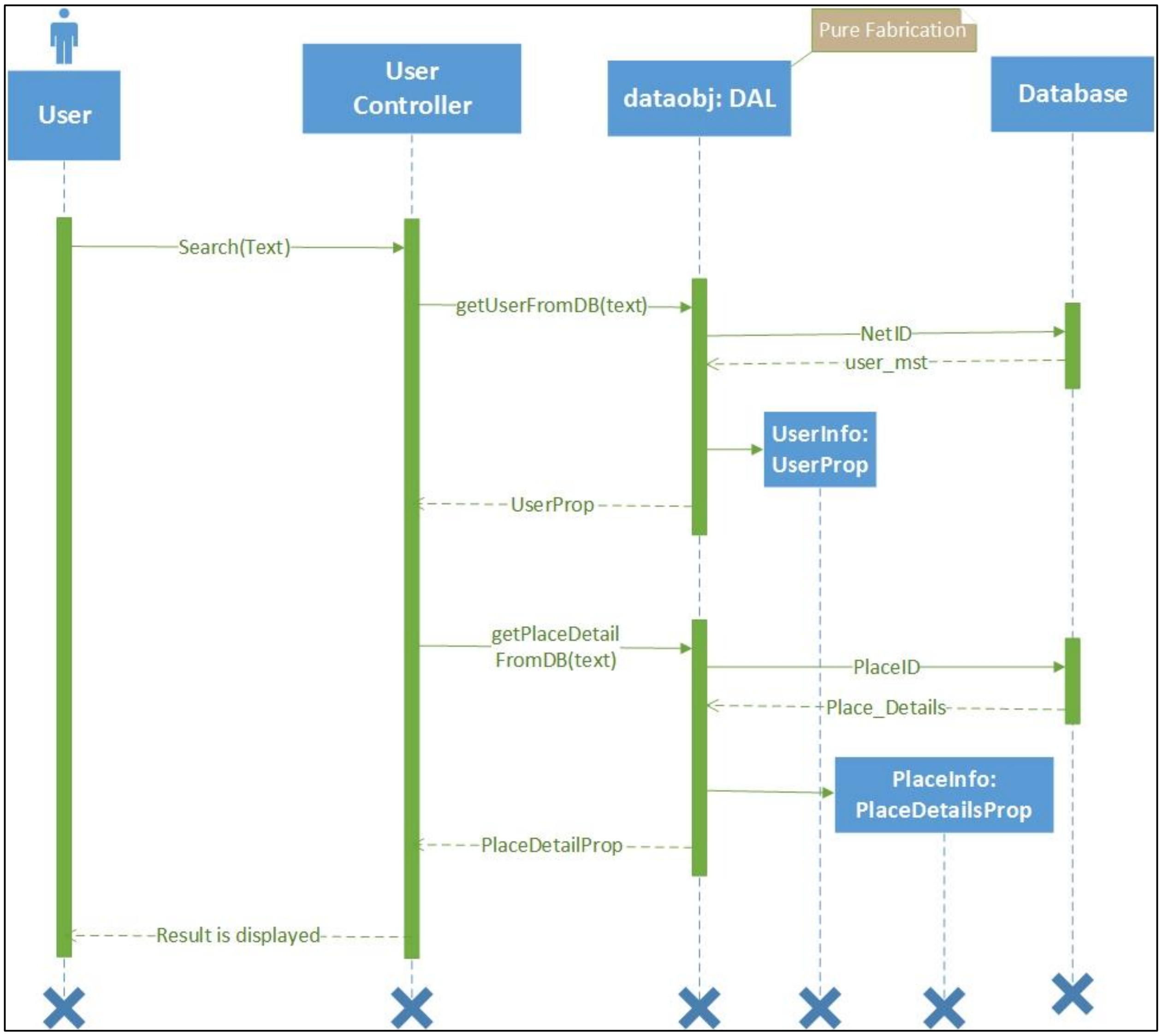


Figure 10: Sequence diagram for search

4.2.4.4 Operation Contract 4

- Operation: search(text)
- Cross References:
    - Use Cases : Search for users and other different locations
- Preconditions:
    - The user is logged in and validated by the application.
- Post-conditions:
    - The getUserByIDFromDB(text) (or) getUserByNameFromDB(text) (or) getPlaceByNameFromDB(text) (or) getPlaceByIDFromDB(text) instance was updated on the map.
    - These instances are associated with searching for users or locations.
    - The user or location details which was searched for is displayed.

5. Design

5.1 Architecture Design

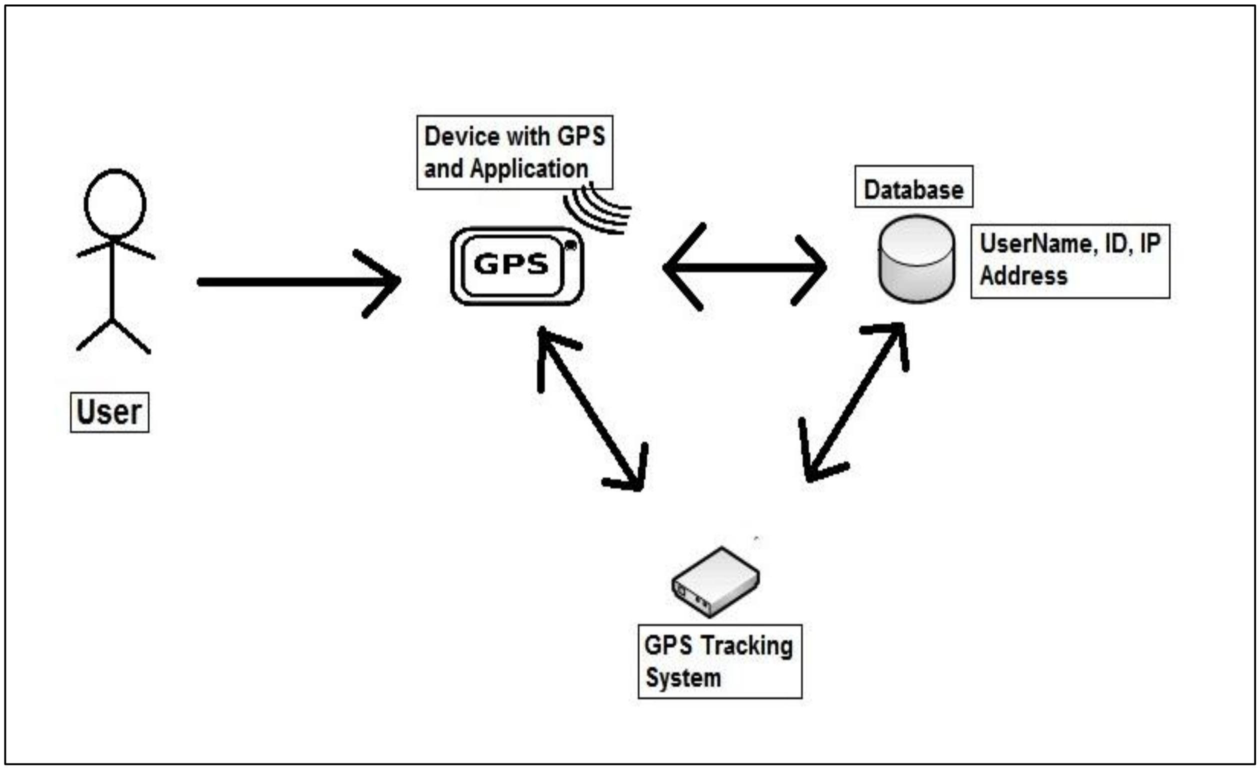


Figure 11: High level architecture

The technologies used for this project are client-server and mobile application. PhoneGap is a mobile development framework, which is our main platform for development. The intermediate platform that we are using is .NET. The general structure of the application is shown above. The system is a 3 tier architecture:

1. Client side: Mobile Application
2. Server side: GPS Tracking Interface and domain logic layer
3. Database

The basic functionality of the system is explained below:
The mobile application is accessed by the user. He logs into the app or creates his account if he is a new user. The validations are done by PhoneGap. Once the user logs in, he/she can access the map (integrated to google maps). Here, the user's GPS location coordinates are taken using the code written on the .NET framework, and also the other user's location is displayed on the map using the same platform. All the users' information is stored in the database and can be retrieved whenever the users' information needs to be displayed.

5.2 Design Model

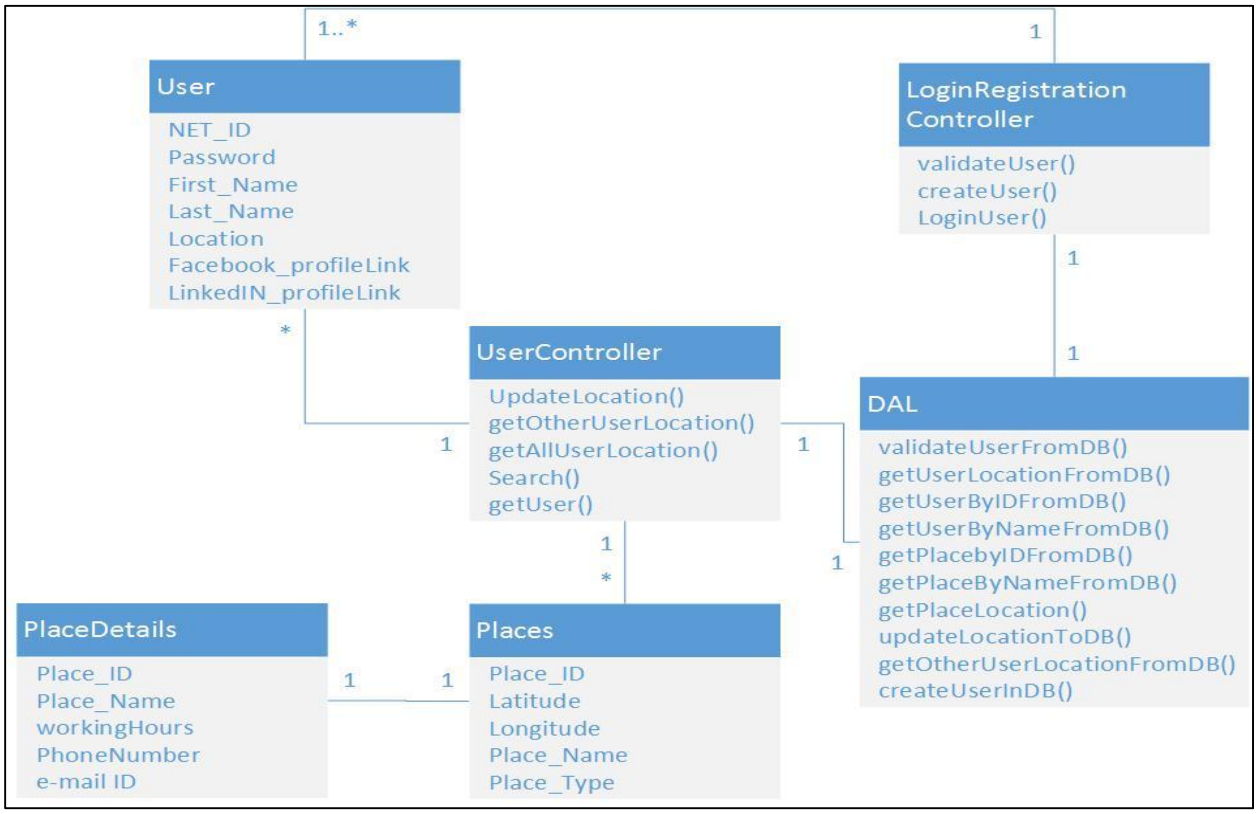


Figure 12: Class diagram

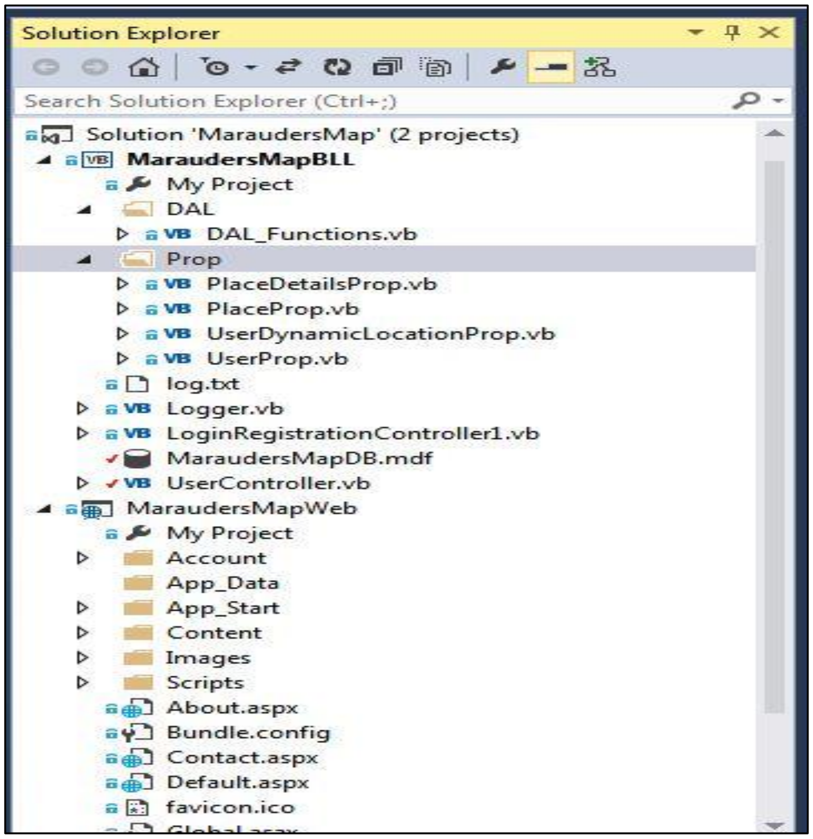
5.3 3-Tier Architecture

Three-tier architecture is a client-server architecture in which the functional process logic, data access, computer data storage and user interface are developed and maintained as independent modules on separate platforms. Three-tier architecture is a software design pattern and well-established software architecture. The 3 layers of the design are:

a) User Interface Layer/ Presentation Layer: Occupies the top level and displays information related to services available on a website. This tier communicates with other tiers by sending results to the browser and other tiers in the network. Presentation layer objects should be strictly bound to the views. There shouldn't be any logic. These objects should be used only for displaying activities.

b) Business Logic Layer: This is on top of the Presentation Layer. This layer coordinates the application, processes commands, makes logical decisions and evaluations, and performs calculations. It also moves and processes data between the two surrounding layers.

Snap Shot of Code Structure in VB for BLL:



```
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'MaraudersMap' (2 projects)
  MaraudersMapBLL
    My Project
    DAL
      DAL_Functions.vb
    Prop
      PlaceDetailsProp.vb
      PlaceProp.vb
      UserDynamicLocationProp.vb
      UserProp.vb
    log.txt
    Logger.vb
    LoginRegistrationController1.vb
    MaraudersMapDB.mdf
    UserController.vb
  MaraudersMapWeb
    My Project
    Account
    App_Data
    App_Start
    Content
    Images
    Scripts
    About.aspx
    Bundle.config
    Contact.aspx
    Default.aspx
    favicon.ico
```

Figure 13: High level code structure

Functions in our BLL:

- Login & Registration Controller: login(), createUser().
- User Controller: updateLocation(), getOtherUserLocations(), getAllUserLocations(), search(), getUser().

Properties in our BLL:

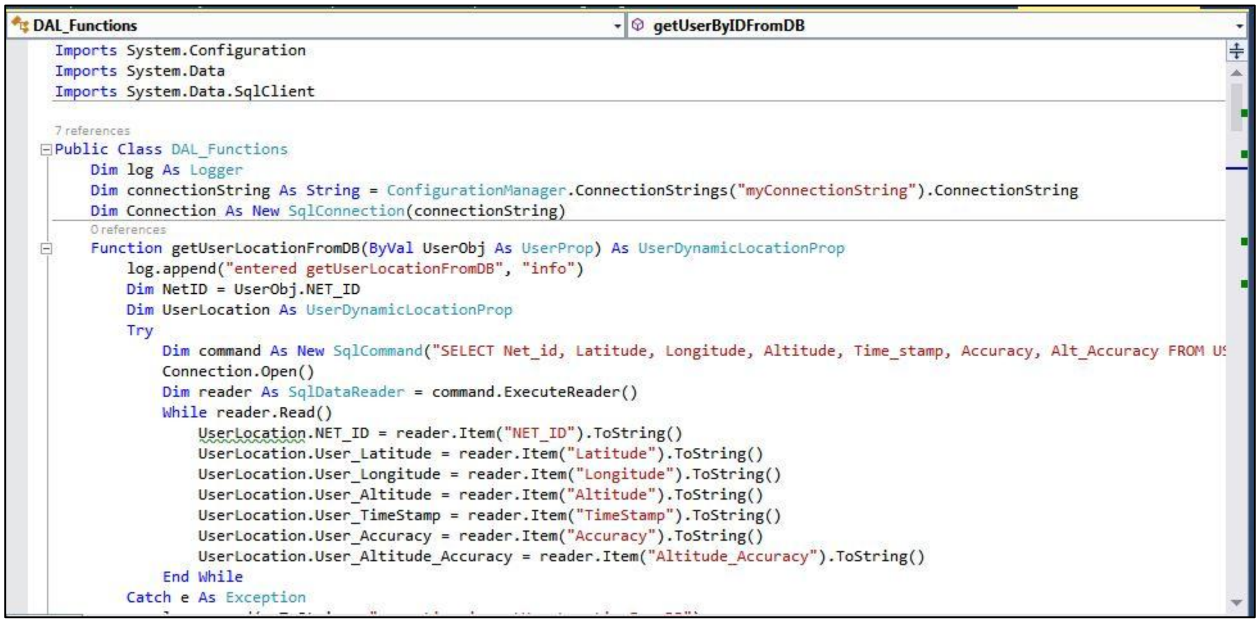
- Place Details Property: Gives Place_ID, Place_Name, Place_Email, Place_Phone, Place_Hours.
- Place Property: Gives Place_ID, Place_Type, Place_Name, Place_Latitude, Place_Longitude.
- User Dynamic Location Property: NET_ID, User_Latitude, User_Longitude, User_Alititude, User_Timestamp, User_Accuracy, User_Altitude_Accuracy.
- User Property: First_Name, Last_Name, NET_ID, Password, Mobile_No, Facebook_ID, LinkedIn_URL.

c) Data Access Layer: Data Access Layer contains methods to connect with database and to perform insert, update, delete, get data from database based on our input data. Here the data is stored and retrieved from the database. The information is then passed onto the business logic layer to be processed and then eventually back to the user. DA Objects should represent your database schema in some way and should be strictly bound to the Database activity.

Functions in our DAL:

- DAL_Fuctions: getUserLocationFromDB(), getUserByIDFromDB(), getUserByNameFromDB(), getPlaceByIDFromDB(), getPlaceLocation(), getPlaceByNameFromDB(), updateLocationToDB(), getOtherUserLocationsFromDB(), getAllUserLocationsFromDB(), createUserInDB().

Snap Shot of DAL_Function.vb



```
Imports System.Configuration
Imports System.Data
Imports System.Data.SqlClient

7 references
Public Class DAL_Functions
    Dim log As Logger
    Dim connectionString As String = ConfigurationManager.ConnectionStrings("myConnectionString").ConnectionString
    Dim Connection As New SqlConnection(connectionString)
    0 references
    Function getUserLocationFromDB(ByVal UserObj As UserProp) As UserDynamicLocationProp
        log.append("entered getUserLocationFromDB", "info")
        Dim NetID = UserObj.NET_ID
        Dim UserLocation As UserDynamicLocationProp
        Try
            Dim command As New SqlCommand("SELECT Net_id, Latitude, Longitude, Altitude, Time_stamp, Accuracy, Alt_Accuracy FROM U
            Connection.Open()
            Dim reader As SqlDataReader = command.ExecuteReader()
            While reader.Read()
                UserLocation.NET_ID = reader.Item("NET_ID").ToString()
                UserLocation.User_Latitude = reader.Item("Latitude").ToString()
                UserLocation.User_Longitude = reader.Item("Longitude").ToString()
                UserLocation.User_Altitude = reader.Item("Altitude").ToString()
                UserLocation.User_TimeStamp = reader.Item("TimeStamp").ToString()
                UserLocation.User_Accuracy = reader.Item("Accuracy").ToString()
                UserLocation.User_Altitude_Accuracy = reader.Item("Altitude_Accuracy").ToString()
            End While
        Catch e As Exception
```

Figure 14: Screenshot of Code for DAL

The main benefits of the N-tier/3-tier architectural style are:

- Maintainability: Each tier is independent of the other tiers, updates or changes can be carried out without affecting the application as a whole.
- Scalability: Tiers are based on the deployment of layers, scaling out an application is reasonably straightforward.
- Flexibility: Each tier can be managed or scaled independently, flexibility is increased.
- Availability: Applications can exploit the modular architecture of enabling systems using easily scalable components, which increases availability.

5.4 Package Diagram

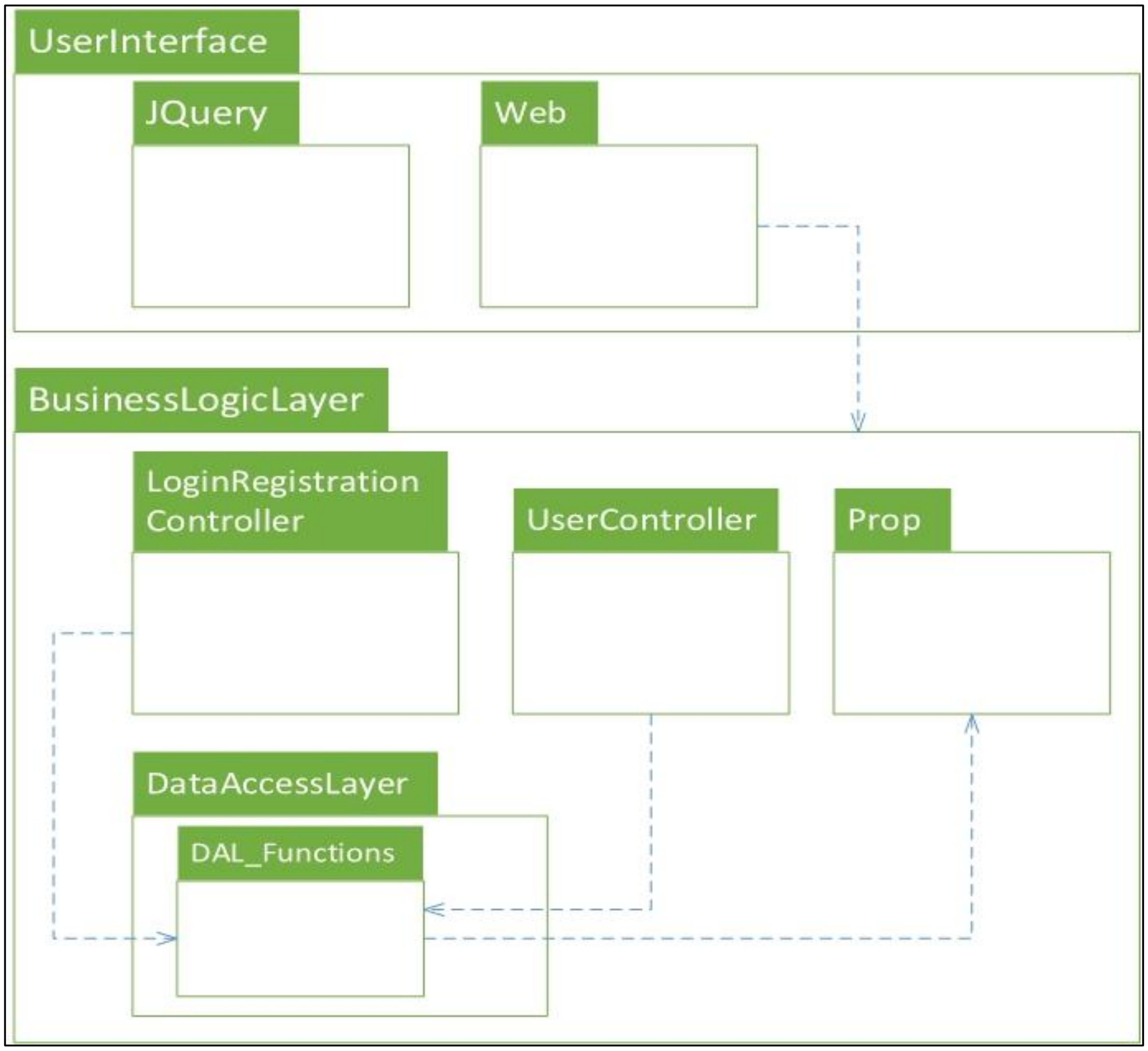


Figure 15: Package Diagram

5.5 GRASP Patterns

5.5.1 Creator:

A class that is responsible for creating objects is called the creator. In our design the login & registration controller is the creator as it creates the user instances. (Figure 4)

5.5.2 Controller:

The Controller pattern assigns the responsibility of dealing with system events to a non-UI class that represents the overall system or a use case scenario. A Controller object is a non-user interface object responsible for receiving or handling a system event. We have 2 controllers in our design: Login & Registration Controller and User Controller. The login & registration controller handles all events related to the login and registration of the user. The user controller handles all the events related to the User, like updating the user's location and getting the user information from the database. (Figure 1)

5.5.3 Information Expert:

Information expert is a principle that assigns responsibilities to objects and also contains all the information in the application. In our design, the Business Logic Layer is the information expert as it contains the 2 controllers which assign responsibilities to objects and the Data Access Layer which contains the information about all the users and locations.

5.5.4 Low Coupling:

Low Coupling is an evaluative pattern, which dictates how to assign responsibilities to support: lower dependency between the classes, change in one class having lower impact on other classes and higher reuse potential. In our design all the classes create their own objects and their creation is not dependent on other classes.

5.5.5 High Cohesion:

High Cohesion is an evaluative pattern that attempts to keep objects appropriately focused, manageable and understandable. High cohesion is generally used in support of Low Coupling. High cohesion means that the responsibilities of a given element are strongly related and highly focused. We have classes that have strong responsibilities like User Controller, Login & Registration Controller, Properties Class, etc. These classes are independent of each other and a change in one class doesn't affect the other.

5.5.6 Pure Fabrication:

A Pure Fabrication is a class that does not represent a concept in the problem domain, specially made up to achieve low coupling, high cohesion, and the reuse potential. Our Data Access Layer shows the pure fabrication phenomenon as it has the properties of a pure fabrication object. (Figure 4, 6, 8 and 10)

5.5.7 Indirection:

The Indirection pattern supports low coupling (and reuse potential) between two elements by assigning the responsibility of mediation between them to an intermediate object. The DAL shows indirection as it is the layer between the BLL and the database.

5.5.8 Diagram for GRASP

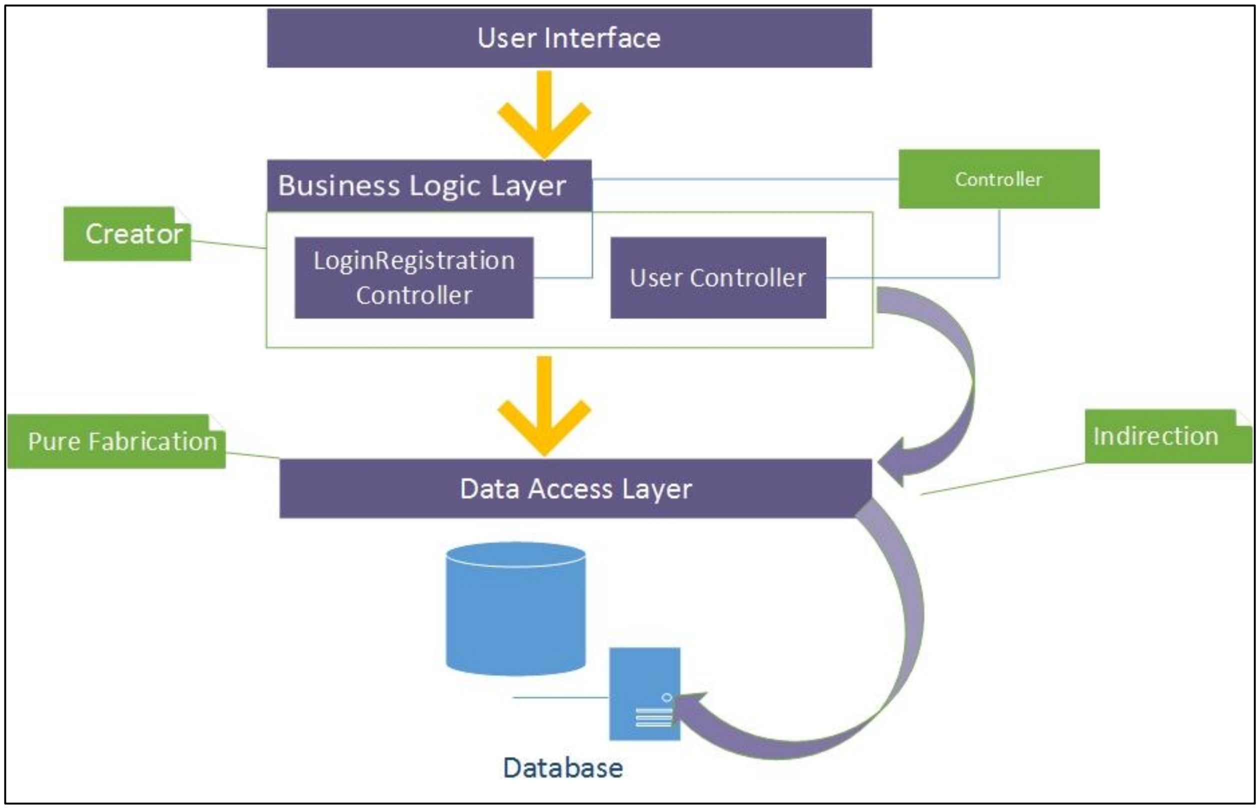


Figure 16: GRASP

5.6 Design Patterns

Observer (Publish – Subscribe)

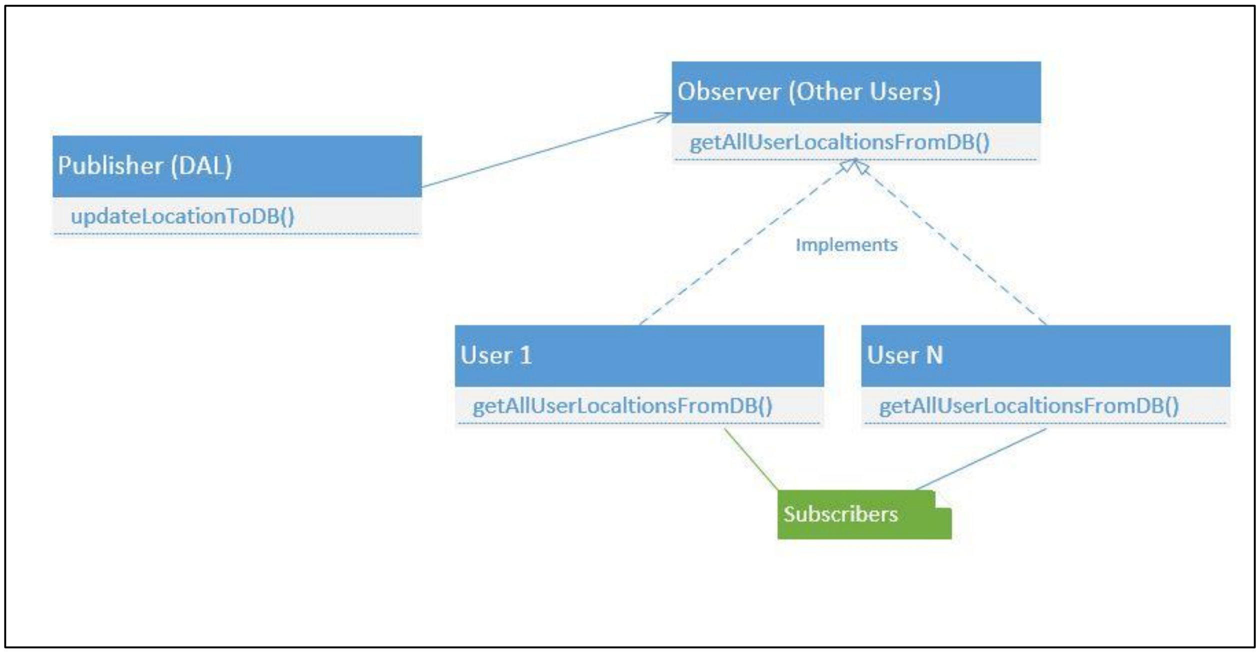


Figure 17: Design Pattern diagram explaining observer pattern

The Observer Pattern is software design pattern in which an object, called the subject, maintains a list of its dependents, called observers, and notifies them automatically of any state changes, usually by calling one of their methods. We have used this pattern in our use case for updating the locations.

Here, DAL is subject (Publisher) which sends all user location updates to all active users.

Active users are the Observers (subscribers) who receive location updates of all other users whose privacy status is off.

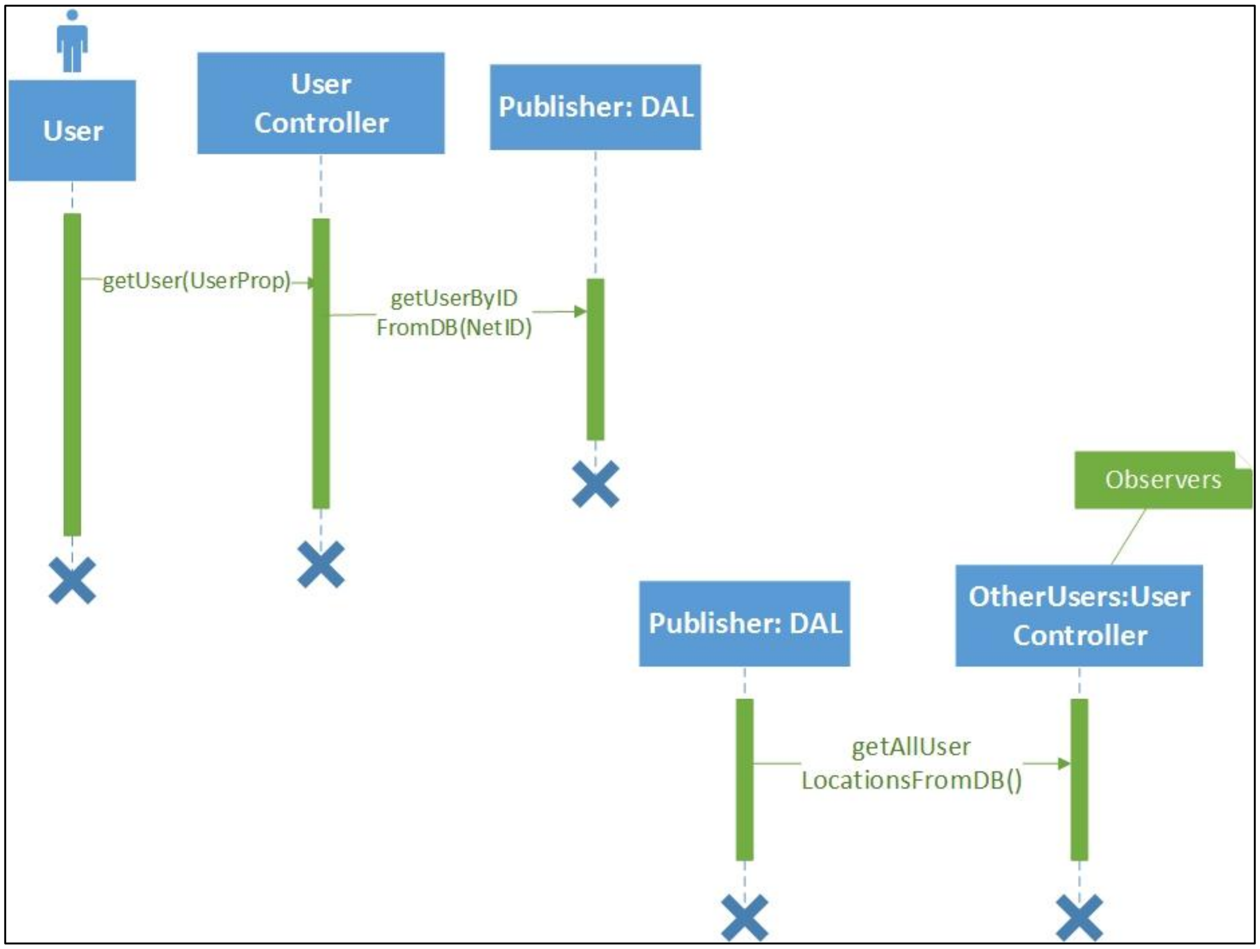

User
User Controller
Publisher: DAL
getUser(UserProp)
getUserByID FromDB(NetID)
Observers
Publisher: DAL
OtherUsers:User Controller
getAllUser LocationsFromDB()

Figure 18: Sequence diagram for observer pattern

Here, location of user is captured and sent to user controller which is sent to DAL and matched by NetID. This information is stored in database for every active user. In return, DAL shares location of every active user with every other active user. Therefore other users are acting as observer here.

Adapter Pattern: No Need

Why? → Marauder's Map does not require to perform same kind of functionality while interacting with different interfaces. Adapters are used when we want this kind of functionality in application like payment method.

Factory Pattern: No Need

Why? → As we are not using any adapter pattern, we don't need any factory pattern to create these adapters.

Singleton Pattern: No Need.

Why? → We don't need any kind of static variable/ class to be defined for the functionality we are working on.

Strategy Pattern: No Need

Why? → Application does not have any dynamically changing variable or characteristics which need strategic approach to handle. Therefore, we don’t have strategy pattern.

Composite Pattern: No Need

Why? → All functionalities have a single possible solution and does not require any decision making for deciding the path/ function to solve the problem. That mean application does not have any conflicting solutions for same transaction.

Façade Pattern: No need

Why? → Application is not having any changing business rules. Façade is a kind of adapter pattern and we don’t need adapter pattern for our application hence façade is also not required.

6. Testing

During the developing phase, we needed to test the functionality of our application at different stages. The modules that were tested are:

1. Login & Registration.
2. View user's location.
3. View other users' location.
4. View different locations on Map.
5. Search for users and locations.
6. Privacy Settings.

The test modules are given below:

Test case number:	**1**
Name of test:	**Login & Registration**
Item/Feature being tested:	**Can the user first register and then login successfully or not.**
Sample Input:	**Allowing the user to register with his details and login with his netId and password.**
Expected Output:	**User successfully completes registration and can login to view the map.**
Actual Output:	**User logs in successfully.**
Remarks:	**Module is working as expected.**

Screen Shots

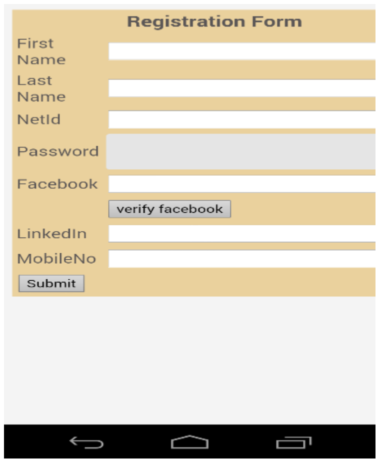


Figure 19: Registration Screen

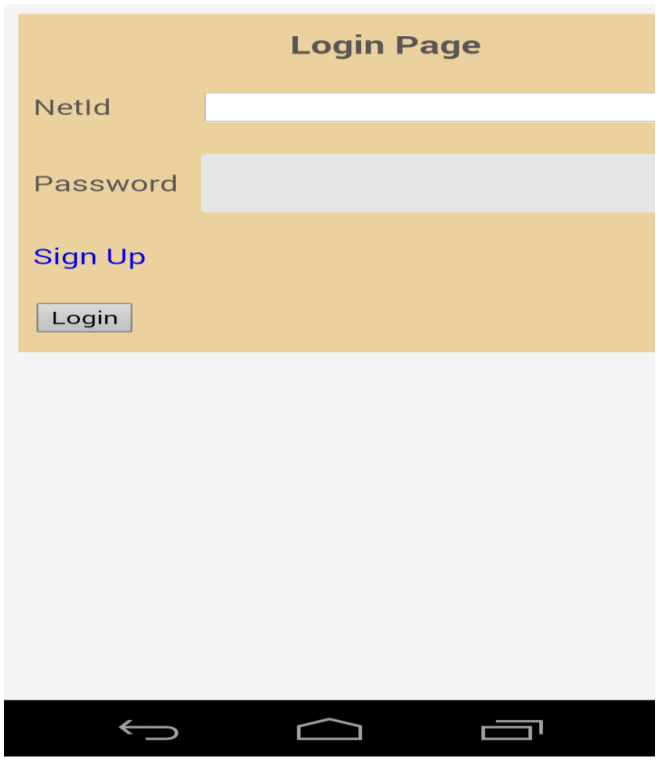


Figure 20: Login Screen

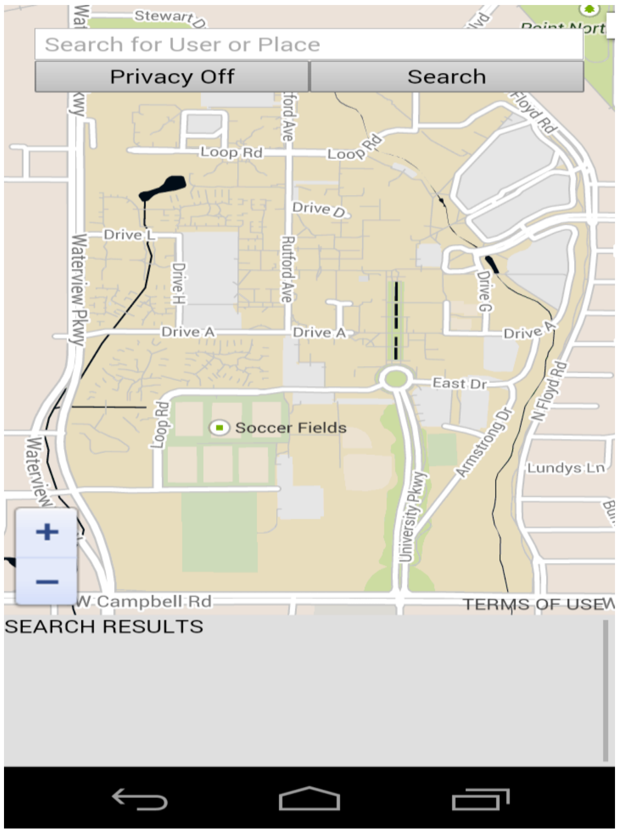


Figure 21: Screen after login

Test case number:	**2**
Name of test:	**View User Location**
Item/Feature being tested:	**Whether the user can see his location on the map or not.**
Sample Input:	**Allowing the user to login and let the app get his GPS location co-ordinates.**
Expected Output:	**User should see his location on the map.**
Actual Output:	**User location can be seen on the map.**
Remarks:	**Module is working as expected.**

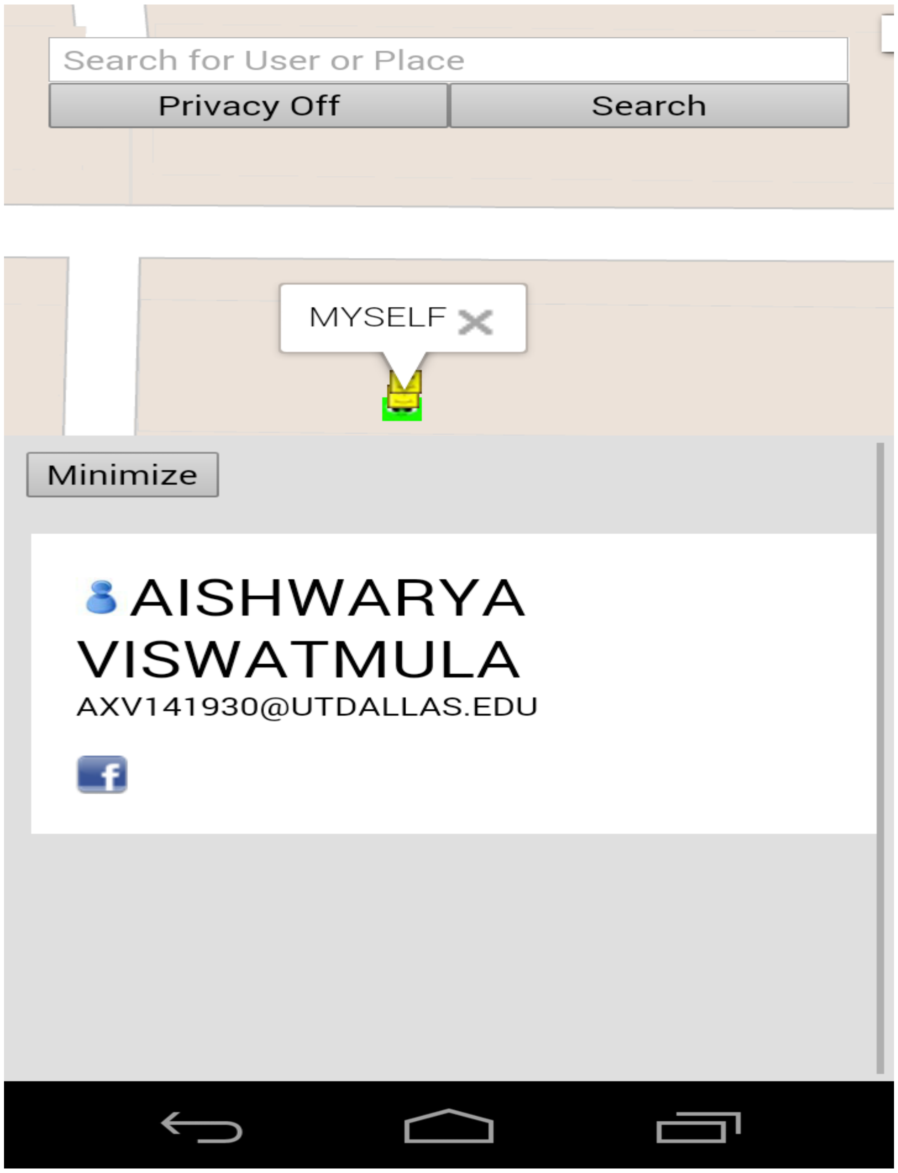
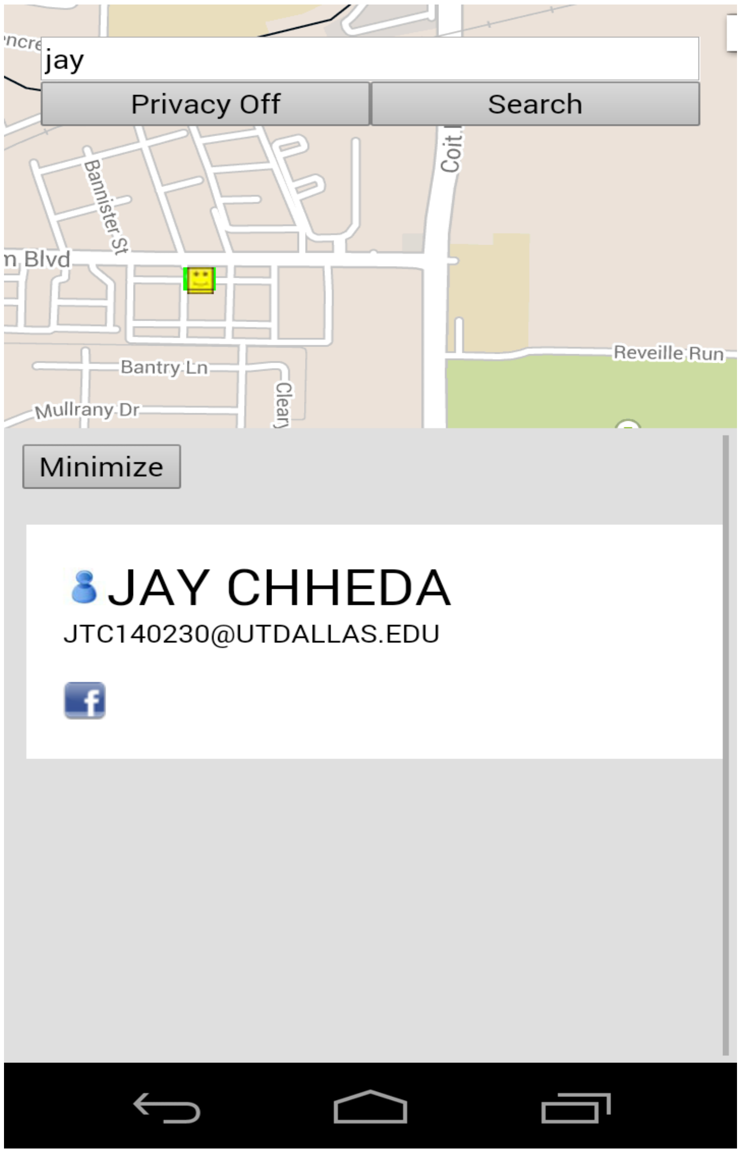


Figure 22: Other User

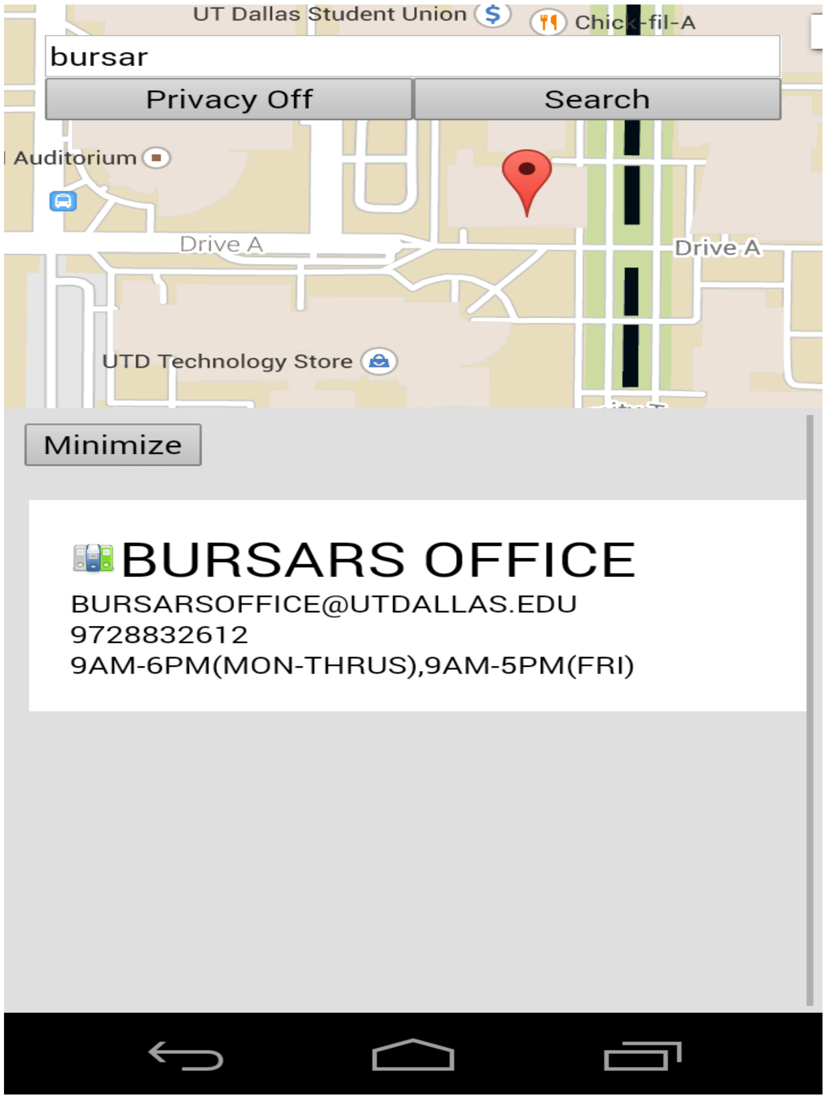
Test case number:	**3**
Name of test:	**View other users' location.**
Item/Feature being tested:	**Whether the user can view the location of other users or not.**
Sample Input:	**Allowing the many users to login and view the map.**
Expected Output:	**User should be able to see the location of other users (who are using the app).**
Actual Output:	**User can view other users.**
Remarks:	**Module is working as expected.**



JAY CHHEDA
JTC140230@UTDALLAS.EDU

Figure 23: View other user's location

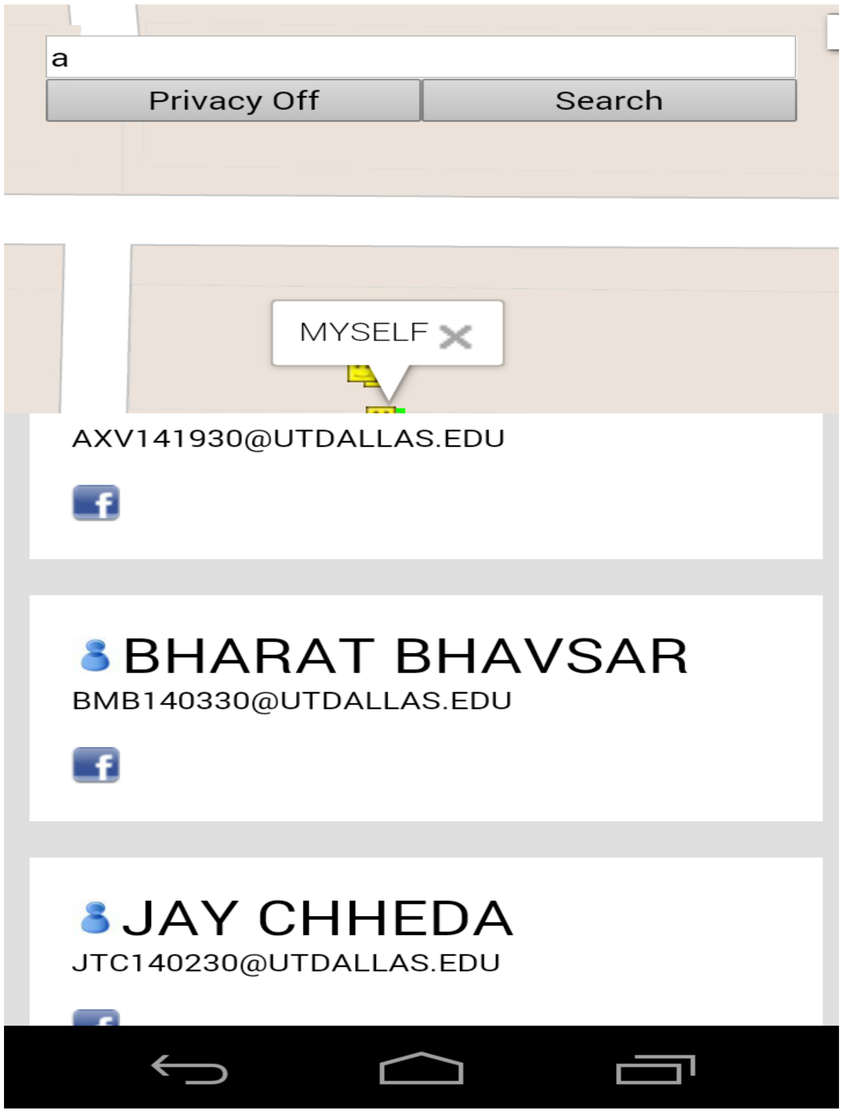
Test case number:	**4**
Name of test:	**View different locations on Map**
Item/Feature being tested:	**Whether the user is able to browse the different locations on Map.**
Sample Input:	**Allowing the user to login with his NetID.**
Expected Output:	**User should login successfully and access the first page with marker on different places around UTD campus.**
Actual Output:	**User is able to browse different locations.**
Remarks:	**Module is working as expected.**



BURSARS OFFICE
BURSARSOFFICE@UTDALLAS.EDU
9728832612
9AM-6PM(MON-THRUS),9AM-5PM(FRI)

Figure 24: Location

Test case number:	**5**
Name of test:	**Search for Users and Locations**
Item/Feature being tested:	**Whether the user find other users and different locations on the map.**
Sample Input:	**Type a user name or location name in the search box.**
Expected Output:	**User should find the user or location on the map.**
Actual Output:	**User can view the other user or location details.**
Remarks:	**Module is working as expected.**



AXV141930@UTDALLAS.EDU

BHARAT BHAVSAR
BMB140330@UTDALLAS.EDU

JAY CHHEDA
JTC140230@UTDALLAS.EDU

Figure 25: Search

Test case number:	**6**
Name of test:	**Privacy Settings**
Item/Feature being tested:	**The user's location shouldn't be updated, but he should be able to see others location.**
Sample Input:	**Turn privacy settings on.**
Expected Output:	**User's location is not updated, but he can view others location.**
Actual Output:	**User's location is not updated, but he can view others location.**
Remarks:	**Module is working as expected.**

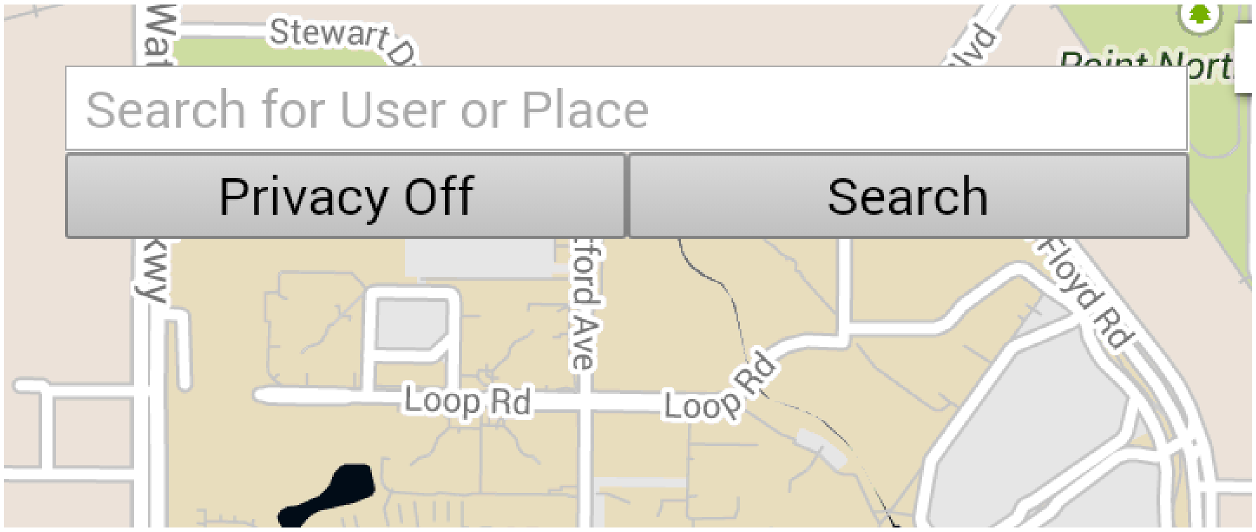


Figure 26: Privacy Off

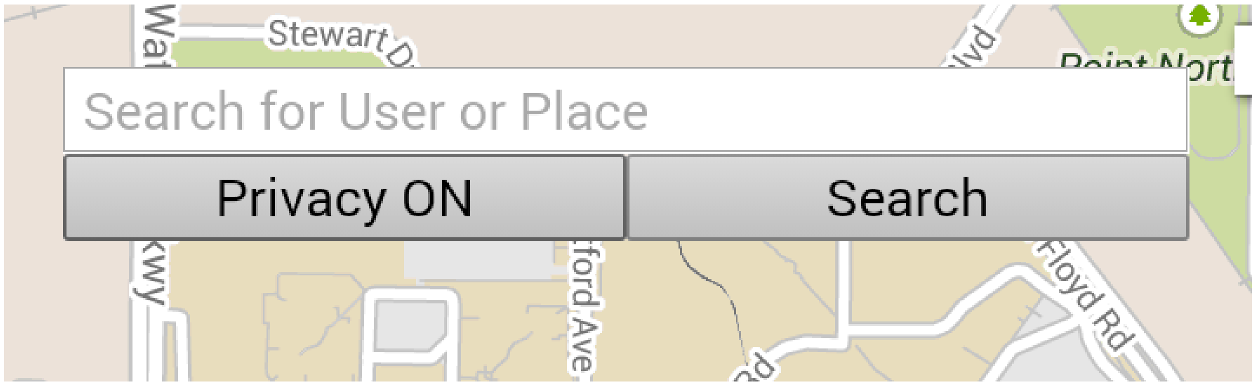
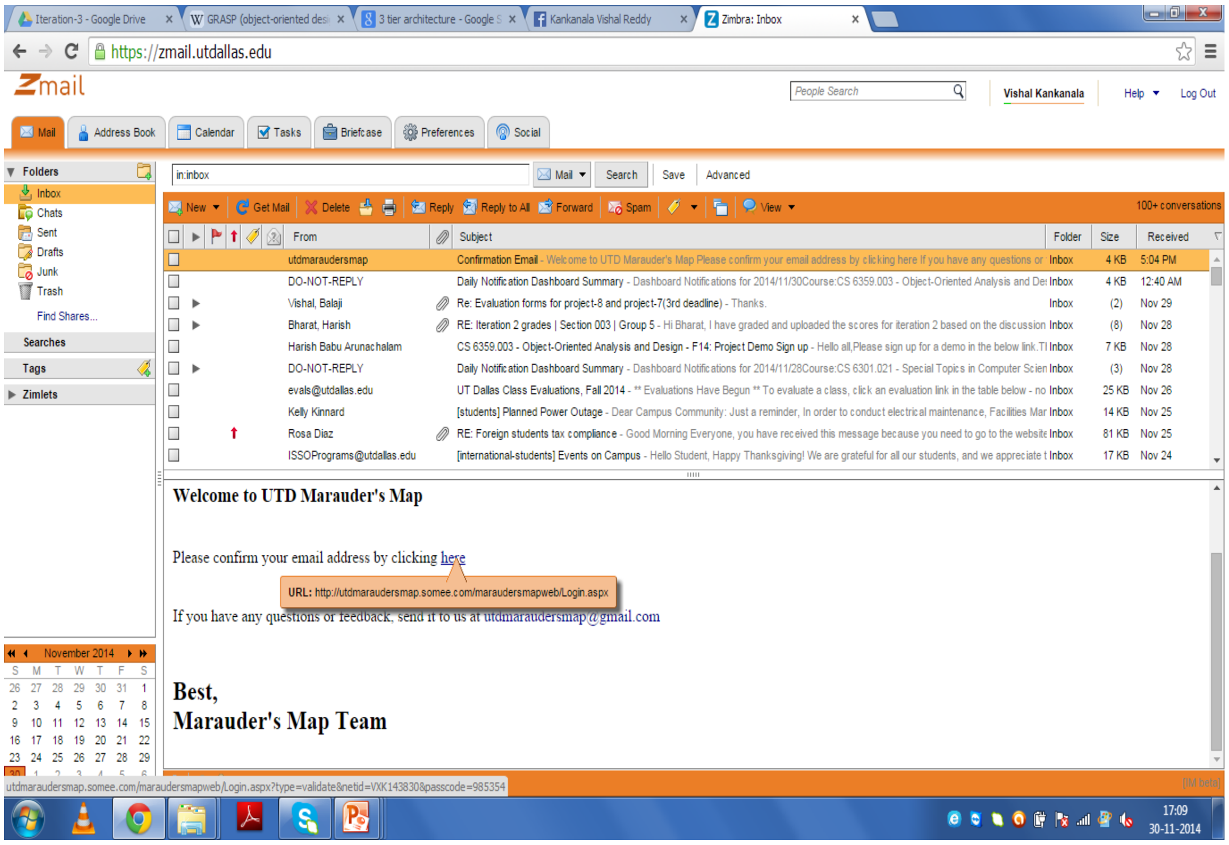


Figure 27Privacy On

Confirmation mail:



From	Subject	Folder	Size	Received
utdmaraudersmap	Confirmation Email - Welcome to UTD Marauder's Map Please confirm your email address by clicking here If you have any questions or	Inbox	4 KB	5:04 PM
DO-NOT-REPLY	Daily Notification Dashboard Summary - Dashboard Notifications for 2014/11/30Course:CS 6359.003 - Object-Oriented Analysis and De	Inbox	4 KB	12:40 AM
Vishal, Balaji	Re: Evaluation forms for project-8 and project-7(3rd deadline) - Thanks.	Inbox	(2)	Nov 29
Bharat, Harish	RE: Iteration 2 grades | Section 003 | Group 5 - Hi Bharat, I have graded and uploaded the scores for iteration 2 based on the discussion	Inbox	(8)	Nov 28
Harish Babu Arunachalam	CS 6359.003 - Object-Oriented Analysis and Design - F14: Project Demo Sign up - Hello all,Please sign up for a demo in the below link.Tl	Inbox	7 KB	Nov 28
DO-NOT-REPLY	Daily Notification Dashboard Summary - Dashboard Notifications for 2014/11/28Course:CS 6301.021 - Special Topics in Computer Scien	Inbox	(3)	Nov 28
evals@utdallas.edu	UT Dallas Class Evaluations, Fall 2014 - ** Evaluations Have Begun ** To evaluate a class, click an evaluation link in the table below - no	Inbox	25 KB	Nov 26
Kelly Kinnard	[students] Planned Power Outage - Dear Campus Community: Just a reminder, In order to conduct electrical maintenance, Facilities Mar	Inbox	14 KB	Nov 25
Rosa Diaz	RE: Foreign students tax compliance - Good Morning Everyone, you have received this message because you need to go to the website	Inbox	81 KB	Nov 25
ISSOPrograms@utdallas.edu	[international-students] Events on Campus - Hello Student, Happy Thanksgiving! We are grateful for all our students, and we appreciate t	Inbox	17 KB	Nov 24

Welcome to UTD Marauder's Map

Please confirm your email address by clicking here

URL: http://utdmaraudersmap.somee.com/maraudersmapweb/Login.aspx

If you have any questions or feedback, send it to us at utdmaraudersmap@gmail.com

Best,
Marauder's Map Team

utdmaraudersmap.somee.com/maraudersmapweb/Login.aspx?type=validate&netid=VXK143830&passcode=985354

Figure 28: Confirmation mail for registration

7. Plans & Achievements

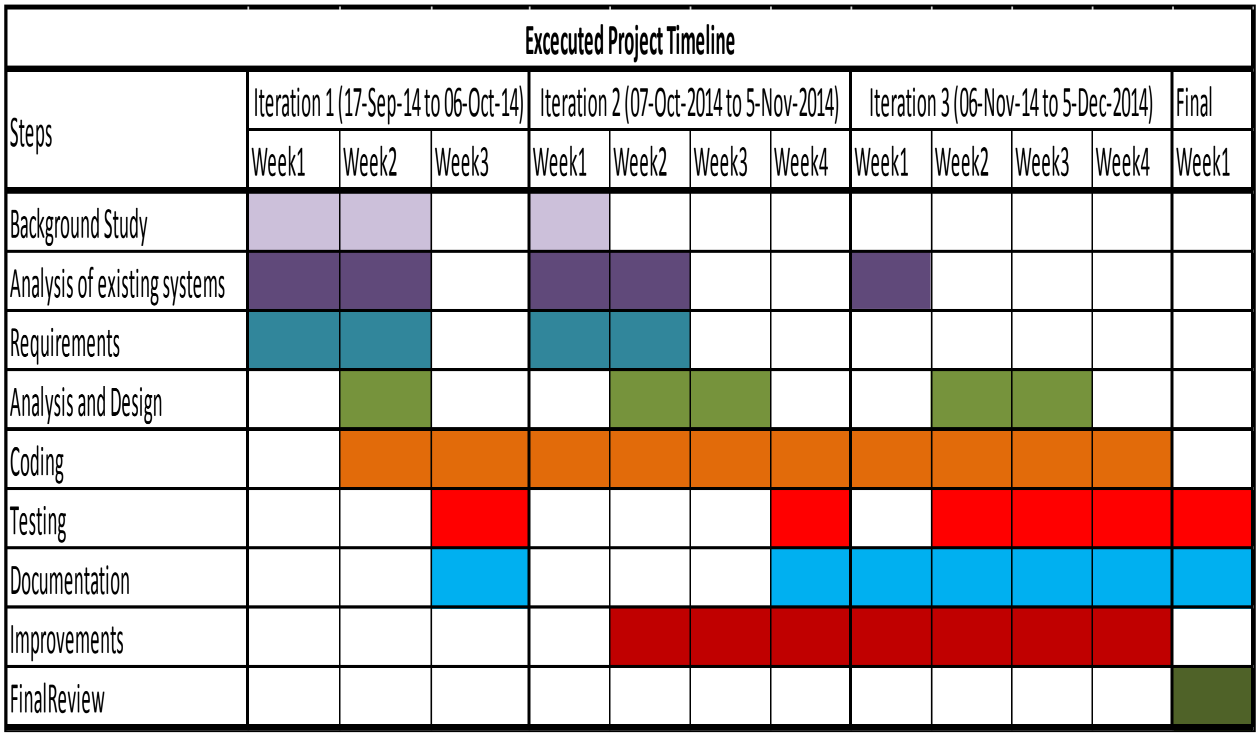
As we are following Agile Methodology, we had the liberty to make changes in our plans for each of the iterations. In the inception phase we had made a timeline showing our plan of action throughout the project. Now here is another timeline which shows our actual progress through all the iterations, which we can compare and tell how close we are to the initial plan of action that we made.

<u>Proposed Timeline</u>:

Proposed Project Timeline												
Steps	Iteration 1 (17-Sep-14 to 06-Oct-14)			Iteration 2 (07-Oct-2014 to 5-Nov-2014)				Iteration 3 (06-Nov-14 to 5-Dec-2014)				Final
	Week1	Week2	Week3	Week1	Week2	Week3	Week4	Week1	Week2	Week3	Week4	Week1
Background Study	🟪	🟪		🟪	🟪			🟪	🟪			
Analysis of existing systems	🟪	🟪		🟪	🟪			🟪	🟪			
Requirements	🟦	🟦		🟦	🟦			🟦	🟦			
Analysis and Design		🟩	🟩		🟩	🟩			🟩	🟩		
Coding		🟧	🟧		🟧	🟧	🟧		🟧	🟧	🟧	
Testing			🟥			🟥	🟥			🟥	🟥	🟥
Documentation				🟦			🟦	🟦			🟦	🟦
Improvements					🟥	🟥	🟥	🟥	🟥	🟥	🟥	
FinalReview												🟩

Figure 29: Planned time line

Executed Timeline:



Excecuted Project Timeline												
Steps	Iteration 1 (17-Sep-14 to 06-Oct-14)			Iteration 2 (07-Oct-2014 to 5-Nov-2014)				Iteration 3 (06-Nov-14 to 5-Dec-2014)				Final
	Week1	Week2	Week3	Week1	Week2	Week3	Week4	Week1	Week2	Week3	Week4	Week1
Background Study	🟪	🟪		🟪								
Analysis of existing systems	🟪	🟪		🟪	🟪			🟪				
Requirements	🟦	🟦		🟦	🟦							
Analysis and Design		🟩			🟩	🟩			🟩	🟩		
Coding		🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	
Testing			🟥				🟥		🟥	🟥	🟥	🟥
Documentation			🟦				🟦	🟦	🟦	🟦	🟦	🟦
Improvements					🟥	🟥	🟥	🟥	🟥	🟥	🟥	
FinalReview												🟩

Figure 30: Actual timeline

The lessons we have learnt while doing this project are:

- The advantages of using Agile methodology is evident, as we felt it is far more flexible than the waterfall methodology.
- We learnt how to complete the work for each iteration in the given time constraint and learnt that this way is more efficient and productive compared to working without iterations.
- While we had a few hard situations where we couldn't finish the work that we proposed that we would do in the next iterations, we learnt how to simplify things and show a simplified product rather than showing nothing.
- As we worked as a team, we learnt to respect each other's time and effort put into the project and have always been united as a team in times of distress.
- The implementation of everything that we have learnt in class has help us improve in our understanding of all the concepts like use cases, design patterns, interaction diagrams, grasp patterns, etc.. We have learnt how to constructively go about the analysis and design of the project.

- Feedback has played a major role in shaping our project. The feedback given after each of the iterations was taken into consideration and we modified our application and documentation accordingly.

- What we thought, and what we could do:

  - ✓ We can track people in real time mode, while they move around the campus and show their details.
  - ✓ We can find important places in the campus and give details of the location like working hours and contact details.
  - ✓ Privacy settings have been implemented, where if privacy is ON then the person's location won't be updated on the map, only his last location can be seen when his privacy was OFF.
  - ✓ We can search for people and locations on the map.
  - ✓ People can connect with each other through their Facebook and LinkedIn profile Id's given in each users details.
  - ❖ We thought of including group settings where everyone can create their own group and can track their friends, but couldn't implement it.
  - ❖ We thought of creating a separate profile/login for professors and students, but didn't implement it.
  - ❖ We wanted to include many details in the map, like all the classroom locations, but couldn't implement it.

8. Conclusion

Through the UTD Marauder's map we wanted to create a new application which would interest the students and staff members to use it, to connect with each other and search for people and locations. We wanted to make this application a surreal experience for the users so that they enjoy using it. We learnt how to apply the Object Oriented Analysis and Design concepts in this project, which has helped increase our knowledge in this area.

9. Prototypes

Similar maps have been created by many students from various universities. Some of the project names are given below:

1. The Digital Marauder's Map: A New Threat to Location Privacy :- Xinwen Fu, Nan Zhang, Pingley, A., Wei Yu , Jie Wang, Wei Zhao
2. Harry Potter's Marauder's Map: Localizing and Tracking Multiple Persons-of-Interest by Nonnegative Discretization :- Shoou-I Yu, Yi Yang, Alexander Hauptmann (Carnegie Mellon University)
3. Marauder's Map@ Olin College(Android App) :- Greg Marra