Bilkent University

Department of Computer Engineering

# CS 492-Senior Design Project 2

*Project short-name: Locum*

Final Report
Spring 2019

Alper Şahıstan, Ata Gün Öğün, Demir Topaktaş, Deniz Sipahioğlu, Orkun Alpar

Supervisor: Fazlı Can
Jury Members: H. Altay Güvenir, Hamdi Dibeklioğlu
Innovation Expert: Mustafa Sakalsız

Final Report
March 5, 2019

## Contents

# List of Figures

# CS 492- Low Level Design Report

*Project short-name: Locum*

## 1 Abstract

People forget things every day. Some people take notes on paper when they go shopping, and some of them take these notes on their phones, and cross out the items that they buy. They both might forget to buy some, and drive near a shop while doing so. Locum is designed to remind people of what they've forgotten, according to their location.

This report provides information about the final version of the Locum application. It includes information on the System Overview, Locum's Algorithmic and Application Design, Final System and Architecture Design, Impact of Engineering Solutions, Tools and Technologies Used, Engineering Solutions and Contemporary Issues followed by a brief User Manual and Class Diagrams.

Keywords: Alarm, Application, Special Location, GPS, Groups, Location, Notes, Reminder, Shared Lists, Shopping, Stationary

## 2 Introduction

The purpose of this project is to help people get over their issue of "forgetting" by reminding those things they want to be reminded of, according to their location.

The project idea came from an everyday issue of forgetting. Some people take notes whenever they go shopping, and cross-out the items on the list when they buy them. Sometimes they forget to buy these items, and drive right next to the shops, and still don't buy them. For situations like this Locum is there to remind them to buy their items when they drive somewhere close to a shop.

The project requires careful planning in both the design and analysis stages, because the system that can provide users with this kind of information needs to be thought of carefully. We are aiming to make a basic prototype of the project, and then add many features that make this project different from others. In the first phase, we will design the UI and some basic features that gets what we want done in the first place, "reminding the user when he goes to a specific location". In the next phase, we are planning to add features like "group planning" and "adding customized locations". On the last step, the project will be extended for commercial use. In the senior project, only the first and second phases will be completed.

Consider, one needs to buy a notebook from a stationary, but he forgets it even when he is at a stationary. There is a stationary on his way back home, but he has other things on his mind, and the "notebook" is not one of them at that moment. He can take notes to Locum, writing "notebook", and Locum will remind every time he goes anywhere near a stationary.

Nowadays, it is possible to track the exact location of the user, and with the help of labels of certain places such as shops or pharmacies, the user can drive to these places with the help of location services. Even if this technology exists, there is no service that the user can take notes about tasks (like a grocery list) and be reminded when they are close to a location where they can complete these tasks, like shops.

Locum aims to help people be reminded of tasks they can do at several places, and notify them when they are close to any of these places. They will be able to choose active hours for the application to work at, and set different priorities to these tasks.

This report provides information about the final version of the Locum application. It includes information on the System Overview, Locum's Algorithmic and Application Design, Final System and Architecture Design, Impact of Engineering Solutions, Tools and Technologies Used, Engineering Solutions and Contemporary Issues followed by a brief User Manual and Class Diagrams.

## 2.1. Engineering and Report Writing Standards

This report follows the UML guidelines for the class interfaces, diagrams, and subsystem decompositions. This guideline was selected to demonstrate the said diagrams in a clear manner, and since it is the guideline taught in Bilkent University CS classes. To generate these diagrams, Visual Paradigm Version 14 was used. For the references, the report follows the MLA standards. The MLA standard is also taught in Bilkent University.

## 2.2. Use of New Tools and Technologies

**Room Persistence Library**: Room persistence library provides abstraction over SQLite which helped us query SQLite statements to the local database very easily. You can specify Java objects as entities in Room and then save/fetch that object into/from the database which is very easy and reliable to use. Entities in Room corresponds to tables in database. You can also create an interface that has methods for accessing databases as well.

**Cloud Platforms**

-**Amazon Web Services EC2:** We used Amazon Web Services EC2 to deploy our NLP flask server to the cloud. EC2 services provide machines that are fast, secure, available and easy to configure. We choose to use Ubuntu instance with 16GB RAM and 25GB storage because we need to have capacity for using Natural Language Processing models. Moreover, our cloud Ubuntu machine is fast and when Android host posts note to the server, it processes and returns the result very quickly.

-**AWS RDS:** We chose to use Amazon RDS (Relational Database Service) because it is secure,easy to administer, available and fast. All information related to Locum is stored in RDS. Monitoring is easy and we can see its availability and response times. It is easy to connect Mysql Workbench to RDS, in order to see our tables.

-**Microsoft Azure:** Our NodeJS server is deployed on Microsoft Azure. We chose to deploy NodeJS server to Azure since Azure has a special service for Node applications. Azure helped us monitor our server and debug and deploy to the cloud in a fast way.

**Express NodeJS:** Our main server is written in Nodejs with Express framework. Express is a lightweight, minimalist framework that provides lots of features for web applications. It is easy to implement a server with this framework. Http request routing is well designed, fully functional and it has a very good error handling mechanism. We can easily debug our server. The MySQL features it has also enables us to query SQL statements to our database easily and reliably.

**Flask:** We used flask micro framework in order to implement our python server which is responsible for Natural Language processing. It has a good http routing and it enabled us to have high computing in server side. Its minimal and simple structure helped us to have a simple python server that satisfies requests very quickly.

**Google Places API:** The Places API is used when returning information about places that use HTTP requests. Places are defined here as establishments, geographic locations, or prominent points of interest. We have used Places API for fetching location data like pharmacies, banks, supermarkets, etcetera within a given circle.

**Geofencing API:** The Geofencing API intelligently uses the device sensors to accurately detect the location of the device in a battery-efficient way. The API allows us to know when a user enters or leaves an active Geofence.

**NLP Libraries/Technologies**

-**Word2Vec:** For processing text, our main model was Word2Vec. Word2Vec is a group of related models that are used to produce word embeddings. These models are shallow, two-layer neural networks that are trained to reconstruct linguistic contexts of words. Word2Vec takes as its input a large corpus of text and produces a vector space, typically of several hundred dimensions, with each unique word in the corpus being assigned a corresponding vector in the space.

**-Natural Language Toolkit (NLTK):** For English pre-processing, we have used Natural Language Toolkit (NLTK). NLTK is a leading platform for building Python programs to work with natural language data. It consists of a suite of text processing libraries for classification, tokenization, stemming, tagging, lemmatization and parsing.

**-Zemberek:** For Turkish pre-processing, we have used Zemberek. Zemberek is a Java library for processing Turkish text. Similar to NLTK, it provides tokenization, tagging, lemmatization, and parsing.

## 2.3. Implementing Creative and Impactful Solutions

Locum application achieves something that was not achieved before. It combines everyday note-taking with Natural Language Processing and Geofencing technologies to create a smooth user experience while being as functional and as easy to use as possible.

In order to implement the complex structure of Locum, we have made the implementation in two separate servers. One of them is for features that do not require complex computations such as authentication, saving/fetching notes to/from global database, managing collaborators and the other server is for Natural Language Processing which requires complex computation. What makes this different from other applications lies in its combination of different servers with very quick response times. Moreover, these servers are on cloud which makes them accessible from anywhere at any time.

Android sends/receives JSON objects to/from servers. This communication is done with HTTP protocol, and all networking is done in threads on Android side. If we didn’t use threads here, the UI had the risk of blocking the UI and this would be a huge problem. We solved this problem by doing the networking in threads.

Another feature Locum implements is the Geofencing Technology explained in Section 4.1. Geofence Management Algorithm and Implementation.

Wherever the user is, he will be reminded of his notes and the points of interest near him. The creative solution here is Locum computing the nearest points of interests and setting up associated Geofences automatically.

# 3 System Overview

Locum aims to help people be reminded of tasks they can do at several places, and notify them when they are close to any of these points of interest. Remembering these kinds of tasks can be hard sometimes, keeping in mind that people have many things going on their minds all the time. They take notes on paper, on their phones, and hope to remember them when they need to, but this is not an efficient solution.

Some of the features in Locum are available in some apps, but there are no apps that satisfy the need of being reminded of a task when you are close to the place where the task needs to be done. Examples of these are "Google Keep", iOS 11 Update, "Buy me a Pie" and much more. None of these remind the user based on his location.

One might need to drop a book to a friend, buy some medicine from the pharmacy on the way back home, or buy some chicken for dinner, and forget to do these things. Locum will help them remember these things by notifying them when they are close to places where they can do these things.  With the API's described in the Current System section, Locum will merge the scope of all these applications mentioned above.

In addition to note-taking, the app will be used in the planning of social activities. Imagine one plans a barbeque with friends and family. Everybody invited to the barbeque will launch Locum and join a group that was created by one of them. The group homepage will contain a group-list that will be composed of necessary items that the people in the group added for this occasion. They will see which person bought which item (to prevent duplicates), and see if there are any unpurchased items in the list. Locum will remind these unpurchased items whenever a member is near a place where they can obtain these items. People using Locum will no longer depend on pen and paper to remember their tasks.

# 4 Locum's Algorithmic and Application Design

This section demonstrates the technologies that are used in the Locum application, their implementation and how they interact with each other. It consists of two parts, Geofence Management Algorithm and the User Experience/Interface Design. Other API's were also used in Locum's implementation, which are demonstrated in Section 7.3. API's Used.

## 4.1. Geofence Management Algorithm and Implementation

A **geo-fence** is a virtual perimeter for a real-world **geographic** area. The Geofencing API intelligently uses the device sensors to accurately detect the location of the device in a battery-efficient way. The API allows us to know when a user enters or leaves an active Geofence. However, the Geofencing API has a major limitation, it only allows 100 Geofences per device. Ideally, we require a lot more Geofences than 100 since there are a lot more possible points of interests at any given time.

For example, when a user sets a reminder saying "Buy cake ingredients", he wants to be reminded in any supermarket he enters, but there are more than 100 supermarkets in Ankara. Moreover, a user might have several different reminders involving different types of locations, including pharmacies, banks, book stores and many more.

The problem here is obvious: There are thousands of locations we want to Geofence, but we can only have a hundred of them active at any given time. The solution we engineered involves setting a "significant distance" Geofence. When a new reminder is set, the application determines the 99 nearest POIs and sets Geofences for those locations. The last one of the hundred Geofences we are allowed to have is used to determine if a significant distance is travelled by the user. If the user travels further than the 99 nearest points of interests, the app is triggered by the "significant change" Geofence. The app does not notify the user of this trigger, instead it deactivates all previous Geofences and calculates the nearest points of interest with respect to users' current location and sets new Geofences for those locations.

To allow finding the 99 nearest locations of interest as a background activity (possibly when the app is not on, or when there is no internet access), the application needs to have a local database of all possible points of interests.

The initial algorithm we described here went under some changes to allow better battery efficiency and computational optimizations, but the essence of it is still the same. We are not going to discuss all these changes to this algorithm in this report since explaining our over-engineered solution would be too long and complicated.

## 4.2. User Experience and Interface Design

For Locum to be a preferred application above others, a simple yet effective design was necessary. Since Locum is an Android application, we used Android studio which accelerated our UI implementation. Designing Locum also required some design tools such as Adobe Photoshop (Section 7.1.13) and Adobe Illustrator (Section 7.1.14). The logo was designed to both include the location and reminder features of Locum, which was struggling. Creating a smooth flow between the screens was easy since we've already planned the flow in the previous reports.

At first, we implemented the UI before starting the NLP and Geofence implementation to make our work easier as we progressed through the project. A problem we faced was that we had many ideas on how several user interactions should function, and couldn't decide on which was the best, what should we include and what should we exclude. After many tests we decided on using the Swipe down action for refreshing, clicking a view for entering a list and completing a task, and long pressing a view for other options to come up. This consistency made the app easy to use.

Another problem we faced was how to display the tags generated by the NLP, and how these tasks should be edited by the user. We thought it we put all the tags, the view would be very complicated, but we also needed the functionality for the user to edit the tags. Since the user can not add custom tags by himself, he needed to see all the tags somewhere. We decided to display the tags generated by the NLP on top, and put other tags in a scrollable view.

Also, the API we used to generate the tags, Places API, didn't include all the tags we wanted like small shops around town. Because of this, the user wouldn't be informed when he is near a small shop that might help him complete the task.

While designing the Alarm page, at first we thought a simple alarm, only informing that there are list items to be bought would be enough. Then we thought it would be very beneficial to see where the user is at by using a map view. This would allow the user to see where he is and the roads around him in a bird's eye view. We thought text based information along the map would be really helpful for this particular screen, so we also decided to show which exact tasks could be completed, because that task might not be necessary at that time, so the user could want to ignore it and continue what he's doing. This screen can be seen in Section 9.12. Push Notification and Alarm Screens.

# 5 Final System and Architecture Design

Locum's structure consists of several subsystems, and they must work together for the application to work seamlessly. This section demonstrates the subsystems, the interaction between them, the classes they consist of and the functions of these classes. A  larger look to the diagrams is available in Appendix B.

Note that these diagrams do not include all the support and scaffolding classes. Only the essential classes are included in this part.

At the topmost level, Locum is composed of two parts, Client and Server. Server has three components; Request Handler, Data Layer and Search Tools. Client also has three components; View, Presenter and Model. In this section of the report, both the Client and the Server sides of our application will be analyzed generally. How the classes are divided into different packages will be discussed. Individual explanations for the classes are provided in Section 3, Class Interfaces.

## 5.1. Packages

### 5.1.1. Client

Client is the user end of our application. It is simply the user with his Android device. We have used Model-View-Presenter pattern in our application which works as follows: Inputs are welcomed by the View. View knows about (communicates with) Presenter, which in turn communicates with View through an interface. Unlike MVC, there's a one-to-one connection between View and Presenter. Presenter manipulates the Model. Model notifies the Presenter about the changes in itself.

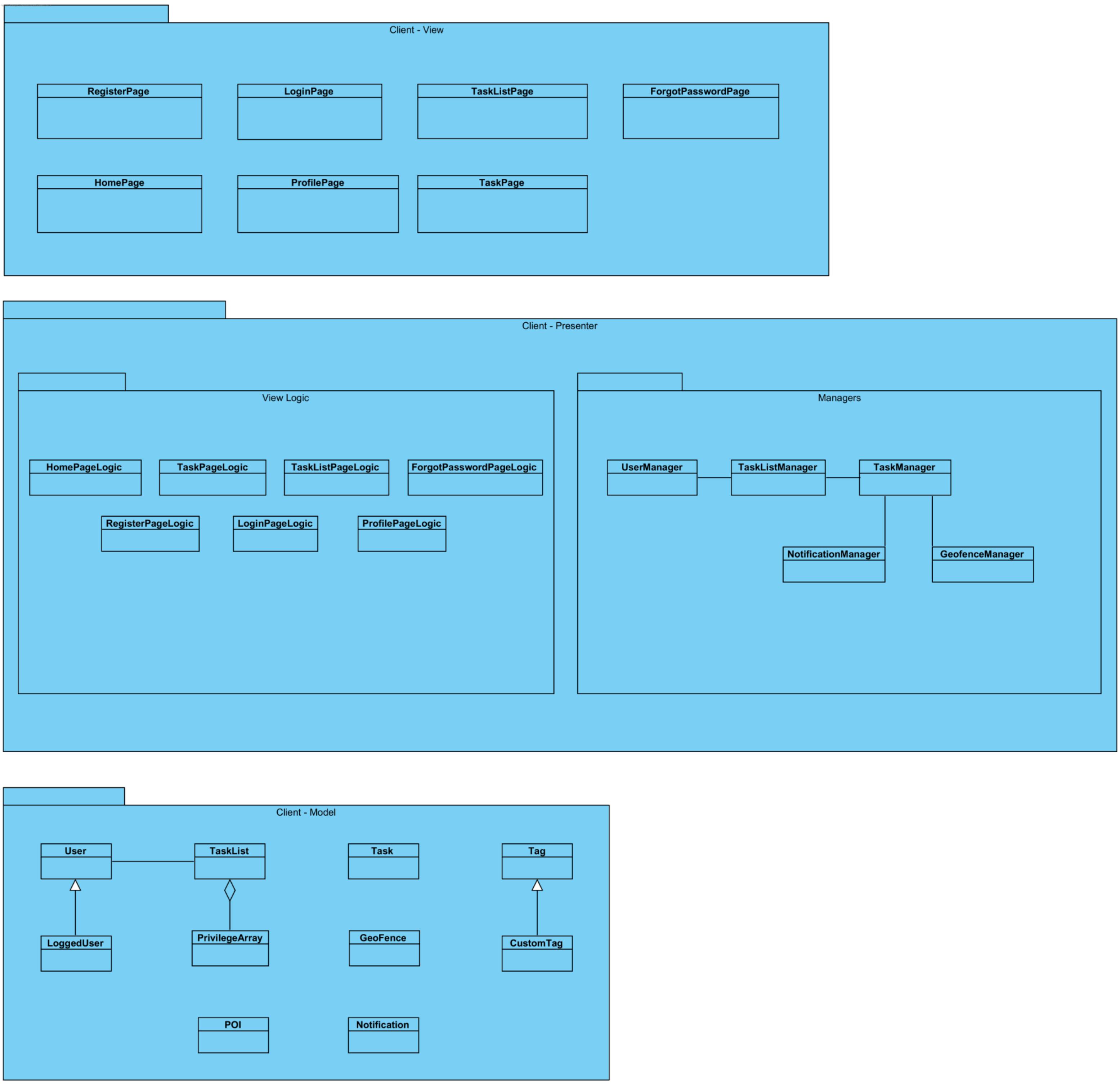


***Figure 1: Client Package Diagram***

### View

View component of the client side is the Android UI which the user interacts with. We have 7 different Android Activities which are simply different pages of the application. Namely; Home page, TaskList page, Task page, Login page, Register page, Profile page and lastly ForgotPassword page. Task page consists of a single task whereas TaskList page contains a list of tasks, for example a shopping list. Home page consists of all the lists the user owns or is a part of (collaborative lists). Login, Register, Profile and ForgotPassword are self-explanatory.

### Presenter

Presenter has two sub-packages. View Logic and Managers. View Logic contains methods that are called from View, that part is used for the creation of objects and it is communicating with the Android activities mentioned above. Managers subpacket has static methods that use the objects created with the help of the methods in View Logic.

### Model

Model is the local database of our application which holds information about the user, tasks, POIs, privileges, et cetera.

### 5.1.2. Server

Server side of our application has three components: Request Handler, Data Layer and Search Tools.

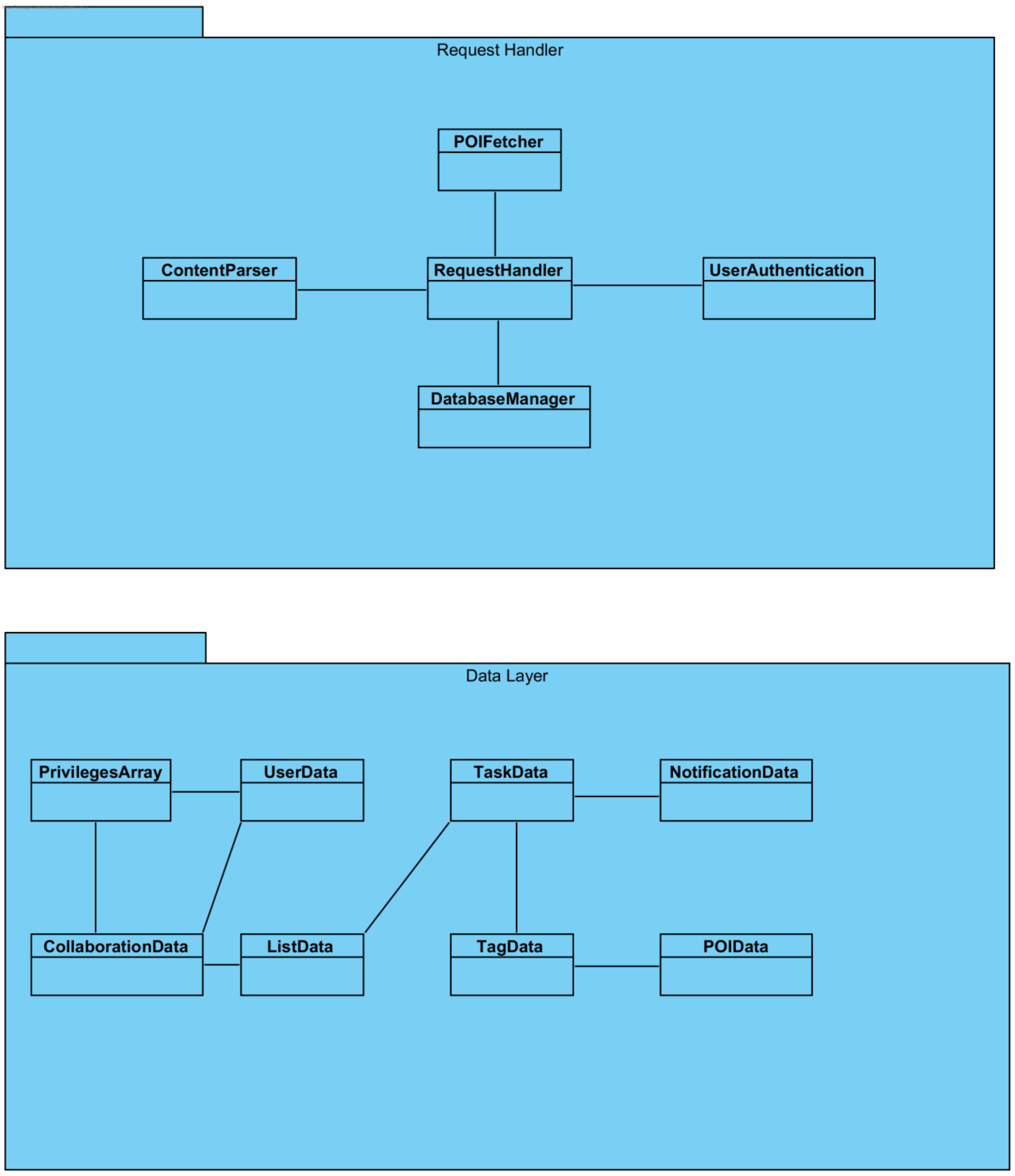


***Figure 2: Server Package Diagram***

**Request Handler**

Request Handler, as the name suggests, handles requests that are given to the application server. That's the topmost layer of our server architecture. We have 5 different classes inside Request Handler, namely; POIFetcher, RequestHandler, ContentParser, UserAuthentication, and DatabaseManager.

**Data Layer**

Data Layer is the database located in the server. It holds information related to users, tasks, lists, et cetera.

**Search Tools**

Search Tools stand for APIs that'll be used for searching the web in order to satisfy the user requests, searching the given queries and mining information.

## 5.2. Class Interfaces

The "UML 2 Class Diagramming Guidelines" found in the Agile Modelling website (provided in the references) advises to omit scaffolding code in class interfaces. According to the same source, "Scaffolding code refers to the attributes and operations required to implement basic functionality within your classes, such as the code required to implement relationships with other classes". Thus, the default constructors; the attributes and methods implied by aggregations, compositions, generalizations and specializations are not included in this part to provide clarity to the reader. For similar reasons, getter and setter methods for private attributes are not explicitly shown. All classes have public getter and setter methods for their private attributes/fields (If not explicitly stated otherwise).

### 5.2.1. Client Package

#### 5.2.1.1. View Package

View component of the client side has pages, children of Android.Activity

Android.Activity has 7 methods that need to be overridden by the developer.

Here, we are showing what these methods are for by presenting a class interface for the HomePage. The same will apply to other 6 pages as well. Namely, TaskPage, TaskListPage, ProfilePage, RegisterPage, LoginPage, and ForgotPasswordPage.

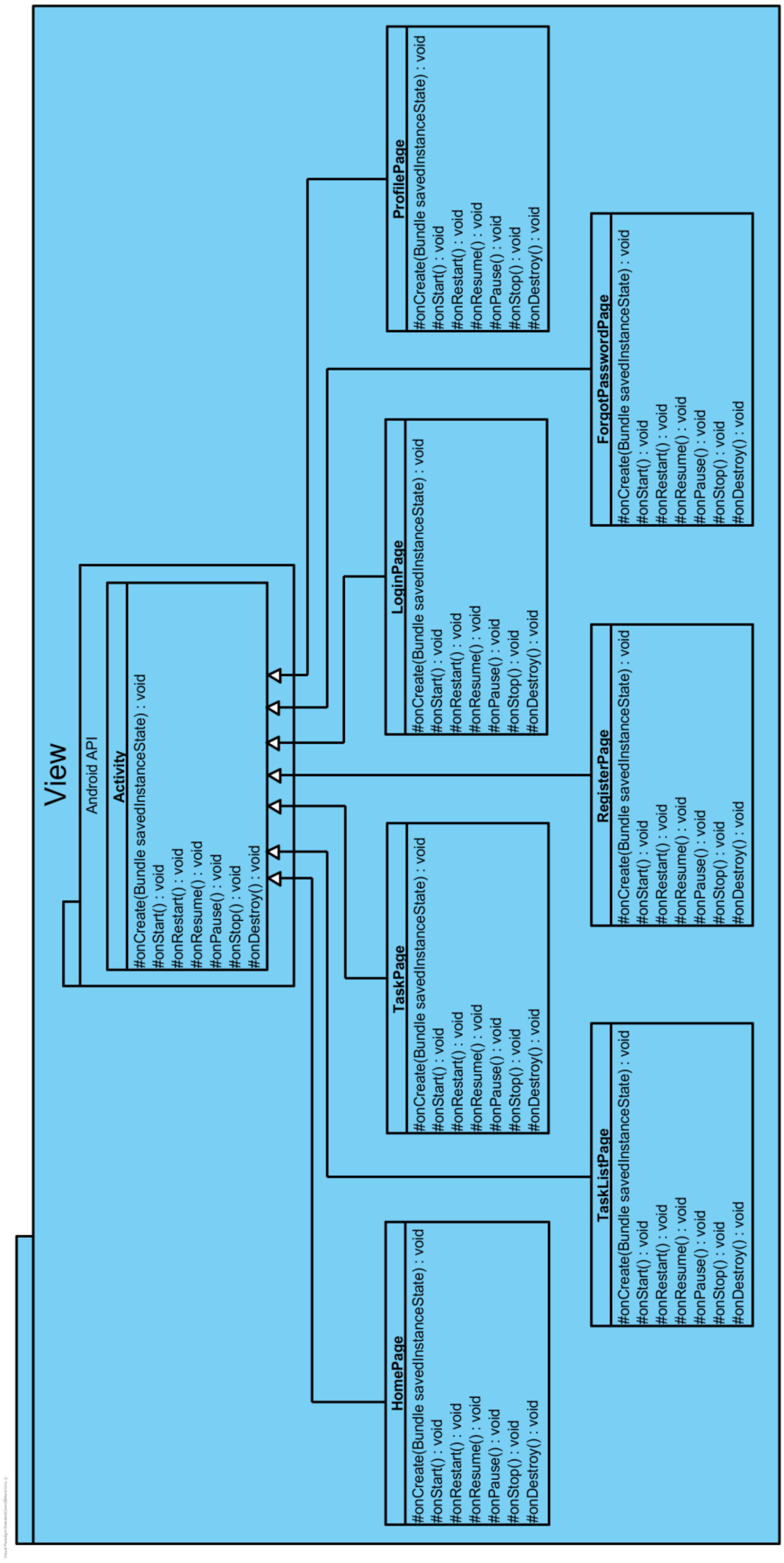


***Figure 3: View Package Class Diagram***

### 5.2.1.2. Presenter Package

The Presenter Package includes the View Logic Package and the Managers Package.

### View Logic Package

This package represents the underlying logic in View. Each View Logic class here is responsible for associated View class. It is the bridge between User Interface and Backend side of the system. It uses Managers Package's services in order to satisfy user operations.

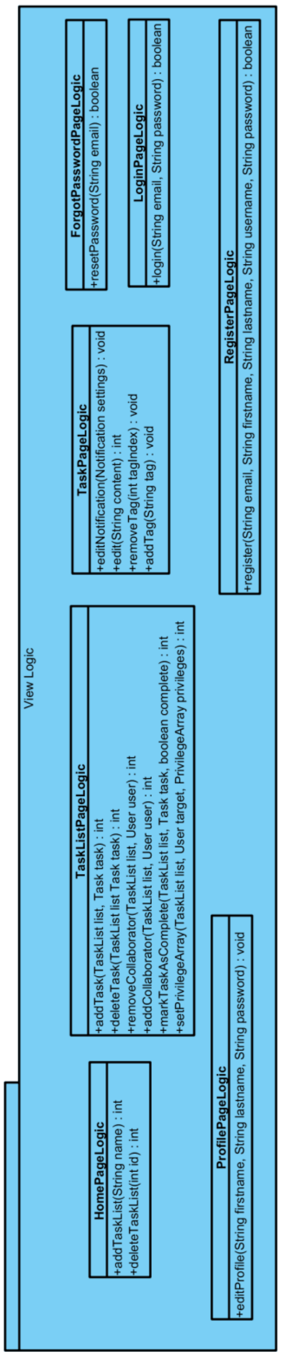

View Logic
ForgotPasswordPageLogic
+resetPassword(String email) : boolean
LoginPageLogic
+login(String email, String password) : boolean
RegisterPageLogic
+register(String email, String firstname, String lastname, String username, String password) : boolean
TaskPageLogic
+editNotification(Notification settings) : void
+edit(String content) : int
+removeTag(int tagIndex) : void
+addTag(String tag) : void
TaskListPageLogic
+addTask(TaskList list, Task task) : int
+deleteTask(TaskList list Task task) : int
+removeCollaborator(TaskList list, User user) : int
+addCollaborator(TaskList list, User user) : int
+markTaskAsComplete(TaskList list, Task task, boolean complete) : int
+setPrivilegeArray(TaskList list, User target, PrivilegeArray privileges) : int
HomePageLogic
+addTaskList(String name) : int
+deleteTaskList(int id) : int
ProfilePageLogic
+editProfile(String firstname, String lastname, String password) : void

*Figure 4: View Logic Package Class Diagram*

## Managers Package

Manager subpackage manages CRUD operations. Its services are called by the View subpackage on the same hierarchy.

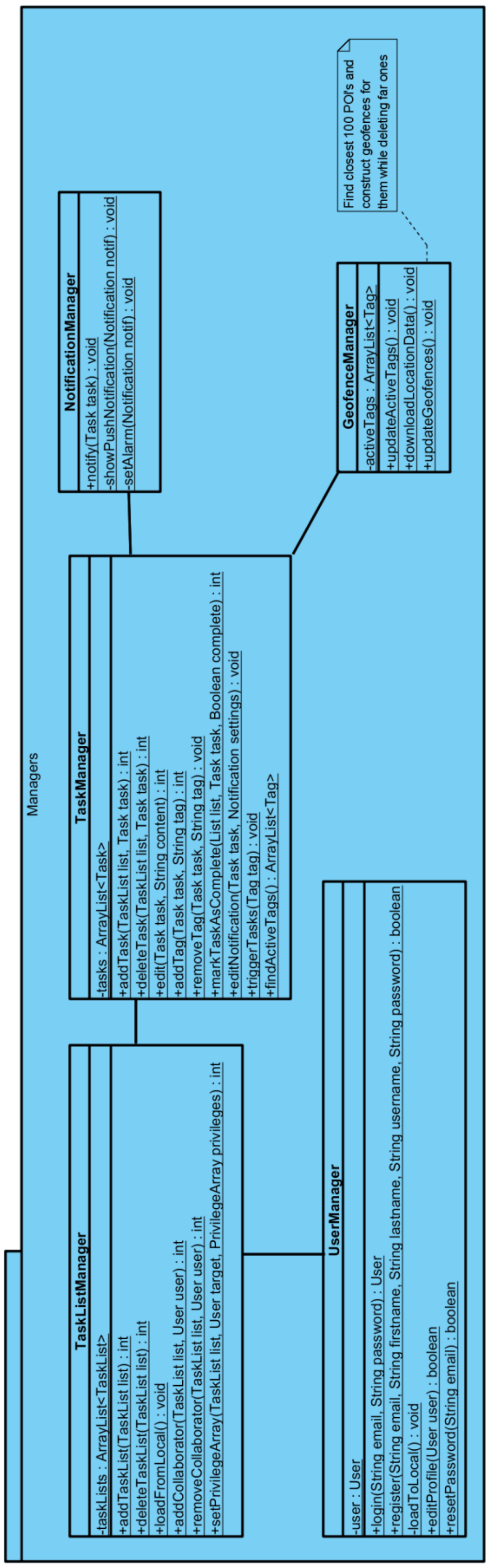

Managers
NotificationManager
+notify(Task task) : void
-showPushNotification(Notification notif) : void
-setAlarm(Notification notif) : void
Find closest 100 POI's and construct geofences for them while deleting far ones
GeofenceManager
-activeTags : ArrayList<Tag>
+updateActiveTags() : void
+downloadLocationData() : void
+updateGeofences() : void
TaskManager
-tasks : ArrayList<Task>
+addTask(TaskList list, Task task) : int
+deleteTask(TaskList list, Task task) : int
+edit(Task task, String content) : int
+addTag(Task task, String tag) : int
+removeTag(Task task, String tag) : void
+markTaskAsComplete(List list, Task task, Boolean complete) : int
+editNotification(Task task, Notification settings) : void
+triggerTasks(Tag tag) : void
+findActiveTags() : ArrayList<Tag>
TaskListManager
-taskLists : ArrayList<TaskList>
+addTaskList(TaskList list) : int
+deleteTaskList(TaskList list) : int
+loadFromLocal() : void
+addCollaborator(TaskList list, User user) : int
+removeCollaborator(TaskList list, User user) : int
+setPrivilegeArray(TaskList list, User target, PrivilegeArray privileges) : int
UserManager
-user : User
+login(String email, String password) : User
+register(String email, String firstname, String lastname, String username, String password) : boolean
-loadToLocal() : void
+editProfile(User user) : boolean
+resetPassword(String email) : boolean

***Figure 5: Managers Package Class Diagram***

### 5.2.1.3. Model Package

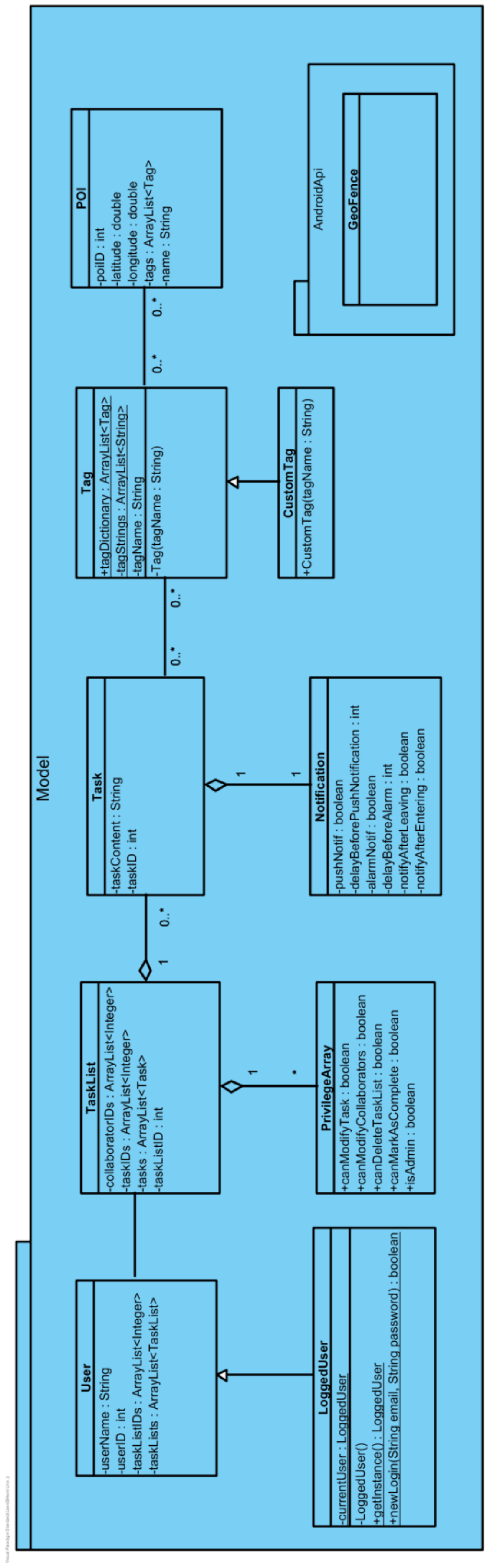

Model
POI
-poiID : int
-latitude : double
-longitude : double
-tags : ArrayList<Tag>
-name : String
0..*
0..*
Tag
+tagDictionary : ArrayList<Tag>
-tagStrings : ArrayList<String>
-tagName : String
-Tag(tagName : String)
CustomTag
+CustomTag(tagName : String)
AndroidApi
GeoFence
0..*
0..*
Task
-taskContent : String
-taskID : int
1
1
Notification
-pushNotif : boolean
-delayBeforePushNotification : int
-alarmNotif : boolean
-delayBeforeAlarm : int
-notifyAfterLeaving : boolean
-notifyAfterEntering : boolean
0..*
1
TaskList
-collaboratorIDs : ArrayList<Integer>
-taskIDs : ArrayList<Integer>
-tasks : ArrayList<Task>
-taskListID : int
1
*
PrivilegeArray
+canModifyTask : boolean
+canModifyCollaborators : boolean
+canDeleteTaskList : boolean
+canMarkAsComplete : boolean
+isAdmin : boolean
User
-userName : String
-userID : int
-taskListIDs : ArrayList<Integer>
-taskLists : ArrayList<TaskList>
LoggedUser
-currentUser : LoggedUser
-LoggedUser()
+getInstance() : LoggedUser
+newLogin(String email, String password) : boolean

***Figure 6: Model Package Class Diagram***

## 5.2.2. Server Package

### 5.2.2.1. Request Handler Package

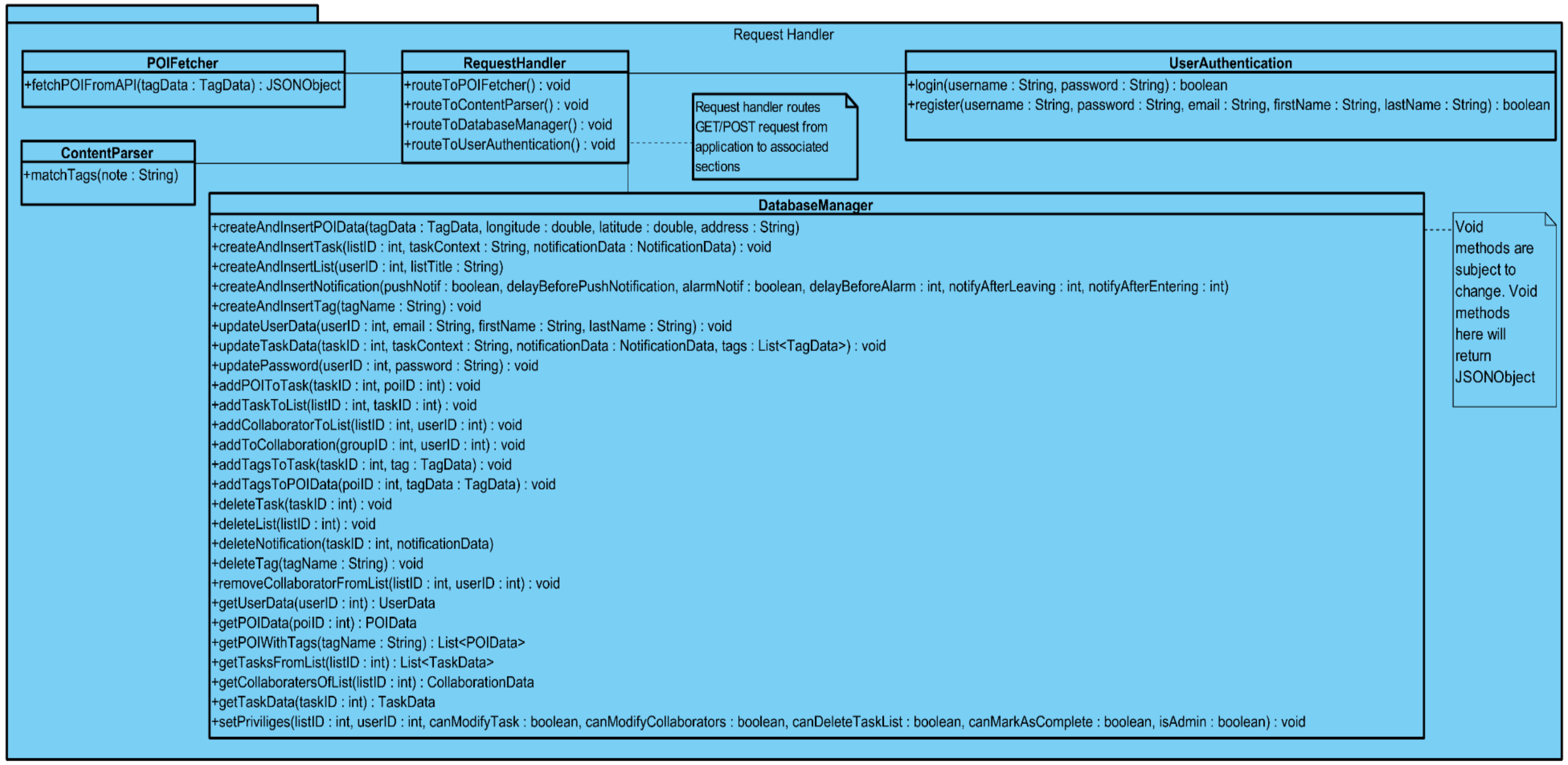

Request Handler
POIFetcher
+fetchPOIFromAPI(tagData : TagData) : JSONObject
ContentParser
+matchTags(note : String)
RequestHandler
+routeToPOIFetcher() : void
+routeToContentParser() : void
+routeToDatabaseManager() : void
+routeToUserAuthentication() : void
Request handler routes GET/POST request from application to associated sections
UserAuthentication
+login(username : String, password : String) : boolean
+register(username : String, password : String, email : String, firstName : String, lastName : String) : boolean
DatabaseManager
+createAndInsertPOIData(tagData : TagData, longitude : double, latitude : double, address : String)
+createAndInsertTask(listID : int, taskContext : String, notificationData : NotificationData) : void
+createAndInsertList(userID : int, listTitle : String)
+createAndInsertNotification(pushNotif : boolean, delayBeforePushNotification, alarmNotif : boolean, delayBeforeAlarm : int, notifyAfterLeaving : int, notifyAfterEntering : int)
+createAndInsertTag(tagName : String) : void
+updateUserData(userID : int, email : String, firstName : String, lastName : String) : void
+updateTaskData(taskID : int, taskContext : String, notificationData : NotificationData, tags : List<TagData>) : void
+updatePassword(userID : int, password : String) : void
+addPOIToTask(taskID : int, poiID : int) : void
+addTaskToList(listID : int, taskID : int) : void
+addCollaboratorToList(listID : int, userID : int) : void
+addToCollaboration(groupID : int, userID : int) : void
+addTagsToTask(taskID : int, tag : TagData) : void
+addTagsToPOIData(poiID : int, tagData : TagData) : void
+deleteTask(taskID : int) : void
+deleteList(listID : int) : void
+deleteNotification(taskID : int, notificationData)
+deleteTag(tagName : String) : void
+removeCollaboratorFromList(listID : int, userID : int) : void
+getUserData(userID : int) : UserData
+getPOIData(poiID : int) : POIData
+getPOIWithTags(tagName : String) : List<POIData>
+getTasksFromList(listID : int) : List<TaskData>
+getCollaboratersOfList(listID : int) : CollaborationData
+getTaskData(taskID : int) : TaskData
+setPriviliges(listID : int, userID : int, canModifyTask : boolean, canModifyCollaborators : boolean, canDeleteTaskList : boolean, canMarkAsComplete : boolean, isAdmin : boolean) : void
Void methods are subject to change. Void methods here will return JSONObject

***Figure 7: Request Handler Package Class Diagram***

#### 5.2.2.2. Data Layer Package

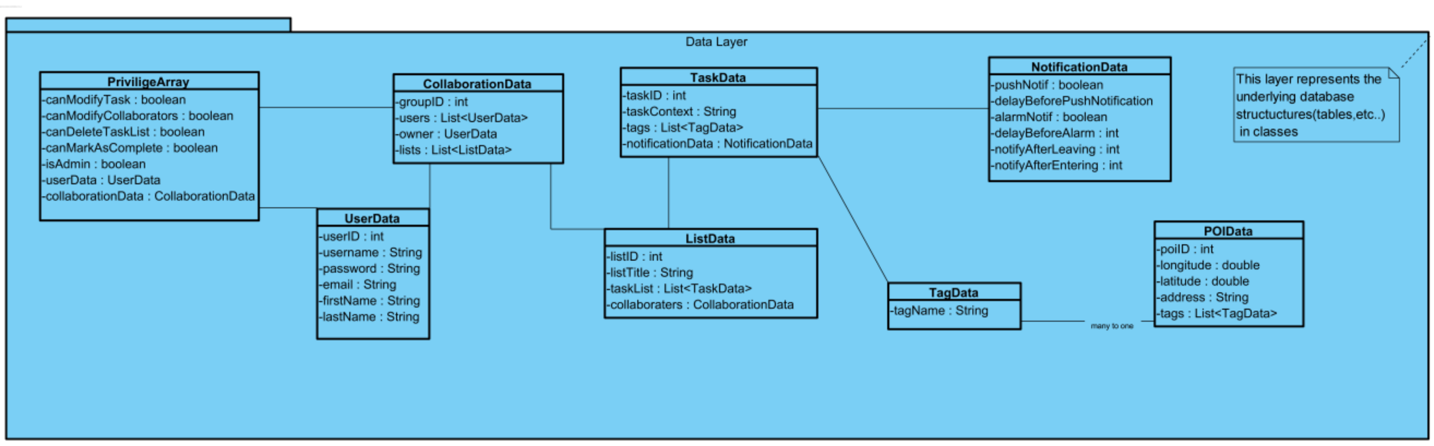


*Figure 8: Data Layer Package Class Diagram*

### 5.2.3. Search Tools Package

Search Tools stand for APIs that'll be used for searching the web in order to satisfy the user requests, searching the given queries and mining information.

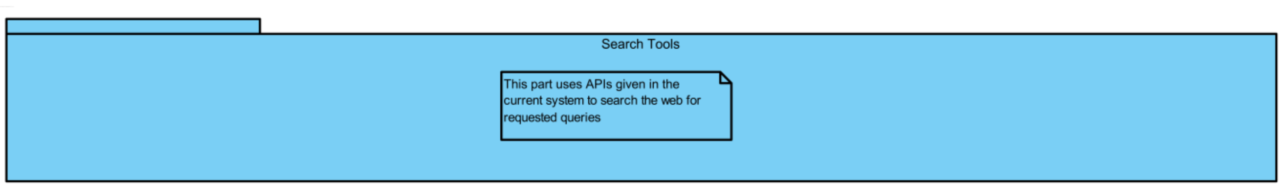

Search Tools
This part uses APIs given in the current system to search the web for requested queries

*Figure 9: Search Tools Package Class Diagram*

# 6 Impact of Engineering Solutions

Locum brings a different approach to everyday problems and their outdated solutions, and it has a huge impact on many areas, since it can be used by all people of all ages, all the time. The discovered implications are as follows:

## 6.1. Economic Impacts

- Our application allows the users to accomplish their intended purchases with ease, as soon as possible. Therefore, it can be said that it has a positive effect on overall economy.
- In addition to that, it reduces the amount of time/money spent on accomplishing a task, which is economically beneficial to our users.

## 6.2. Environmental Impacts

- Our application allows the users to accomplish their intended purchases with ease, as soon as possible. Therefore, it can be said that it has a positive effect on overall economy.
- In addition to that, it reduces the amount of time/money spent on accomplishing a task, which is economically beneficial to our users.

## 6.3. Social Impacts

- With the help of Locum, users will not have to allocate another time for doing the tasks they have forgotten (e.g. remembering you hadn't buy eggs and driving back to the market). Instead, they will be reminded by our application.
- Our application allows people to collaborate on different lists, which increases social interaction among others. This makes it easier for families to have shared grocery lists, so whenever a family member is available and near a point of interest, the others won't have to go get them.

## 6.4. Political Impacts

- Locum application is unlikely to create any political impact.

## 6.5. Impacts on Security

- At the first glance, it may seem that we are tracking the user, which is against personal security. However, we are keeping all user information in hashed format, and Geofencing technology allows us to use user location data as low as possible.

## 6.6. Impacts on Health and Safety

- As it is intended, Locum has the location data for hospitals and pharmacies. Given that our user needs to buy some medicine, or needs to see a doctor, our application could remind them. Locum reminds them when a user needs to buy medicine when they are near a pharmacy, or take their medicine when they get home. This is useful in terms of health.

## 7 Tools and Technologies Used

In this section, the development tools and frameworks, library resources and the API's used during Locum's development will be demonstrated briefly.

## 7.1. Development Tools and Frameworks

Thanks to some Development Tools and Frameworks, the implementation of Locum was accelerated in a great manner. In this part, these development tools is briefly explained and their usage in Locum is demonstrated.

### 7.1.1. Android Studio

Locum is an Android application, and Android Studio was used in developing it. Android Studio is widely known for developing mobile applications for Android. It made it easier to test the application both on available phones (by installing APK files) and it on all kinds of emulators on which helped testing in all API levels and brands as desired. The IDE of Android Studio is User Friendly and creating Activity UI's is made very easy. Code completion also accelerated the progress, and since it was linked to git-kraken, source control was made easier to manage.

### 7.1.2. MobaXterm

We used MobaXterm for accessing Amazon EC2 Ubuntu computer. This tool helped us to securely connect our python server that is running on Ubuntu. We easily deployed our server program, NLP models and datasets to the server using MobaXterm. We also used MobaXterm for testing the server and debugging it.

### 7.1.3. Github

GitHub is again a widely known git platform and is used for version control. We had to find a way to manage the source code of Locum. Our team consists of five people, and each of us had different responsibilities requiring different branches, so we had to find a way to work on the code together remotely. To manage this we used Android Studio along with GitHub and GitKraken.

### 7.1.4. GitKraken

We used GitKraken along with GitHub and Android Studio to manage the source code of Locum. GitKraken is a client that enables being a more productive Git user thanks to its intuitive interface and experience, merge conflict editor, built-in code editor and task tracking features. This helped us a lot since we had to work on different branches and surely some conflicts occurred, and we managed them easily, and reverted our commits when we felt necessary.

### 7.1.5. Express NodeJS

Our main server is written in NodeJS with Express framework. Express is a lightweight, minimalist framework that provides lots of features for web applications. It is easy to implement a server with this framework. Http request routing is well designed, fully functional and it has a very good error handling mechanism. We can easily debug our server. The MySQL features it has also enables us to query SQL statements to our database easily and reliably.

### 7.1.6. Flask

We used flask micro framework in order to implement our python server which is responsible for Natural    Language processing. It has a good http routing and it enabled us to have high computing in server side. Its minimal and simple structure helped us to have a simple python server that satisfies requests very quickly.

### 7.1.7. Amazon Web Services EC2

We used Amazon Web Services EC2 to deploy our NLP flask server to the cloud. EC2 services provides machines that are fast, secure, available and easy to configure. We choose to use Ubuntu instance with 16GB RAM and 25GB storage because we need to have capacity for using Natural Language Processing models. Moreover, our cloud Ubuntu machine is fast and when Android host posts note to the server, it processes and returns the result very quickly.

### 7.1.8. AWS RDS

We chose to use Amazon RDS(Relational Database Service) because it is secure,easy to administer, available and fast. All information related to Locum is stored in RDS. Monitoring is easy and we can see its availability and response times. It is easy to connect Mysql Workbench to RDS, in order to see our tables.

### 7.1.9. Microsoft Azure

Our NodeJS server is deployed on Microsoft Azure. We chose to deploy NodeJS server to Azure since Azure has a special service for Node applications. Azure helped us monitor our server and debug and deploy to the cloud in a fast way.

### 7.1.10. MySQL

We used MySQL in our database. It provides a secure and reliable database management. We also used MySQL Workbench for accessing our RDS database remotely.

### 7.1.11. SQLite

Locum uses Room which is built on top of SQLite. SQLite is used in the client side of Locum and is used to query certain statements to local database.

### 7.1.12. Postman

We used Postman for checking Http requests and responses. This helped us in debugging our servers and also test our servers.

#### 7.1.13. Photoshop

Adobe Photoshop is used by designers to create artwork. Adobe Photosh op is used in Locum in designing some UI components. The use of  Adobe Photoshop was combined with Adobe Illustrator during designing artwork for Locum.

#### 7.1.14. Illustrator

Adobe Illustrator is used by designers to create vector images. Adobe Illustrator is used in Locum for the logo and in designing some UI components. The use of Adobe Illustrator was combined with Adobe Photoshop during designing artwork for Locum.

### 7.2. Library Resources

In this section, the library resources used in the development of Locum, namely the Room Persistence Library, Google Play Services and Android Libraries will be demonstrated.

#### 7.2.1. Room Persistence Library

Room persistence library provides abstraction over SQLite which helped us query SQLite statements to the local database very easily. You can specify Java objects as entities in Room and then save/fetch that object into/from the database which is very easy and reliable to use. Entities in Room correspond to tables in database. You can also create an interface that has methods for accessing databases as well.

#### 7.2.2. Google Play Services

Locum is published on Google Play Services so that users can use Locum. It can be adjusted globally upon user request.

#### 7.2.3. Android Libraries

Locum uses AndroidX library which is a new and improved version of Android Support Library. It included necessary features such as widgets, Animations, layouts and views.

### 7.3. API’s Used

In this section, the API’s used in the development of Locum will be demonstrated. Locum highly depends on the following API’s functioning and being wired correctly. The following API’s saved a huge amount of development time for the team.

#### 7.3.1. Google Maps API

Google Maps API is used whenever a mapview is displayed, na mely add Custom Location and Alarm Screens in Sections 9.9. Add Task with Customized Location Screen and 9.12. Push Notification and Alarm Screens.

### 7.3.2. Google Places API

The Places API is used when returning information about places that use HTTP requests. Places are defined here as establishments, geographic locations, or prominent points of interest. We have used Places API for fetching location data like pharmacies, banks, supermarkets, etc within a given circle.

### 7.3.3. Geofencing API

The Geofencing API intelligently uses the device sensors to accurately detect the location of the device in a battery-efficient way. The API allows us to know when a user enters or leaves an active Geofence.

## 7.4. NLP Libraries and Technologies

### 7.4.1. Word2Vec

For processing text, our main model was Word2Vec. Word2Vec is basically a couple of related models that are used to produce word embeddings. These are shallow models, two-layer neural networks that are trained to reconstruct linguistic word contexts. Word2Vec takes as its input a large corpus of text and produces a vector space, typically of several hundred dimensions, with each unique word in the corpus being assigned a corresponding vector in the space.

Word2Vec let us compute similarities between words. As expected, sim(aspirin, pharmacy) > sim(aspirin, car_repair) so we "understand" that aspirin is something that could be bought from a pharmacy. However, as user input is unstructured either in English or Turkish, we needed to do some pre-processing.

### 7.4.2. NLTK

For English pre-processing, we have used Natural Language Toolkit (NLTK). NLTK is used when building Python applications working with natural language data. It has libraries classification, tokenization, stemming, tagging, lemmatization and parsing.

### 7.4.3. Zemberek

For Turkish pre-processing, we have used Zemberek. Zemberek is a Java library for processing Turkish text. Similar to NLTK, it provides tokenization, tagging, lemmatization, and parsing.

# 8 Engineering Solutions and Contemporary Issues

This section will describe major issues that we've encountered and our solutions to address these issues. We will briefly explain our design choices and how we tackle these issues with these choices. Moreover, we will also explain trade-offs and give information about how we implemented Locum in order to address issues.

## 8.1. Language and Communication

Locum supports both English and Turkish reminders. UI items such as menus, fields, etc.. are all in English, but the user can create lists, notes in Turkish as well. Locum can generate associated tags both for Turkish and English notes using Natural Language Processing. For example, if the user specificies "Tavuk al" as a note, Locum can generate associated tag such as "supermarket" or "convenience store". Especially Turkish language processing is hard and the accuracy of it is still low compared to English because of scarcity of good models for Turkish processing.The other issue is distinguishing between Turkish and English notes. For that purpose, we decided to let user choose notes' language because of accuracy issues that we faced with language detection. This made the language processing more reliable with the cost of requiring one more user input.

We have used NLTK for English and Zemberek for Turkish. These libraries are used for preprocessing the text. For example, when user types in something like "I need to withdraw 3000 Euros as soon as possible", NLTK library first does tokenization and then positional tagging. It neglects everything other than verbs and nouns. After having a set of verbs and nouns, we use the lemmatizer for reducing them even to a more plain form. What we eventually have from the initial sentence is ["need", "withdraw", "euro"].

After "reducing" the sentence, we do similarity score computation of these tokens with each of our POI types. For achieving that, we've used Word2Vec (see 7.3.4). We return the top k=5 results to the device.

## 8.2. Internet Connection Issues

Locum requires user to have continuous internet connection for accessing all functionalities of the application. Functionalities such as authentication, adding collaborators, adding/removing lists and tasks all require the user to have a consistent internet access. This introduces a bottleneck to the system since internet connection quality and speed varies from location to location. However, Locum also uses Androids' local database in order to lower internet requests and providing the user a smoother experience. It fetches important information from the server and then saves it into the local database so that the user can use this information whenever he needs, without waiting response from the server. Even when there is no internet, the user can benefit from the location-based reminders of Locum.

The other issue about networking is that our database and server are in separate networks. Our Azure server is accessing AWS MySQL server and this might slightly delay the system. However, delays caused by this are not severe and they do not affect the overall user experience.

## 8.3. Speed Issues

The speed issue is that large and important information are fetched during only authentication and by the refresh operation, so that the user does not feel like he is waiting all the time. Here, we had to decide between user experience and synchronization between user tasks.

In our NLP server, as language models are large and take a long time to load, we have chosen an always-on approach, where models are loaded and waiting for user input all the time. That way, when user types in a new task, the only time spent is the analysis of that task.

## 8.4. Security Issues

In terms of security, Locum does not hold user's location in server the side. All operations that include user's location are kept in the client side. Moreover, we are using AWS's RDS service explained in 7.1.8. AWS RDS, in the MYSQL database server, and Amazon guarantees the security of our database.

## 8.5. Platform Development Issues

The team had different phones from different brands, some android, some iOS, working on different API's. We decided on developing only for Android devices because Android provides reliable Geofencing support and location services. Moreover, networking in Android is fast and its Room library provides reliable data storage.

Not only this, we struggled finding a minimum API level for the project at first. The minimum API level we had was 21 at first, but we decided between the two trade-offs "usability" and "ease of use" and used some components available for API 23 or higher, so we updated the minimum API level to 23.

In the end, Locum works on only Android devices with API level greater than 23 which corresponds to 85% percent of Android phones.

## 8.6. User Consent Issues

Our application highly depends on the location of the user, so it requires the consent of the user in sharing his location information with the application. At the first launch of the application, the user is given an option if he is going to share his information. If he doesn't want to share it, then he won't be using the location based reminder features of Locum.

The application allows the user to register using a valid email and password, therefore the user should also not have a problem about sharing his email with the application.

## 8.7. Gender Issues

Locum is an application that can be used by everyone, so it is not focused on a single gender. The gender of the user is irrelevant to the context of Locum, so it is not necessary while registering to the application. All the users are identified by their usernames.

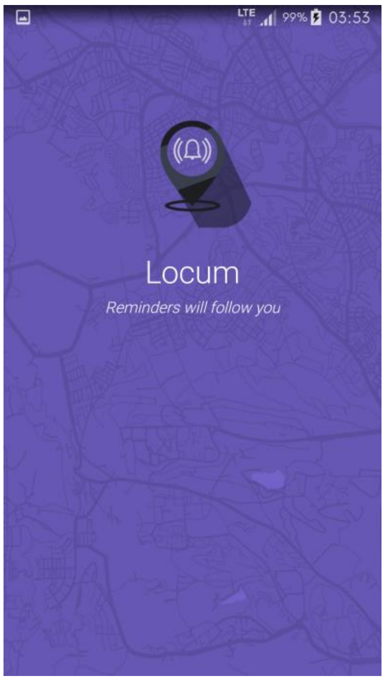
## 8.8. Future Work

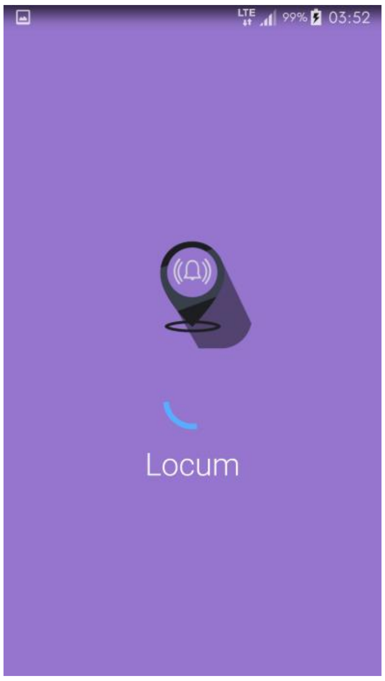
For future work, we can improve the security of application by encrypting all user related information and changing protocol to HTTPS. We can also change server side checks as well. Moreover, we can have more servers with a load balancer to increase the scalability of our application. We can also add more language support such as Turkish, Spanish, German etc. We can also improve our tag generation with user feedback. Additionally, we can improve the accuracy of out NLP system.

# 9 Appendix A – User Manual

In this part, the user manual will be presented along with the user interfaces, how the user interacts with the app and functionalities of each interface. Each screen shown below, is a functional view of the application.

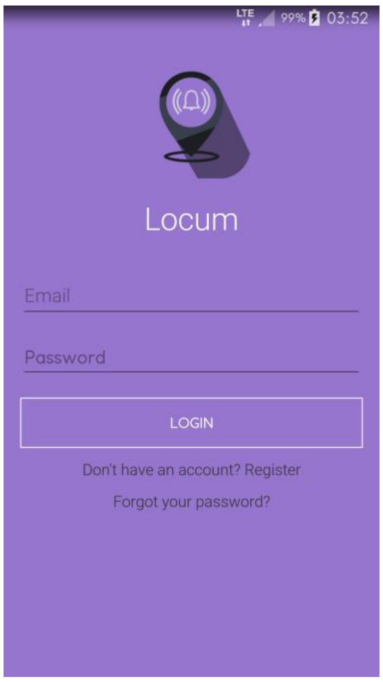
## 9.1. Splash Screen

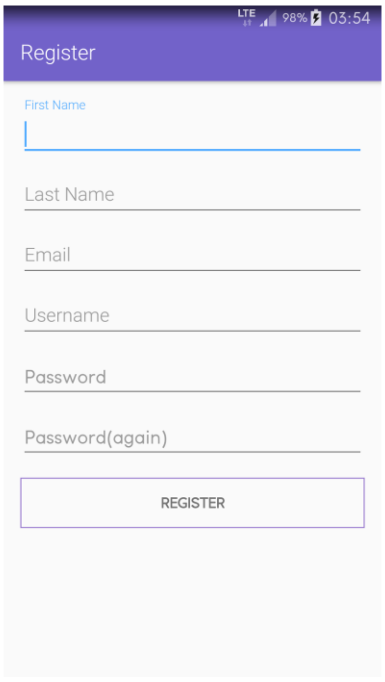


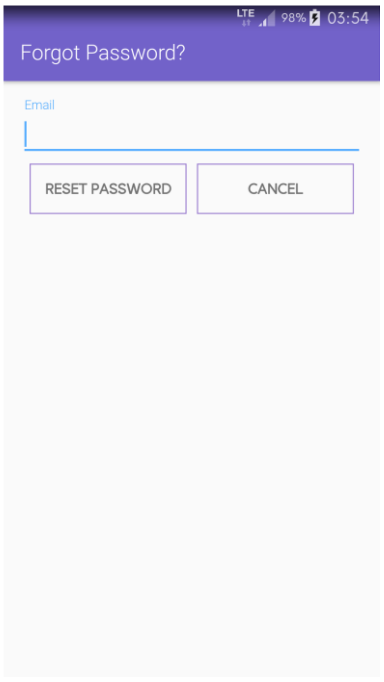


The screen on the left will be presented to the user while the application is loading and during automatic login. The screen on the right is presented after the user is logging in manually.

## 9.2. Log In, Register and Forgot Password Screens





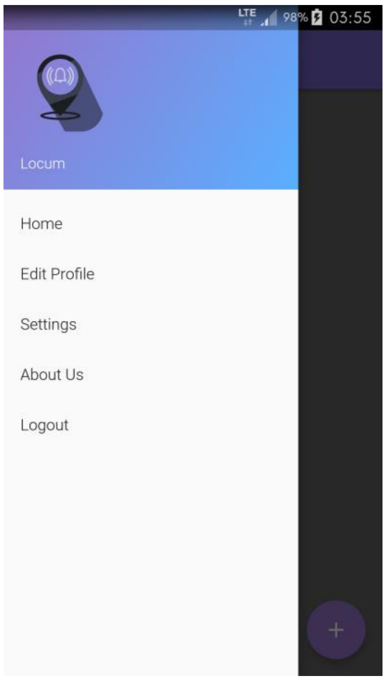


On the left is the Login screen of the application. It asks the user his username and password. There is an option to register, which redirects to the Register Screen.

On the middle, is the Register Screen of the application. It asks the user his first name, last name, email, username and password. The email must be a valid email (must include the "@" sign), and the password shouldn't be shorter than 4 characters.

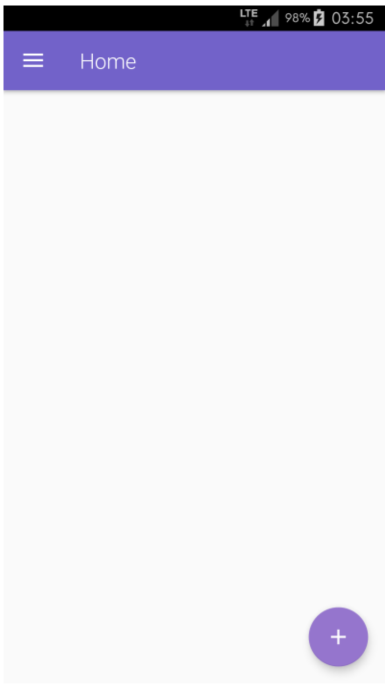
On the right is the Forgot Password screen. Here, the user can enter his email address for a mail to be sent to him to reset his password.

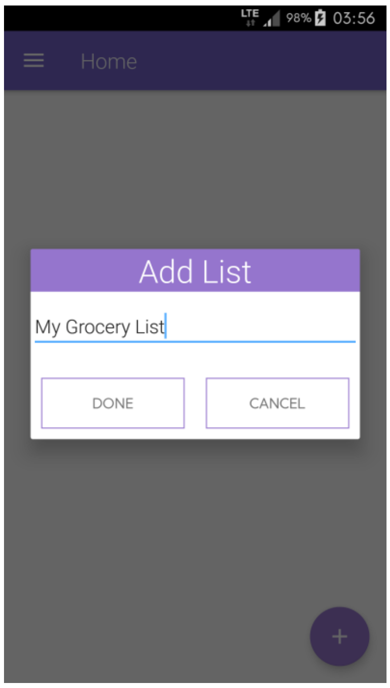
## 9.3. Navigation Drawer Screen



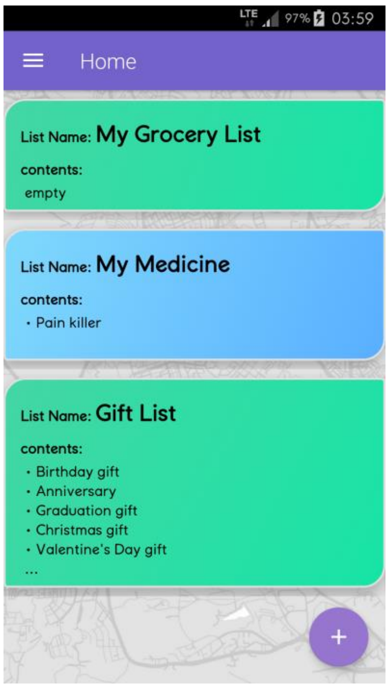
There is a menu icon on the top right corner of the views, which is opens the Navigation Drawer Screen. The menu opens once the user clicks that icon, without changing the current screen the user is in at. In the menu, there are options to go to the Home Page, Edit Profile, Settings and About Us Screens. The user can log out by choosing that option from the Navigation Drawer.

## 9.4. Home Page/Task List and Add List Screens





My Grocery List



List Name: My Grocery List
contents:
empty

List Name: My Medicine
contents:
• Pain killer

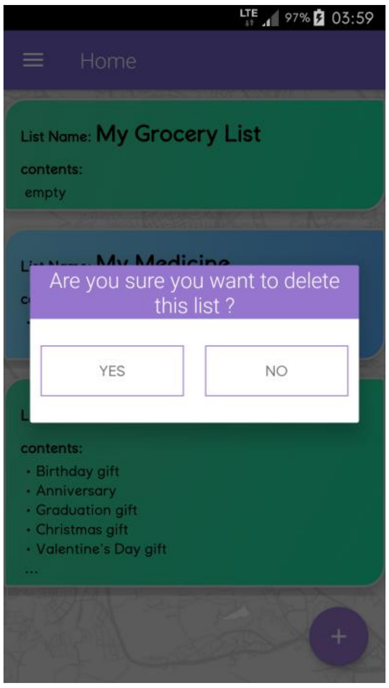
List Name: Gift List
contents:
• Birthday gift
• Anniversary
• Graduation gift
• Christmas gift
• Valentine's Day gift
...

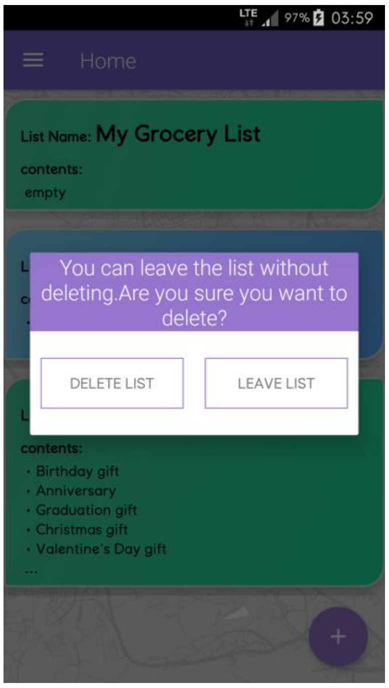
On the left is the Home Page/Task List Screen, which presents the user lists he owns. This screen is initially empty, and it fills up when the user creates a list using the "+" button. Once he adds tasks to that list, he can see up to five tasks under the list title. He can also see if the tasks are complete by seeing if there is a checkmark at the end of these tasks.

On the middle is the pop-up that comes up when the "+" button on the lower right corner is pressed, which enables the user to enter the list name, and be pressing the done button, the list is created.

On the right is the final view of the home page, filled with task-lists and tasks under these.

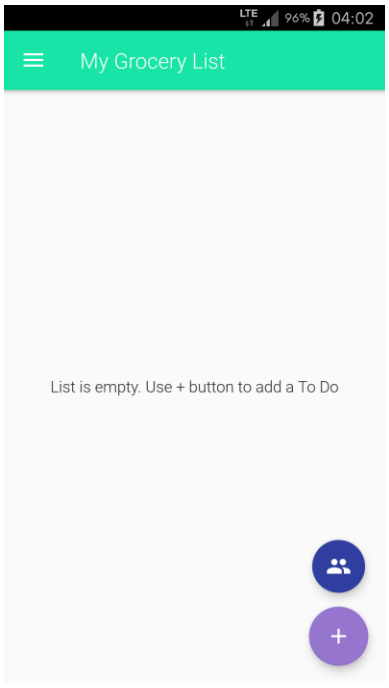
## 9.5. Delete List Screen

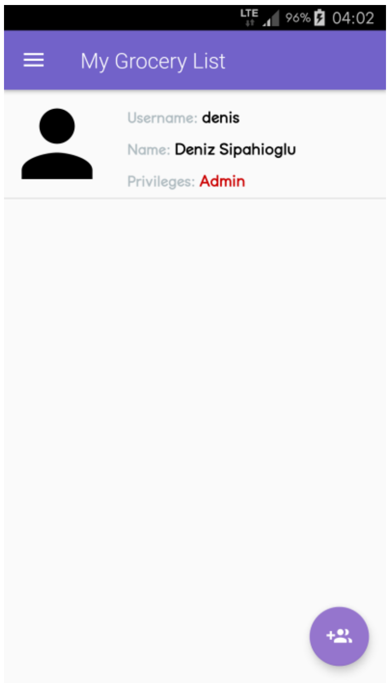


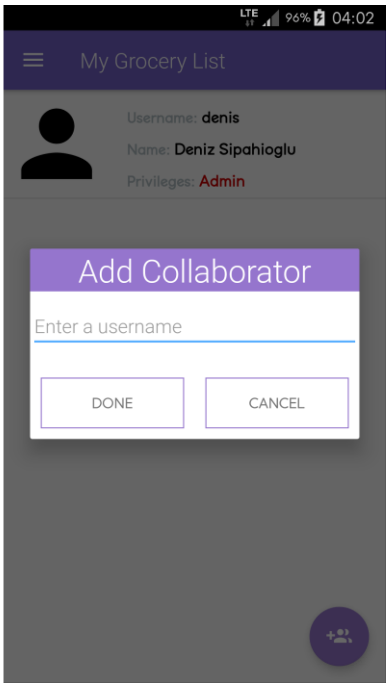


These screens are presented when the user long presses a task-list, and options to delete the list or leave the list come up. If that user is the owner of that list, deleting the list means also wiping it for the collaborators. If the user is only a collaborator, an option to leave the list instead is presented. This way, a collaborator that no longer needs that list can leave without affecting other people.

## 9.6. Collaborators Screen, Add Collaborator Screen



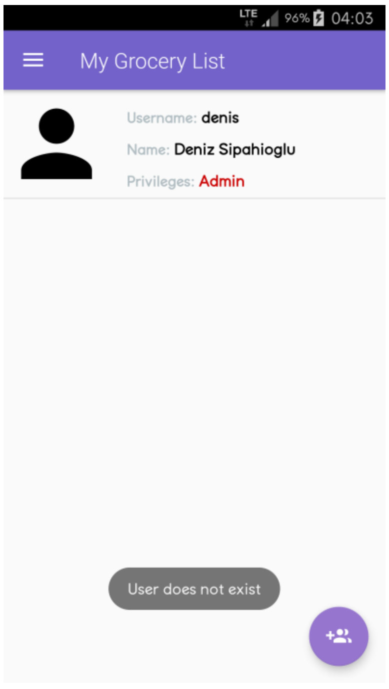




On the left is the task screen. When the user presses the blue collaborator button on the bottom right corner of the task screen, the second image shows up.

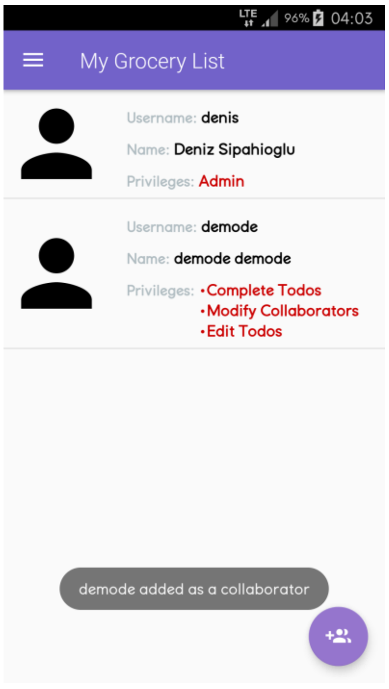
By pressing the blue collaborators button, the user can see the collaborator list as seen on the middle, and add collaborators using the "+" button on the bottom right.

As it can be seen from the figure on the right, the user can add collaborators by typing their usernames in. If that user does not exist, a toast pops up acknowledging the user that the other user does not exist, which is demonstrated below.



Username: denis
Name: Deniz Sipahioglu
Privileges: Admin

User does not exist

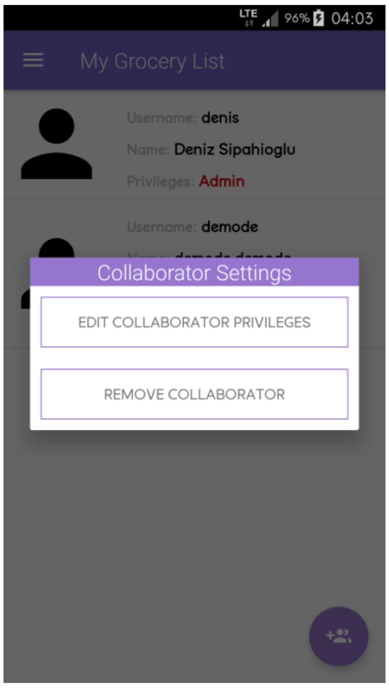


Username: denis
Name: Deniz Sipahioglu
Privileges: Admin

Username: demode
Name: demode demode
Privileges: •Complete Todos
•Modify Collaborators
•Edit Todos

demode added as a collaborator

## 9.7. Collaborator Settings, Edit and Remove Collaborator


96% 04:03

My Grocery List

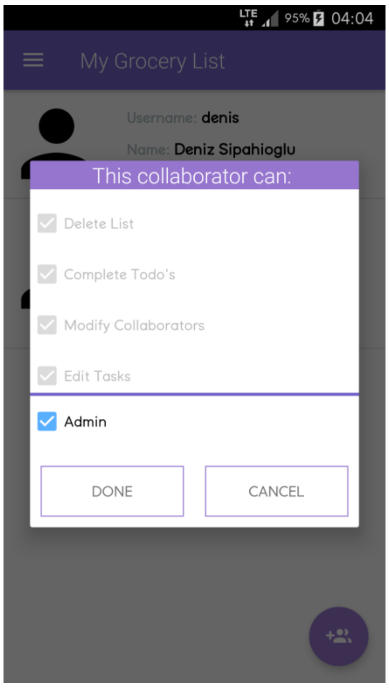
Username: denis
Name: Deniz Sipahioglu
Privileges: Admin

Username: demode
Name: demode demode

Collaborator Settings

EDIT COLLABORATOR PRIVILEGES

REMOVE COLLABORATOR


95% 04:04

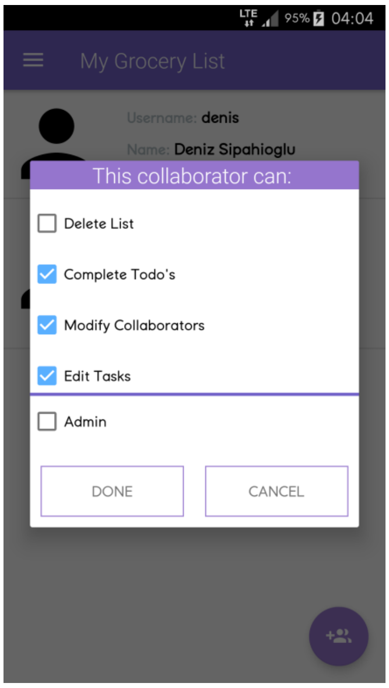
My Grocery List

Username: denis
Name: Deniz Sipahioglu

This collaborator can:

- [x] Delete List
- [x] Complete Todo's
- [x] Modify Collaborators
- [x] Edit Tasks
- [x] Admin

DONE CANCEL


95% 04:04

My Grocery List

Username: denis
Name: Deniz Sipahioglu

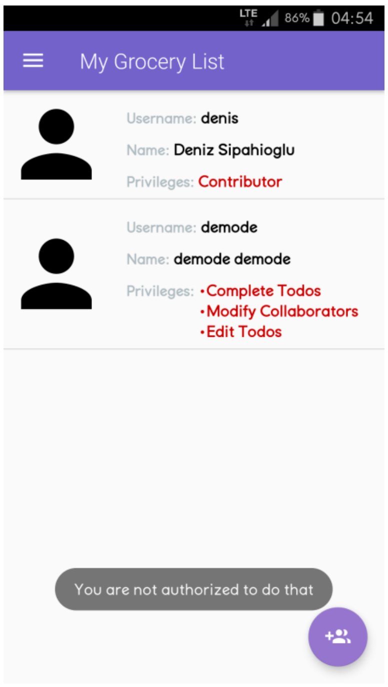
This collaborator can:

- [ ] Delete List
- [x] Complete Todo's
- [x] Modify Collaborators
- [x] Edit Tasks
- [ ] Admin

DONE CANCEL

The figure on left shows the collaborators screen where the user can edit collaborator privileges by long-pressing on a collaborator and remove collaborator if he has the privileges to do so.

The second and third figures shows the view inside the edit collaborator pop-up, where the user can choose if the user of subject can Delete Lists, Complete Tasks, Modify Collaborators and Edit Tasks. If the user of subject can to these, entire he automatically becomes an admin. Again, if only the admin option is selected, all other privileges are selected automatically.


86% 04:54

My Grocery List

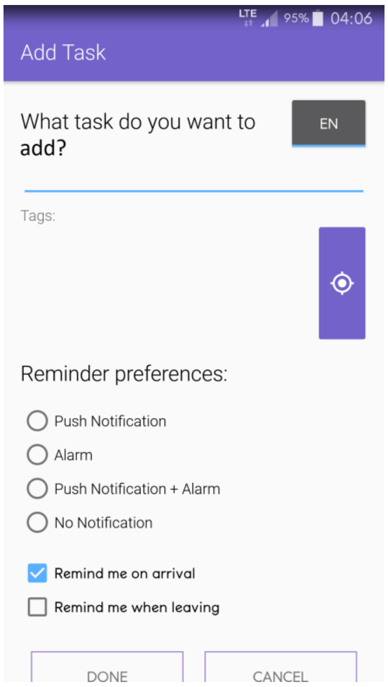
Username: denis
Name: Deniz Sipahioglu
Privileges: Contributor

Username: demode
Name: demode demode
Privileges: •Complete Todos
•Modify Collaborators
•Edit Todos

You are not authorized to do that

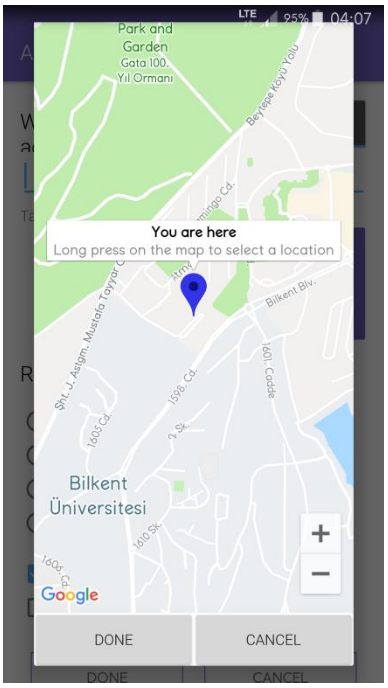
This figure shows when a collaborator can be removed. If the current user doesn’t have the privilege to modify collaborators, a toast pops up acknowledging the user.

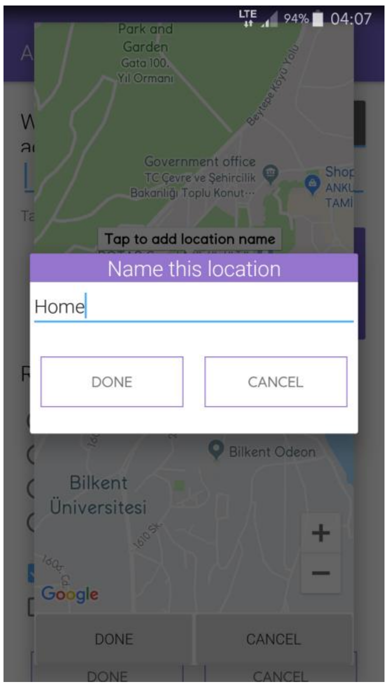
## 9.8. Add Task Screen



This screen will allow the user to add a new task to one of his lists. He needs to type in what he would like to do, location tags will be automatically generated. He can remove irrelevant tags by simply by deselecting undesired tags, and add tags by selecting the ones he wants. He can set reminder preferences and select either one of push notification, alarm, both, or no notification at all. Being reminded on arrival/when leaving can also be selected. The user can see the selections he made on the task screen as icons on the right of the task.

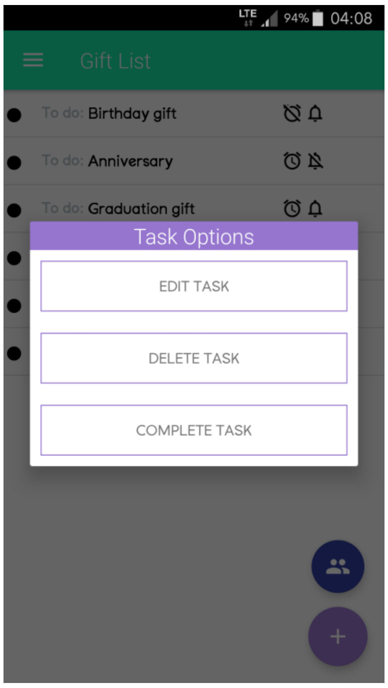
## 9.9. Add Task with Customized Location Screen





On this screen the user can select if he wants to be reminded at a custom location, he can press the location button on the right, then a popup will be shown for him to pin the location he wants. On this view, he can long press on a location to select that location, and name the location as he wishes. After this, when he presses the done button, he will be reminded of that task in that custom location from that time on.

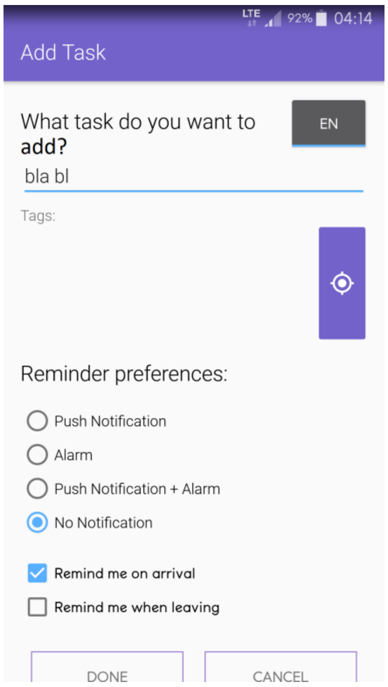
## 9.10. Task Options Screen (Edit, Delete, Complete)


94% 04:08

Gift List

- To do: Birthday gift
- To do: Anniversary
- To do: Graduation gift

Task Options


92% 04:14

Add Task

What task do you want to add? bla bl

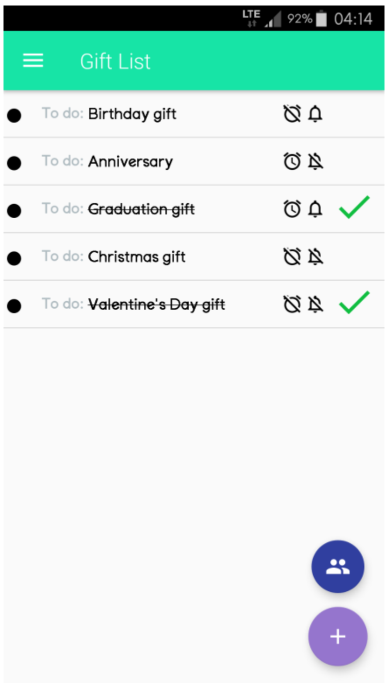
Reminder preferences:

- ( ) Push Notification
- ( ) Alarm
- ( ) Push Notification + Alarm
- (●) No Notification

- [x] Remind me on arrival
- [ ] Remind me when leaving

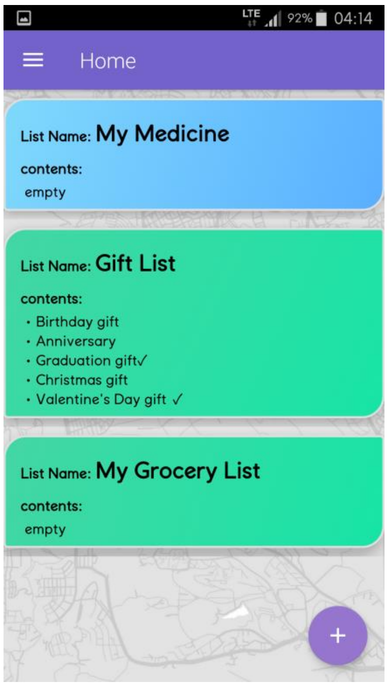
On the tasks page, once the user presses and holds down a specific task, a pop-up window entitled Task Options will be opened, and it'll offer the user 3 options: Edit Task, Delete Task and Complete Task. In the Edit Task option, the Add task screen shows up, filled in with that task details, and the user can change the task there. The delete task option deletes that task from the list. The complete task button is explained in the Complete Task Screen section.

## 9.11. Complete Task Screen


92% 04:14

Gift List

- To do: Birthday gift
- To do: Anniversary
- To do: ~~Graduation gift~~ ✓
- To do: Christmas gift
- To do: ~~Valentine's Day gift~~ ✓


92% 04:14

Home

List Name: My Medicine
contents:
empty

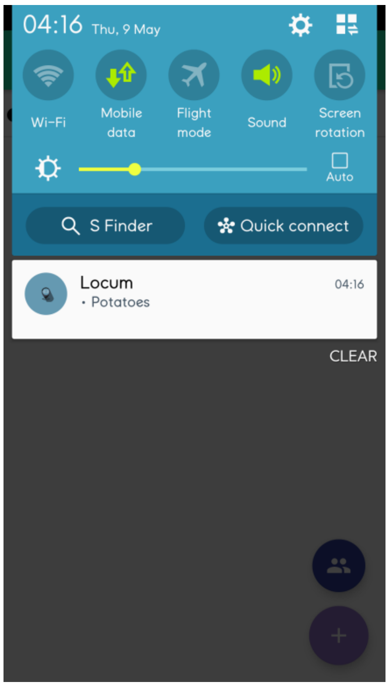
List Name: Gift List
contents:
- Birthday gift
- Anniversary
- Graduation gift✓
- Christmas gift
- Valentine's Day gift ✓

List Name: My Grocery List
contents:
empty

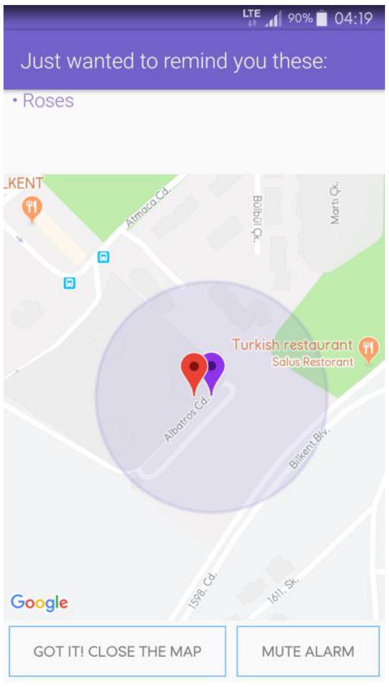
There are two ways a user can complete a task.

By pressing on the task once, or by long pressing and choosing the Complete Task option. When a task is completed, it is striked out and a check icon appears on the right of the task. The task also has a tick on its right on the list screen.

## 9.12. Push Notification and Alarm Screens



Locum
• Potatoes
04:16

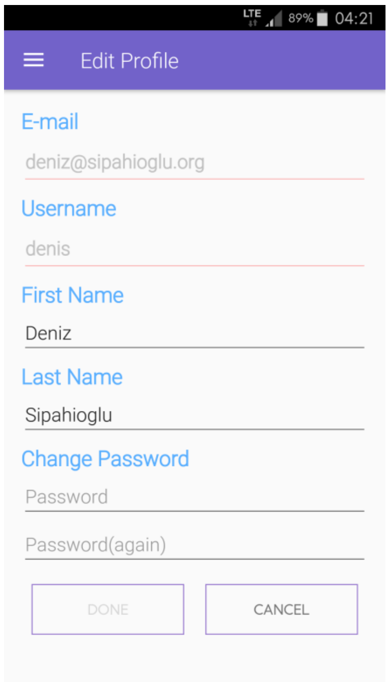


Just wanted to remind you these:
• Roses

The screen on the left demonstrates when the Push Notification option is selected to be reminded of a task.

The screen on the right demonstrates when the Alarm option is selected to be reminded of a task. Here, the user can select to stop the alarm

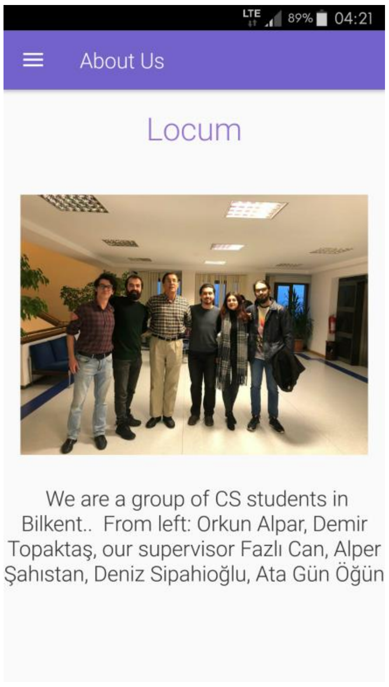
## 9.13. Edit Profile Screen



E-mail
deniz@sipahioglu.org
Username
denis
First Name
Deniz
Last Name
Sipahioglu
Change Password
Password
Password(again)

This screen will allow the user to check his profile information and also give him the ability to change his name, last name and password. The username and e-mail cannot be changed.

## 9.14. About Us Screen

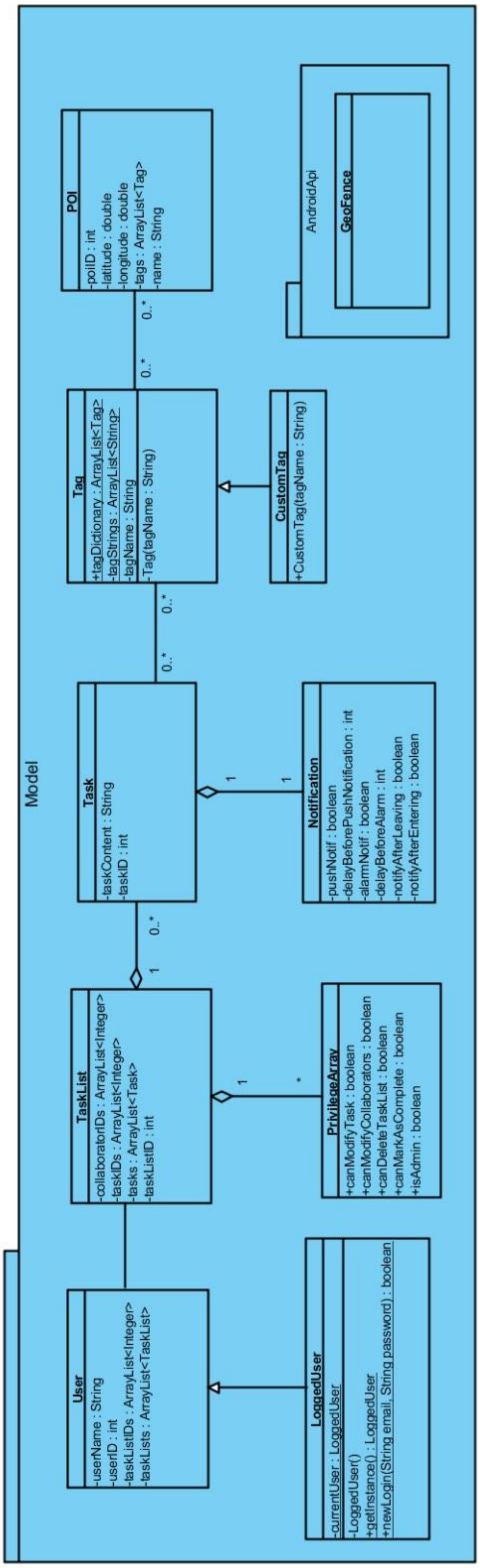


We are a group of CS students in Bilkent.. From left: Orkun Alpar, Demir Topaktaş, our supervisor Fazlı Can, Alper Şahıstan, Deniz Sipahioğlu, Ata Gün Öğün

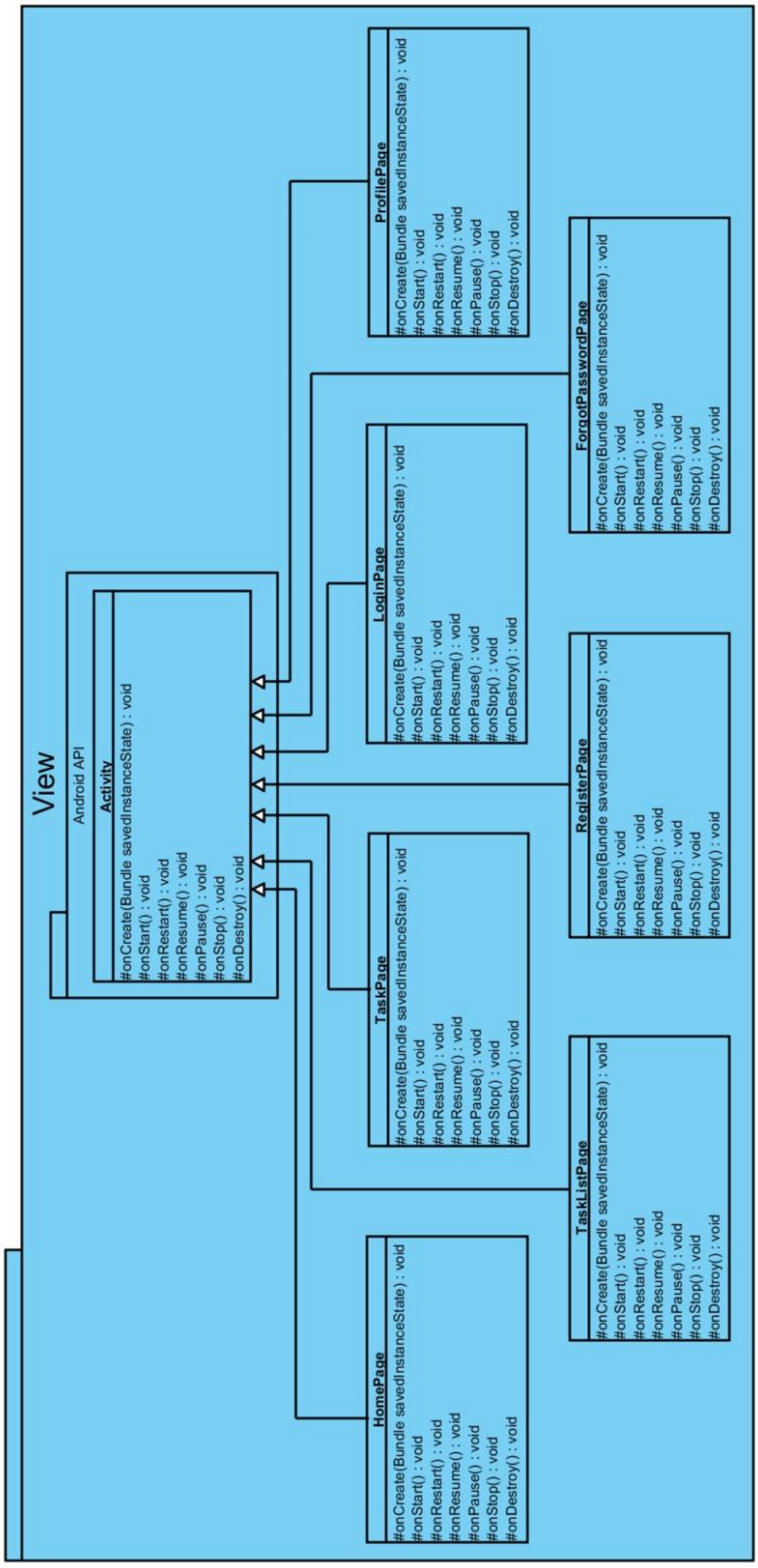
This screen has information on the creators of this app (us) and gives a brief explanation of who they are.

# 10 Appendix B – Class Diagrams

## Model

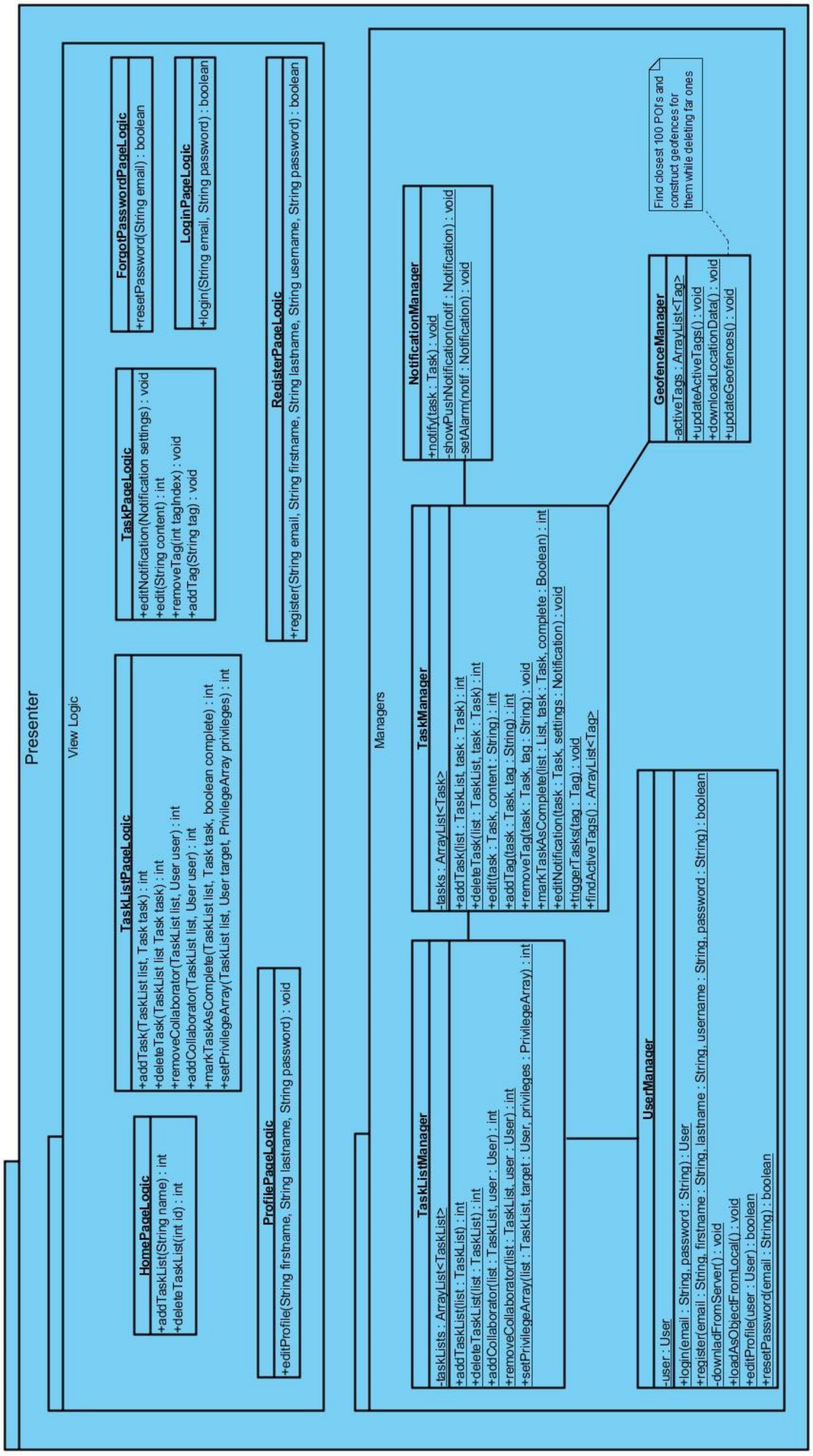

Model
POI
-poiID : int
-latitude : double
-longitude : double
-tags : ArrayList<Tag>
-name : String
0..*
0..*
Tag
+tagDictionary : ArrayList<Tag>
-tagStrings : ArrayList<String>
-tagName : String
-Tag(tagName : String)
CustomTag
+CustomTag(tagName : String)
AndroidApi
GeoFence
0..*
0..*
Task
-taskContent : String
-taskID : int
1
1
Notification
-pushNotif : boolean
-delayBeforePushNotification : int
-alarmNotif : boolean
-delayBeforeAlarm : int
-notifyAfterLeaving : boolean
-notifyAfterEntering : boolean
1
0..*
TaskList
-collaboratorIDs : ArrayList<Integer>
-taskIDs : ArrayList<Integer>
-tasks : ArrayList<Task>
-taskListID : int
1
*
PrivilegeArray
+canModifyTask : boolean
+canModifyCollaborators : boolean
+canDeleteTaskList : boolean
+canMarkAsComplete : boolean
+isAdmin : boolean
User
-userName : String
-userID : int
-taskListIDs : ArrayList<Integer>
-taskLists : ArrayList<TaskList>
LoggedUser
-currentUser : LoggedUser
-LoggedUser()
+getInstance() : LoggedUser
+newLogin(String email, String password) : boolean

**View**



View
Android API
Activity
#onCreate(Bundle savedInstanceState) : void
#onStart() : void
#onRestart() : void
#onResume() : void
#onPause() : void
#onStop() : void
#onDestroy() : void
ProfilePage
#onCreate(Bundle savedInstanceState) : void
#onStart() : void
#onRestart() : void
#onResume() : void
#onPause() : void
#onStop() : void
#onDestroy() : void
LoginPage
#onCreate(Bundle savedInstanceState) : void
#onStart() : void
#onRestart() : void
#onResume() : void
#onPause() : void
#onStop() : void
#onDestroy() : void
TaskPage
#onCreate(Bundle savedInstanceState) : void
#onStart() : void
#onRestart() : void
#onResume() : void
#onPause() : void
#onStop() : void
#onDestroy() : void
HomePage
#onCreate(Bundle savedInstanceState) : void
#onStart() : void
#onRestart() : void
#onResume() : void
#onPause() : void
#onStop() : void
#onDestroy() : void
ForgotPasswordPage
#onCreate(Bundle savedInstanceState) : void
#onStart() : void
#onRestart() : void
#onResume() : void
#onPause() : void
#onStop() : void
#onDestroy() : void
RegisterPage
#onCreate(Bundle savedInstanceState) : void
#onStart() : void
#onRestart() : void
#onResume() : void
#onPause() : void
#onStop() : void
#onDestroy() : void
TaskListPage
#onCreate(Bundle savedInstanceState) : void
#onStart() : void
#onRestart() : void
#onResume() : void
#onPause() : void
#onStop() : void
#onDestroy() : void

**Presenter**


Presenter
View Logic
HomePageLogic
+addTaskList(String name) : int
+deleteTaskList(int id) : int
ProfilePageLogic
+editProfile(String firstname, String lastname, String password) : void
TaskListPageLogic
+addTask(TaskList list, Task task) : int
+deleteTask(TaskList list Task task) : int
+removeCollaborator(TaskList list, User user) : int
+addCollaborator(TaskList list, User user) : int
+markTaskAsComplete(TaskList list, Task task, boolean complete) : int
+setPrivilegeArray(TaskList list, User target, PrivilegeArray privileges) : int
TaskPageLogic
+editNotification(Notification settings) : void
+edit(String content) : int
+removeTag(int tagIndex) : void
+addTag(String tag) : void
RegisterPageLogic
+register(String email, String firstname, String lastname, String username, String password) : boolean
ForgotPasswordPageLogic
+resetPassword(String email) : boolean
LoginPageLogic
+login(String email, String password) : boolean
Managers
TaskListManager
-taskLists : ArrayList<TaskList>
+addTaskList(list : TaskList) : int
+deleteTaskList(list : TaskList) : int
+addCollaborator(list : TaskList, user : User) : int
+removeCollaborator(list : TaskList, user : User) : int
+setPrivilegeArray(list : TaskList, target : User, privileges : PrivilegeArray) : int
TaskManager
-tasks : ArrayList<Task>
+addTask(list : TaskList, task : Task) : int
+deleteTask(list : TaskList, task : Task) : int
+edit(task : Task, content : String) : int
+addTag(task : Task, tag : String) : int
+removeTag(task : Task, tag : String) : void
+markTaskAsComplete(list : List, task : Task, complete : Boolean) : int
+editNotification(task : Task, settings : Notification) : void
+triggerTasks(tag : Tag) : void
+findActiveTags() : ArrayList<Tag>
NotificationManager
+notify(task : Task) : void
-showPushNotification(notif : Notification) : void
-setAlarm(notif : Notification) : void
GeofenceManager
-activeTags : ArrayList<Tag>
+updateActiveTags() : void
+downloadLocationData() : void
+updateGeofences() : void
Find closest 100 POI's and construct geofences for them while deleting far ones
UserManager
-user : User
+login(email : String, password : String) : User
+register(email : String, firstname : String, lastname : String, username : String, password : String) : boolean
-downladFromServer() : void
+loadAsObjectFromLocal() : void
+editProfile(user : User) : boolean
+resetPassword(email : String) : boolean

# 11 Glossary

In this part, some terms found in the report will be explained briefly. Then the development strategy and methodology and the server and deployment technologies will be demonstrated.

Collaboration/Group list: The lists that belong to a group of people rather than a single individual Geofencing: A virtual geographic boundary that triggers a response with the movement of the user
Items: The content to be obtained in order to accomplish a task
Notification: Alerting the user when a point of interest is triggered
Points of Interest: A location which is possibly related to a task
Reminder data: The location details and content of tasks
Task: A piece of work that the user wants to be reminded of

## 11.1. Development Strategy and Methodology

One of the most simple and intuitive ways to develop software is the incremental approach. Incremental development allows for developers to continually hone, refine and polish their product owing to its flexible nature and hybrid composition of ideas coming from waterfall and iterative approaches. When a satisfactory amount of goals are met, the project is deployed and user feedback is collected. This feedback is taken into account with the long term goals of the project and is reflected on the next increment of the project. Functionalities added to the product increase ease of use and over time can help developers anticipate user requests and act proactively. The app is released in a base form and then incremented using updates and fixes. In these incrementions, new functionalities are added while keeping the overall experience the same.

This strategy requires good planning and design and well defined module interfaces. Thankfully, our team was very good at sticking to the the plan and attending to all meetings we planned. Everyone devoted an astonishing amount of time to this project. This development strategy allowed us to create an early prototype of the Locum application. After testing this prototype we added other functionality depending on their priority and the dependence of other features.

## 11.2. Server and Deployment Technology

We have two servers. One is written in Nodejs and it is deployed on Microsoft Azure. For  deploying to Azure, we use webhooks from Github repository to Azure. Azure has a well monitoring feature so that we can check whether our server is running healthy or checking response times. Other server is deployed to Amazon Ubuntu cloud machine and runs on Flask. Amazon also offers good monitoring so that our machine is working reliable. Moreover, we deploy our Flask server using Moba xTerm.

## 12 References

Locum: Latin for "Location"

"About IOS 11 Updates." *IPhone Service Pricing - Apple Support*, 9 July 2018, https://support.apple.com/en-us/HT208067. Date Accessed: 2 October 2018.

"Activity | Android Developers." Android Developers, 23 January 2019, developer.android.com/reference/android/app/Activity#ActivityLifecycle. Date Accessed: 18 February 2019.

"Adobe Illustrator." Buy Adobe Illustrator | Vector Graphic Design Software, www.adobe.com/products/illustrator.html. Date Accessed: 8 May 2018.

"Adobe Photoshop." Buy Adobe Photoshop | Best Photo, Image, and Design Editing Software, www.adobe.com/products/photoshop.html.

Ahmet A. A. "Zemberek-NLP." GitHub, 17 Apr. 2019, github.com/ahmetaa/zemberek-nlp. Date Accessed: 8 May 2018.

"Amazon Web Services (AWS) - Cloud Computing Services." Amazon, Amazon, aws.amazon.com/. Date Accessed: 8 May 2018.

"Amazon Relational Database Service (RDS) – AWS." Amazon, Amazon, aws.amazon.com/rds/. Date Accessed: 8 May 2018.

"API Development Environment," Postman, https://www.getpostman.com/. Date Accessed: 8 May 2018.

"API Reference | Android Developers." *Android Developers*, 24 July 2018, https://developer.android.com/reference/. Date Accessed: 1 October 2018.

"Best Git Client 2019 - Features | GitKraken." GitKraken.com, www.gitkraken.com/git-client#integrations. Date Accessed: 8 May 2018.

"Build Software Better, Together." GitHub, github.com/. Date Accessed: 8 May 2018.

Ceta, Noel. "All You Need to Know About UML Diagrams: Types and 5 Examples." Tallyfy, Tallyfy, 14 Sept. 2018, tallyfy.com/uml-diagram/. Date Accessed: 8 May 2018.

"Code of Ethics." *Code of Ethics | National Society of Professional Engineers*, July 2018, www.nspe.org/resources/ethics/code-ethics. Date Accessed: 14 October 2018.

"Create and Monitor Geofences | Android Developers." *AndroidDevelopers*, 8 November 2018, https://developer.android.com/training/location/geofencing#java. Date Accessed: 15 November 2018.

"Download Android Studio and SDK Tools." Android Developers, developer.android.com/studio. Date Accessed: 8 May 2018.

"Flask." Welcome | Flask (A Python Microframework), flask.pocoo.org/. Date Accessed: 8 May 2018.

Friesen, Jeff. "Getting Started with Android Library Projects, Part 1." SitePoint, 19 Aug. 2013, www.sitepoint.com/getting-started-with-android-library-projects-part-1/. Date Accessed: 8 May 2018.

"Geocoding API Usage and Billing." *Google Developers, Google, 1* October 2018, https://developers.google.com/maps/documentation/geocoding/usage-and-billing#pricing-and-billing-changes. Date Accessed: 14 October 2018.

"Geofencing API | Google Developers." *Google Developers, Google,* developers.google.com/location-context/geofencing. Date Accessed: 25 December 2018.

Google Maps, Google, maps.google.com/. Date Accessed: 8 May 2018.

"Google Play Services - Apps on Google Play." Google, Google, play.google.com/store/apps/details?id=com.google.android.gms&hl=en. Date Accessed: 8 May 2018.

Jaggavarapu, Manoj. "Presentation Patterns : MVC, MVP, PM, MVVM." *Manoj Jaggavarapu WordPress*, 2 May 2012, https://manojjaggavarapu.wordpress.com/2012/05/02/presentation-patterns-mvc-mvp-pm-mvvm/. Date Accessed: 25 December 2018.

"Meet Google Keep – Save Your Thoughts, Wherever You Are." *Google Search, Google*, www.google.com/keep/. Date Accessed: 1 October 2018.

"Microsoft Azure Cloud Computing Platform & Services." Microsoft Azure Cloud Computing Platform & Services, azure.microsoft.com/. Date Accessed: 8 May 2018.

Mobatek. "MobaXterm Free Xserver and Tabbed SSH Client for Windows." Free Xserver and Tabbed SSH Client for Windows, mobaxterm.mobatek.net/. Date Accessed: 8 May 2018.

"MySQL." The World's Most Popular Open Source Database, www.mysql.com/. Date Accessed: 8 May 2018.

"Natural Language Toolkit." Natural Language Toolkit - NLTK 3.4.1 Documentation, www.nltk.org/. Date Accessed: 8 May 2018.

"Nearby Places for Android | ArcGIS for Developers." *System Requirements | ArcGIS API for JavaScript 4.8*, https://developers.arcgis.com/example-apps/nearby-android/. Date Accessed: 1 October 2018.

"Node.js Web Application Framework." Express, expressjs.com/. Date Accessed: 8 May 2018.

"Overview | Places API | Google Developers." Google, Google, developers.google.com/places/web-service/intro. Date Accessed: 8 May 2018.

"Principle of Least Privilege." Wikipedia, Wikimedia Foundation, 26 May 2018, https://en.wikipedia.org/wiki/Principle_of_least_privilege. Date Accessed: 14 October 2018.

"Room Persistence Library | Android Developers," Android Developers. [Online]. Available: https://developer.android.com/topic/libraries/architecture/room. Date Accessed: 8 May 2018.

Rouse, Margaret, and Matthew Haughn. "What Is Constraint (Project Constraint)? - Definition from WhatIs.com." *WhatIs.com*, March 2015, https://whatis.techtarget.com/definition/constraint-project-constraint. Date Accessed: 14 October 2018.

SQLite Home Page, www.sqlite.org/index.html. Date Accessed: 8 May 2018.

“UML 2 Class Diagramming Guidelines.” Agile Design, agilemodeling.com/style/classDiagram.htm. Date Accessed: 18 February 2019.

“Visual Paradigm.” Ideal Modeling & Diagramming Tool for Agile Team Collaboration, www.visual-paradigm.com/. Date Accessed: 8 May 2018.

“Word2vec.” Wikiwand, www.wikiwand.com/en/Word2vec. Date Accessed: 8 May 2018.

“Your Ultimate MLA Format Guide.” MLA Format and MLA Citations - Your BibMe Guide to MLA Citing, www.bibme.org/mla. Date Accessed: 8 May 2018.