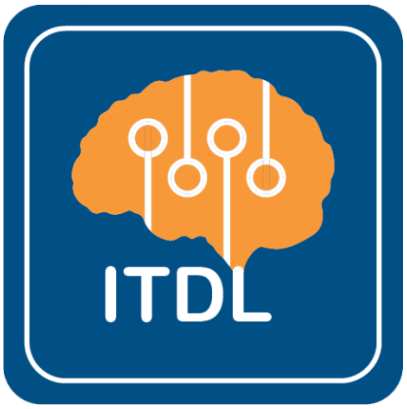
Cairo University
Faculty of Computers and Information
Computer Science Department

Graduation project 2015/2016



Intelligent To-Do List

Dr. Abeer El-Korani

ID	Name	Dept
20120080	Esraa Salem Ahmed	CS
20120456	Yasmin Abdel Latif Mohammad	CS
20120196	Samah Khaled Abd El-Hakem	CS
20120211	Shimaa Mohammed Hassan	CS
20120455	Yasmin Ehab Ahmed	CS

Contents

list of figures

list of tables

Acknowledgement

Apart from the efforts of us, the success of any project depends largely on the encouragement and guidelines of many others. We take this opportunity to express our gratitude to the people who have been instrumental in the preparation of this project.

We would like to show our greatest appreciation to our Supervisor **Dr. Abeer El-Korani**. We can't say thank you enough for her support and help.

At last, we would like to thank our **Eng. Eman Negm and Eng. Laila Mostafa** for their encouragement and guidance.

Chapter 1: Introduction

1.1 Introduction

In our daily life, there are many tasks we have to do and goals we want to achieve in seek of make progress in our lives. Because of we are overwhelmed with tasks, sometimes we feel that we need more than 24 hours to accomplish all our tasks. We may forget some of them even if the context (location and time) is accessible to finish a certain task or wasting time is lower priority tasks all of this take us off track. Another concern is that when you try to create a new habit or to improve yourself this need to do the same task continually without interruption until it turns to a real habit. You need something to remind you with your goals and tasks, help you manage your time and prioritizing your tasks. And this is the goal and importance of our project which will be your To Do list organizer but in intelligent way as we will discuss in coming chapters.

1.2 Motivation

Our application will help the users to manage their daily tasks in order to

- Boost their productivity
- Save time
- Organizing and prioritizing their tasks

1.3 Problem definition

In this project we use social media and recommendation technologies to provide an intelligent To-Do List application. The application is designed to record the user daily activities and use them, in addition to the user's Twitter posts, to learn the user preferences. Beyond the normal functionalities of a To-Do list application, like reminding users with their scheduled activities, our application will also try to recommend other activities to the user according to their location and preferences.

We have three main components in our system:

- **Notification component:**

  We specify 4 types of notes (Ordinary note – Meeting note – Deadline note – Shopping notes)

  In this component, there are the main operations of the To-Do list like add, update, delete and show notes. Each type of notes will be handled in different way and all of them will cooperate in the user behavior analysis and know the user preferences and interests which will be used in personalizing activities to the user.

  Notifications are based on four parts

  1) Shopping notes

     The user will be notified when there is a near store that sells the products that the user wrote them in the shopping notes.

  2) Meeting notes

     The user will be notified before the meeting with a specific duration. And user can view the meeting agenda.

  3) Deadlines notes

     The user will be notified from time to time with progress bar to track her/his achievement at a specific task.

  4) Ordinary notes

     This type of notes is used to detect the user habits (periodic actions)

- **Recommendation component:**

  This component is interested more about understanding the user and giving the user recommendations for offers, places or some actions s/he might be interested in.

  In this part we will use social media contents from Facebook or Twitter in addition to the user's notes in knowing the preferences and its ratio. After analyzing these inputs, we will use a ranking algorithm to output a number that represent the percentage of the user interest of specific category.

  According to these ratios we will recommend the user some offers and stores that he might interested in.

- **Offers website**

  This website helps stores owners to creates account for their stores and add their offers. They will specify the offers content, offers categories and periods, store location on google maps. We will use this part to recommend the To-Do list app user of some offers that he may interest and match his preferences. These offers will be displayed to the user according to his preferences order.

1.4 Project Objective (suggested solution)

- Analyzing and categorizing the user notes to get needed information to understand the user interests and preferences.
- Being able to recommend things to the customer that can please him.
- Help the user to manage his time and perform the tasks that have the greatest impact of his/her progress.

1.5 Suggested Solution

Intelligent-To-Do-List android application aims to helps user organize his tasks and responsibilities by making use of the nature language processing operation like text analysis, text categorization to enhance the application and connecting to twitter and facebook to learn the user behavior better. So, user enters his notes. According to the note type system will react in specific way. In shopping note, system will remind user of the product that he wants to buy. with help of store offers component system personalized these offers according to the user's interests and preferences for example if user is interested in reading and there are offers in some bookstores system will tells user about these offers. In Meeting notes, user will be notified before the meeting time by at least one day before. So user can prepare the meeting agenda. In deadline notes, system will encourage user to accomplish his tasks by displaying a progress bar represents the user

progress percentage. In ordinary notes, system will learn more about user behaviors in order to predict user's periodic actions.

- The system is divided to two parts first "Store offers" as a web application part which helps to process the second part "To-Do-List" as an Android application part.

1. **Website part "Store Offers":**

   this part is responsible for fill the database with shopping offers where the store owner will create an account for his store, enters its location, enters the store category ex: sports, food, clothes, electronics...etc. and adds offers. The second part will personalize these offers according to the user preferences.

2. **Android part "Intelligent To-Do-List" abstract view:**

   The goal of this application is that user can record his notes, which are 4 types (deadline notes, meeting notes, shopping notes, ordinary notes)

   This is the core application, consists of two main components.

   1) **Notification engine** processes the first 2 type of notes represented in

      a. **meeting notification:**

         This notification reminds the user before the meeting to remember the appointment and meeting agenda.

      b. **deadline notifications:**

         This notification reminds the user about his tasks which has a specific deadline. System can track the user's productivity where user can specify the progress percentage of his task and adjust it till he finishes it.

   2) **Recommender engine**

      a. **offer recommender engine**

         System learns the user preferences by analyzing user's notes history. According to these preferences system will personalize offers to the user.

      b. **habit recommender engine**

         System will infer the user habits by analyzing user's notes history.

1.6 Report Organization (summary of the rest of the report)

Later at the rest of the document we are going to talk in:

- Chapter 2 about the closest applications to our application (ITDL) and what makes us different.
- Chapter 3 about the project specifications (Functional and Non-functional requirements), then our application's class diagrams (Android part and Web part), then use case diagrams and tables, then the sequence diagrams and at the end of this chapter we are going to represent our test cases and the expected output.
- Chapter 4 about our UI at the Android part and the Web part.
- Chapter 5 about our Conclusion and Suggested Future Work.

1.7 Stakeholders

We have 2 stakeholders:

- Normal Users → Android application users who add notes
- Store Owner → Offer web page who posting store offers

Chapter 2: Related work

Google Keep

An app that lets you set reminders for what you captured in your mind and reminds you at the right place and right time. You can add notes, images, voice memos to Google keep and share them with your family and friends. https://support.google.com/keep/

Todoist

Todoist turns your Android phone into a powerful task manager for personal, professional, and shared projects.

http://todoist.com/android

Wunderlist

Supports timed reminders, recurring to-dos (although its recurring feature is definitely lacking), separate reminders from the due date of the task, notes and additional info associated with your to-dos, shared to-dos with others, multiple categories, and more. You can star important tasks (but that's as close to priority as you'll get) https://www.wunderlist.com/

What makes us different

- **Our application notifies user by messages based on information not only info given by user but also based on information that the application learns from tracking the user behavior from previous notes, tweets and his location.**

Table 1 Difference Between our App and Other Apps

	Intelligent to do list	Other apps
Create and edit notes	True	True
Set reminders	True	True
Share notes	False	True
Access tasks everywhere (IOS - Android - Web)	False (future work)	True
Detect periodic action and recommend it to user to make him keep track with his goals even if he didn't enter it	True	False
Takes user preferences and refining them using user social media content	True	False
Notify user by the nearest stores to do "Shopping notes"	True	False
Recommend user by offers according to his preferences as a part of the shopping notes	True	False
Notify the user about the deadline notes multiple time before the deadline and make him edit his progress rate in order not to forget his deadlines and to work on them	True	false

Chapter 3: System Design

3.1 Project specification

3.1.1 Functional Requirements

Abstract view of our system components

the system is divided to two parts first “Store offers” as a web application part which helps to process the second part “To-Do-List” as an Android application part.

The second part has 2 main components Notes components and Recommendation components

So we organize the system functional requirement to 3 parts:

1- Functional requirements of the Note components:

1. Sign up

   Enter user preferences (after sign up user is asked to enter his initial rating for some defined categories ex: reading, music, health, sports, etc…) these information is required in the recommendation part.

- **Functions that triggered by the user**

2. Login

   2.1 login by Facebook (to help in extracting his posts to be analyzed further)

3. Logout
4. Update profile
5. Add note (meeting, deadline, shopping, ordinary)
6. Update note (meeting, deadline, shopping, ordinary)

   For every type of notes there are different fields inputs ex:in Meeting notes user will enter meeting title, meeting agenda, meeting time. For shopping notes the product that the user wants to buy is required

7. Delete note
8. Note is done

   When user finish certain note it is removed from his current notes and saved to the history notes

9. show current notes

10. show history notes (includes notes which marked as done previously)

- **dynamic notification services**

11. fire notification based on Meeting note: when meeting note is entered we set the android system alarm to fire a notification to user to remind him the meeting by 1 day before it and before the transport estimated time in order user not to be late for his meeting when user click on the notification he will be redirected to the meeting information details.
12. Fire notification based on deadline note: when user enters a deadline note system calculates the remaining period from now to the deadline date and fire a periodic notification to remind the user about his deadline when user click on the notification he will be redirected to android activity asks him to edit his progress percentage.
13. There is a synchronization function that runs in the background its role is to synchronize date between the database hosted on the user mobile and the database server on Google App Engine server which we use to host the system web services that runs complex logic. We use it in case user enters some notes and his mobile is not connected to the internet. To make data consistence to be use in further analysis and processing.

2- Functional requirements of Recommendation engine:

Most of this part functions are related to the system. Functions are not triggered by the user they are running in the background according to time based threads. The main role of this part is that to extract user social network content from twitter or Facebook and his recorded notes to be analyzed to refine his preferences weights which user entered them previously as initial weights. That helps in case of cold user who does not have contents on the social media. Or in the beginning of the user install the app on his mobile and does not have notes. We used a ranking algorithm “Analytical Ranking Processing” which is used to refine the initial ratios of the user preferences weights and choose the most preferred category after sorting the output initial ratios.

1. The ranking methodology:

The ranking algorithm is running as a thread in the background of the main application each week this short period is more preferred to analyze user behavior

Steps for user interests refining:

1. Extract user content (user notes- twitter or Facebook) for only previous week. We use tweets if user does not log in by Facebook. We store the last date the app runs the algorithm and extract the content for only this period from last update to today
2. Use text categorization tool ‘Alchemy API’ as a Natural Language Processing application to know the category which each text (note – tweet – post) belongs to.

   For the project scope we predefine list of categories:

   - Health and Fitness
   - Art and Entertainment
   - Style and Fashion
   - Education
   - Technology and Computing
   - Religion and Spirituality
   - Food and Drink
   - Reading
   - Pets
   - Sports
   - Movies
   - Music
3. we defined a significance weights of each input source ex: if there are 2 statements in sports category and one statement comes from users note and the other comes from user Facebook post we state that the text from user note has more significance rate than the text which comes from Facebook

   so from this step we get the count of each category for each input source for example (tables date are dummy data just for illustration):

a. category frequency of the user content is the first input to the ranking algorithm

Category	User Notes	User FB Posts	User tweets
Sports	3	2	1
Fashion	4	5	0
……	…..	…..	….

b. input source significance is the second input for the ranking algorithm

We set the input source significance like this

	Notes source	Facebook source	Twitter source
significance	50%	25%	25%

c. user initial weights is the third input for the ranking algorithm

As we mentioned previously user has list of initial weights he enters them after sign up, to solve the cold user problem

Category	Initial weights
health and fitness	40%
art and entertainment	10%
style and fashion	50%
education	30%
technology and computing	80%
religion and spirituality	60%
food and drink	15%
Reading	20%
Pets	0
sports	0
movies	40%
music	0%

Then run the algorithm and the output will be similar to the previous table with the new ratios

As we mentioned this service runs weekly it checks first for the new user content for the previous week before it starts working if it works and gets the updated weights service will fire a notification to the user to inform him that

there are updates to your preferences and display it in descending order. And the user has the choice whether to confirm these updates or not. If user accepts to update his preferences, then these rations are updated in the database.

All of the previous work will help in personalizing the offers to the user according his preferences list.

2. **Show preferred offers to the user:**

   We use the output of the ranking algorithm represented in updated weights of the user categories and extract the offers if any then notify the user only by the offers that matches his top 3 preferences. When user clicked on the notification a list of suggested offers details is viewed.

   This functionality is running in the background of the app every day evening to notify the user of the tomorrows offers if any.

3. **Show nearest stores functionality**

   We handle the user shopping notes by notify the user by the nearest stores that sell the product he wants to buy; this is also a function that run in the background. When user clicked on the notification a list nearest stores details are viewed

4. **Show suggested list of actions:**

   By analyzing the user history of notes we suggest some notes that are not in his current to do list that he may interest to do or it is a habit for him that will help the user to keep tracks of his goal in indirect way. A list of suggested notes is viewed in form of check boxes Then he marks the check boxes of the notes that he is interested in to be added to his current To-Do list.

3- Functional requirements of Store Offers part:

Website part "Store Offers":

this part is responsible for fill the database with shopping offers where the store owner will create an account for his store, enters its location, enters the store category ex: sports, food, clothes, electronics...etc. and adds offers. The second part will personalize these offers according to the user preferences. The main functions as follows:

1. sign up
2. sign in
3. update profile
4. sign out
5. add offer
6. update offer
7. delete offer

Expected data:

1. Offer category ex:(Arts and Entertainments, Movies, Music, Food and drinks, Technology, Sports, Health, Religion, Education, Pets and animals, Fashion, Reading).
2. Store name
3. Store address (manager adds the location by google maps)
4. Offer content
5. Duration (start time - end time)

3.1.2 Non-Functional Requirements

- User requirements:
  - User should be able to navigate to system without difficulties, system functions name should be self-explained to the user
  - System support English language
- User interface requirements:
  - The application design should not confuse the user to perform the functionality.

- Colors should be comfortable for the user eye.

- Reusability requirements

  The ability to reuse gives the ability to build larger things from smaller parts, and being able to identify commonalities among those parts. In our application:
  In implementing our application, we consider the use of SOLID principles of software engineering. We divide the system into components and each class has a single responsibility. We use Model-View-Controller design pattern which helps to extend the system and modify it. We use specific naming convention rules across the team that helps understand the code of each of us.

- Compatibility requirements

  The application should be compatible with the android device

- Accessibility requirement:

  User can use the application in both case if it is connected to the internet or not. And system will synchronize the data between local mobile database and server as a service in the background.

3.1.3 Used Technologies

In our project we use MVC design pattern. And follow the design principles to allow us modify and extend the functionality easily. The backend logic and database are hosted on Google app engine and we deploy the system functions as web services. And just invoke these web services from the android front end. This methodology was organized and ease dividing tasks across the team.

- Google App Engine:

  We use it to host backend of the system functions web services.

  Google App Engine (GAE) is a platform as a service (PaaS) cloud computing platform for developing and hosting web applications in Google-managed data centers. App Engine offers automatic scaling for web applications—as the number of requests increases for an application, App Engine automatically allocates more resources for the web application to handle the additional demand. Google App Engine is free up to a certain level of consumed resources. Fees are charged for additional storage

- Alchemy API

  This API performs NLP functionalities like sentiment analysis, text similarity, and entity extraction. We used it in the categorized the text to know what it is represent (sports, fashion, art, health, technology…)

  There are some other APIs like Uclassify, Dandelion API, Aylien API.

- Facebook API:

  Is used in login by Facebook functionality and extracting posts and check ins if the user that we user to analyze and learn the user behavior

- Twitter API

  Is used in extracting user tweets to be analyzed in case user not choose to login by Facebook

- Githup

  That helps in source code management across the team. We use it to make versions of the code and upload the latest worked version after adding some functionalities.

3.2 System Architecture

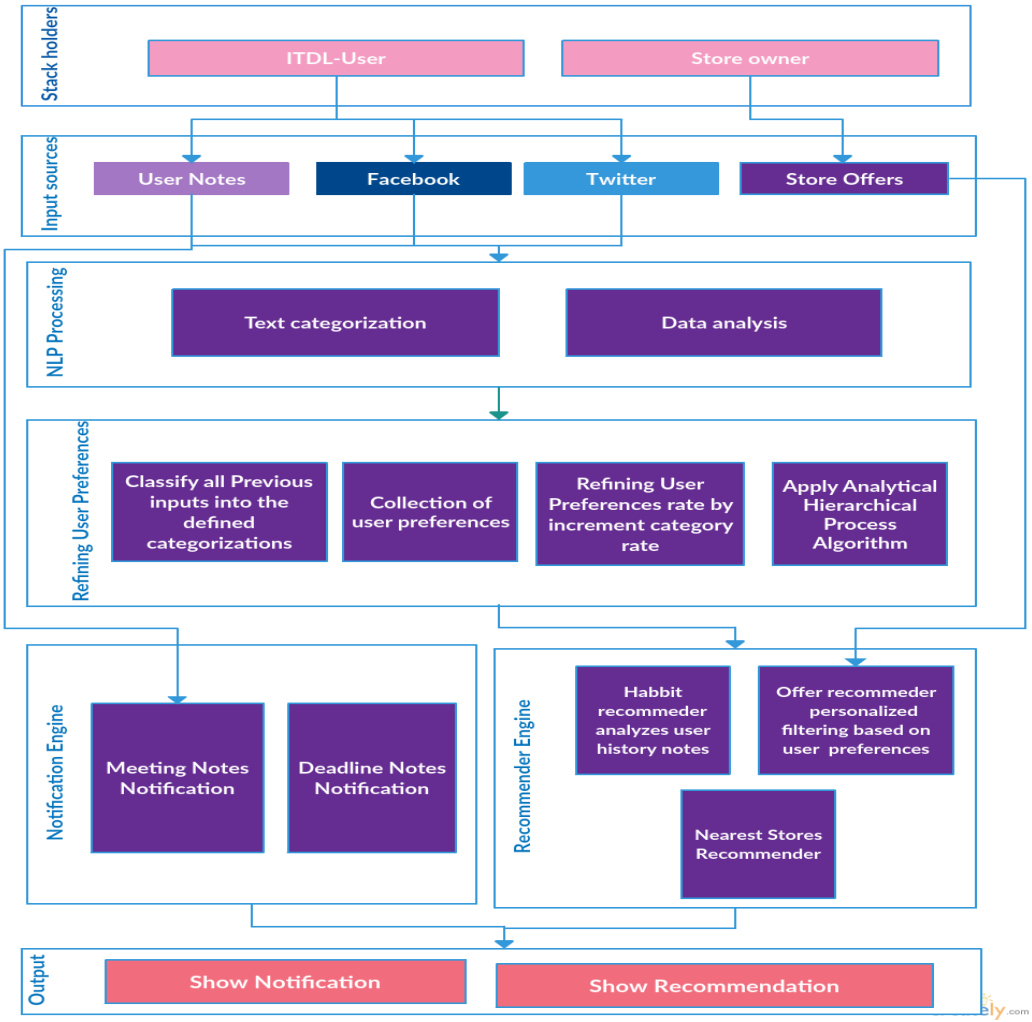


Figure 1 System Architecture

This is the system architecture and workflow of our application. We have 2 main stakeholders Intelligent To-Do List Application (ITDL-User) and the stores owners

We start to collect the application inputs for ITDL-User we get his notes and connect to Facebook and Twitter APIs to extract user tweets, posts and check ins if any.

Next stage is processing these inputs and analyze them by using text categorization and history analysis to help the application learns the user behavior. And refining his preferences to enhance the recommendation engine

Finally, the output of the system in form of notification of the time related notes and a recommendation related to system suggestions based on the information that system learns analyzing user data like suggesting offers, nearest stores and periodic actions.

3.3 Backend Class diagram

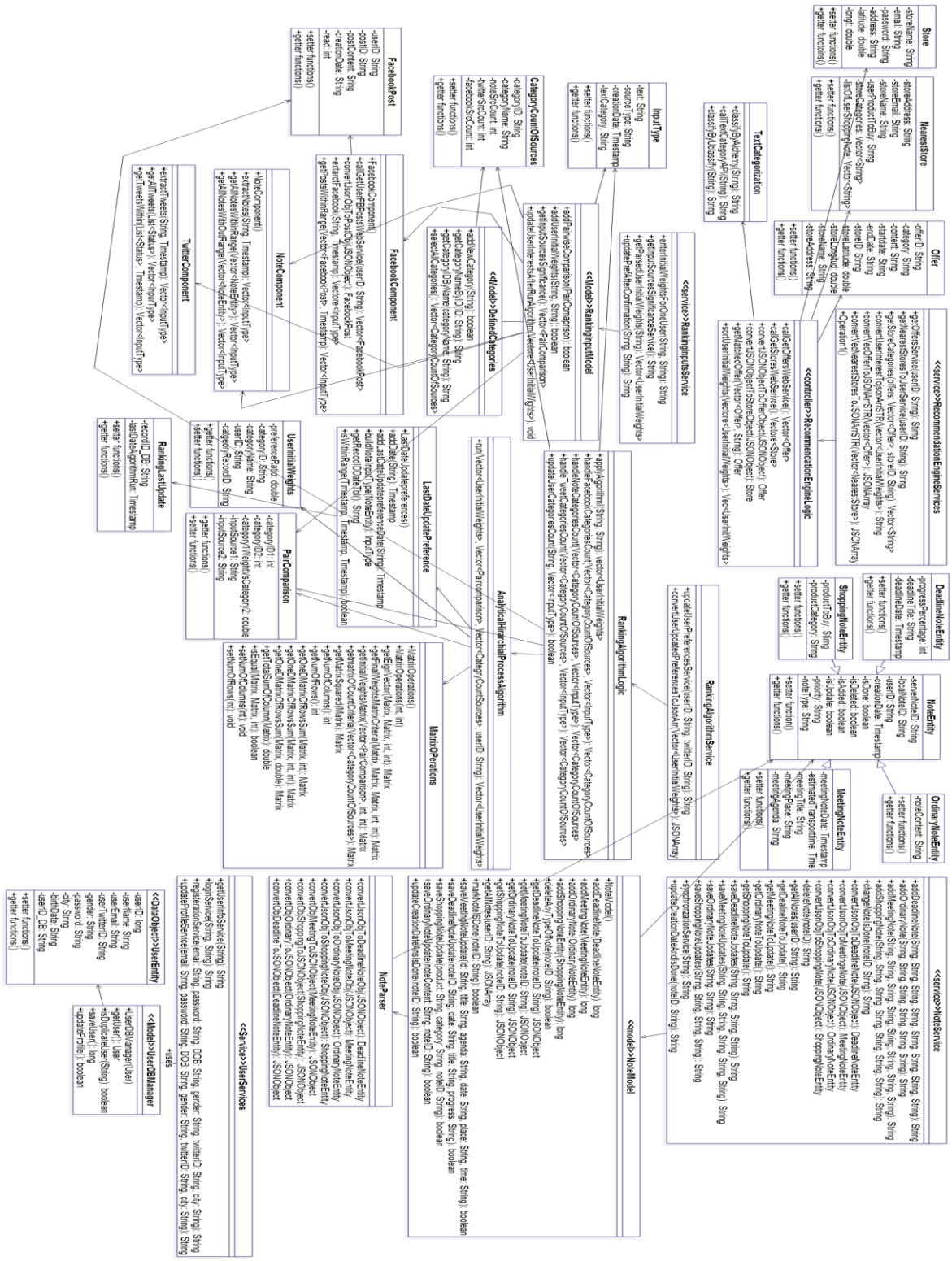


Figure 2 Backend Class Diagram

3.4 Backend Subsystem

3.4.1 User Subsystem

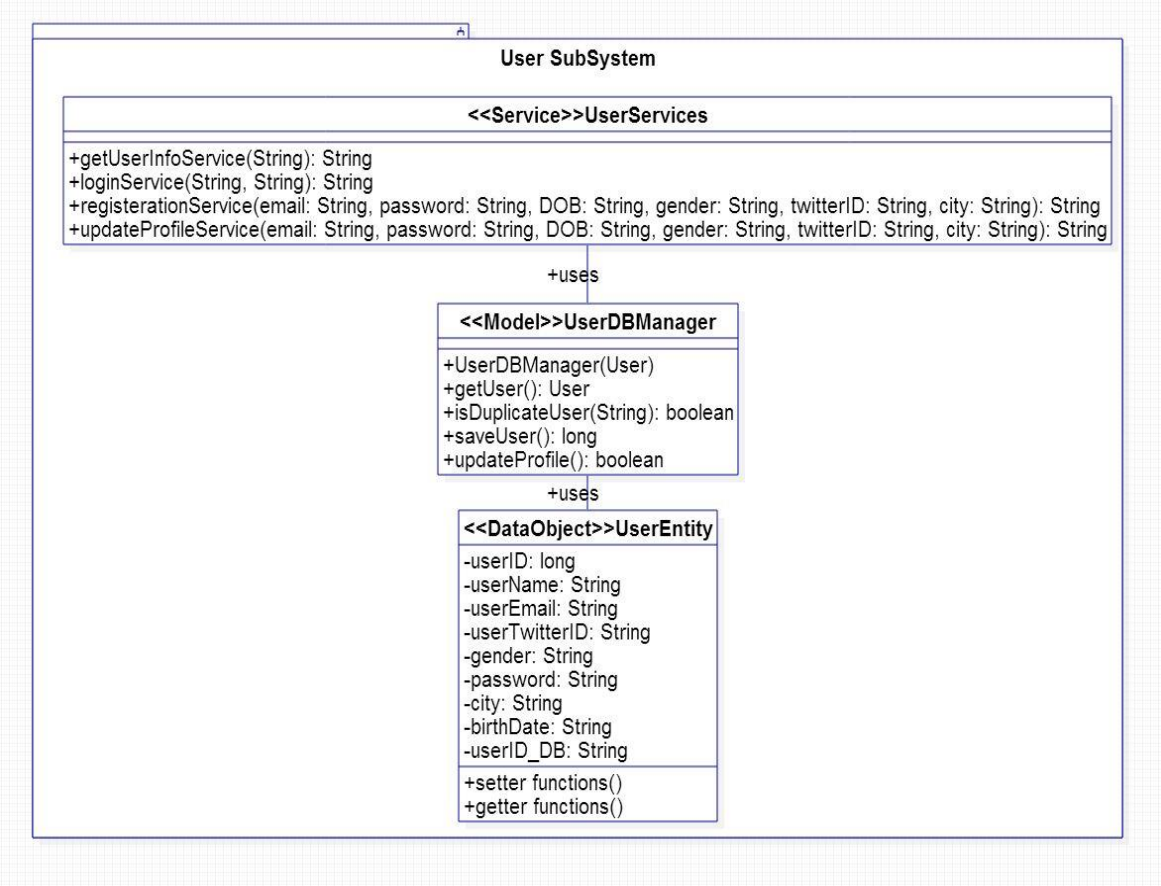


Figure 3 Backend User Subsystem

3.4.2 Note Subsystem

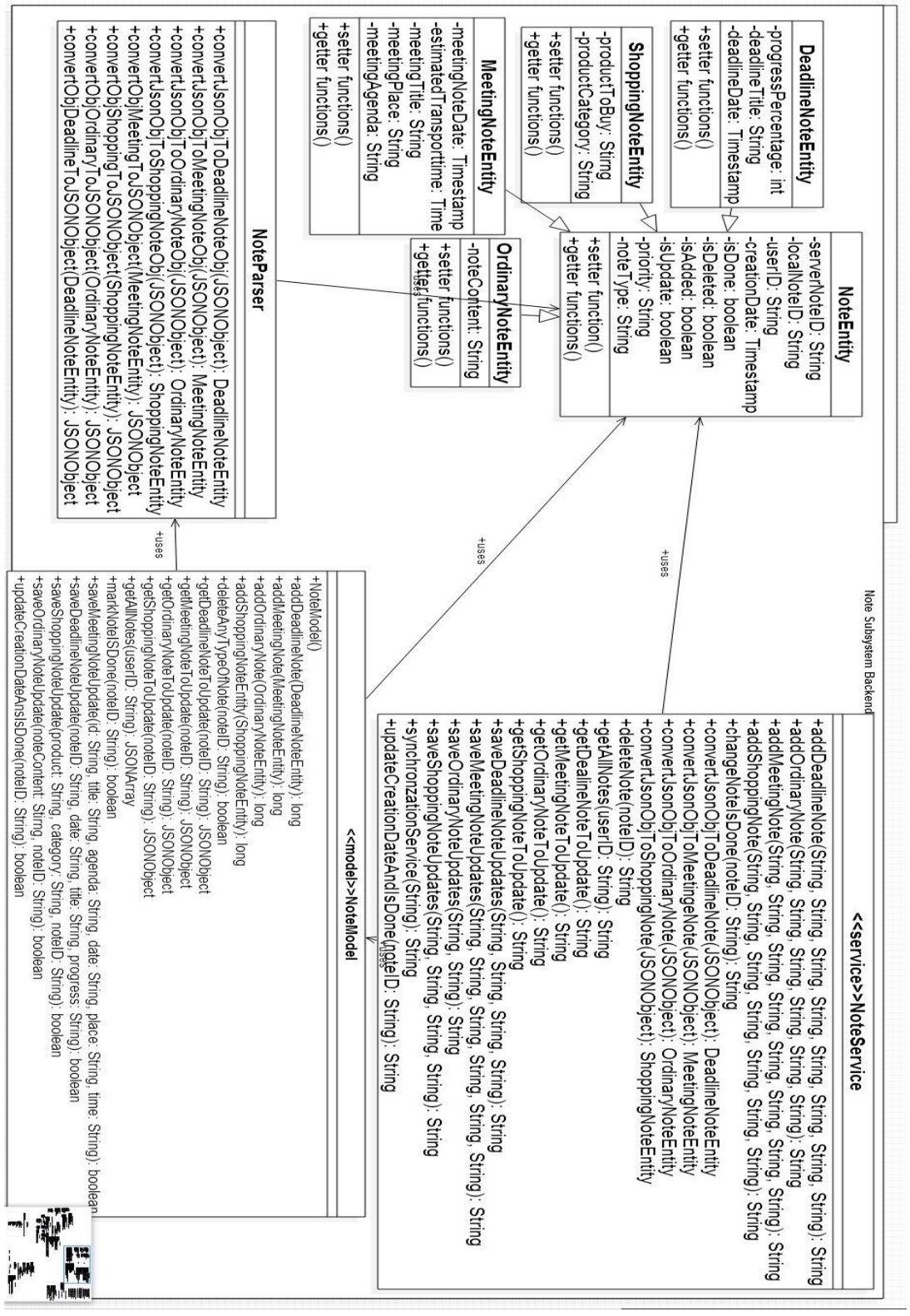


Figure 4 Backend Note Subsystem

3.4.3 Ranking & Update Preferences Subsystem

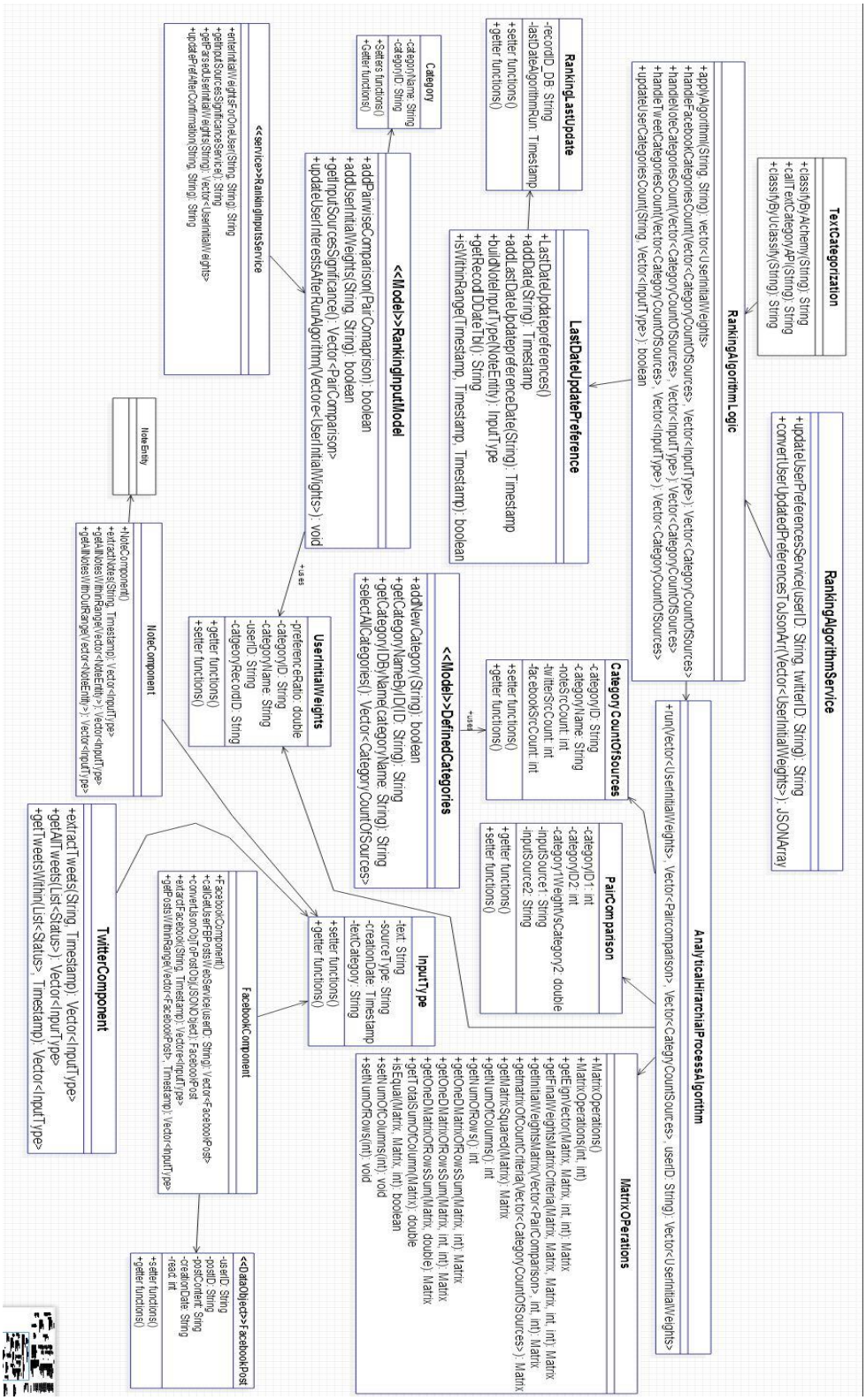


Figure 5 Backend Ranking & update preferences Subsystem

3.4.4 Recommendation Subsystem

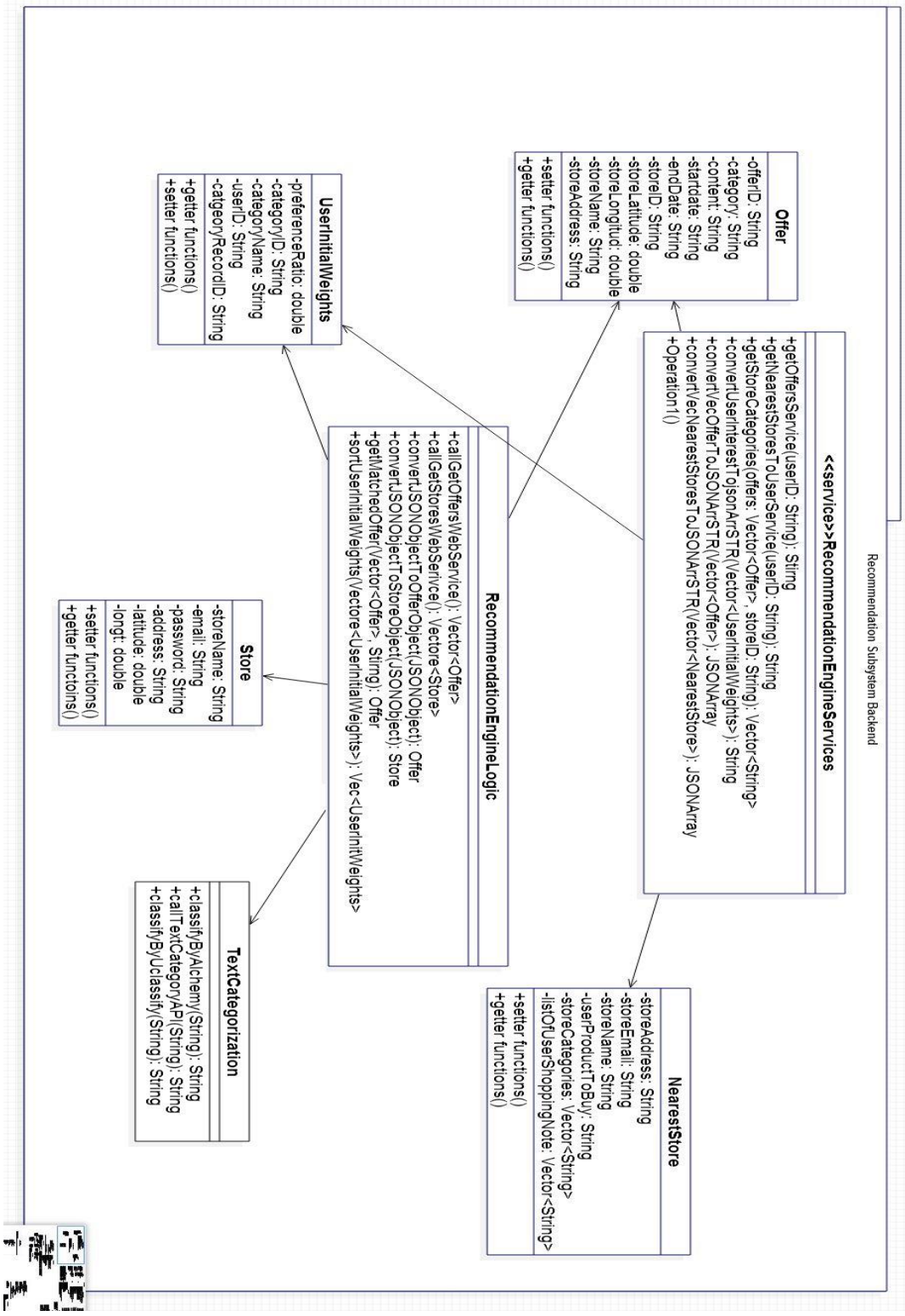


Figure 6 Backend Recommendation Subsystem

3.5 Frontend Class Diagram

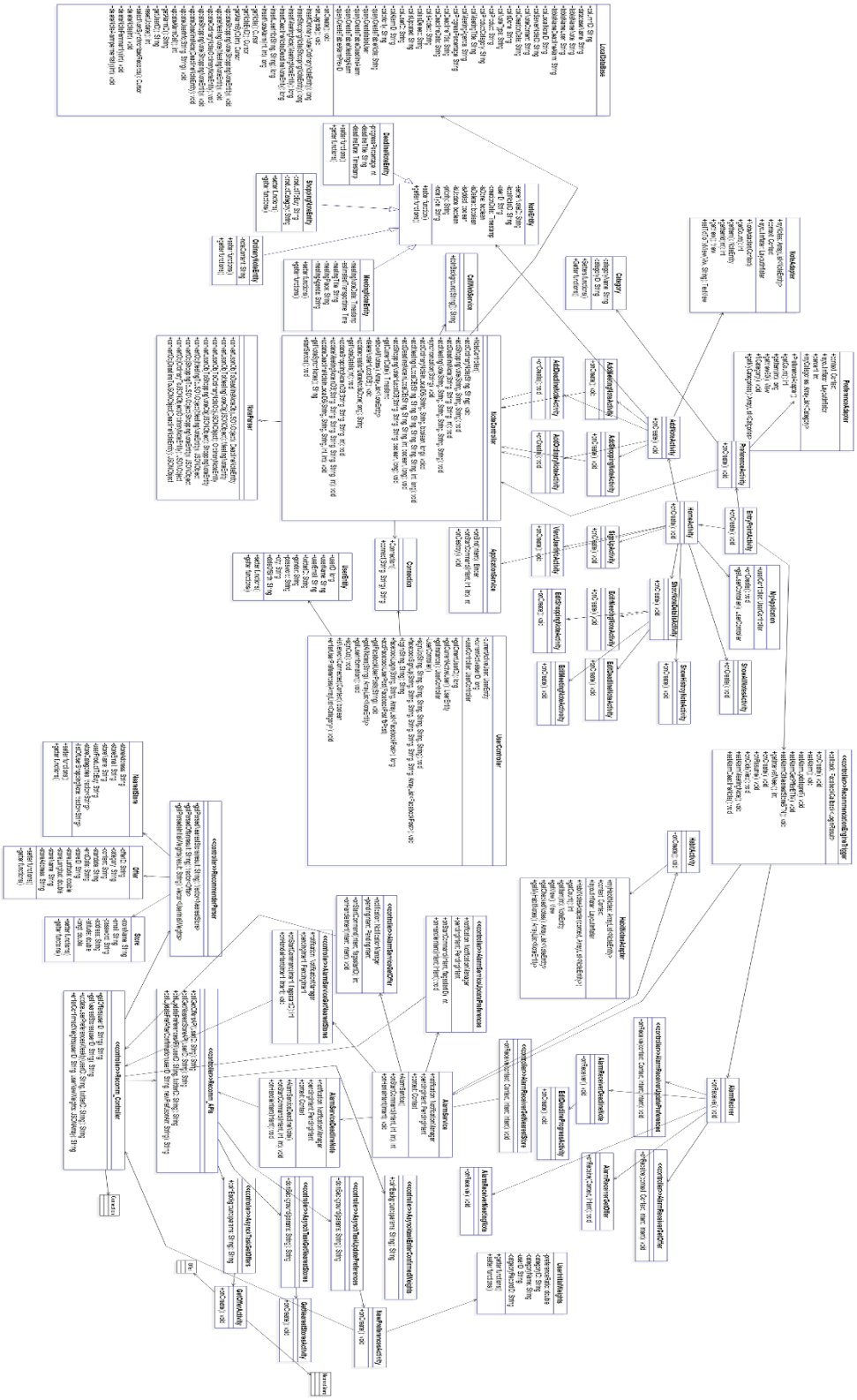


Figure 7 Frontend Class Diagram

3.6 Web Part Class Diagram

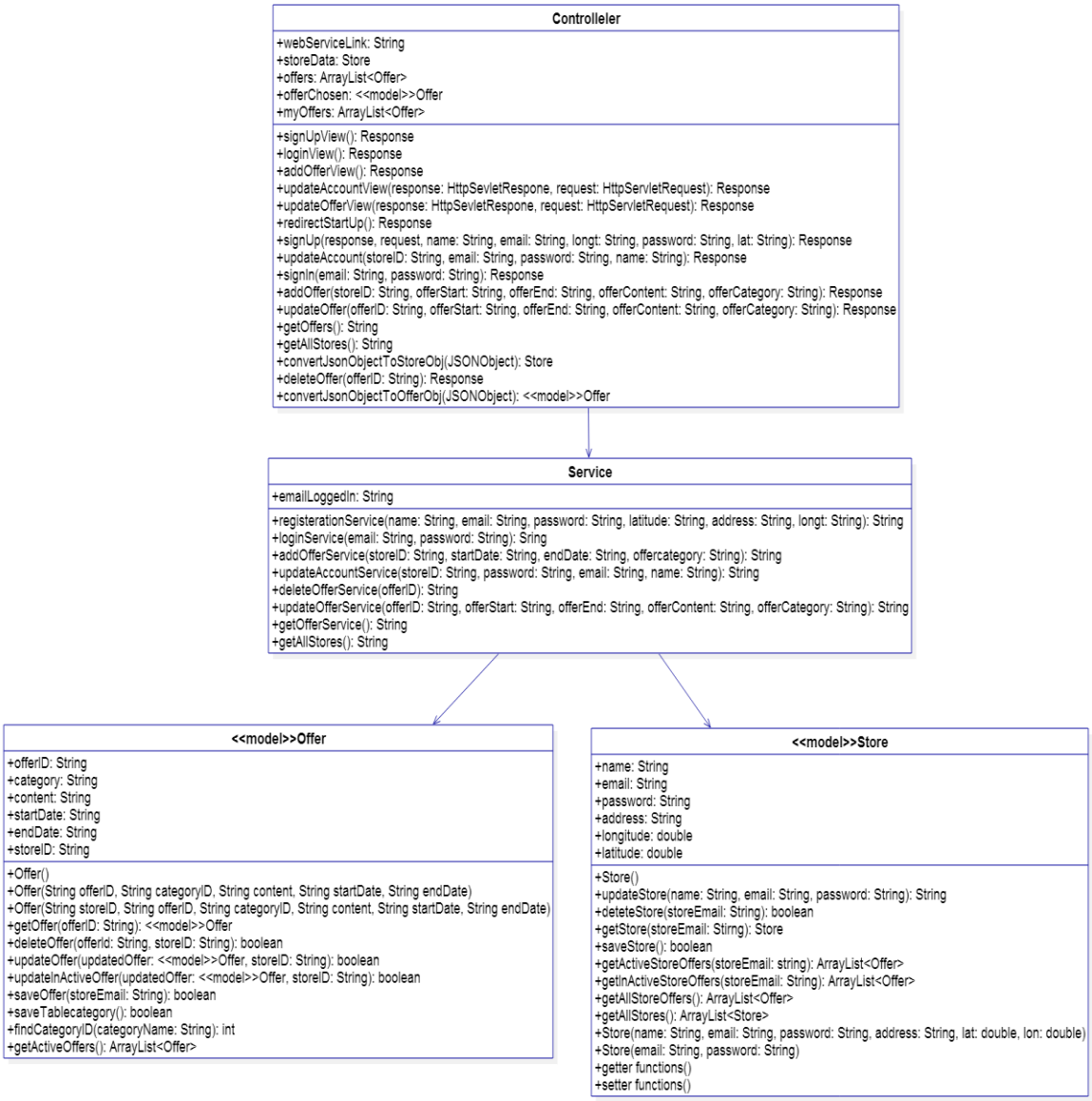


Figure 8 Web Part Class Diagram

3.7 Use Case Diagram

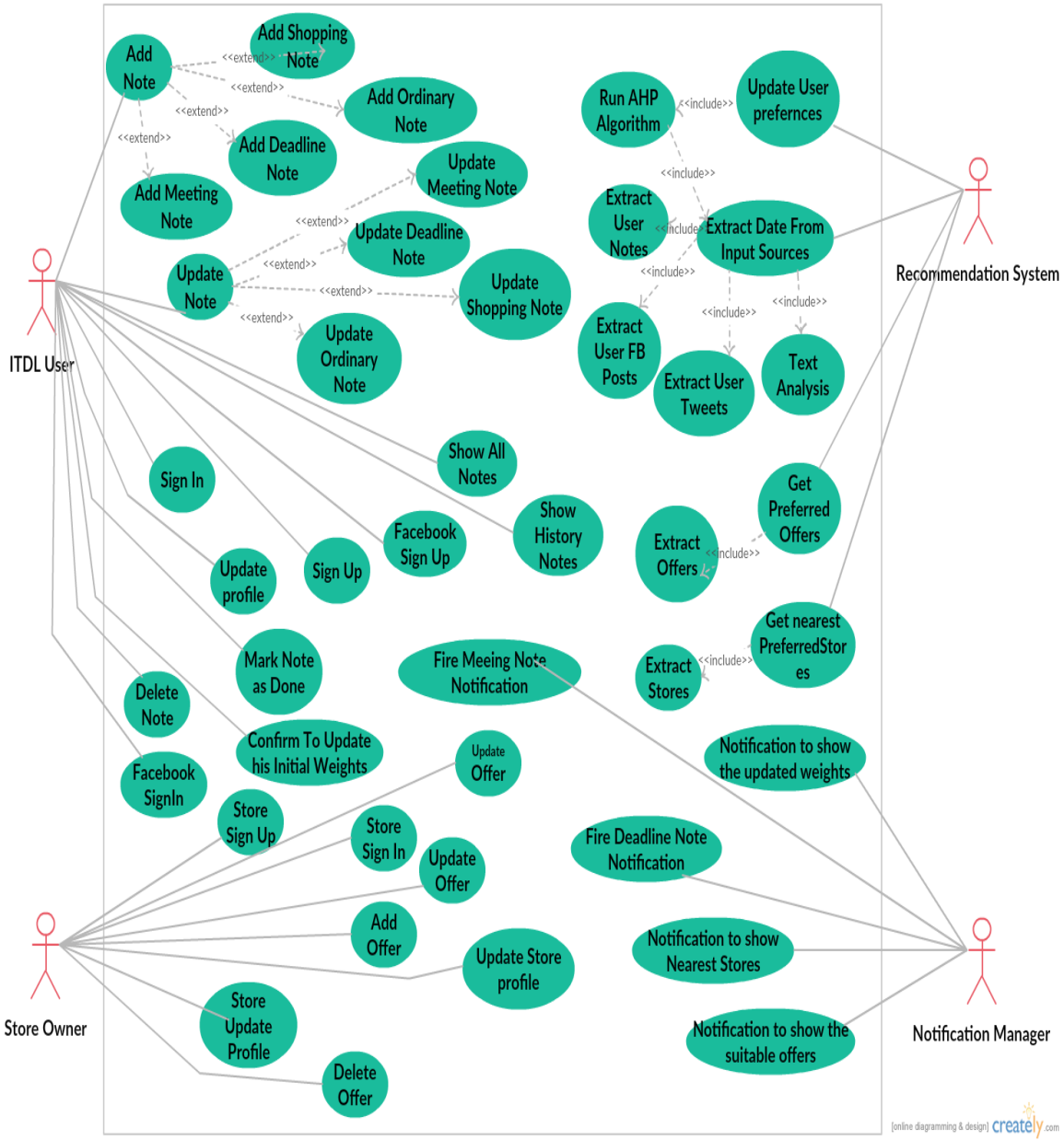


Figure 9 Use Case Diagram

3.8 Use Case Tables

3.8.1 Use case tables Notes Part

Table 2 - Use Case Table - Sign up User

ID	UC-1.1	
Title	Sign up	
Description	The user enters his email, name, city and create a new account on the application	
Actor	ITDL User	
Pre-conditions	User should have an email. And there are no duplicate emails.	
Post-conditions	Users should enter his preferences list after sign up successfully	
Flow of events	**User Action**	**System Action**
	1- User writes his email, name, preferences, location	
		2- System checks the email duplicates in the database. if the entered email is duplicate. Error message will be displayed. Else user account will be created successfully and redirect to enter initial preferences activity.
	3- Users enters his initial preferences.	
		4- System creates an account on the application for the user and saves his info and his preferences to the database
Exceptions	The user enters a wrong formatted email -The system will ask him to write a right formatted one	

Table 3 - Use Case Table - Login User

ID	UC-1.2
Title	Log in
Description	The user logs in to the application
Actor	ITDL User

Pre-conditions	The user already has an account on the application	
Post-conditions	User can use the application and create notes	
Flow of events	**User Action**	**System Action**
	1-User enters his email and password	
		2-System checks if the email and password are in the database
	3-User logs in to the application	
Exceptions	If the user entered a wrong email or wrong password. -System shows an error message to the user and ask him to enter the right information	
Includes	-	

Table 4 - Use Case Table - Facebook Login User

ID	UC-1.2.1	
Title	Facebook Log in	
Description	The user logs in to the application	
Actor	ITDL User	
Pre-conditions	The user already has a Facebook account and logged in on his device. And there is internet connection	
Post-conditions	User can use the application and create notes	
Flow of events	**User Action**	**System Action**
	1- User press login by Facebook	
		2- Facebook API will check the user information if correct. Will redirect him to the application home page. Else error message will be displayed.
	3- User logs in to the application	
Exceptions	If the user does not setup Facebook application on his device. Or not logged in. and if there is no an internet connection which is required to the Facebook API	
Includes	-	

Table 5 - Use Case Table – Add Ordinary Note

ID	UC-1.3	
Title	Add note (Add ordinary note)	
Description	The user adds an ordinary note in his application	
Actor	ITDL User	
Pre-conditions	The user has the application on his mobile, has an account on it and already logged in	
Post-conditions	The note will be saved to the database and the user will be able to update and delete it	
Flow of events	**User Action**	**System Action**
	1- the user chooses to add an ordinary note	
	2- the user writes his note and press save	
		3- if device is not connected to the internet the system will save his note to the local database. And when it connects to the internet the background thread will synchronize the data between local database and the server.
Exceptions	-	
Includes	-	

Table 6 - Use Case Table – Add Shopping Note

ID	UC-1.4
Title	Add note (Add shopping note)
Description	The user can add a shopping note that will help him to remember the things he wants to buy
Actor	ITDL User
Pre-conditions	The user has the application on his mobile, has an account on it and already logged in
Post-conditions	The note will be saved to the database and the user will be able to update and delete it

Flow of events	**User Action**	**System Action**
	1- the user chooses to add a shopping note	
	2- User add the information of the note, what he wants to buy	
		3- if device is not connected to the internet the system will save his note to the local database. And when it connects to the internet the background thread will synchronize the data between local database and the server.
Exceptions	-	
Includes	-	

Table 7 - Use Case Table – Add Meeting Note

ID	UC-1.5	
Title	Add note (Add meeting note)	
Description	The user can add a meeting note that will help him to remember the meeting and its time	
Actor	ITDL User	
Pre-conditions	The user has the application on his mobile, has an account on it and already logged in. and meeting date is after the meeting creation date.	
Post-conditions	The note will be saved to the database and the user will be able to update and delete it	
Flow of events	**User Action**	**System Action**
	1- User choose to add a meeting note	
	2- User add the information of the note, place/time/the person he's going to meet and the topic of the meeting	

		3- if device is not connected to the internet the system will save his note to the local database. And when it connects to the internet the background thread will synchronize the data between local database and the server.
		4- system will set an alarm for the meeting date once note is created. System will fire a notification before the meeting date by one day and before the meeting by the transport estimated time.
		5- system will add these alarms IDs in alarm table with the note ID in order to update these alarms or delete it when note itself is updated and deleted
Exceptions	The user enters the meeting time of the note that's before the current time. System shows an error message and ask the user to set a right time	
Includes	-	

Table 8 - Use Case Table – Add Deadline Note

ID	UC-1.6
Title	Add note (Add deadline note)
Description	The user can add a deadline note that will help him to remember the things he wants to accomplish and gives him a notification when the deadline is about to end
Actor	ITDL User
Pre-conditions	The user has the application on his mobile, has an account on it and already logged in. and the deadline date is after the note creation date.
Post-conditions	The note will be saved to the database and the user will be able to update and delete it

	User Action	System Action
Flow of events	1- User choose to add a deadline note	
	2- User add the information of the note, end time/a small description	
		3- if device is not connected to the internet the system will save his note to the local database. And when it connects to the internet the background thread will synchronize the data between local database and the server.
		4- System will set an alarm for the deadline date once note is created. System will fire a notification when the deadline is about to come
		5- System will add these alarms IDs in alarm table with the note ID in order to update these alarms or delete it when note itself is updated and deleted
Exceptions	The user enters the end time of the note that's before the current time. -System shows an error message and ask the user to set a right time	
Includes	-	

Table 9 - Use Case Table – Delete Note

ID	UC-1.7
Title	Delete note
Description	The user can delete a note he had already made
Actor	ITDL User
Pre-conditions	The user has the application on his mobile, has an account on it and already logged in, and has already made a note on the application
Post-conditions	User can use the rest of the application where he can add note/edit note/ update note/ delete note

	User Action	System Action
Flow of events	1- User click on the notes tab	
		2- System retrieve all the notes in the database
	3- The user choose the note he wants to delete and press the delete button	
		4- the system will mark a flag isDeleted to 1 instead of 0 that will help in the synchronization function which will selects the deleted notes from local DB then delete them from the server in case of internet connection is available then delete it from the local DB permanently.
Exceptions	-	
Includes	-	

Table 10 - Use Case Table – Update Note

ID	UC-1.8	
Title	Update note	
Description	The user can update a note he had already made	
Actor	ITDL User	
Pre-conditions	The user has the application on his mobile, has an account on it and already logged in, and has already made a note on the application	
Post-conditions	User can use the rest of the application where he can add note/edit note/ update note/ delete note	
Flow of events	**User Action**	**System Action**
	1- User click on the notes tab	
		2- System retrieve all the notes in the database

	3- The user chooses the note he wants to update and press the update button	
		4- System retrieve the detailed information of the note from the database and show it to the user
	5- User update the information he wants and press the update button	
		6- System marks the isUpdated flag in local DB to 1 instead of 0 to synchronize it when it is connected to the internet. Else it will update it in both server and local DB.
Exceptions	-	
Includes	-	

Table 11 - Use Case Table – Update User Profile

ID	UC-1.9	
Title	Update profile	
Description	User will be able to update his profile and changes some information in it	
Actor	ITDL User	
Pre-conditions	User should have an account on the application and already logged in	
Post-conditions	User can use the rest of the application where he can add note/edit note/ update note/ delete note	
Flow of events	**User Action**	**System Action**
	1- User choose to update his account	
		2- System retrieve the information of the user from the database and shows it to him

	3- Users may change his email/Username/Password/Preferences	
		4- System changes the updated information of the user in the database
Exceptions	The user enters a wrong password -System won't confirm the changes the user made to his account	
Includes	-	

Table 12 - Use Case Table – Show all current notes

ID	UC-1.10	
Title	Show all current notes	
Description	User can display all his notes	
Actor	ITDL User	
Pre-conditions	User should have an account on the application and already logged in. and have some notes	
Post-conditions	-	
Flow of events	**User Action**	**System Action**
	1- User choose to show his current notes	
		2- System retrieves the notes whose is Done flag is 0
Exceptions	There are no notes	
Includes	-	

Table 13 - Use Case Table – Show history note

ID	UC-1.11	
Title	Show history notes	
Description	User can display notes which pasts	
Actor	ITDL User	
Pre-conditions	User should have an account on the application and already logged in. and have some notes	
Post-conditions	-	
Flow of events	**User Action**	**System Action**

	1- User choose to show his history notes	
		2- System retrieves the notes whose is Done flag is 1
Exceptions	There are no notes	
Includes	-	

Table 14 - Use Case Table – Mark a note as finished

ID	UC-1.12	
Title	Mark a note as finished	
Description	**User mark note as done note**	
Actor	ITDL User	
Pre-conditions	User should have an account on the application and already logged in. And have some notes.	
Post-conditions	-	
Flow of events	**User Action**	**System Action**
	1- User choose to show his current notes	
		2- System retrieves the notes whose is Done flag is 1
	3- user press on Done button	
		4- system edits isDone flag to 1 instead of 0 in local DB and synchronize it with the server.
Exceptions	There are no notes	
Includes	-	

Table 15 - Use Case Table – Show preferred offers

ID	UC-1.13
Title	Show preferred offers
Description	**System will recommend user some offers according to user's preferences list**
Actor	Recommendation Engine

Pre-conditions	There are some offers and user initial weights. There is an internet connection in order to run this web service	
Post-conditions	-	
Flow of events	**User Action**	**System Action**
		1- system extracts some offers from the offers website
		2- system extracts user list of preferences
		3- system selects the most suitable offers for certain user according to his preferences list.
		4- system fires a notification to the user to inform him to see the list of preferred offers.
	5- User clicks on the notification	
		6- a list of offers details is viewed to the user.
Exceptions	-	
Includes	-	

Table 16 - Use Case Table – Show list of nearest stores to user

ID	UC-1.14	
Title	Show list of nearest stores to user	
Description	**List of stores is displayed to the user according to his shopping notes list.**	
Actor	Recommendation Engine	
Pre-conditions	User should have an account on the application and already logged in. And have some Shopping notes. There is an internet connection in order to run this web service	
Post-conditions	-	
Flow of events	**User Action**	**System Action**
		1- system extracts user shopping notes.
		2- System extracts stores that have the same categories of the products that user wants to buy.

		3- system fires a notification to the user to inform him to see the list of some stores that sells some products he may interest in.
	4- User clicks on the notification	
		5- a list of stores details is viewed to the user.
Exceptions	-	
Includes	-	

Table 17 - Use Case Table – Update user preferences

ID	UC-1.15	
Title	Update user preferences	
Description	ITDL user enters a lists of initial preferences at the beginning of using the system. ITDL app tries to learn the system behavior. And refine his preferences list. To enhance the recommendation to the user by analyze his note contents and some of his social media content	
Actor	Recommendation Engine	
Pre-conditions	User should have a content from at least one input source notes ,posts or tweets. There is an internet connection in order to run this web service	
Post-conditions	Update the user preferences list in the database when user confirms these updates	
Flow of events	**User Action**	**System Action**
		1- system extract user notes, posts and tweets if any for previous week only
		2- system analyze the user's data
		3- system runs the Ranking algorithm
		4- system fires a notification to the user to inform him that his preferences were updated
	5- User clicks on the notification and	

	choose yes to update his preferences or no for not updating them	
		6- if user press yes system will update his preferences in the database and save the last date when the algorithm runs else let the previous rates as it is.
Exceptions	-	
Includes	-	

Table 18 - Use Case Table – Habit detection

ID	UC-1.16	
Title	User habits	
Description	Show list of suggested actions that the user may interest to do them. That by analyzing his notes history	
Actor	Recommendation system	
Pre-conditions	There is an internet connection in order to run this web service.	
Post-conditions		
Flow of events	**User action**	**System action**
		1- system extracts the notes of the same current day for the last 2 weeks
		2- system fire notification to user that there is list of suggested actions
	3- user clicks the notification	
		4- system display list of suggested action in form of check boxes
	5- user checks the notes that he may to do it	
		6- system adds the checked notes to the current list of notes. And delete the other unchecked notes from the habit table.

Exceptions	-
Includes	-

3.8.2 Use case tables Web Part

Table 19 - Use Case Table - Signup Store

ID	UC-2.1	
Title	Sign up	
Description	Store owner will create a new account for his store	
Actor	Store owner	
Pre-conditions	Store owner should have a store that sells something and have an email for his store	
Post-conditions	Store owner will have an account for his store	
Flow of events	**User Action**	**System Action**
	1- Store owner writes his email/Store name/Category/Password	
		2- System saves his information to the database
Exceptions	Store owner write an email with wrong format -System asks the owner to enter a right formatted email Store owner writes an email that already exists in the database for another account. -System asks the owner to write a new email.	
Includes	-	

Table 20 - Use Case Table - Login Store

ID	UC-2.2	
Title	Log in	
Description	The store owner logs in to the system so he would be able to perform some functions through it	
Actor	Store owner	
Pre-conditions	The store owner has an account on the system and already logged in	
Post-conditions	The store owner can add/update/delete an offer	
Flow of events	**User Action**	**System Action**

	1- Store owner enters the email and password for the account	
		2- System checks the email and password and verify them
Exceptions	Store owner entered a wrong email or password -System shows an error message and ask the store owner to enter the right information	
Includes	-	

Table 21 - Use Case Table - Update Store Profile

ID	UC-2.3	
Title	Update profile	
Description	The store owner will be able to update his account information	
Actor	Store owner	
Pre-conditions	The store owner already has an account on the system	
Post-conditions	The store owner can use the functions of the system	
Flow of events	**User Action**	**System Action**
	1- Store owner changes the information of the account	
		2- System asks the store owner to enter his password in order to accept his changes
	3- Store owner enters his password	
		4- The system verify the changes and save them to the database
Exceptions	The store owner enters a wrong password -System won't confirm the changes the user made to his account	
Includes	-	

Table 22 - Use Case Table - Add Offer

ID	UC-2.4
Title	Add offer
Description	The store owner adds a new offer that exists in his store

Actor	Store owner	
Pre-conditions	The store owner already has an account on the system and already logged in	
Post-conditions	The store owner offer can be promoted to the users by the application according to the information of the offer	
Flow of events	**User Action**	**System Action**
	1- Store owner add a new offer and adds the product name/category/price/start date/ end date	
		2- System adds the offer to the database with all its information
Exceptions	The store owner adds an offer that's already exists in the database. -The system shows an error message that the offer is already exists.	
Includes	-	

Table 23 - Use Case Table - Update Offer

ID	UC-2.5	
Title	Update offer	
Description	The store owner updates an offer that exists in his store and in the offers database on the system	
Actor	Store owner	
Pre-conditions	The store owner already has an account on the system and already logged in and already made an offer that he wants to edit	
Post-conditions	The store owner offer can be promoted to the users by the application according to the information of the offer	
Flow of events	**User Action**	**System Action**
	1- The store owner chooses the offer he wants to update	
		2- The system retrieves its information and shows it to the store owner

	3- Store owner changes the information he wants and press the update offer button	
		4- System saves the new information of the offer to the database and deletes the old one
Exceptions	-	
Includes	-	

Table 24 - Use Case Table - Delete Offer

ID	UC-2.6	
Title	Delete offer	
Description	The store owner deletes an offer that exists in his store and in the offers database on the system	
Actor	Store owner	
Pre-conditions	The store owner already has an account on the system and already logged in and already made an offer that he wants to delete	
Post-conditions	The offer no longer exists in the database of the system	
Flow of events	**User Action**	**System Action**
	1- The store owner chooses the offer he wants to delete	
		2- System deletes the offer from the database
Exceptions	-	
Includes	-	

3.9 Sequence Diagrams

Android Part

SignupUser

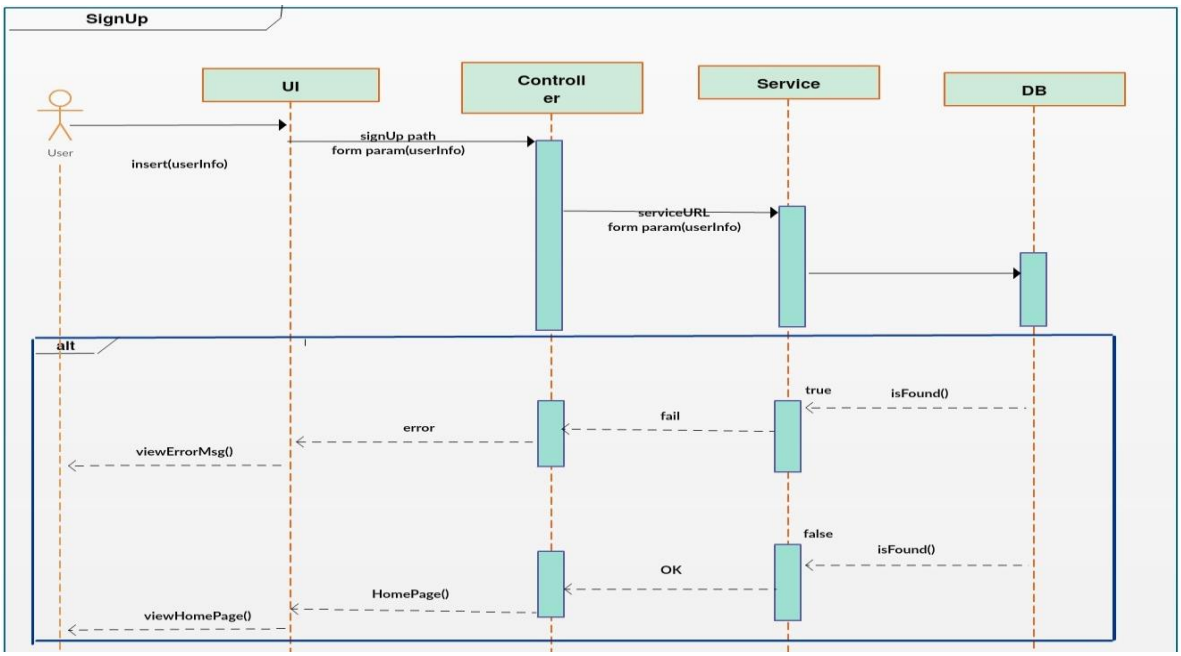


Figure 10 - Sequence Diagram - Signup User

LoginUser

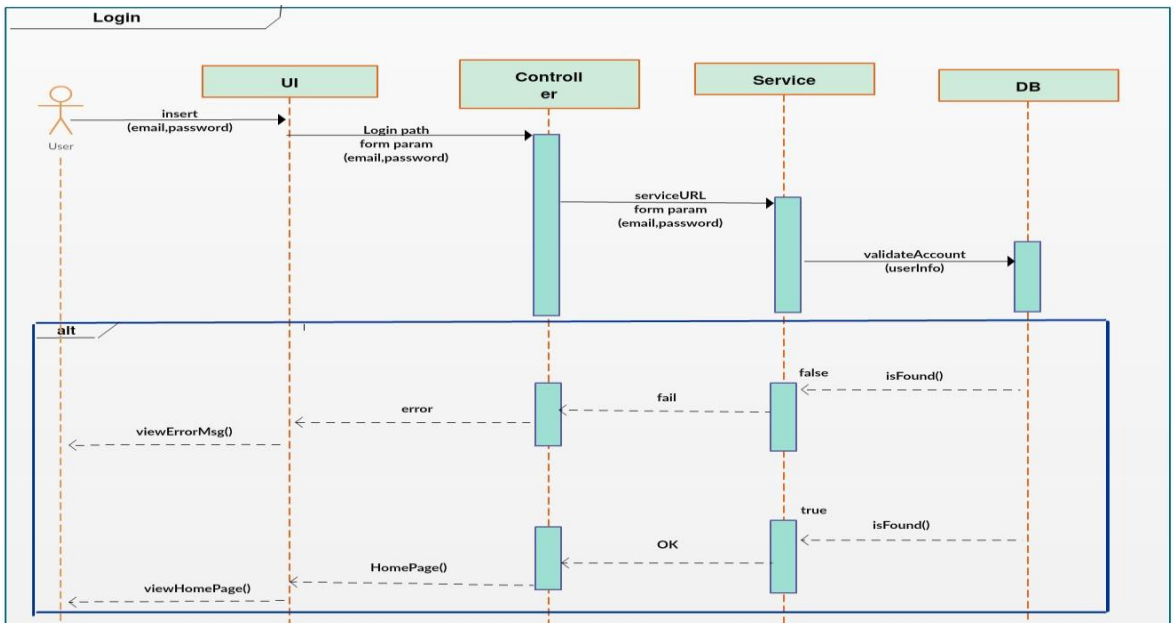


Figure 11 - Sequence Diagram - Login User

AddOrdinaryNote

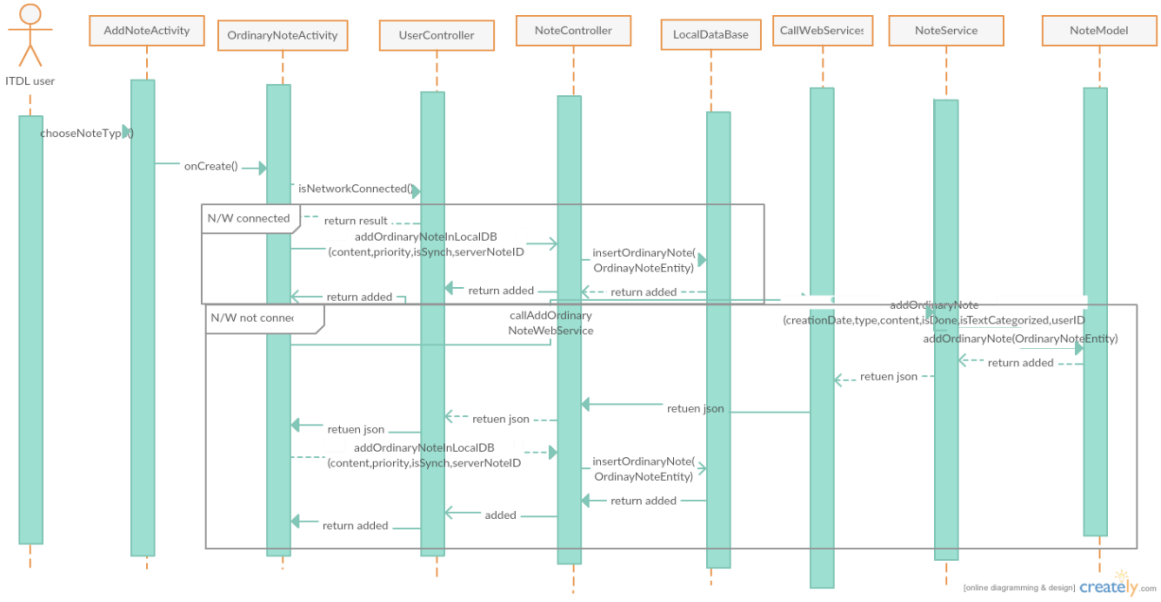


Figure 12 - Sequence Diagram – AddOrdinaryNote

AddShoppingNote

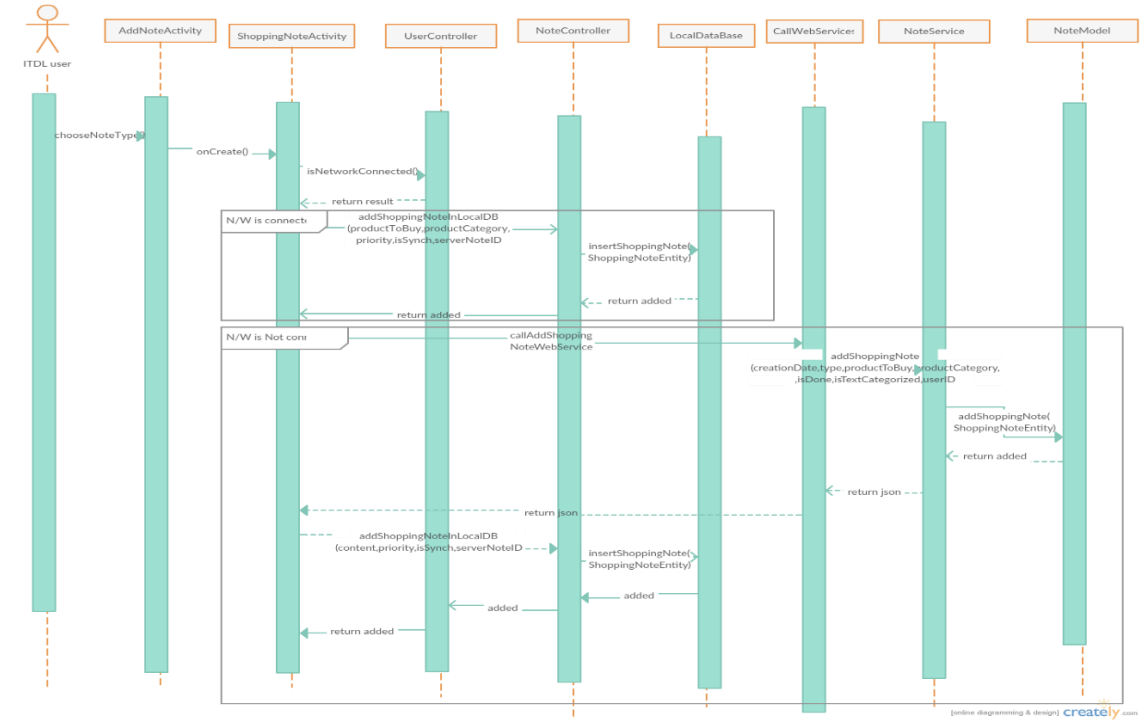


Figure 13 - Sequence Diagram – AddShoppingNote

AddMeetingNote

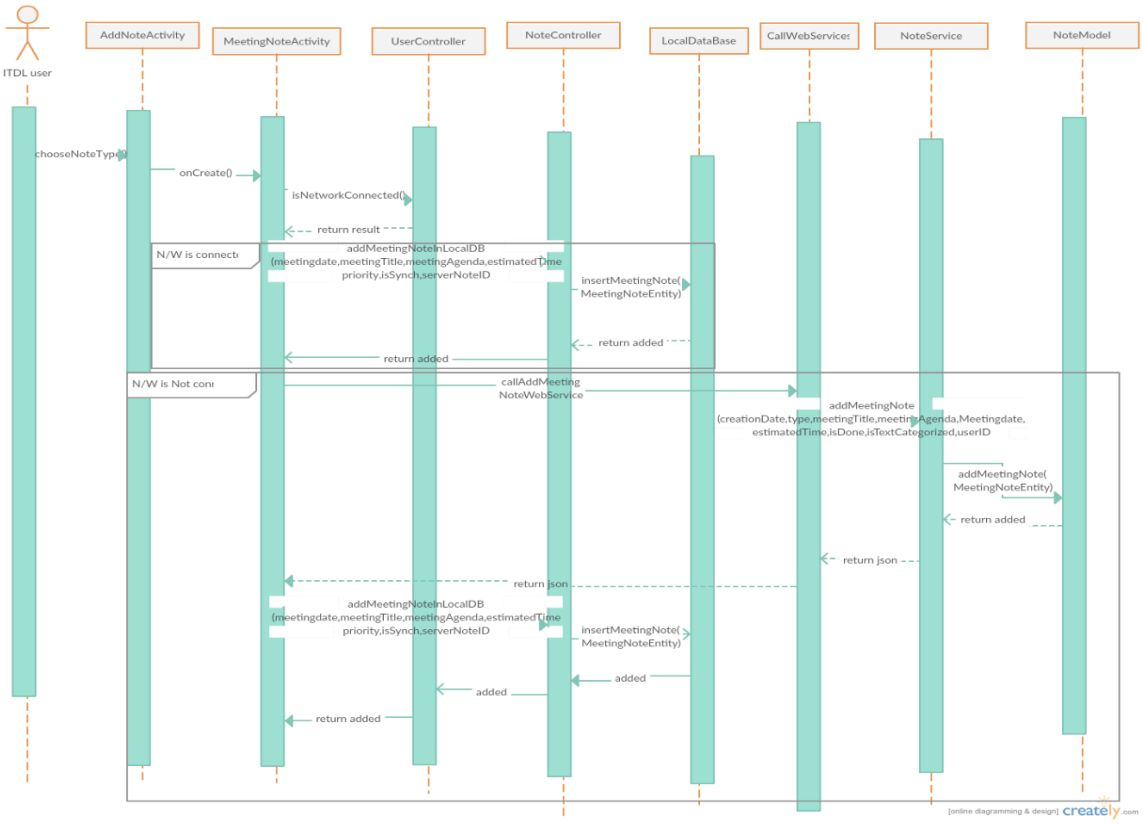


Figure 14 - Sequence Diagram - AddMeetingNote

AddDeadlineNote

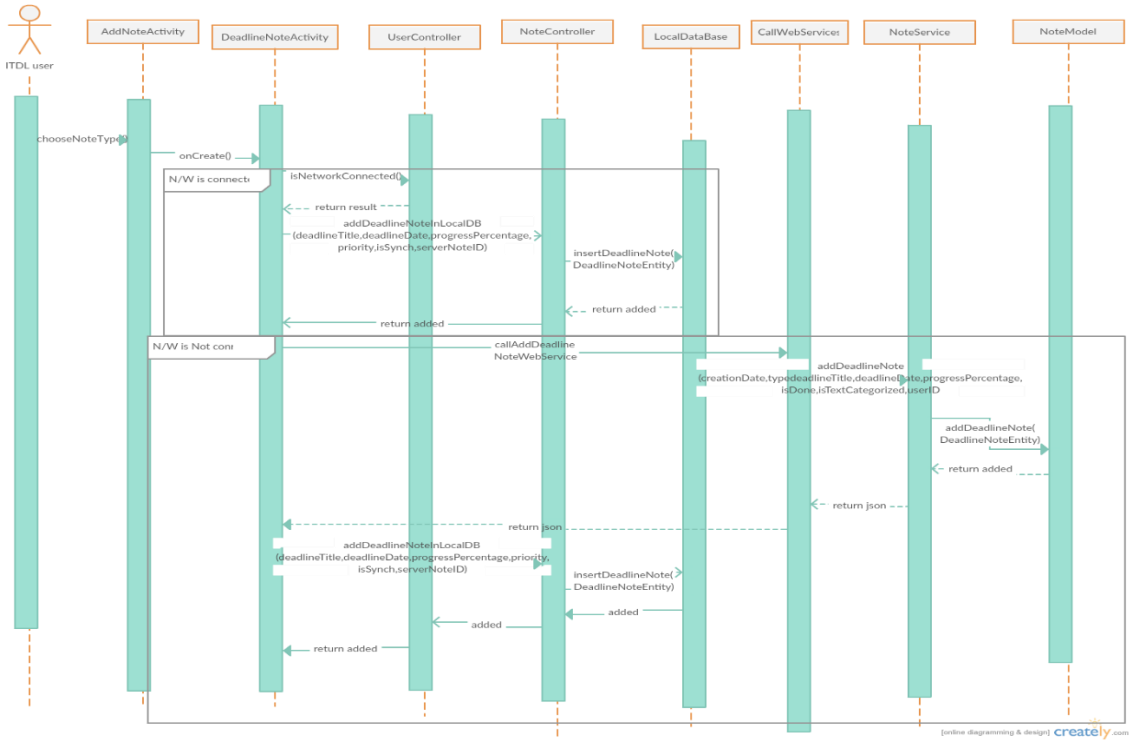


Figure 15 - Sequence Diagram - AddDeadlineNote

UpdateNote

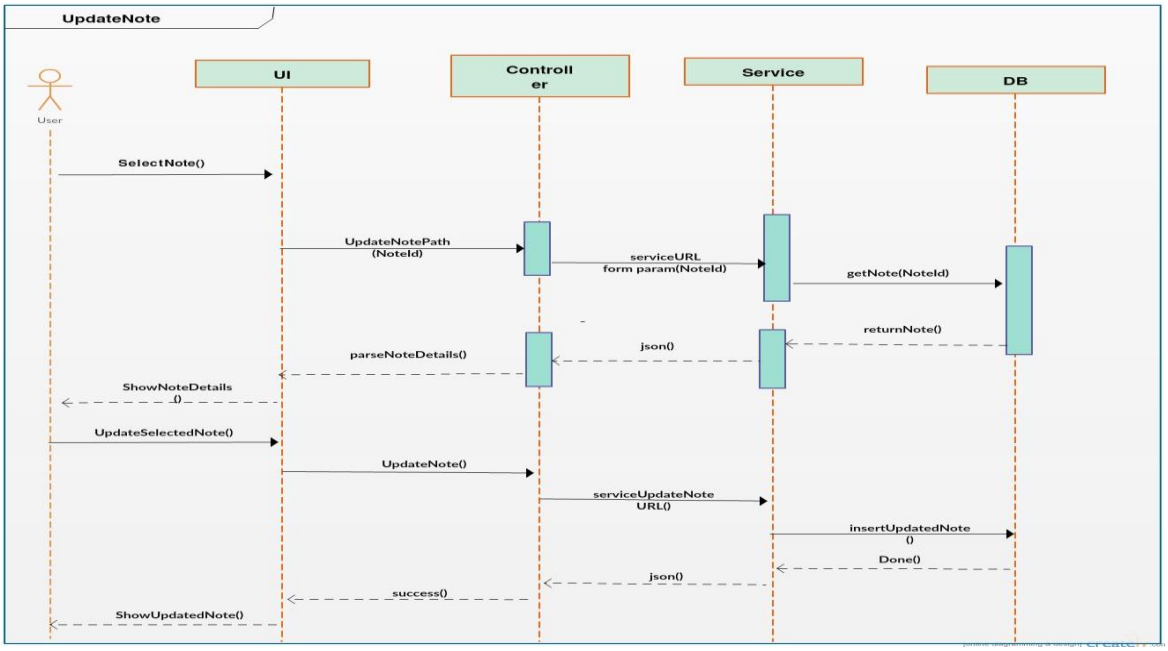


Figure 16 - Sequence Diagram - UpdateNote

DeleteNote

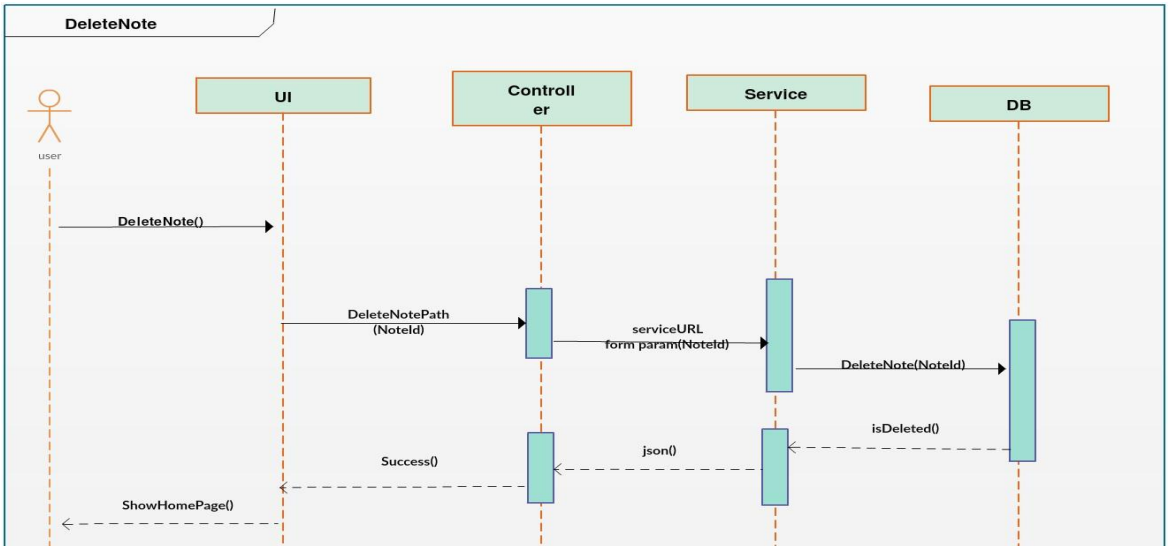


Figure 17 - Sequence Diagram - DeleteNote

Show current and history notes

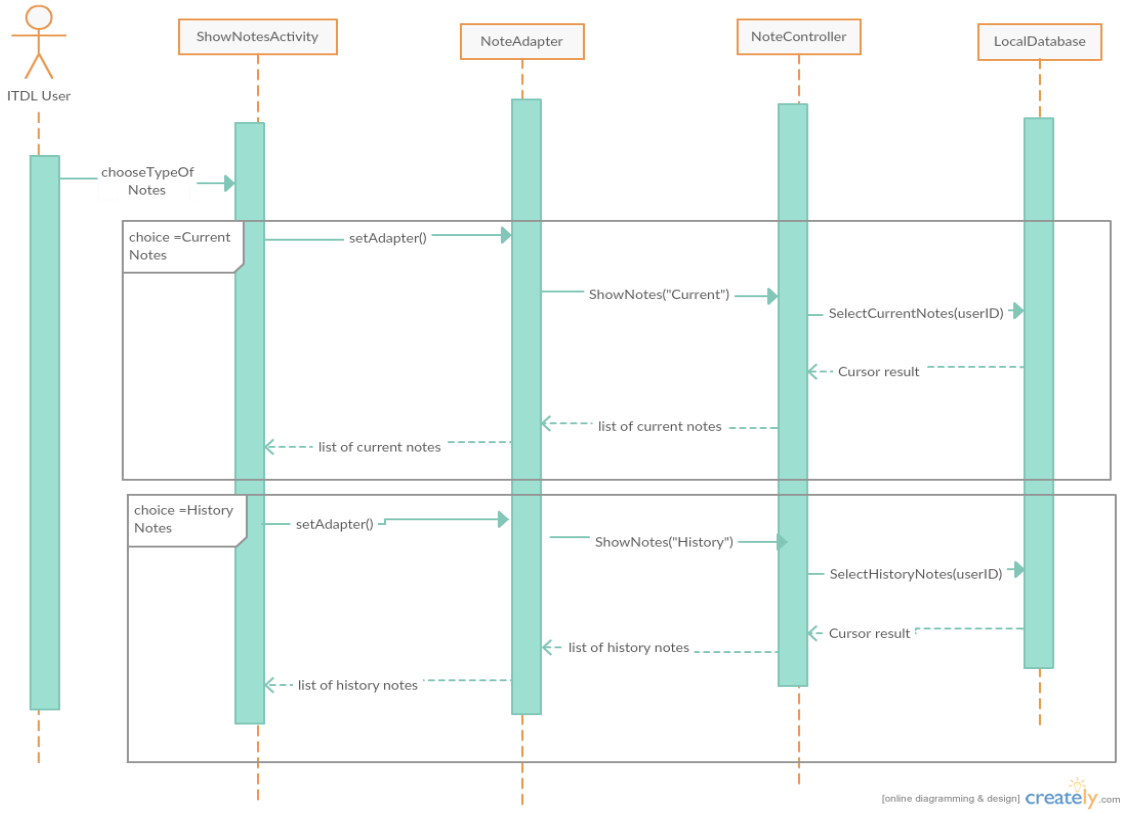


Figure 18 – Sequence Diagram – Show current and history notes

Update preferences front end

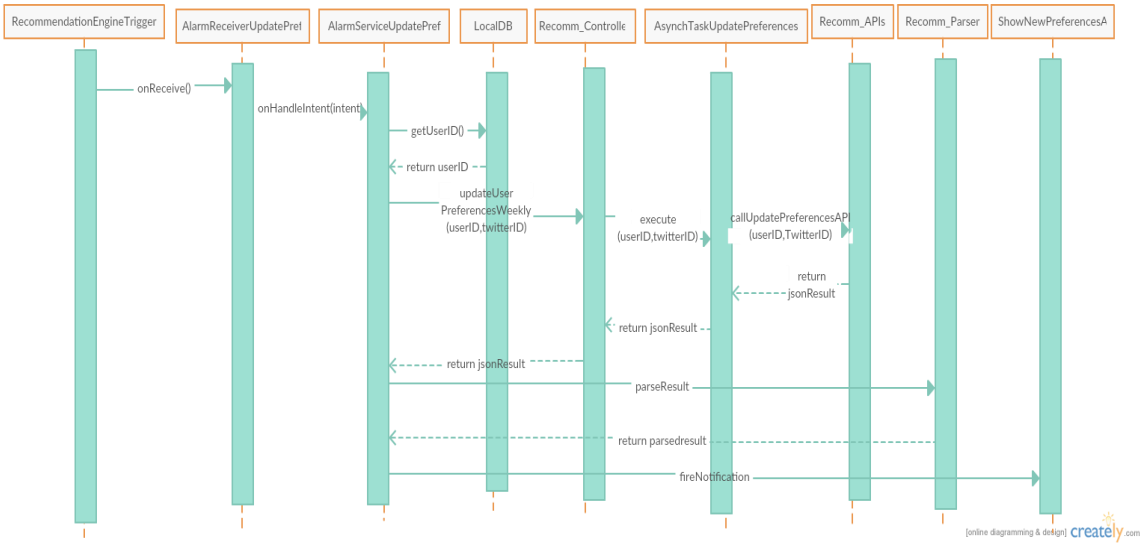


Figure 19 – Sequence Diagram – UpdatePrefFrontEnd

Update preferences backend

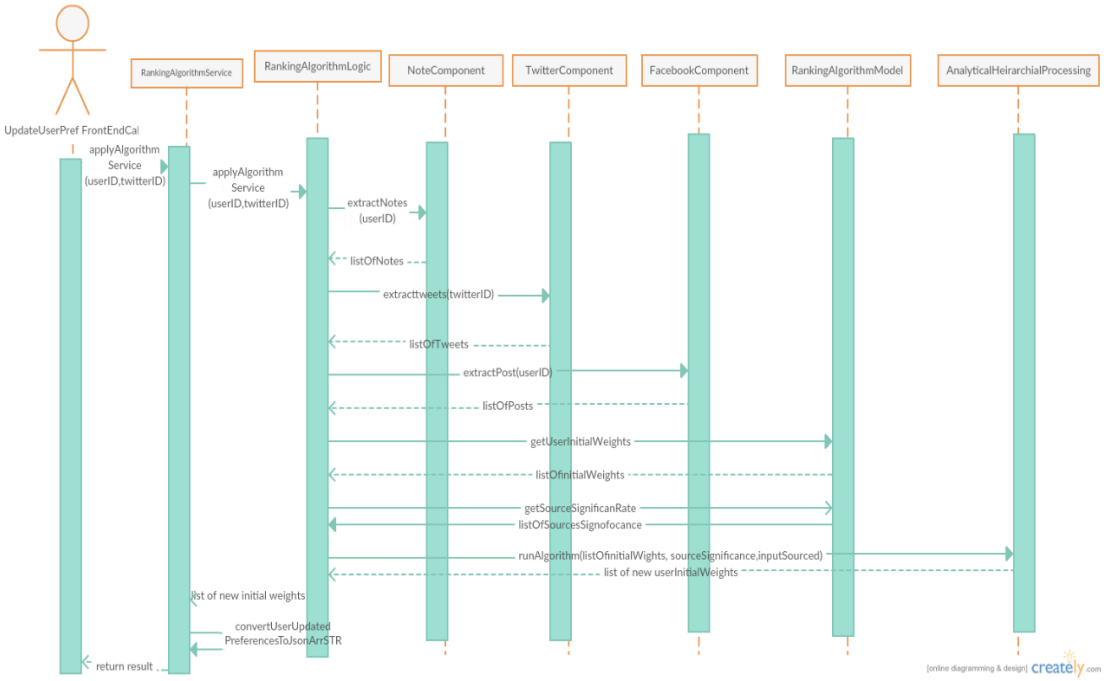


Figure 20 – Sequence Diagram – Update preferences backend

getNeareastStores backend and frontend

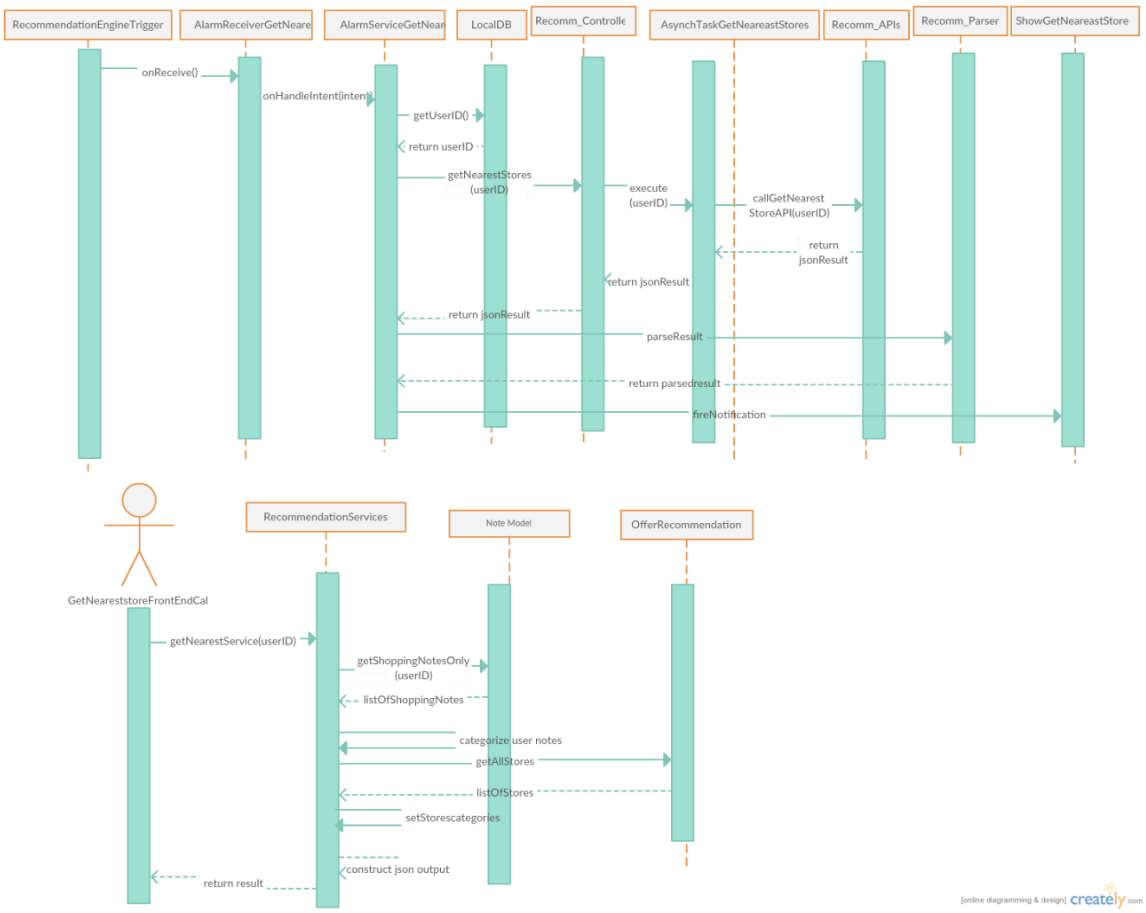


Figure 21 – Sequence Diagram – getNeareastStores

getOffers backend and frontend

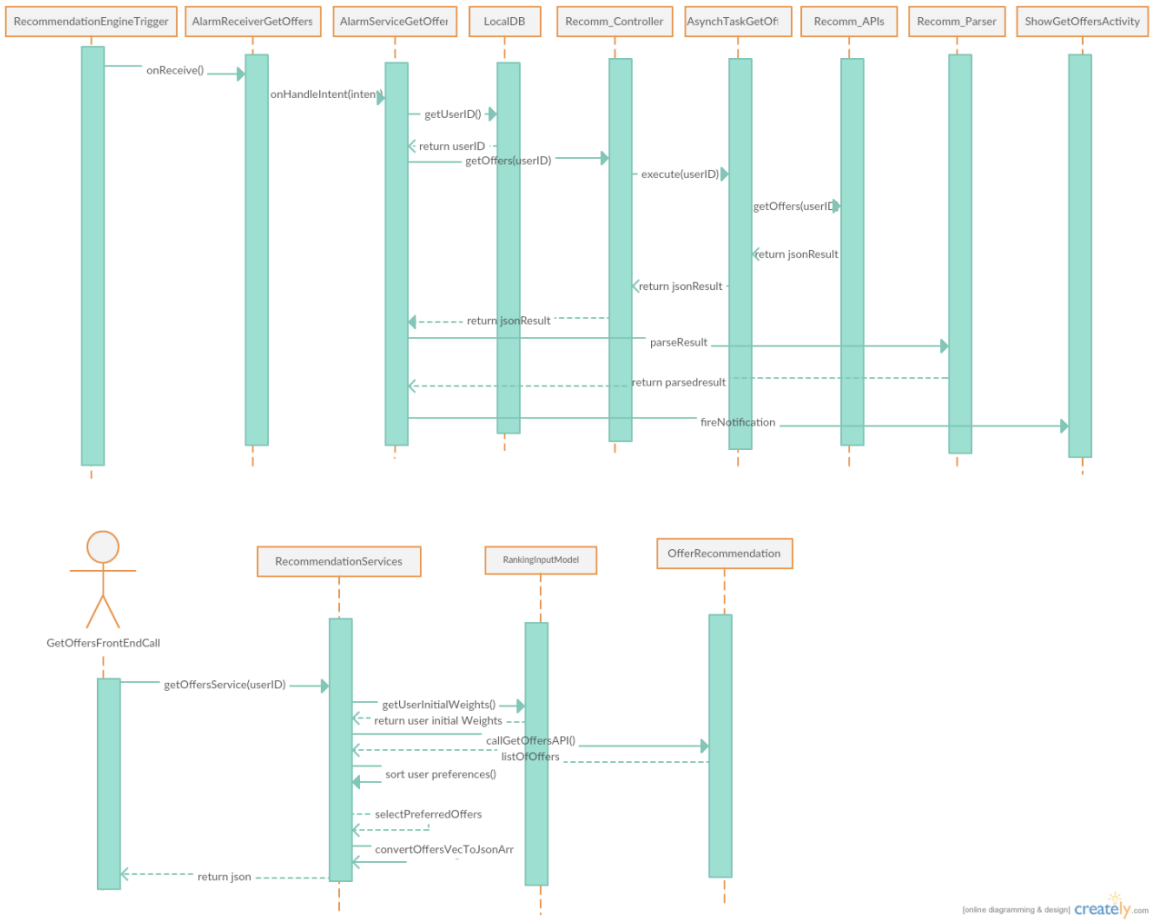


Figure 22 – Sequence Diagram – getOffers

Web Part

SignupStore

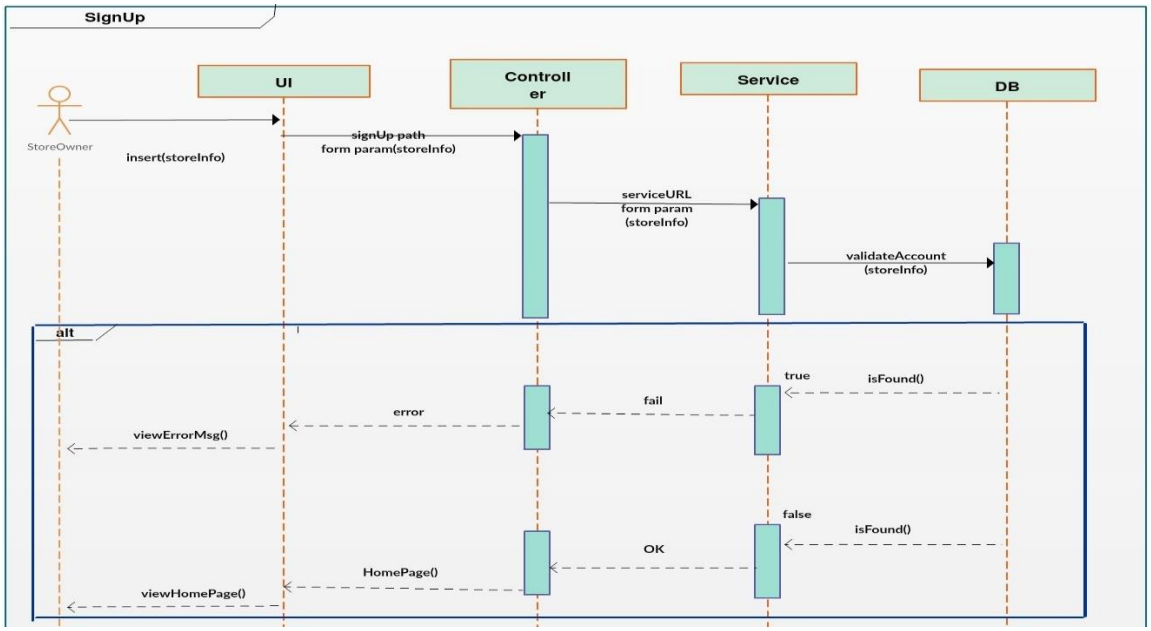


Figure 23 - Sequence Diagram - SignupStore

LoginStore

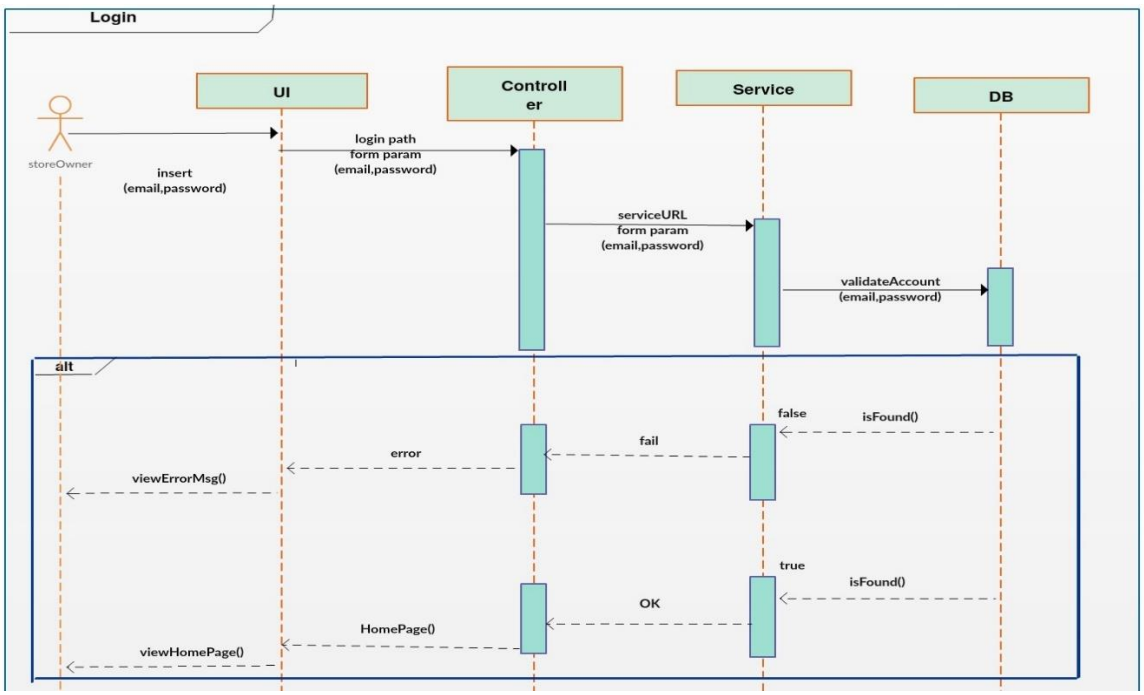


Figure 24 - Sequence Diagram - LoginStore

AddOffer

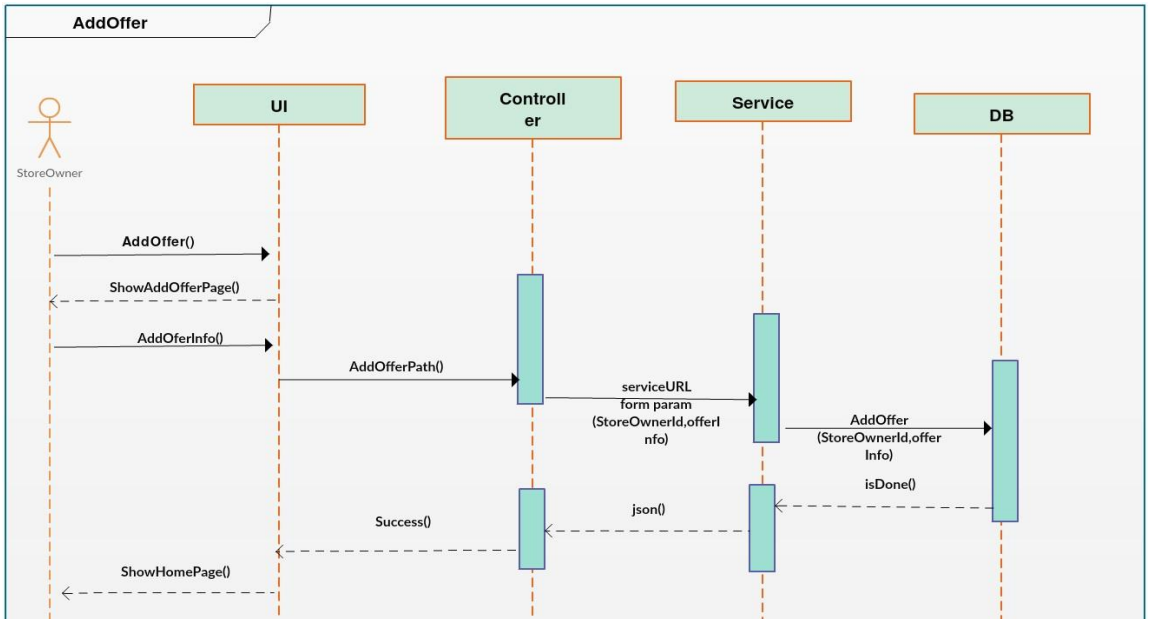


Figure 25 - Sequence Diagram - AddOffer

UpdateOffer

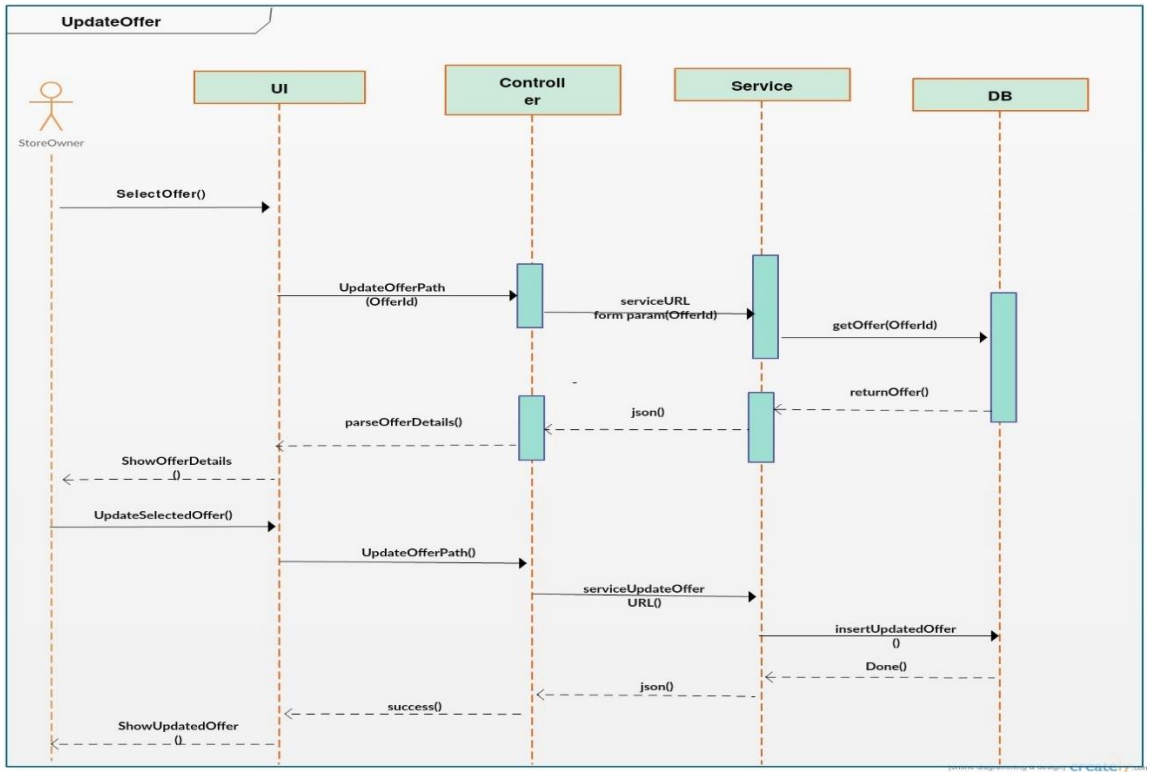


Figure 26 - Sequence Diagram - UpdateOffer

DeleteOffer

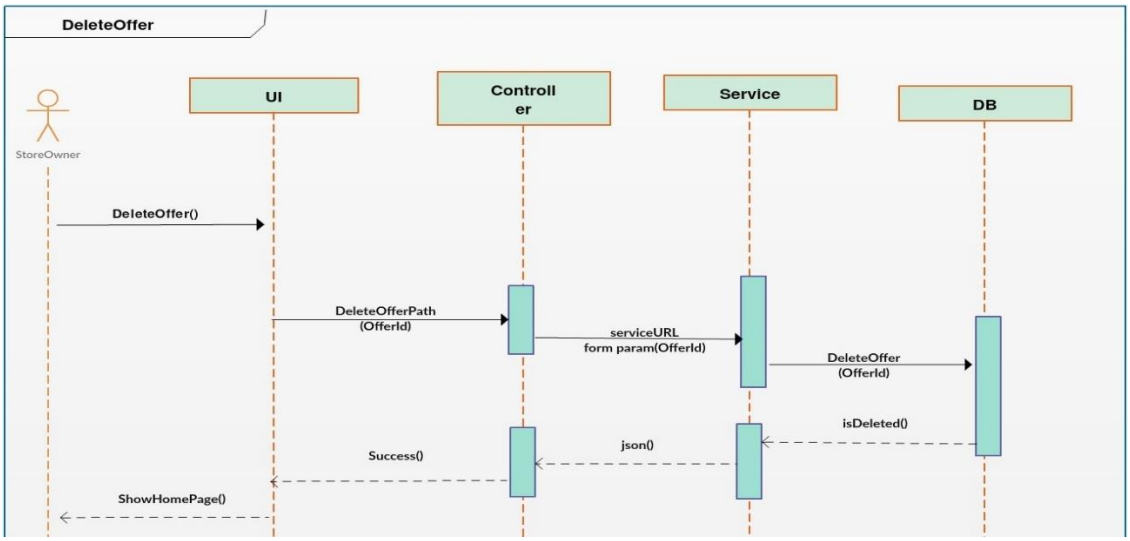


Figure 27 - Sequence Diagram – DeleteOffer

UpdateStoreProfile

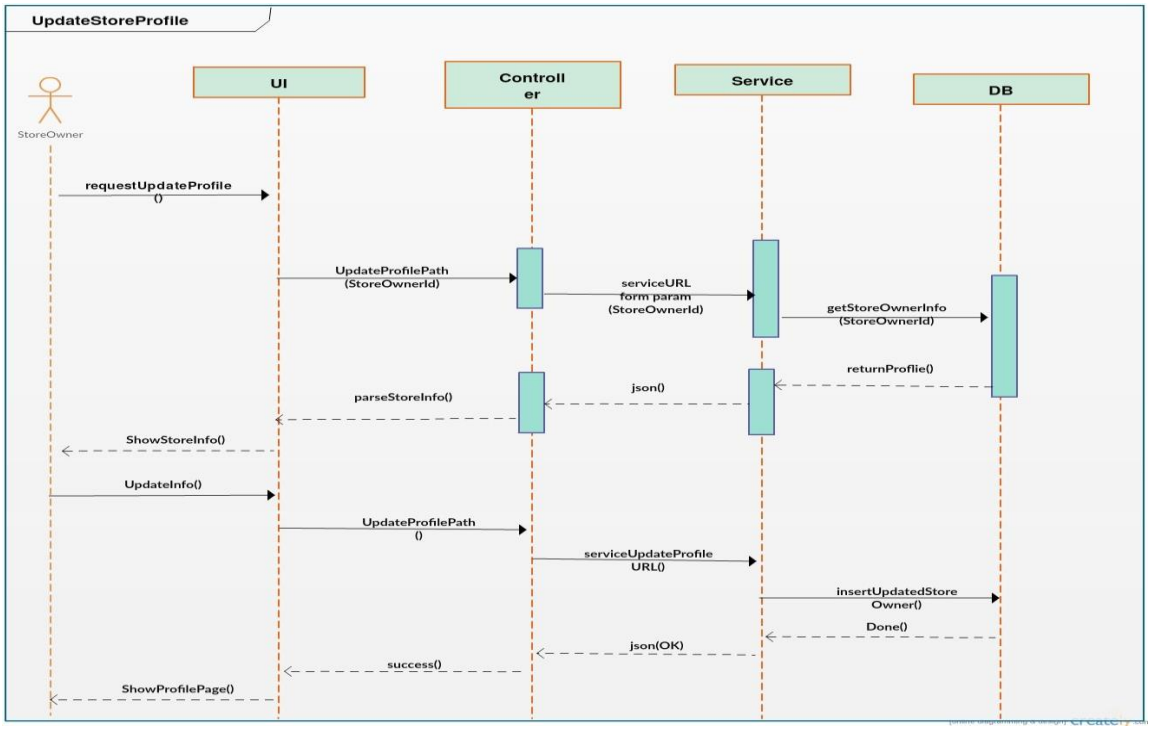


Figure 28 - Sequence Diagram - UpdateStoreProfile

3.10 System test cases.

Web Part

Table 25 - Signup Store Method Inputs

Input	Partitions
Email	E1: Email Exist in DB E2: Email Not Exist in DB E3: Email Not Given
Password	E4: Password size < 6 E5: Password Not Given E6: Password size >= 6
Name	E7: Name Not Given E8: Name >= 6 E9: Name size < 6
Coordinates	E10: Geolocation Permission Denied E11: No Answer to Geolocation for long time E12: No Internet Connection E13: Geolocation not supported by the browser E14: Geolocation Permission Allowed

Table 26 - Signup Store Method Test Cases

#	Input	Expected Output	Covered
1	Email: Yasmeen.kloub@gmail.com Password: 20120456 Name: Yasmin Coordinates: 30.0012354, 31.1232455	Registration Completed	E2, E6, E8, E14
2	Email: Yasmeen.kloub@gmail.com Password: 123456 Name: Yasmin Coordinates: 30.0012354, 31.1232455	ERROR: Email Already Exist	E1, E6, E8, E14
3	Email: Password: 1234567 Name: Yasmin Coordinates: 30.0012354, 31.1232455	ERROR: Email Text Field Must be given	E3, E6, E8, E14

4	Email: Yasmeen.kloub@yahoo.com Password: 12345 Name: yasmin Coordinates: 30.0012354, 31.1232455	ERROR: Password Must be at least 6 characters	E2, E4, E8, E14
5	Email: Yasmin.kloub@gmail.com Password: Name: Yasmin Coordinates: 30.0012354, 31.1232455	ERROR: Password Text Field Must be given	E2, E5, E8, E14
6	Email: yasmin.kloub@yahoo.com Password: 1234567 Name: yasso Coordinates: 30.0012354, 31.1232455	ERROR: User Name Must be at least 6 characters	E2, E6, E9, E14
7	Email: yasso.kloub@gmail.com Password: 1234567 Name: Coordinates: 30.0012354, 31.1232455	ERROR: User Name Text Field Must be given	E2, E6, E7, E14
8	Email: yasso.kloub@yahoo.com Password: 1234567 Name: Yasmin Coordinates: Deny	ERROR: User denied the request for Geolocation.	E2, E6, E8, E10
9	Email: Yasmeen.latif@gmail.com Password: 1234567 Name: Yasmin Coordinates: No Answer to the Request	ERROR: The request to get user location timed out.	E2, E6, E8, E11
10	Email: Yasmeen.latif@yahoo.com Password: 1234567 Name: Yasmin Coordinates: No Internet Connection	ERROR: Location information is unavailable.	E2, E6, E8, E12
11	Email: Yasmin.latif@gmail.com Password: 1234567 Name: Yasmin Coordinates: IE9	ERROR: Geolocation is not supported by this browser.	E2, E6, E8, E13

Table 27 - Login Store Method Inputs

Input	Partitions
Email	E1: Email Exist in DB E2: Email Not Exist in DB E3: Email Not Given
Password	E4: Password Not Right E5: Password Not Given E6: Password Right

Table 28 - Login Store Method Test Cases

#	Input	Expected Output	Covered
2	Email: Yasmeen.kloub@gmail.com Password: 123456	Home Page	E1, E6
1	Email: Yasmin.latif@yahoo.com Password: 20120456	ERROR: Email Doesn't Exist	E2, E6
3	Email: Password: 1234567	ERROR: Email Text Field Must be given	E3, E6
4	Email: Yasmeen.kloub@yahoo.com Password: 12345	ERROR: You Have Entered a Wrong Password	E2, E4
5	Email: Yasmin.kloub@gmail.com Password:	ERROR: Password Text Field Must be given	E2, E5

Table 29 – Add/Update Offer Method Inputs

Input	Partitions
Start Date	E1: End Date > Start Date => Current Date E2: Start Date < Current Date E3: Start Date Not Given
End Date	E4: End Date > Current Date E5: End Date < Start Date E6: End Date < Current Date E7: End Date Not Given
Category	E8: Category is Selected E9: Category Not Selected
Content	E10: Content is Selected, E11: Content Not Selected

Table 30 – Add/Update Offer Method Test Cases

#	Input Assuming Current Date = (2/6/2016)	Expected Output	Covered
1	Start Date: 3/6/2016 End Date: 6/6/1016 Category: Food and Drinks Content: Order 2 Pizza and get one Free	Offer Added Successfully	E1, E4, E8, E10
2	Start Date: 1/6/2016 End Date: 6/6/1016 Category: Food and Drinks Content: Order 2 Pizza and get one Free	ERROR: Start Date Must Be After the Current Date	E2, E4, E8, E10
3	Start Date: End Date: 6/6/1016 Category: Food and Drinks Content: Order 2 Pizza and get one Free	ERROR: Start Date Must Be Given	E3, E4, E8, E10
4	Start Date: 4/6/2016 End Date: 3/6/1016 Category: Food and Drinks Content: Order 2 Pizza and get one Free	ERROR: End Date Must Be After the Start Date	E1, E5, E8, E10
5	Start Date: 4/6/2016 End Date: 1/6/1016 Category: Food and Drinks Content: Order 2 Pizza and get one Free	ERROR: End Date Must Be After the Current Date and the Start Date	E1, E5, E6, E8, E10
6	Start Date: 3/6/2016, End Date: Category: Food and Drinks Content: Order 2 Pizza and get one Free	ERROR: End Date Must Be Given	E1, E7, E8, E10
7	Start Date: 3/6/2016 End Date: 6/6/1016, Category: Content: Order 2 Pizza and get one Free	ERROR: Category Must Be Selected	E1, E4, E9, E10
8	Start Date: 3/6/2016 End Date: 6/6/1016 Category: Food and Drinks, Content:	ERROR: Content Must Be Given	E1, E4, E8, E11

Table 31 - Signup User Method Inputs

Input	Partitions
User object	E1: User object is null. E2: User object is not null.
User Name	E3:UserName is null E4:USerName is not null
User Password	E5: User Password is null E6: User Password is not null
User Email	E7: Email valid E8:Email not valid E9:Email is null
User Gender	E10: gender is null E11:gender is not null
User birth date	E12: birth date<Current date E13 : birth date >=Current date
Twitter Username	E14: Username valid E15:Email not valid

Table 32 - Signup User Method Test Cases

#	Input	Expected Output	Covered
1	User=null	NullPointerException (Assumed)	E1
2	User name ="" Userpassword="1234567" Useremail=ahmed@yahoo.com Usergender="male" userbirthdate="2/1/1990" TwitterUsername="@ahmed"	Failed to sign up	E2,E3,E6,E7, E11 E12,E14
3	User name ="ahmed" Userpassword="" Useremail=ahmed@yahoo.com Usergender="male" userbirthdate="2/1/1990"	Failed to sign up	E2,E3,E5,E7, E11 E12,E14

	TwitterUsername="@ahmed"		
4	User name ="ahmed", Userpassword="1234567" Useremail=a@yahoo.com Usergender="male" userbirthdate="2/1/1990" TwitterUsername="@ahmed"	Failed to sign up	E2,E3,E6,E8, E11 E12,E14
5	User name ="ahmed", Userpassword="1234567", Useremail= "", Usergender="male",userbirthdate ="2/1/1990" TwitterUsername="@ahmed"	Failed to sign up	E2,E3,E6,E9, E11 E12,E14
6	User name ="mona", Userpassword="1234567" Useremail= mona12@yahoo.com Usergender="", userbirthdate="2/1/1990" TwitterUsername="@mona"	Failed to sign up	E2,E3,E6,E9, E10 E12,E14
7	User name ="mona", Userpassword="1234567", Useremail=mona12@yahoo.com Usergender="female ", userbirthdate="2/3/2016" TwitterUsername="@mona"	Failed to sign up	E2,E3,E6,E9, E10 E13,E14
8	User name ="mona", Userpassword="1234567", Useremail=mona12@yahoo.com, Usergender="female", userbirthdate="2/3/1990" TwitterUsername="@mona"	Register Successes	E2,E3,E6,E9, E10 E12,E14

Table 33 - Login User Method Inputs

Input	Partitions
email	E1: email is valid. E2: email is not valid.
password	E3: password is correct. E4: password is not correct.

Table 34 - Login User Method Test Cases

#	Input	Expected Output	Covered
1	Email="gp@yahoo.com", password="123456" (Assumed this mail exist in DB)	logged in Successfully	E1,E3
2	Email="gp12@yahoo.com", password="123456" (Assumed this mail it doesn't exist in DB)	Failed to log in	E2
3	Email="gp@yahoo.com", password="12345689" (Assumed this mail exist in DB but this password isn't correct)	Failed to log in	E4,E3

Table 35 - Add/Update Ordinary Note Method Inputs

Input	Partitions
OrdinaryNote object	E1: OrdinaryNote object is null. E2: OrdinaryNote object is not null.

Table 36 - Add/Update Ordinary Note Method Test Cases

#	Input	Expected Output	Covered
1	Note=null	NullPointerException (Assumed)	E1
2	Note="I want to play Football"	True	E2

Table 37 - Add Meeting Note Method Inputs

Input	Partitions
title	E1: title is null. E2: title is not null.
content	E3: content is null. E4: content is not null.
Date	E5: Date < Current date. E6: Date >= Current date.

Table 38 - Add/Update Meeting Note Method Test Cases

#	Input	Expected Output	Covered
1	Title=null, content="GP Meeting",Date="7/2/2016" (Assumed Current Date="2/2/2016")	NullPointerException (Assumed)	E1,E4,E6
2	Title=GP, content="GP Meeting",Date="7/2/2016" (Assumed Current Date="2/2/2016")	True	E2,E4,E6
3	Title=GP, content=null, Date="7/2/2016" (Assumed Current Date="2/2/2016")	NullPointerException (Assumed)	E3,E2,E6
4	Title=GP, content="GP Meeting",Date="7/2/2016" (Assumed Current Date="2/2/2016")	True	E4,E2,E6
5	Title=GP, content="GP Meeting",Date=="20/1/2016" (Assumed Current Date="2/2/2016")	InvalidParameterException (Assumed)	E5,E2,E4
6	Title=GP, content="GP Meeting",Date="7/2/2016" (Assumed Current Date="2/2/2016")	True	E6,E2,E4

Table 39 - Add/Update Deadline Note Method Inputs

Input	Partitions
DeadlineNote object	E1: DeadlineNote object is null. E2: DeadlineNote object is not null.
Deadline Date	E3: Deadline Date>Current Date E4:Deadline Date <=Current Date
progress	E5: progress<0 E6:progress >100 E7:0<=Progress <=100
title	E8:title is null E9:title is not null
content	E10:content is null E11:content is not null

Table 40 - Add/Update Deadline Note Method Test Cases

#	Input	Expected Output	Covered
1	Content="Assignment is about chapter 1" DeadlineDate=4/2/2016 Current Date=6/2/2016 Progress =0,title="Assignment IA"	Failed to add/Update Note	E2,E4,E7,E9,E11
2	Content="Assignment is about chapter 1" DeadlineDate=9/2/2016 Current Date=6/2/2016 Progress =0,title="Assignment IA"	Note Added/Updated Successfully	E2,E3,E7,E9,E11
3	Content="Assignment is about chapter 1" DeadlineDate=9/2/2016 Current Date=6/2/2016 Progress =-1, title="Assignment IA"	Failed to add/Update Note	E2,E4,E5,E9,E11
4	Content="Assignment is about chapter 1" DeadlineDate=9/2/2016	Failed to add/Update Note	E2,E4,E6,E9,E11

	Current Date=6/2/2016 Progress =120, title="Assignment IA"		
5	Content="" DeadlineDate=9/2/2016 Current Date=6/2/2016 Progress =50, title="Assignment IA"	Failed to add/Update Note	E2,E4,E7,E9,E10
6	Content="Assignment is about chapter 1" DeadlineDate=9/2/2016 Current Date=6/2/2016 Progress =50, title=""	Failed to add/Update Note	E2,E4,E7,E8,E11

Table 41 - Add/Update Shopping Note Method Inputs

Input	Partitions
ShoppingNote object	E1: ShoppingNote object is null. E2: ShoppingNote object is not null.
Category	E3: Category null E4:Category not null
content	E5:content is null E6:content is not null

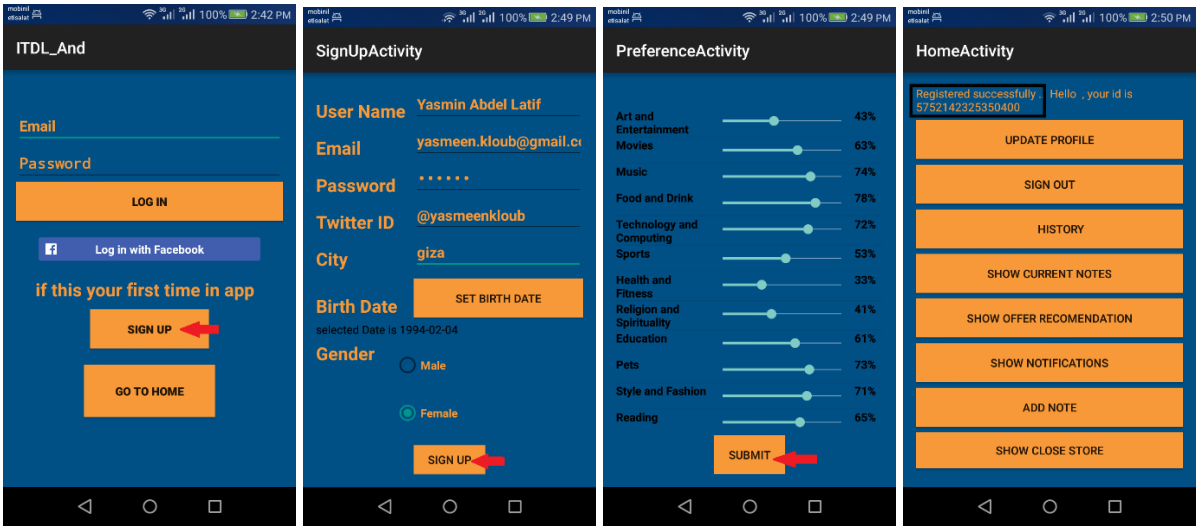
Table 42 - Add/Update Shopping Note Method Test Cases

#	Input	Expected Output	Covered
1	ShoppingNote()	NullPointerException (Assumed)	E1
2	ShoppingNote !=null,Content="milk" Category =" "	Failed to add/Update note	E2,E3,E6
3	ShoppingNote !=null,Content="" Category ="food"	Failed to add/Update Note	E2,E3,E5
4	ShoppingNote !=null,Content="cheese" Category ="food"	Note added/Updated Successfully	E2,E4,E6

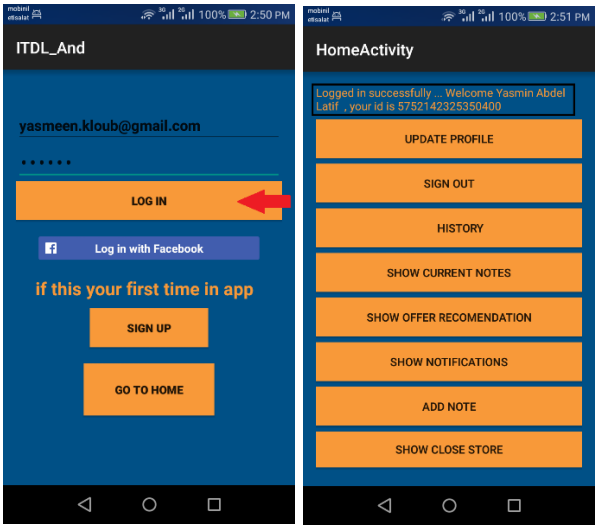
Chapter 4: Screen Shots

4.1 ANDROID APP SCREEN SHOTS

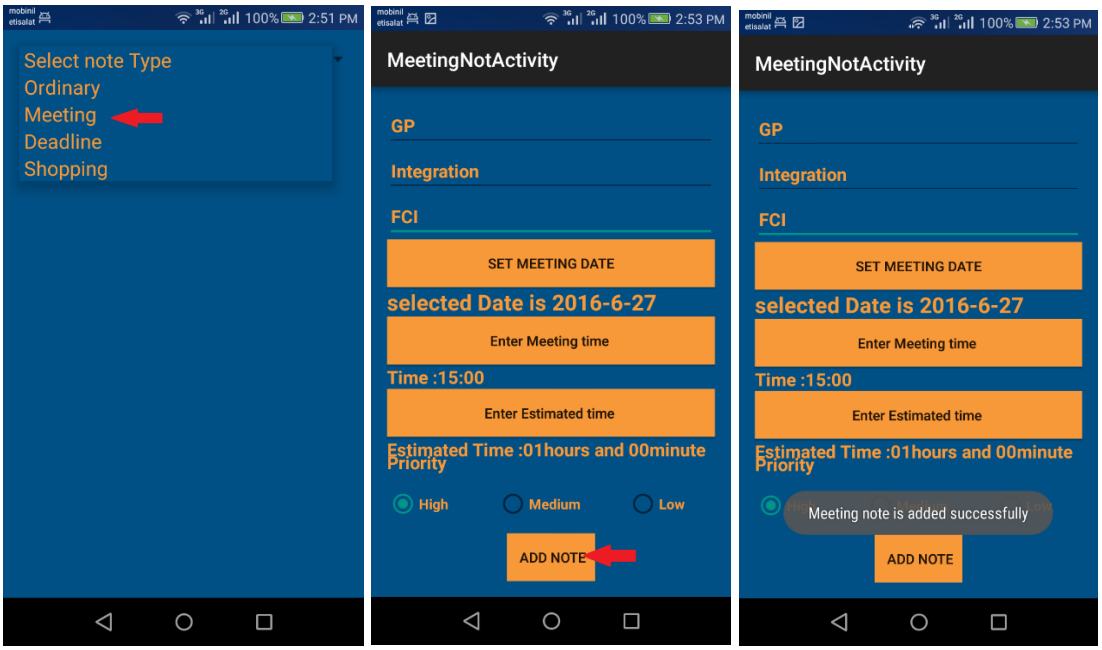
4.1.1 Sign up



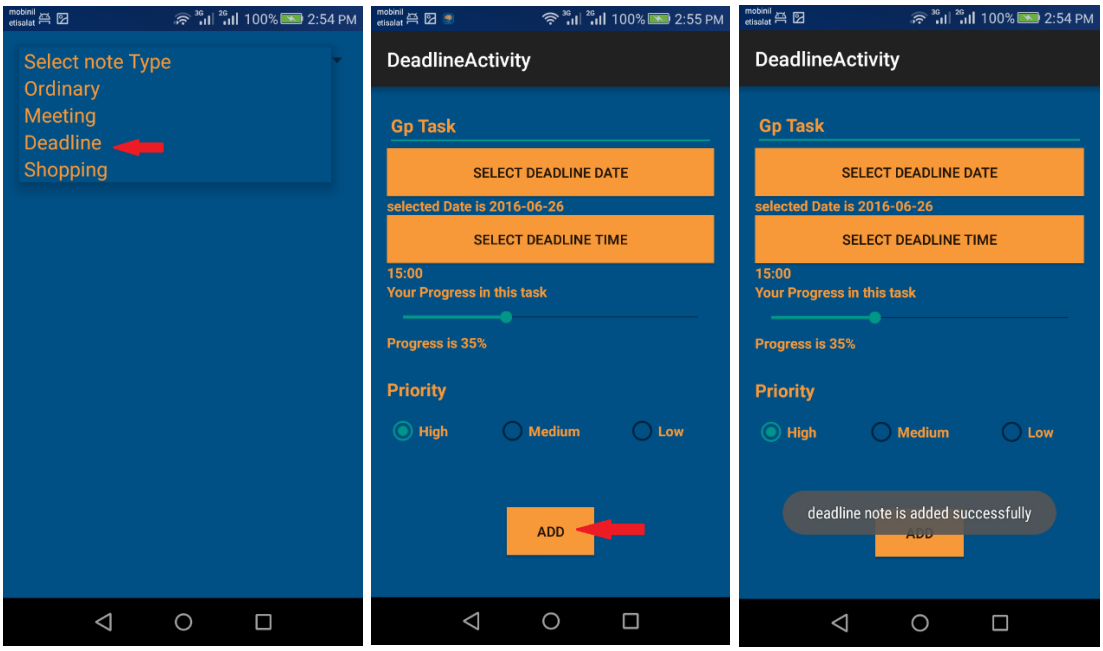
4.1.2 Log in



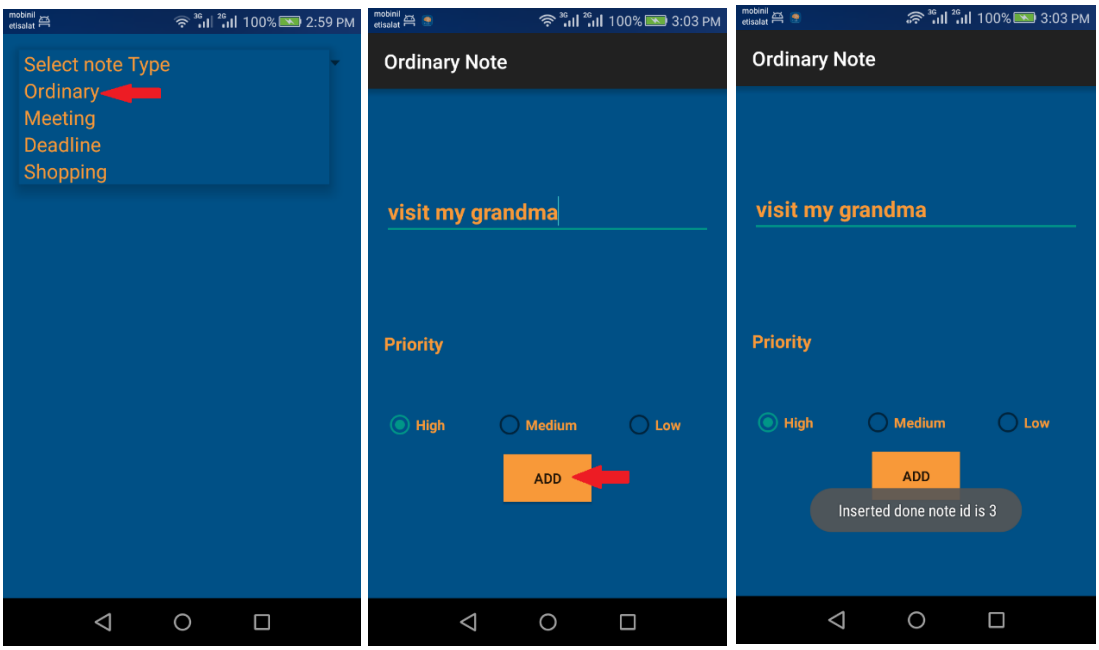
4.1.3 Add Meeting Note



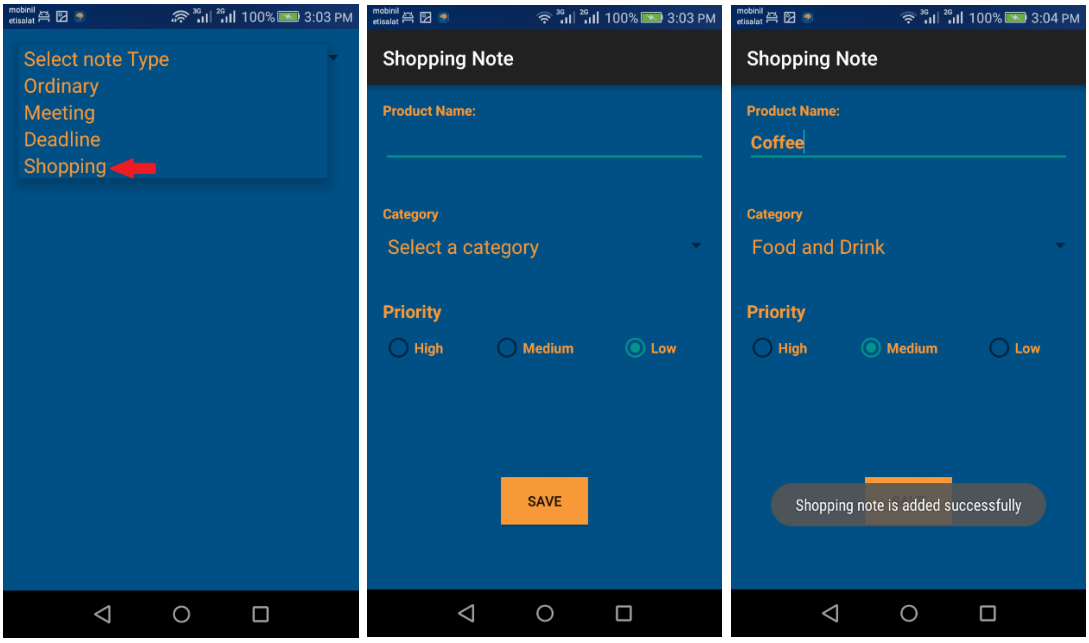
4.1.4 Add Deadline Note



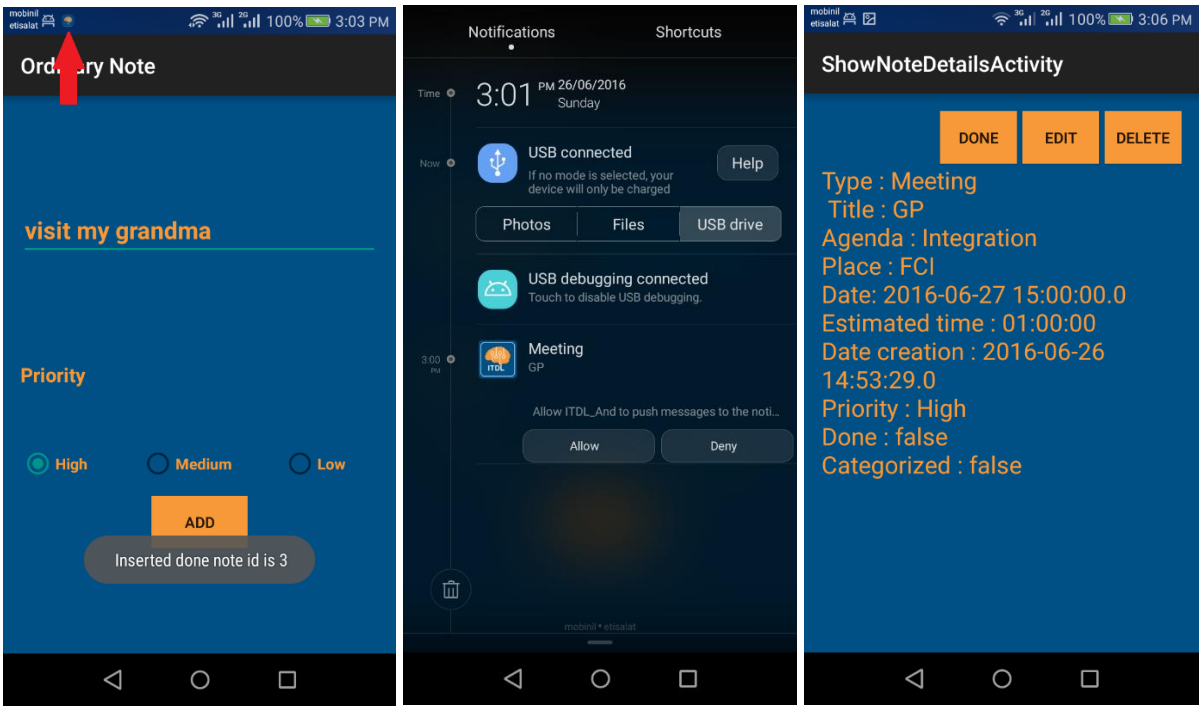
4.1.5 Add Ordinary Note



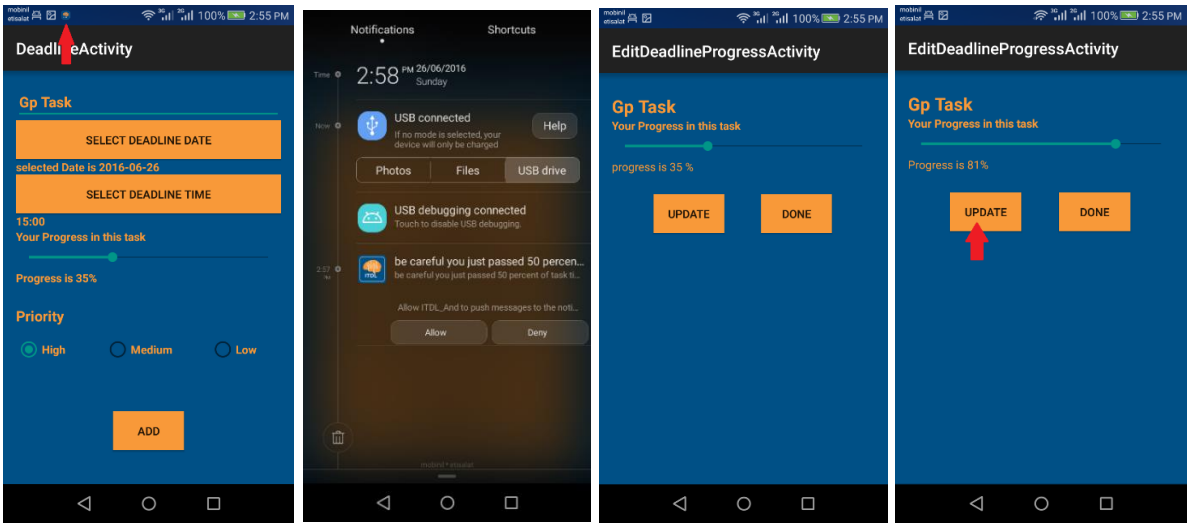
4.1.6 Add Shopping Note



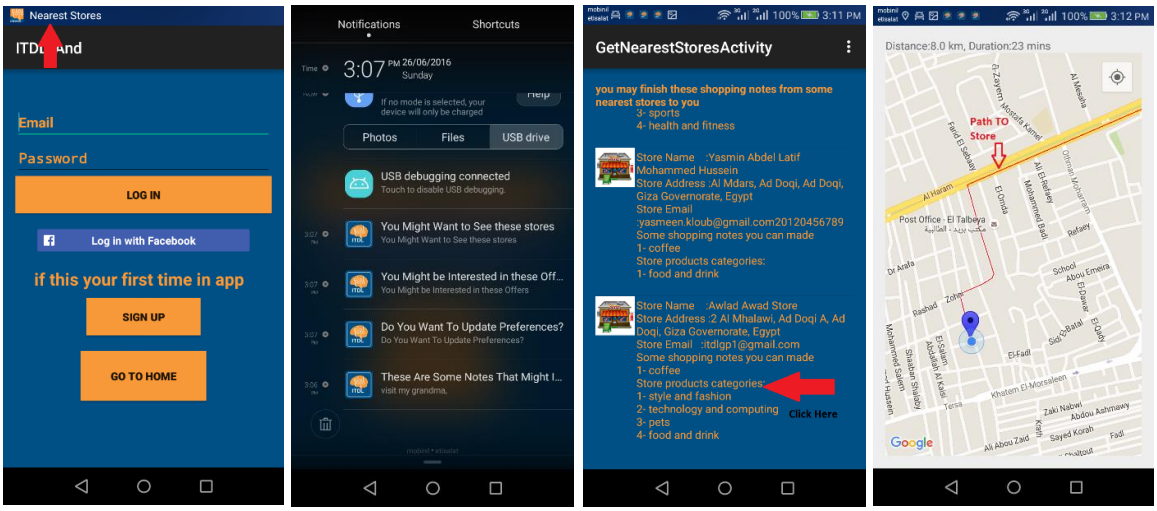
4.1.7 Get Meeting Alarm



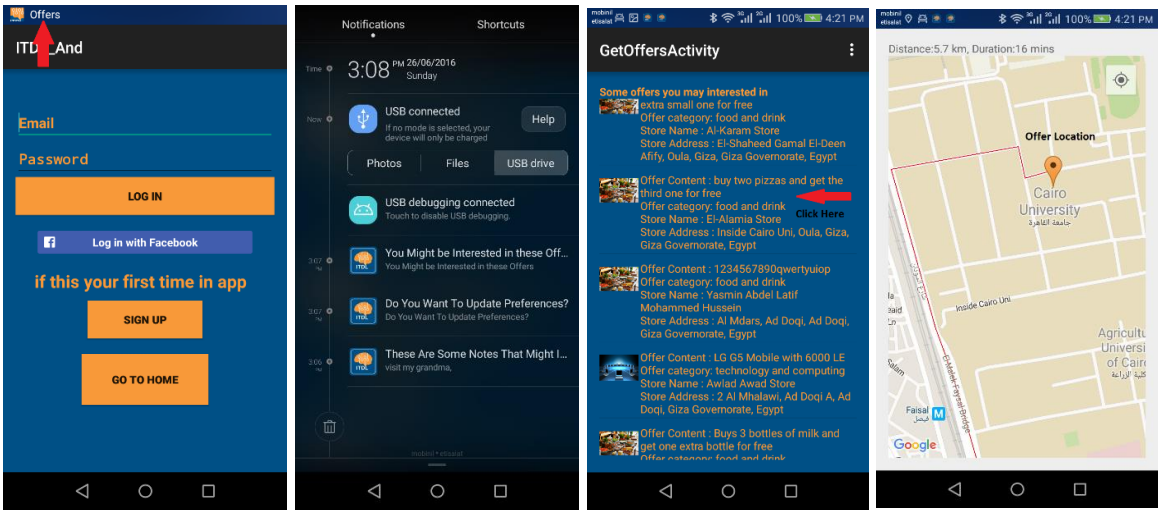
4.1.8 Get Deadline Alarm



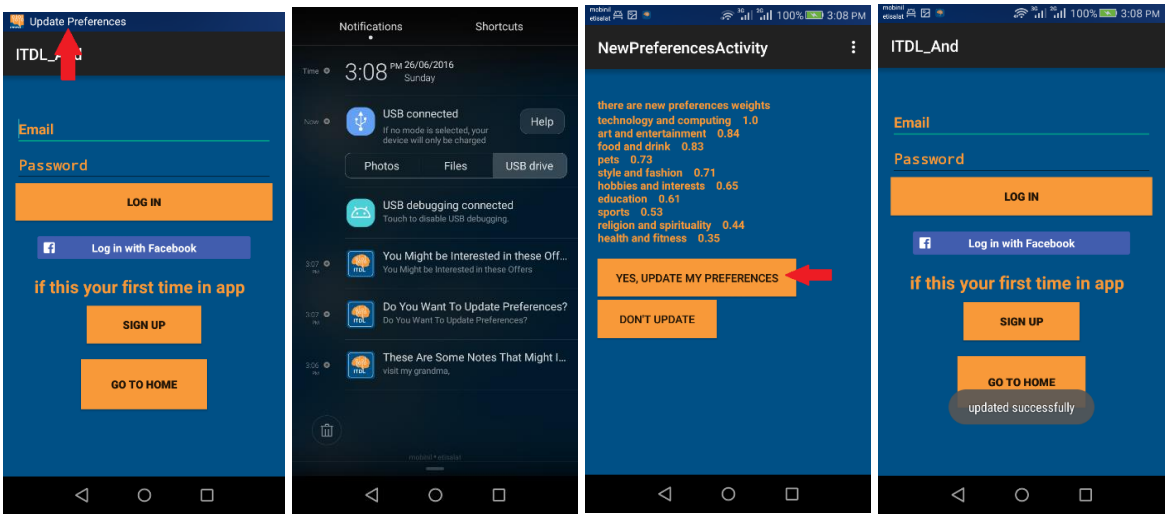
4.1.9 Get Nearest Store Notification



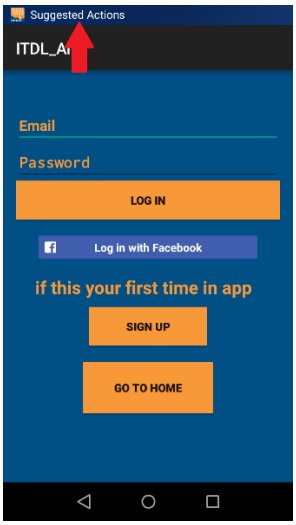
4.1.10 Get Nearest Offers Notification



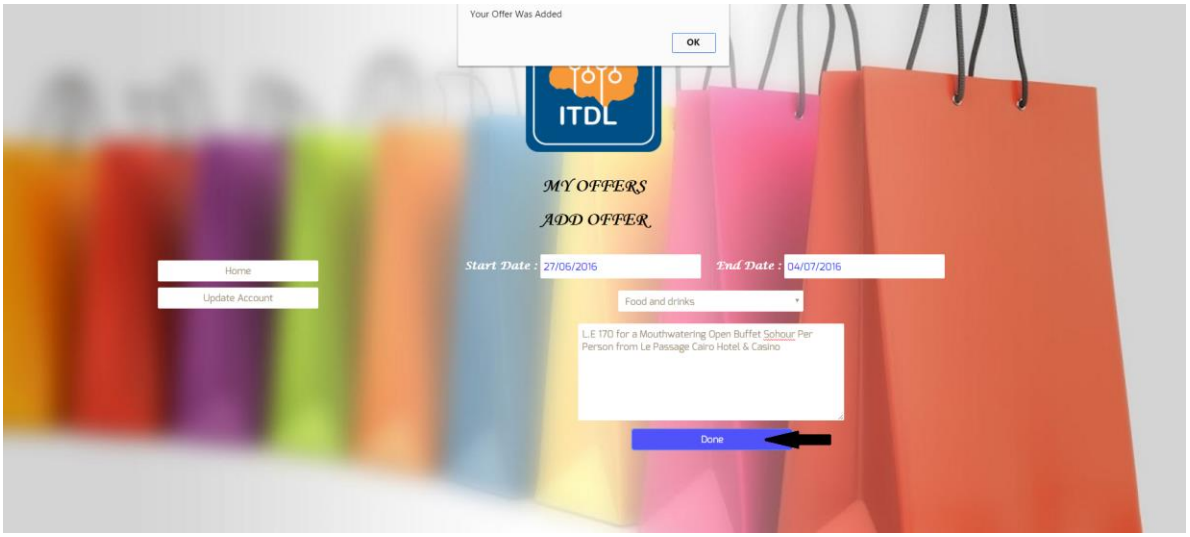
4.1.11 Get Update Preferences Notification



4.1.12 Get Suggested Actions Notification



4.2 WEB SCEENS (ADD OFFER)



Chapter 5

5.1 Conclusion

To sum up our project, Intelligent to do list help user to organize his tasks in intelligent way. Trying to understand the user behavior by analyzing the text from input sources like user daily notes, tweets or Facebook.

Each note type is handled in different way

Deadline note: user is notified multiple times before the deadline and visualize the progress percentage to encourage him finish his work.

Meeting note: system notified the user before the meeting by 1 day and by the transport time

Shopping note: user can organize his products he wants to buy and the system recommend him some stores that he may handle some shopping notes

Ordinary note: user can write any notes that are not in the previous categories. System make use from these notes by detecting the user habits and suggest them some actions to do.

ITDL application always updates the user interests weekly and inform the user by these updates in order not to enforce these updates and according to his action system will update his initial weights or not.

5.2 Future work

This application can be enhanced by adding some features and enhancing the recommendation engine.

- We can add sharing feature in the meeting and deadline notes. So user can make network of friends so we can use collaborative filtering recommendation algorithm. Lenskit tool provide implementation for these algorithms. This is an open source project which can be embedded in our application.
- Implementing web version of the application and synchronize data between the mobile and the web.
- Support Arabic language.

References

https://dev.twitter.com/

http://www.alchemyapi.com/

https://www.uclassify.com/

http://developer.android.com/index.html

http://www.tutorialspoint.com/android/android_tutorial.pdf

http://www.oodesign.com/

https://www.youtube.com/watch?v=s7wmiS2mSXY

http://lenskit.org/

https://en.wikipedia.org/wiki/Analytic_hierarchy_process