

SMALL TASKS AT HAND- INCREMENT 4

Ayinala kausik (Class ID: 4)

Yaswanth Bonda (Class ID: 6)

Tharkin Vankayala (Class ID: 34)

Ravi Teja Yakkala (Class ID: 38)

OBJECTIVES:

- Build the system for many to many user.
- Improve the location service in it.

Existing Services/API:

Google MAPS API:

In this particular increment we have introduced the google maps API so that our application will be user friendly as follows, It gives an option for the user/employer to point the location to where he can locate the address that he is posting.

```
Example: <meta-data
    android:name="com.google.android.geo.API_KEY"
    android:value="API_KEY"/>
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
    android:name="com.google.android.gms.maps.MapFragment"/>
```

MongoLAB API:

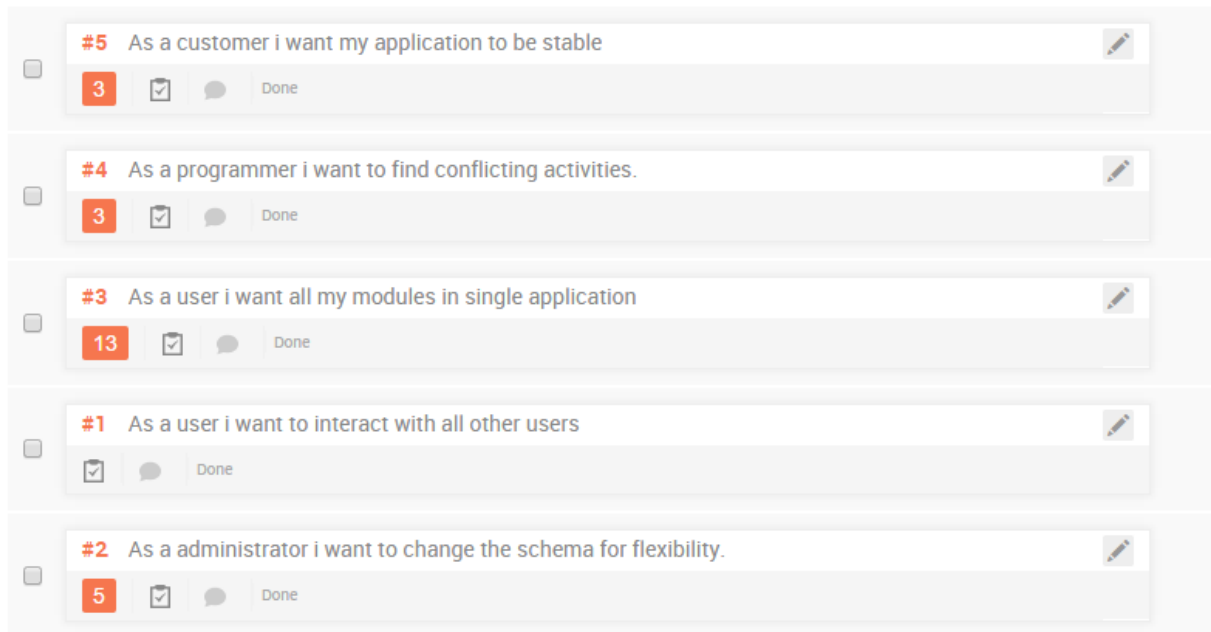
The other API that we have used in the fourth increment is the mongoLAB API by which we will have an access to the mongoDB.

Example: <https://api.mongolab.com/api/1/databases?apiKey=mykey>

Detail design of Services:

User Stories:

So far we developed the job searching and applying and accepting for tasks. In the later section we will integrate those parts. The picture below shows the stories and progress of our project at the end of the fourth iteration.



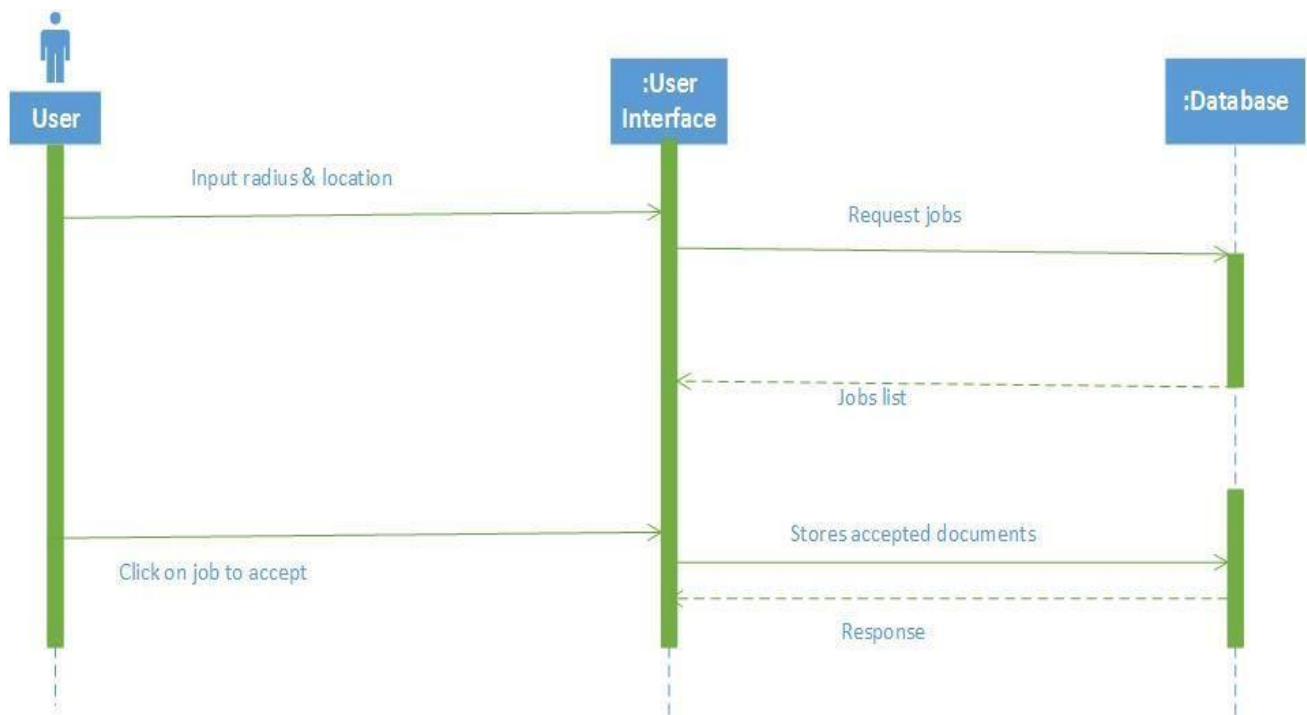
Service Description:

Three services are developed in this increment.

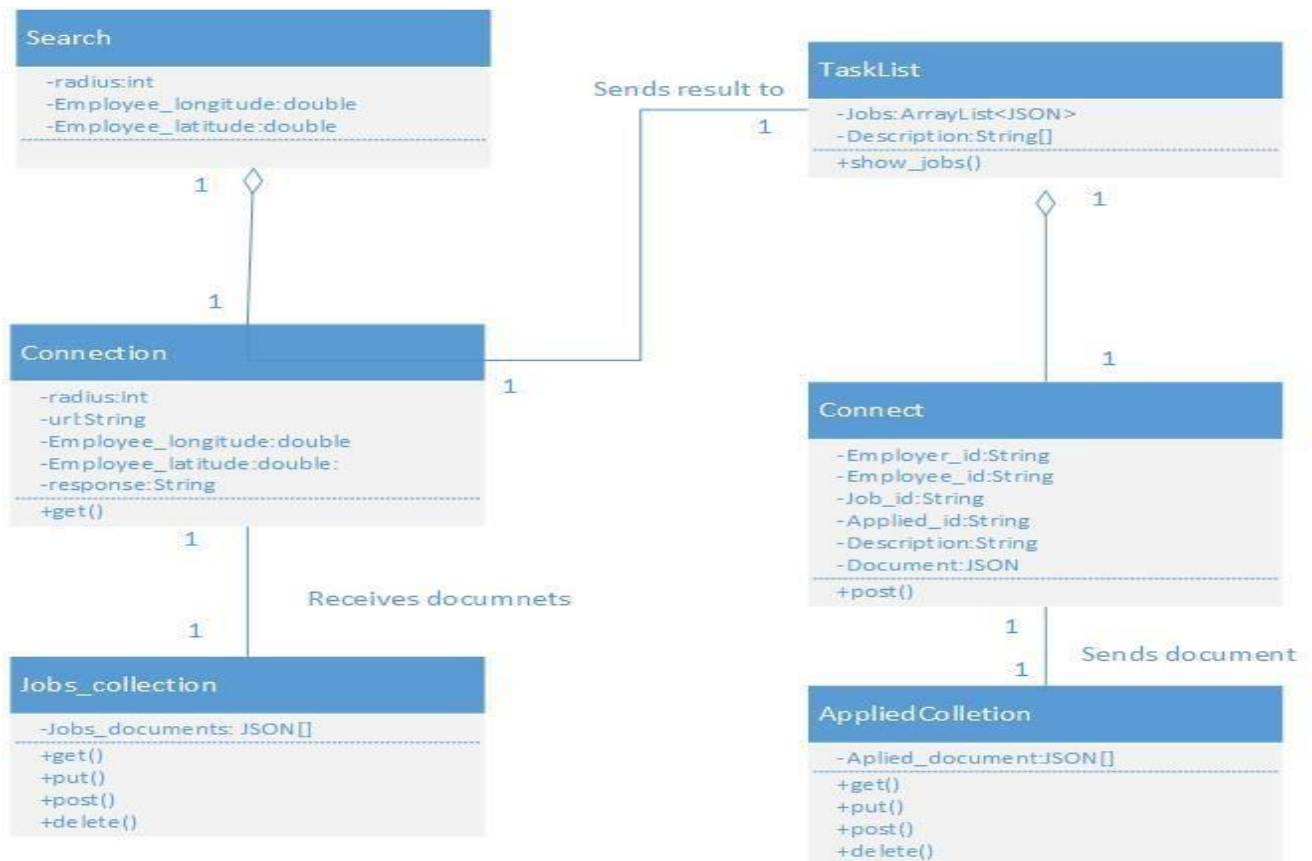
1. Searching for tasks and applying for available tasks:

We have improved this service in this module which we did earlier. With the help of this service the employee can search for the available tasks with in the given radius. The radius is given as the input by the user that too in meters. So by clicking the search button that page will be redirected to the new page which consists the list of tasks that are available in that radius so that he can apply. Mongo DB will act as the catalyst for retrieving the tasks from the job collections. This service is a sub part of employee module. Tasks list will be shown to the user, user can apply for the task by clicking on the task. So the list of tasks will be displayed to the user and he can accept that by clicking on the accept task button.

Sequence diagram:



Class diagram:



Mobile client interface:

Location and radius are taken as the input parameters for this searching the job service. The radius will be given as the input in EditText and location can be retrieved with the help of the search button.

search task

Radius: 1 miles

☒ use my location

☐ select location from maps

SEARCH

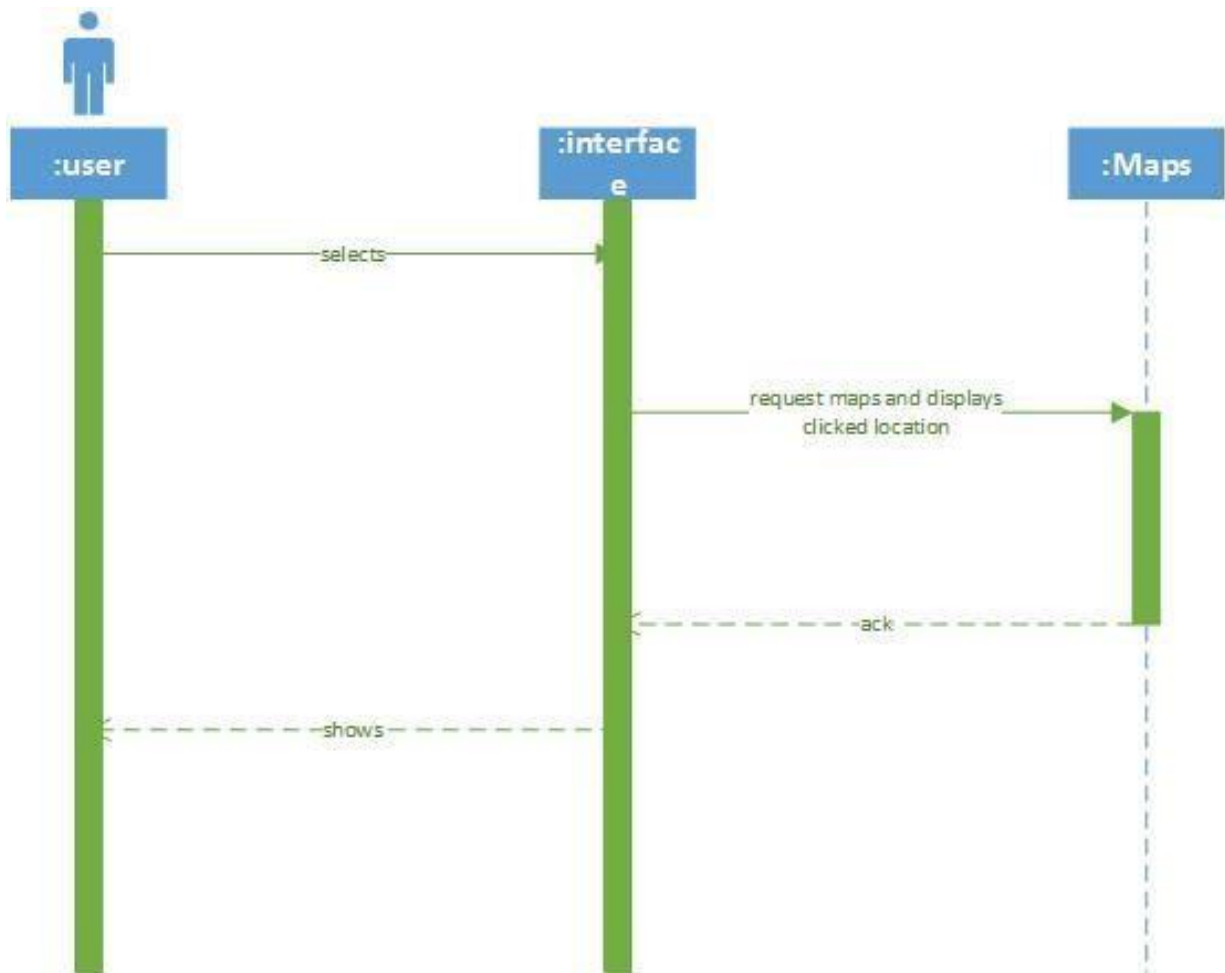
DESIGN OF TEST CASES:

1. The task's location should be stored in the JSON format in the document so that it can use the geospatial feature of MongoDB. MongoDB shell will test this case.
2. The working of our tasks retrieving query is checked by MongoDB shell.
3. REST API provided by mongolab is tested whether it is working or not. We should also test whether the documents are storing in the applied collection after an employee has applied for the task.
4. The order retrieving tasks and the order of posting of tasks both should be equal. Postman chrome extension and MongoDB shell are used to test this case.

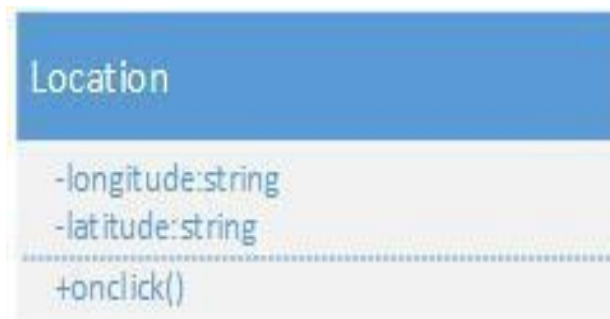
2. Location Service:

Our application has a service of selecting a location to post the job or task by the employer, so we used to do this by using a location provided by the network provider automatically by retrieving from the current location from the device where he is posting the task. Later we got a scenario where a user wants to use our application and needs to post a task from a place different from the place which he is posting the task right now. This case leads us to think in new ways to make our application more user friendly from the employer side. So, we clearly examined different ways to retrieve the location or even to select a particular location from a map or even to take it automatically from the network provider or current location using a GPS from the existing device. Finally, we came up with an option of selecting a particular needed location by the employer where he needs the task location, i.e. we will iterate through all the providers of the location and use the best case in our application to select any location where the employer wants to post the task at.

Sequence diagram:



Class diagram:



Mobile client interface:

This will not have an interface because this will be used in other functionalities of the application.

DESIGN OF TEST CASES:

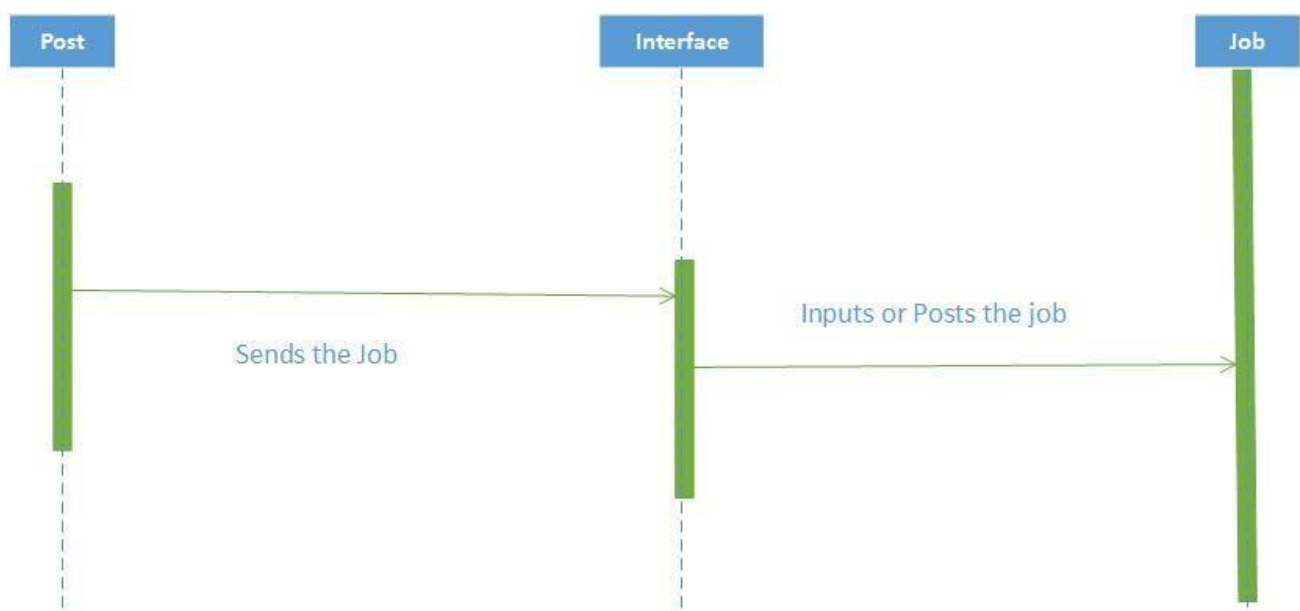
1. Maps should return the co-ordinates of the location where user clicks, is it accurate or not?

3. Posting the Task Information:

In this module we have improved the service which we had earlier as a one to one user interface like one employer and one employee. Earlier the employer will post the task whatever he needs at the location from where he is using our application and that current location would be retrieved from that device. Whenever he posts the task it is visible to the employee. But ultimately all this will be good to use when we have multiple users, so we needed to extend the users on both sides of employers and employee. Thus we extended the service in such a way that a user for example an employer can post his task in our application and it can be viewed by several users like employee who want to do the tasks for them and can search for tasks. So we were successful in extending the service properly from a one to one to many to many users.

Sequence Diagram:

Posting the job information:



Class Diagram:

Posting the job information:



Mobile client interface:

We will post the task like this in our application. An employee will get to post the task in this format while using our application.

The screenshot shows the 'post task' screen of a mobile application. The interface includes the following elements:

- Title:** A text input field containing 'pick up'.
- Description:** A text input field containing 'pick my friend from airport'.
- Estimated Time(hr):** A text input field containing '1'.
- Estimated Price(\$):** A text input field containing '25'.
- Location Selection:** Two radio buttons. The first is selected and labeled 'use my location'. The second is labeled 'select location from map'.
- Buttons:** A dark grey button labeled 'task posted' and a light grey button labeled 'POST TASK'.

The top of the screen shows a status bar with the time 7:27 and various icons. The bottom of the screen shows the Android navigation bar.

DESIGN OF TEST CASES:

1. A user like an employer can post the task when he needs to get it done, then we need to test it and see to that it is visible in the search for every user of employee to be retrieved for a specified location when it is in a given radius entered as an input by the employee.
2. Should check the post or task posted, whether it is posted properly to all the users or visible to all the valid users of our application for all the employee.

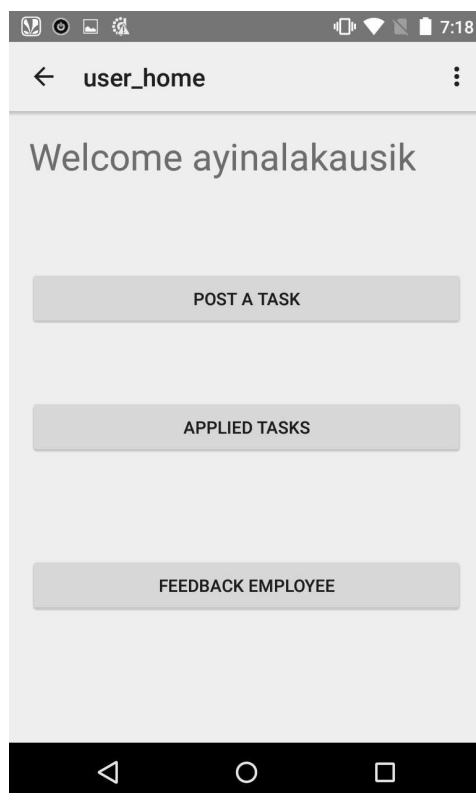
IMPLEMENTATION OF REST SERVICES:

In this iteration we did not get to use any REST services.

IMPLEMENTATION OF USER INTERFACE:

The flow of the posting and searching goes in this order where the employer has a task to post and get it done then he will get to use this part or option in our application.

It looks like this.

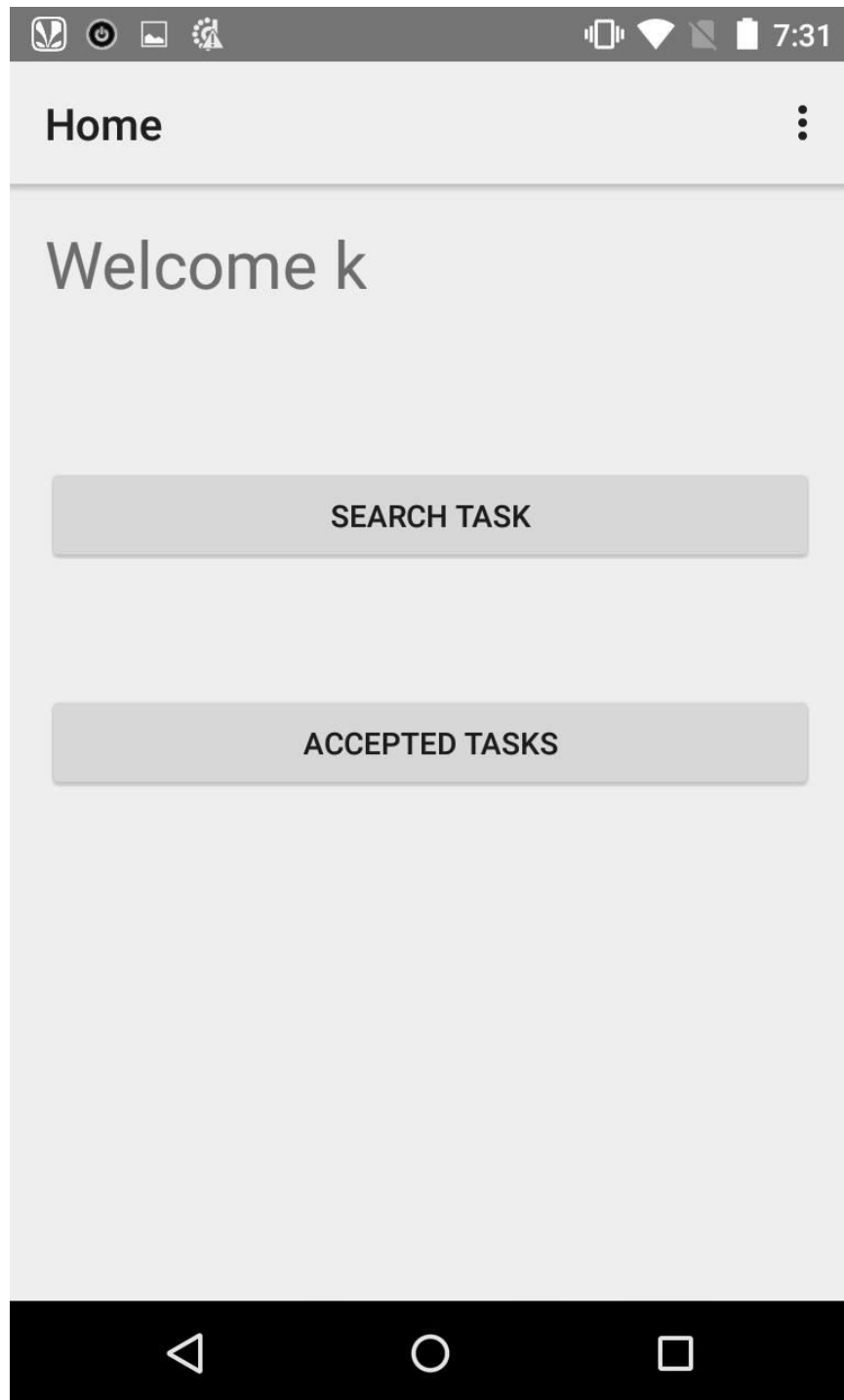


After this page if he clicks the option of “post a task” the he will be directed to this page.

The screenshot shows a mobile application interface for posting a task. The title bar at the top is labeled "post task" and includes a menu icon (three vertical dots) on the right. Below the title bar, there are four input fields: "Title" with the text "pick up", "Description" with the text "pick my friend from airport", "Estimated Time(hr):" with the value "1", and "Estimated Price(\$):" with the value "25". Below these fields are two radio buttons: "use my location" (which is selected, indicated by a filled green circle) and "select location from map" (which is unselected, indicated by an empty circle). At the bottom of the form is a button labeled "POST TASK". The entire form is set against a light gray background. The bottom of the screen shows the standard Android navigation bar with back, home, and recent apps icons.

Like this he can post his task.

Now when the jobs are posted, there can be many users of employee who want to search the task according to their convenience and select the task to do it.



This is the option where a user as an employee get to search for a task to do.

search task

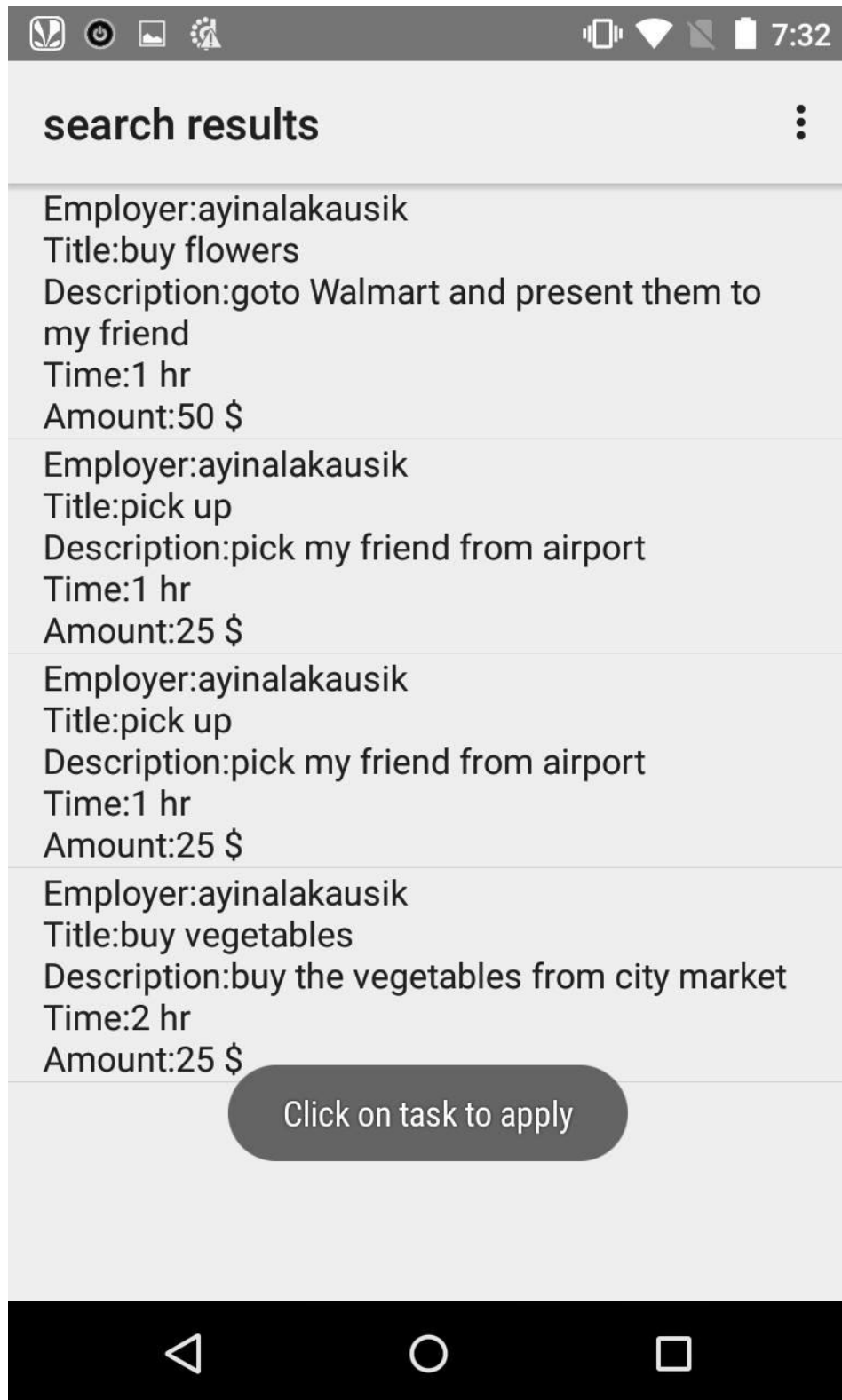
Radius: miles

☒ use my location

☐ select location from maps

SEARCH

Users get to enter their radius to search jobs in the needed locations according to the radius entered.



Then they will get the results as such.

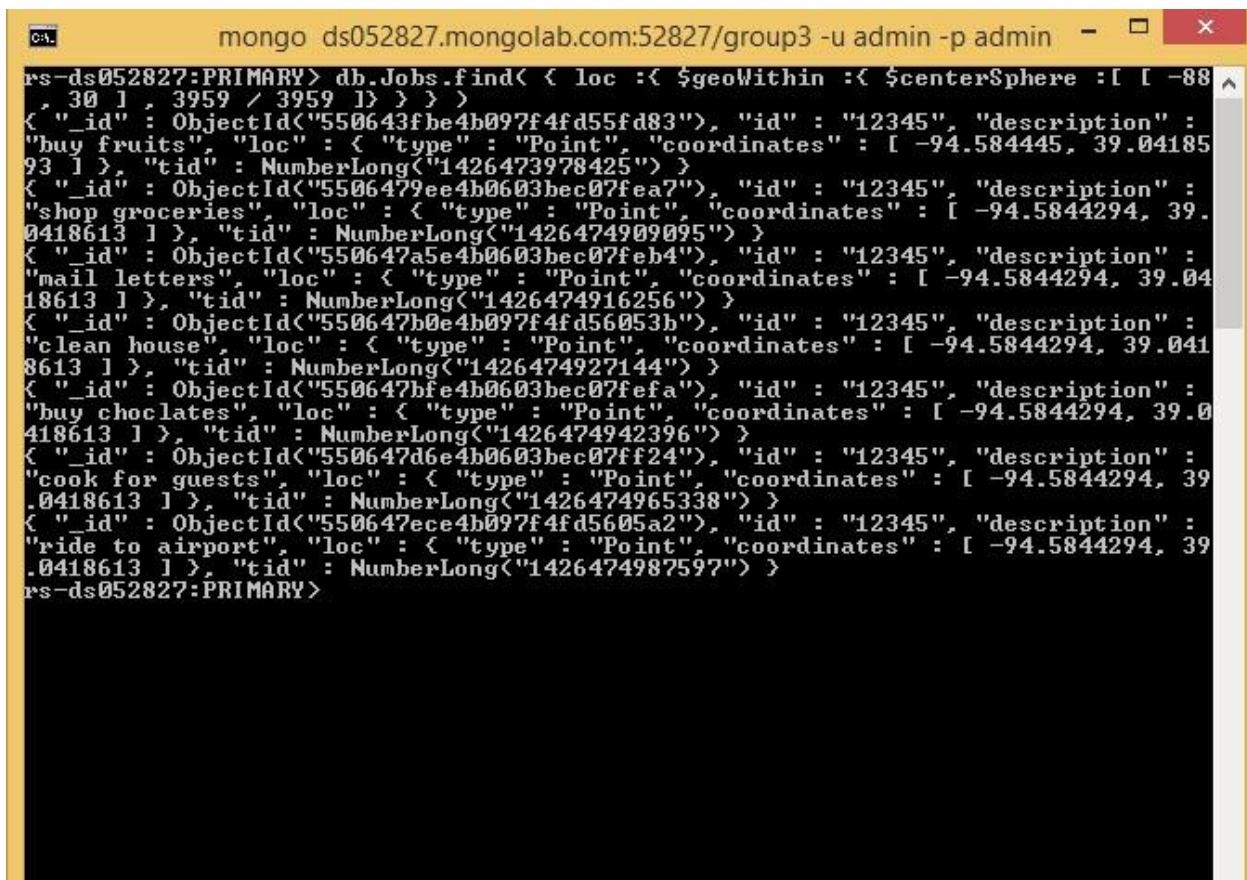
IMPLEMENTATION OF TEST CASES:

Here we get to do the various test cases like REST API were tested using postman chrome extension and functional testing is done using mongoDB shell. Detailed explanation was given in testing.

TESTING:

FUNCTIONAL TESTING:

1. Is our Tasks retrieving query (to find the tasks in given radius) is working or not. We tested by using mongoDB shell. We executed the query in mongoDB shell. In this query we found all the tasks on globe



```
mongo ds052827.mongolab.com:52827/group3 -u admin -p admin
rs-ds052827:PRIMARY> db.Jobs.find( { loc : { $geoWithin : { $centerSphere : [ [ -88
  30 1 , 3959 / 3959 1] ] } } }
{ "_id" : ObjectId("550643fbc4b097f4fd55fd83"), "id" : "12345", "description" :
"buy fruits", "loc" : { "type" : "Point", "coordinates" : [ -94.584445, 39.04185
93 1 ] }, "tid" : NumberLong("1426473978425") }
{ "_id" : ObjectId("5506479ee4b0603bec07fea7"), "id" : "12345", "description" :
"shop groceries", "loc" : { "type" : "Point", "coordinates" : [ -94.5844294, 39.
0418613 1 ] }, "tid" : NumberLong("1426474909095") }
{ "_id" : ObjectId("550647a5e4b0603bec07feb4"), "id" : "12345", "description" :
"mail letters", "loc" : { "type" : "Point", "coordinates" : [ -94.5844294, 39.04
18613 1 ] }, "tid" : NumberLong("1426474916256") }
{ "_id" : ObjectId("550647b0e4b097f4fd56053b"), "id" : "12345", "description" :
"clean house", "loc" : { "type" : "Point", "coordinates" : [ -94.5844294, 39.041
8613 1 ] }, "tid" : NumberLong("1426474927144") }
{ "_id" : ObjectId("550647bfe4b0603bec07fe4a"), "id" : "12345", "description" :
"buy chocolates", "loc" : { "type" : "Point", "coordinates" : [ -94.5844294, 39.0
418613 1 ] }, "tid" : NumberLong("1426474942396") }
{ "_id" : ObjectId("550647d6e4b0603bec07ff24"), "id" : "12345", "description" :
"cook for guests", "loc" : { "type" : "Point", "coordinates" : [ -94.5844294, 39
.0418613 1 ] }, "tid" : NumberLong("1426474965338") }
{ "_id" : ObjectId("550647ece4b097f4fd5605a2"), "id" : "12345", "description" :
"ride to airport", "loc" : { "type" : "Point", "coordinates" : [ -94.5844294, 39
.0418613 1 ] }, "tid" : NumberLong("1426474987597") }
rs-ds052827:PRIMARY>
```

2. Whoever applies for the task first his name should appear first in the employee applicant list. This is done by using mongoDB shell and Postman.
3. Check whether the tables are storing in the database or not. This is done by using MongoLab UI.

DEPLOYMENT:


Scrumdo link: <https://www.scrumdo.com/projects/project/smalltasksathand/iteration/124761>

Github link: <https://github.com/ayinalakaushik/SmallTasksAtHand>

REPORT:

In this iteration we have integrated all modules. We have changed the schema of the database as following.

The schema is as follows



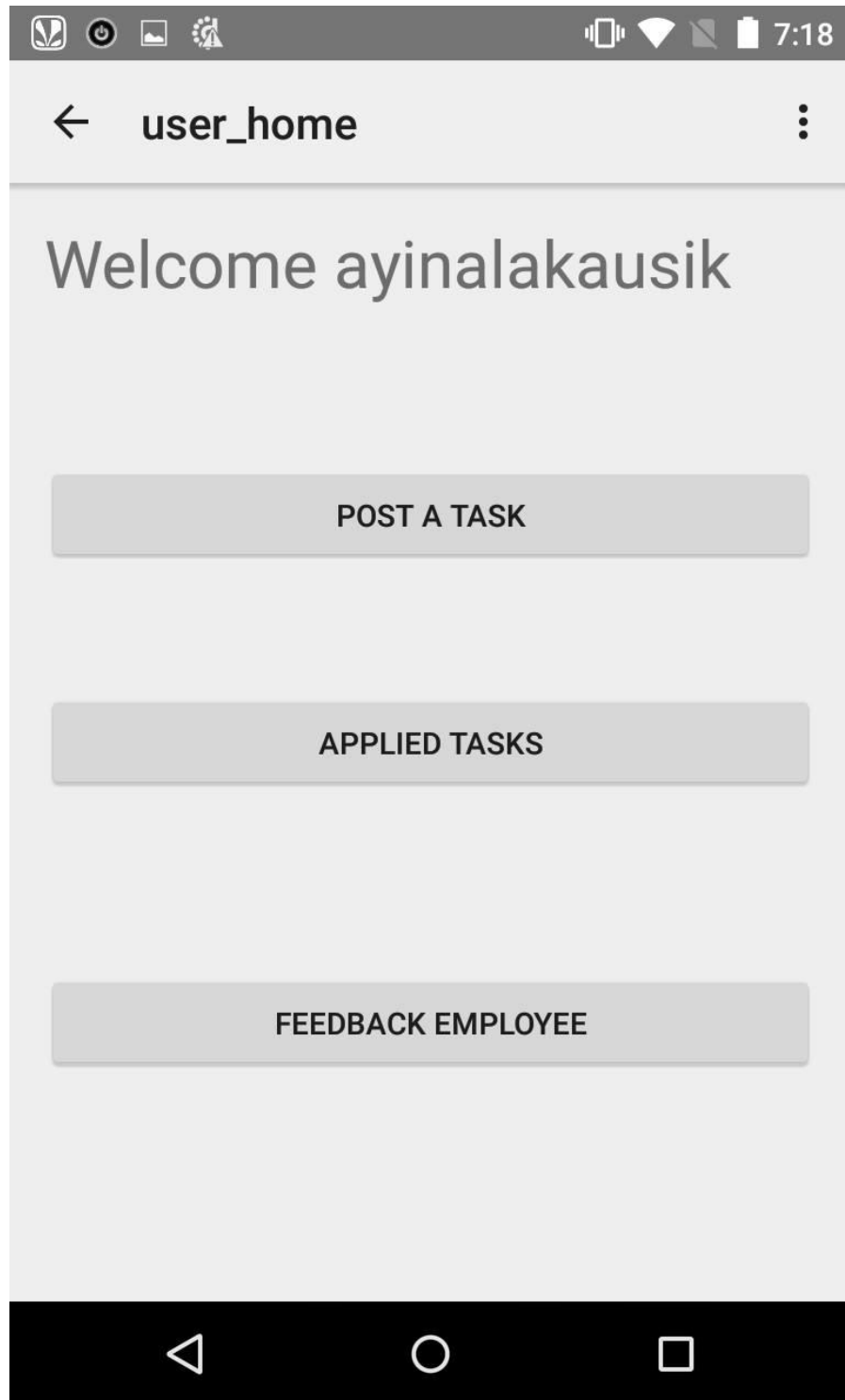
The screenshot shows the MongoLab UI with a table of collections. The table has columns for NAME, DOCUMENTS, CAPPED?, and SIZE. There are six collections listed: accept, applied, feedback, Jobs, recent, and users. Each collection has a corresponding 'X' button on the right.

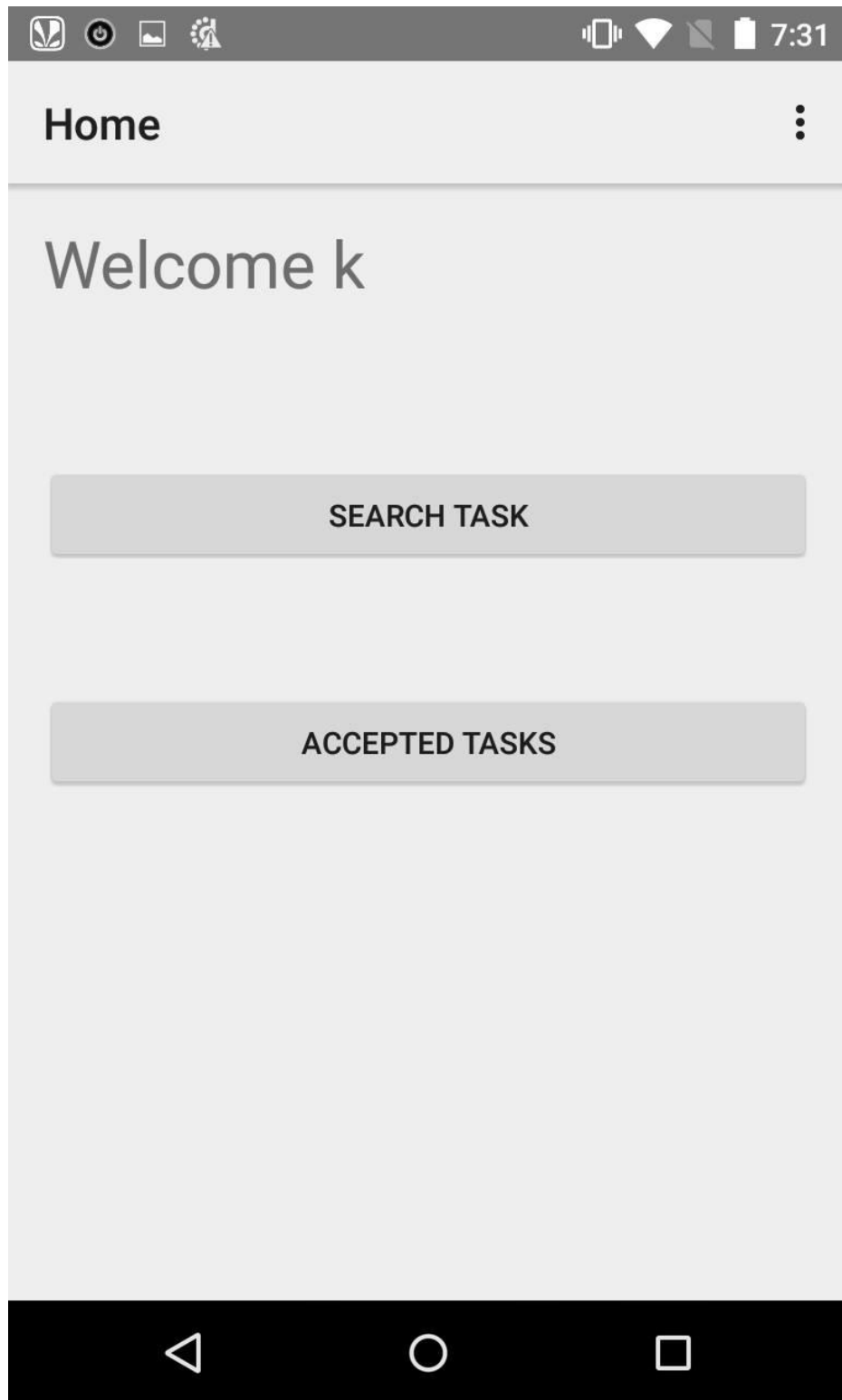
NAME	DOCUMENTS	CAPPED?	SIZE ?	
accept	1	false	8.09 KB	X
applied	3	false	8.69 KB	X
feedback	3	false	8.31 KB	X
Jobs	3	false	8.69 KB	X
recent	59	false	13.44 KB	X
users	7	false	9.75 KB	X

The main problem is we have given same names to classes in different modules. It caused ambiguity. So we have renamed all the files in modules.

As we developed login and register in third iteration it made our application one to one user so we have modified all our modules for many to many user. We have written some activities and home pages for users.

While registering user will say whether he is in employer or not. User will directed to home as he mentioned in registration.





We have written new activities for task applied as we have to display list of employee who applied for that task.



applied lists



buy flowers





7:35

employee applied



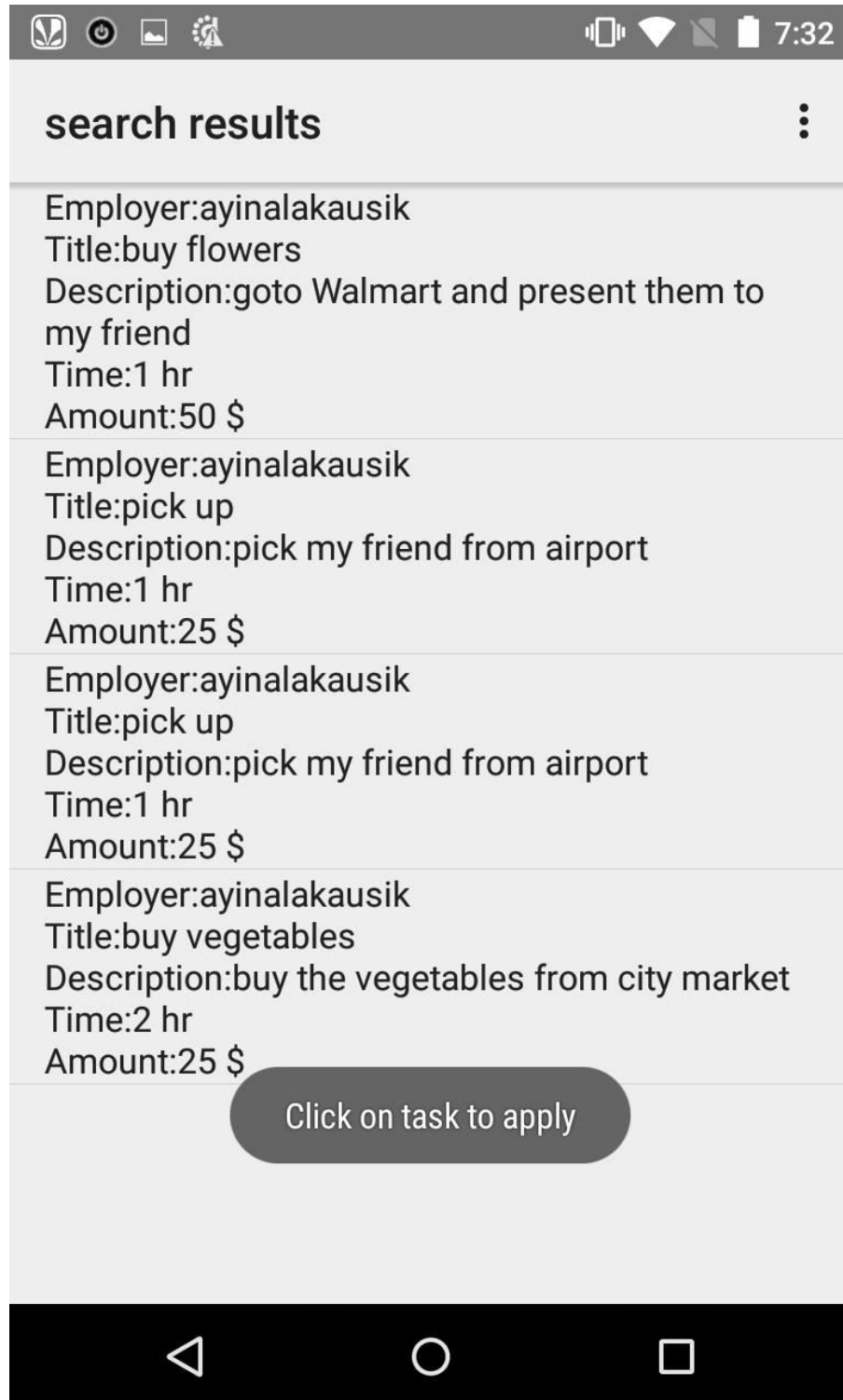
k

click on employee to accept him



We used google maps API in third iteration. For integration we started a new project and added files of previous but we have get new API key for maps.

We have added details and updated them in our UI, like employer name, task price in task result graph



We tried to implement payment service but we are unable to load paypal API.

Project Management:

Work completed:

1. Many to many user interactions.
2. Updating of services.
3. Integration of all services into two modules.

Kaushik - 34%

Yaswanth - 22%

Tharkin - 22%

Ravi Teja - 22%

First Service	Second Service	Third Service	Integration
	Kaushik – Implemented total logic (Time Taken: 6 hours)		Kaushik - Integrated employer module
Yaswanth - Developed the front end for this service, managed the activities with respect to the asynchronous tasks. (Time Taken: 3 hours)		Yaswanth - Developed the front end for this service, managed the activities with respect to the asynchronous tasks. (Time Taken: 2 hours)	Yaswanth - Integrated employer module
Tharkin - Did the testing for this service. (Time Taken: 2 hours)	Tharkin - Did the testing for this service. (Time Taken: 2 hours)	Tharkin - Did the testing for this service. (Time Taken: 2 hours)	Tharkin - Integrated employee module
Ravi Teja - Modified connection class to retrieve the results using GET for jobs collection and to store the jobs applied by POST. (Time Taken: 4 hours)	Ravi Teja - modified connection class to delete a document in collection, to get a document in applied, to post a document into the accepted. (Time Taken: 8 hours)	Ravi Teja - Modified connection class to retrieve the data from accept collection. (Time Taken: 2 hours)	Ravi Teja - Integrated employee module

Works to be Completed: unable to implement PayPal payment service.