

## **SMALL TASKS AT HAND- INCREAMENT 2**

Ayinala kausik (Class ID: 4)

Yaswanth Bonda (Class ID: 6)

Tharkin Vankayala (Class ID: 34)

Ravi Teja Yakkala (Class ID: 38)

## OBJECTIVES:

- Employee should be able to search for the tasks that are available within the given radius.
- After searching for the task he should be able to apply for the task.
- Employer should be able to view the applied list of tasks and he can accept the employee for that specific task.
- After accepting employee should be able to view the accepted list of tasks.

## Existing Services/API:

In the second increment we used the rest API's provided by the mongo labs to connect to mongoDB. These rest API's provides the services PUT, GET, POST and DELETE the data. Below is the rest API URL:

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

Where mykey is the API key provided by mongolab to our account.

In this increment we used the GET, POST and DELETE. To get the list of documents in collection which are there in the database we used the GET rest API by using HttpGet object. To create a document in the database we used the POST rest API by using HttpPost object. To delete a document in the database we used the DELETE rest API by using HttpDelete object.

## 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 second iteration.

|   |  |   |
|---|--|---|
| #6 As a tester i want to test the REST api of mongolab  | Done Tasks   0 Comments mongolab             | 3 |
| #5 As a programmer i want to use get,post,delete documents in our collections using mongolab REST api   | Done Tasks   0 Comments RestAPI Connectivity | 8 |
| #4 As a programmer i want to create front end in android for all the services   | Done Tasks   0 Comments frontend             | 5 |
| #3 As a programmer i want to write a query to retrieve the jobs in the given location   | Done Tasks   0 Comments mongodb query        | 8 |
| #1 As a administrator i want to store the documents in database<br>creating collections in the database. applied,accepted collections will be created in database | Done Tasks   0 Comments database             | 1 |
| #2 As a programmer i want to store the location as JSONArray in the documents   | Done Tasks   0 Comments database             | 5 |

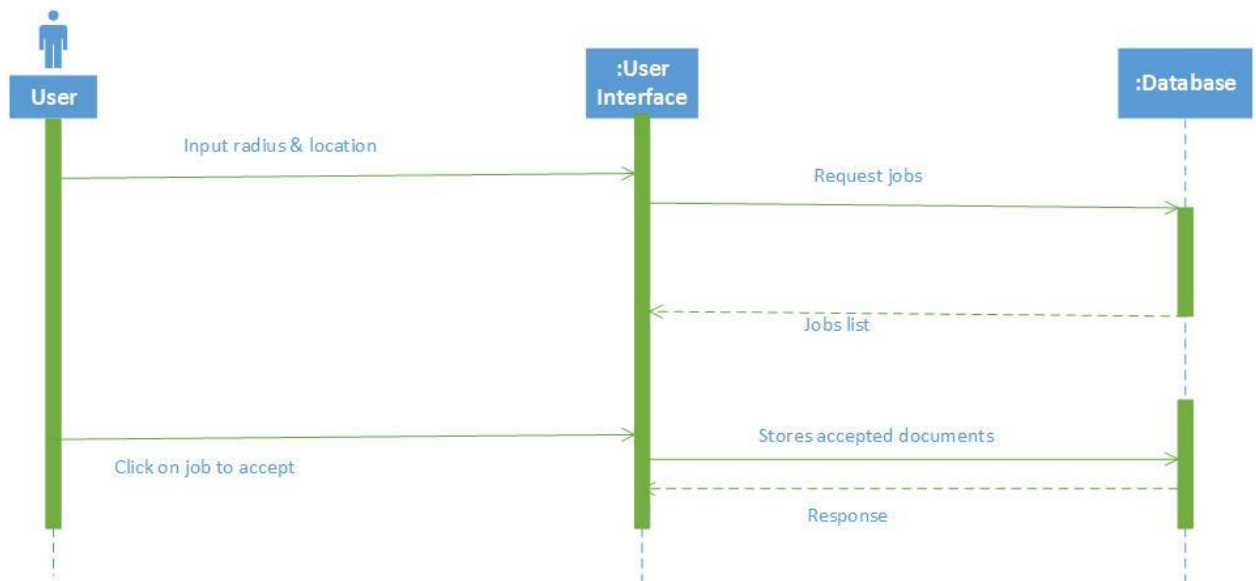
## Service Description:

We have developed three services in this increment.

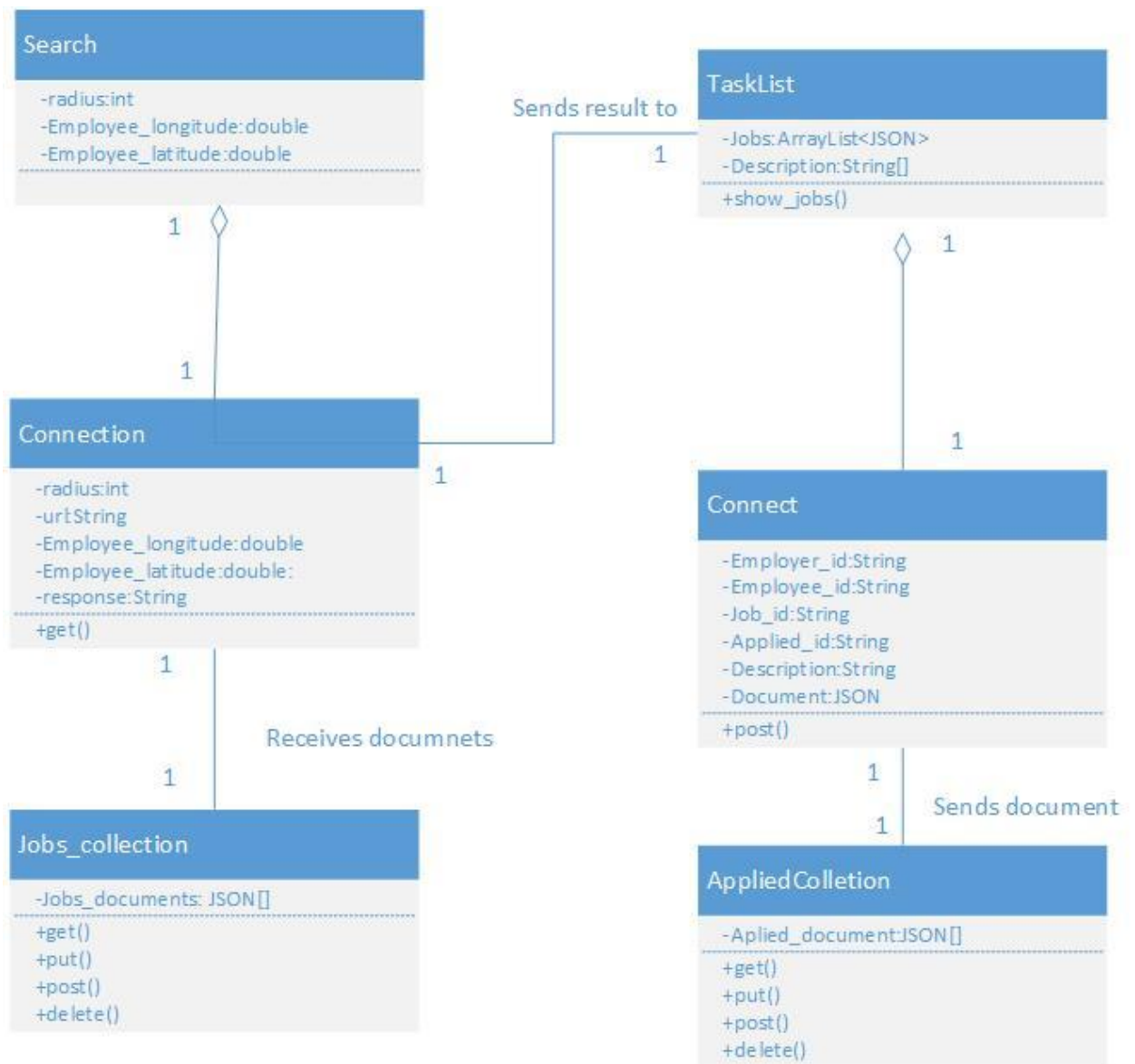
### 1. Searching for tasks and applying for available tasks:

In this service user (employee) can search the tasks that are available in the given radius. User will input the radius in meters in which he want to do the task, after pressing the search button user will be redirected to other activity which contains the list of tasks that are available in the given radius. The tasks are retrieved from the collection Jobs in mongoDB database. This service exists in employee module. Tasks list will be shown to the user, user can apply for the task by clicking on the task.

## Sequence diagram:

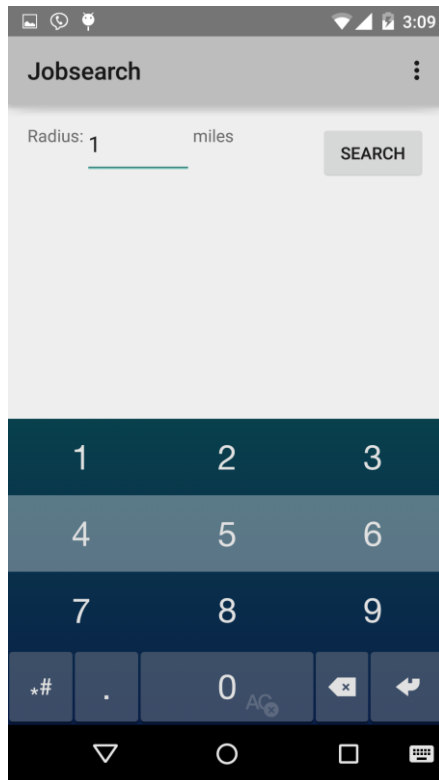


## Class diagram:

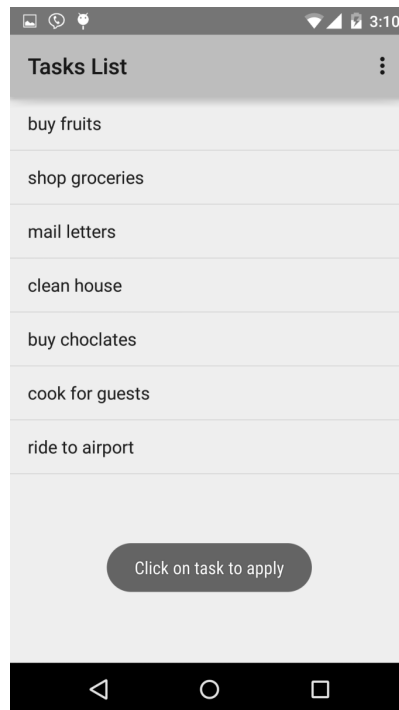


## Mobile client interface:

In this service we have to take input parameter radius and Location. User will input the radius in the EditText and location of the user will be retrieved on clicking the search button.



After retrieving the radius and location, the activity changes to shows the list of available jobs in given radius. We used ListView in android to show the list of jobs.



User can apply for the job by just clicking on the task.

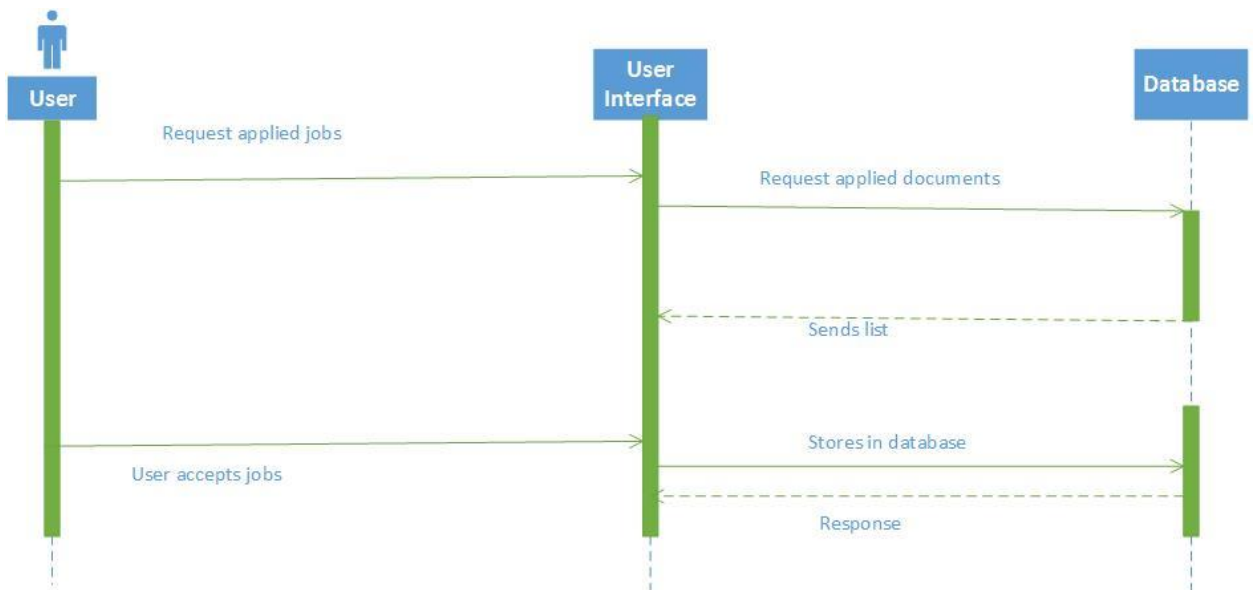
### Design of test cases:

- Location of the task should be stored in JSONArray format in document to use the geospatial feature of mongoDB. This case is tested by using mongoDB shell.
- Is our Tasks retrieving query (to find the tasks in given radius) is working or not. We tested by using mongoDB shell.
- Test whether REST API provided by mongolab or working or not. Test whether the documents are storing in our applied collection after applying for task.
- The order retrieving tasks should be equal to order of posting of tasks. We tested this case by using Postman chrome extension and mongoDB shell.

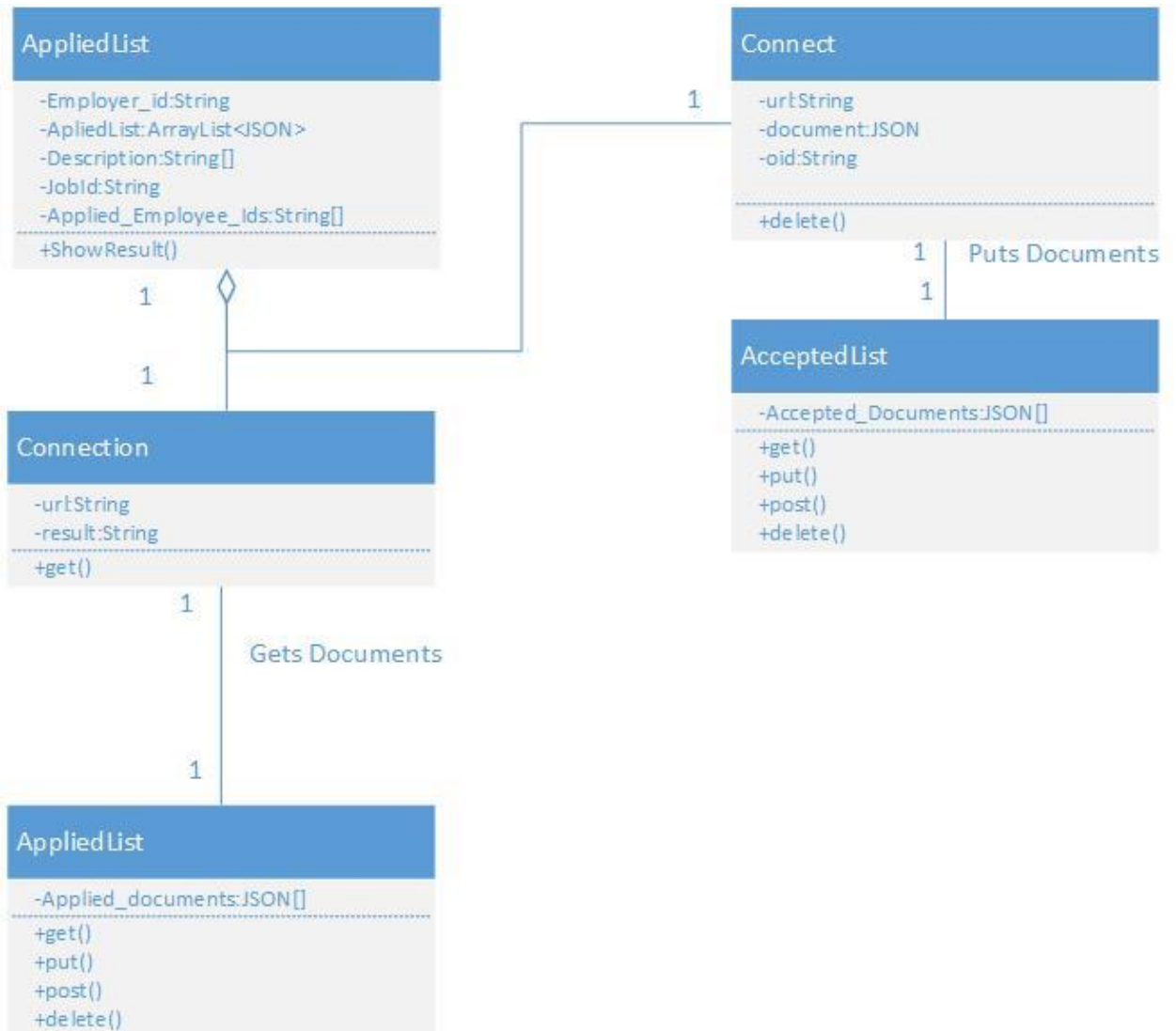
## 2. Viewing applied tasks and accepting the tasks:

After applying for a task by employee the employer should be able to see the applicants list. This service belongs to employer module. Employer can see the list of applicants in this service and employer can accept the applicant. Employer will redirected another activity which contains the list of applicants after pressing button. Employer can accept an application by just clicking on the name.

### Sequence diagram:

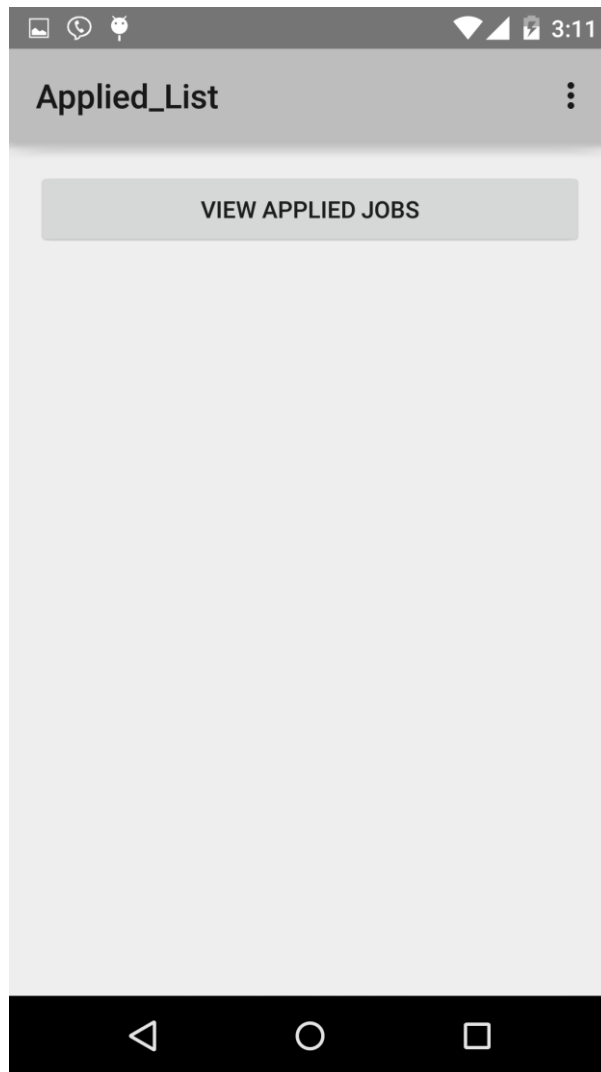


## Class diagram:



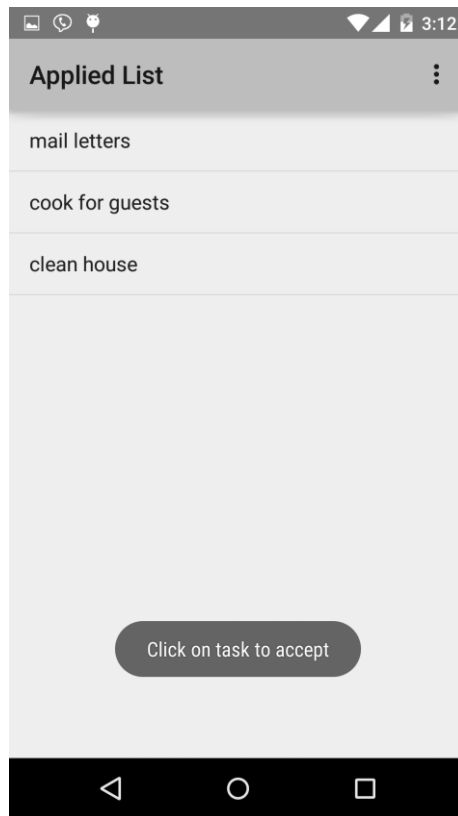
### Design of Mobile Client:

When employer login to his account he can see the button 'view application list'.



On pressing the button employer can view the list of applicants.





Employer can accept an applicant by pressing on the name

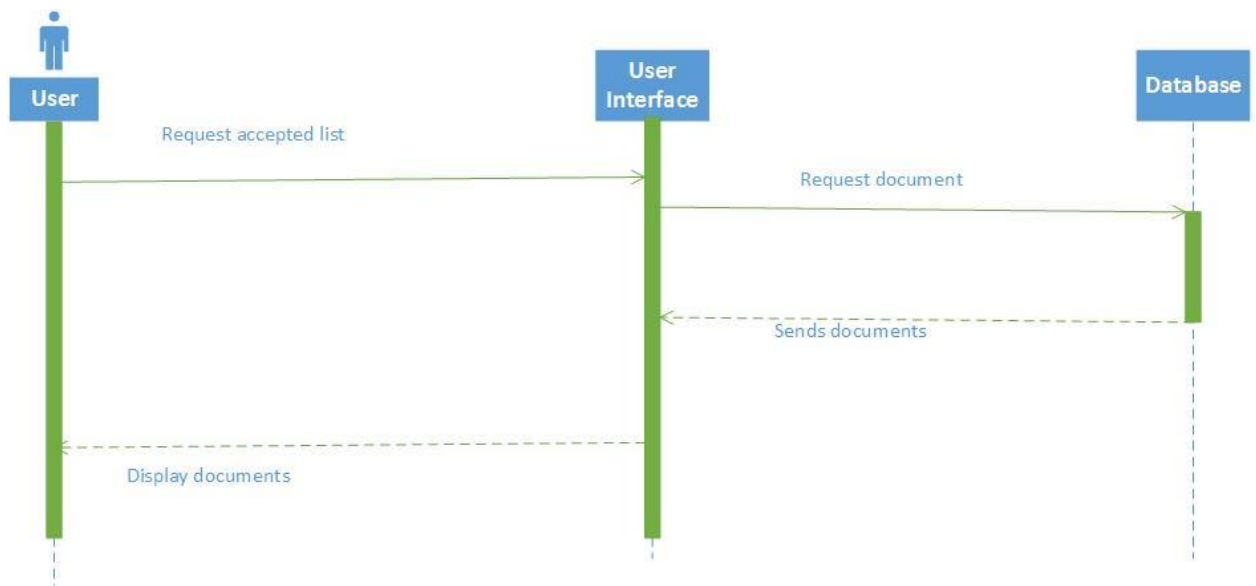
**Design of test cases:**

- Test whether the REST API services provided by the mongolab are working or not to post data into the applied. This is done by using Google chrome extension POSTMAN.
- 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. Viewing employee acceptancy:**

When employer accepts the application of the employee, employee should be able to see whether he was accepted or not. This service belongs to employee module. After logging employee will be able to use service. Employee can see the list of the accepted tasks in this service.

### Sequence diagram:

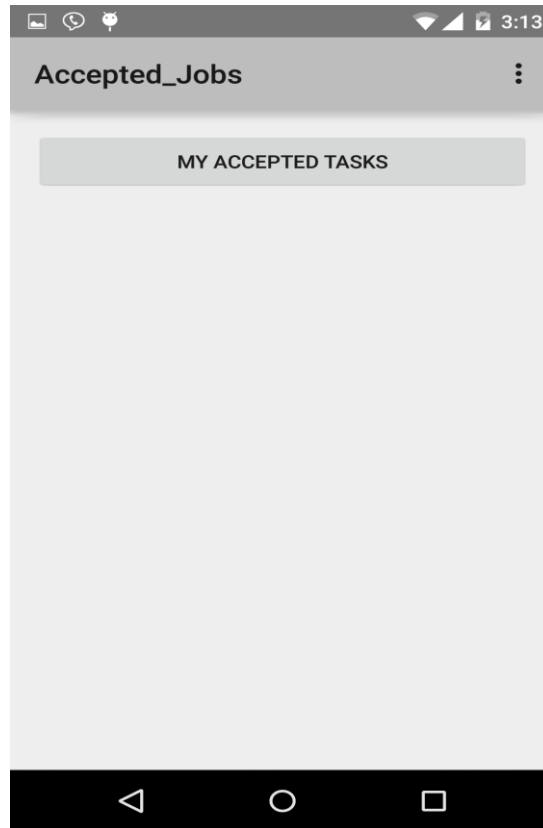


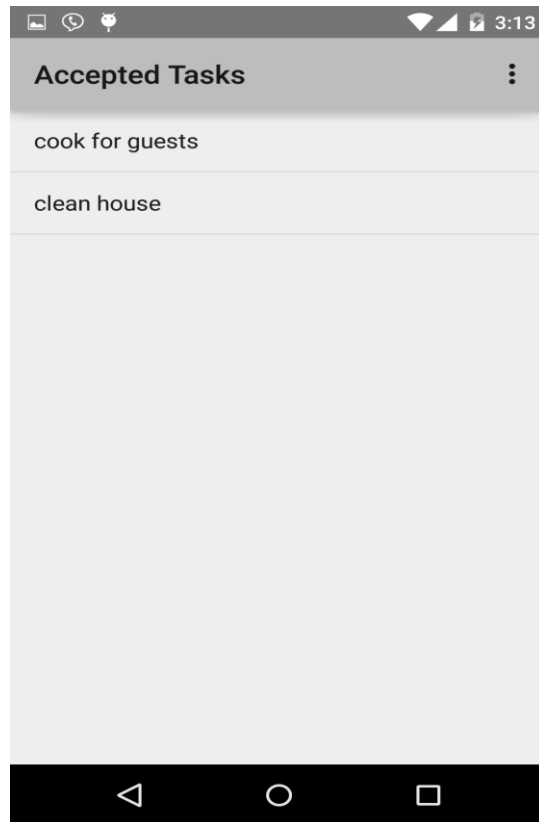
### Class diagram:



### Design of Mobile Client:

When employee presses the 'view accepted tasks' he will be redirected to another activity which contains the list of accepted tasks.





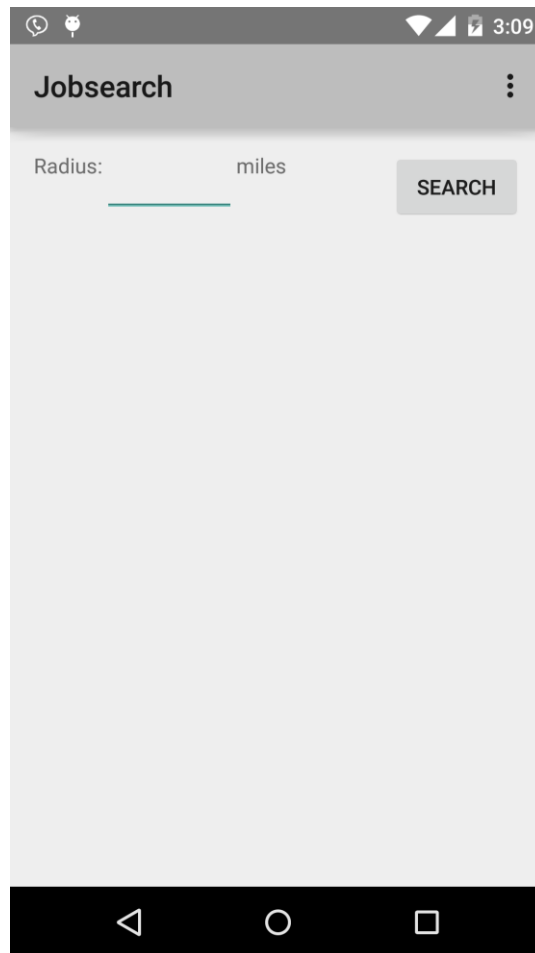
### **Design of test cases:**

Test whether REST API provided by mongolab are working to get data from accepted collection

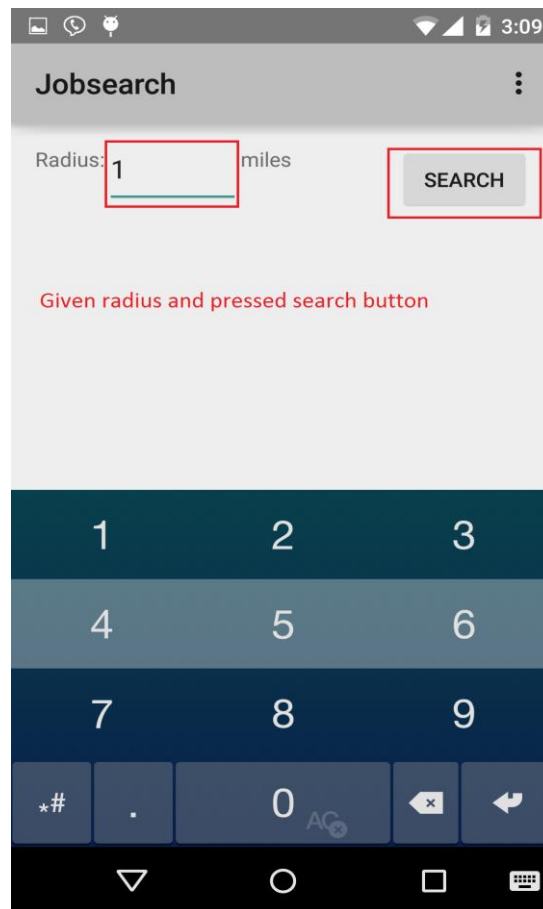
### **Implementation of user interface:**

All services will have user interface in android. In this service we developed three services and all are having user interface

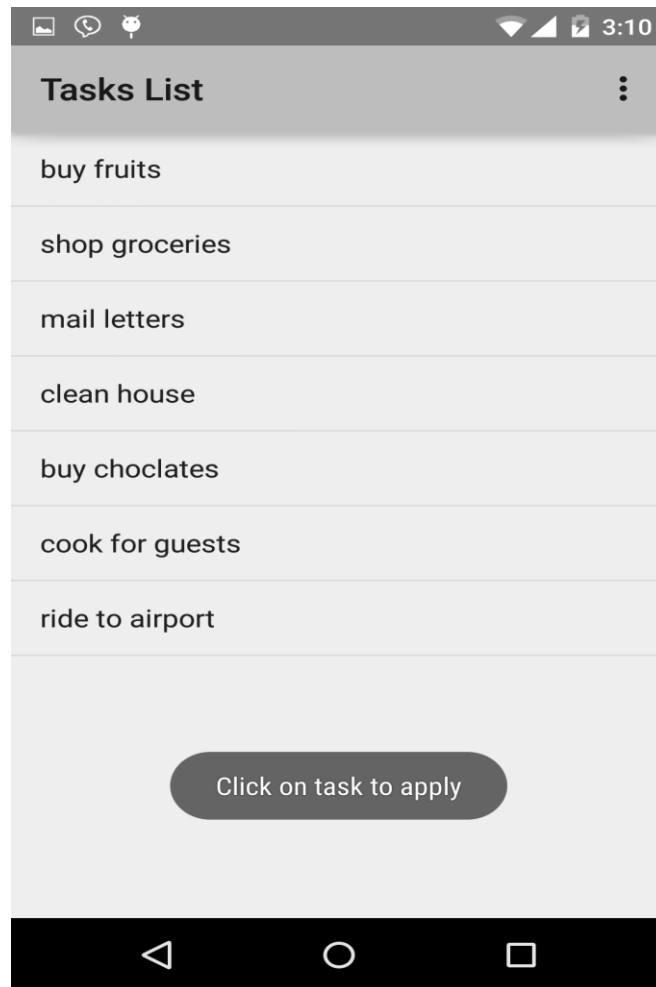
1. Job searching and applying: In this service there will be two activities. In first activity user can input the radius of the task that he want to take. Location of the user will be retrieved automatically when employee presses 'search'.



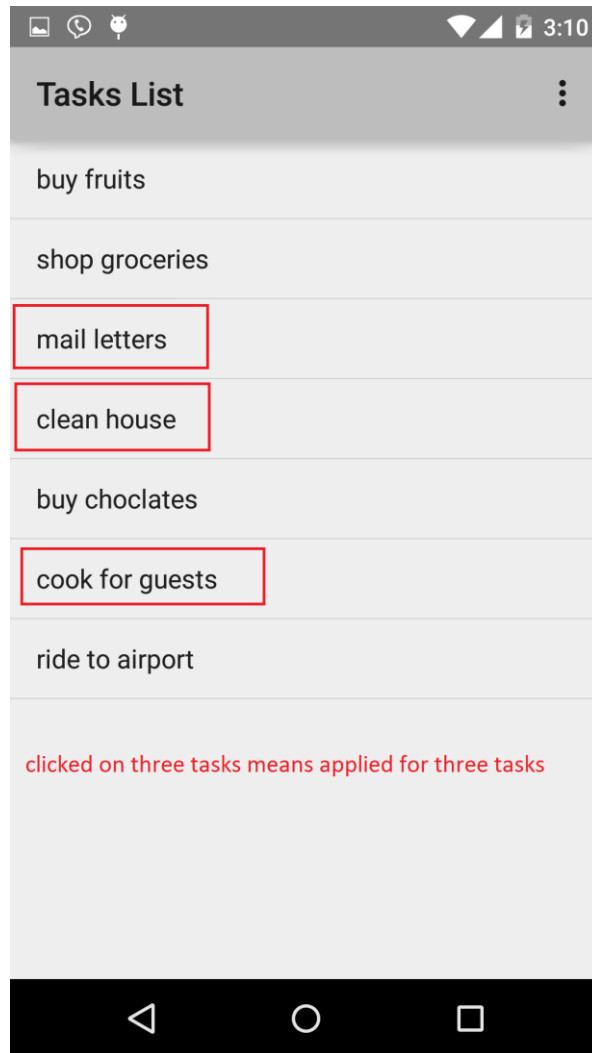
In EditText user can input the radius and he can press search button



Location and radius are passed to the connection class and connection class will pass the query to mongoDB gets the result from collection Jobs. After retrieving results only connection class will call new activity. The new activity will contain the list of available tasks. We used ListView to display the available tasks.

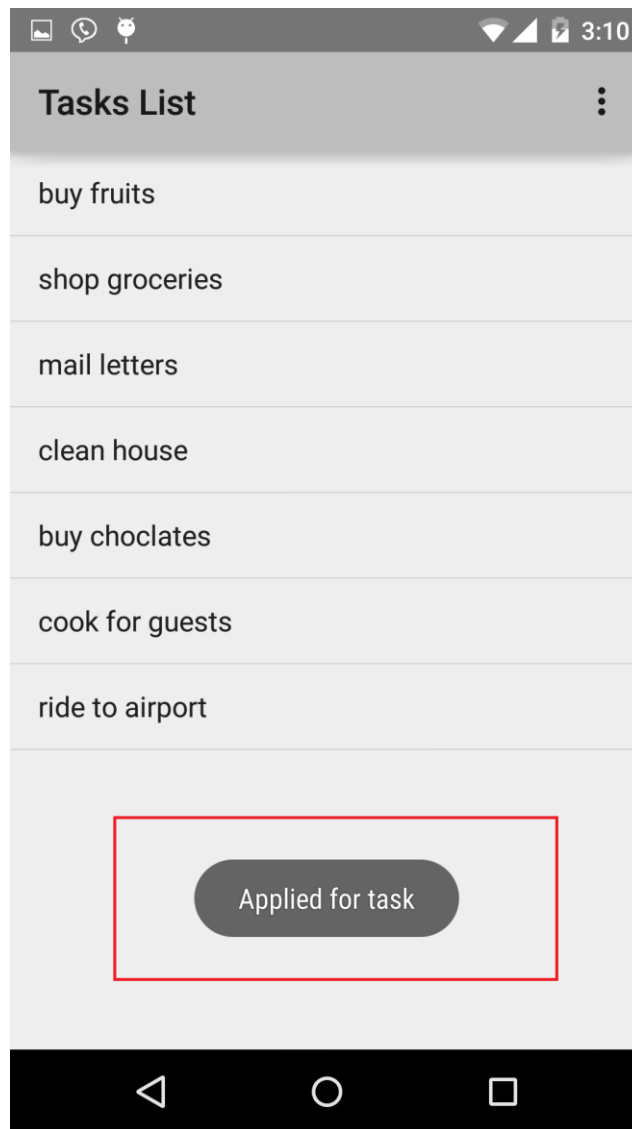


Employee can apply to a job by clicking on the task.



If applying for the task is success then we will get the message 'Applied for Task', otherwise we will not get any message

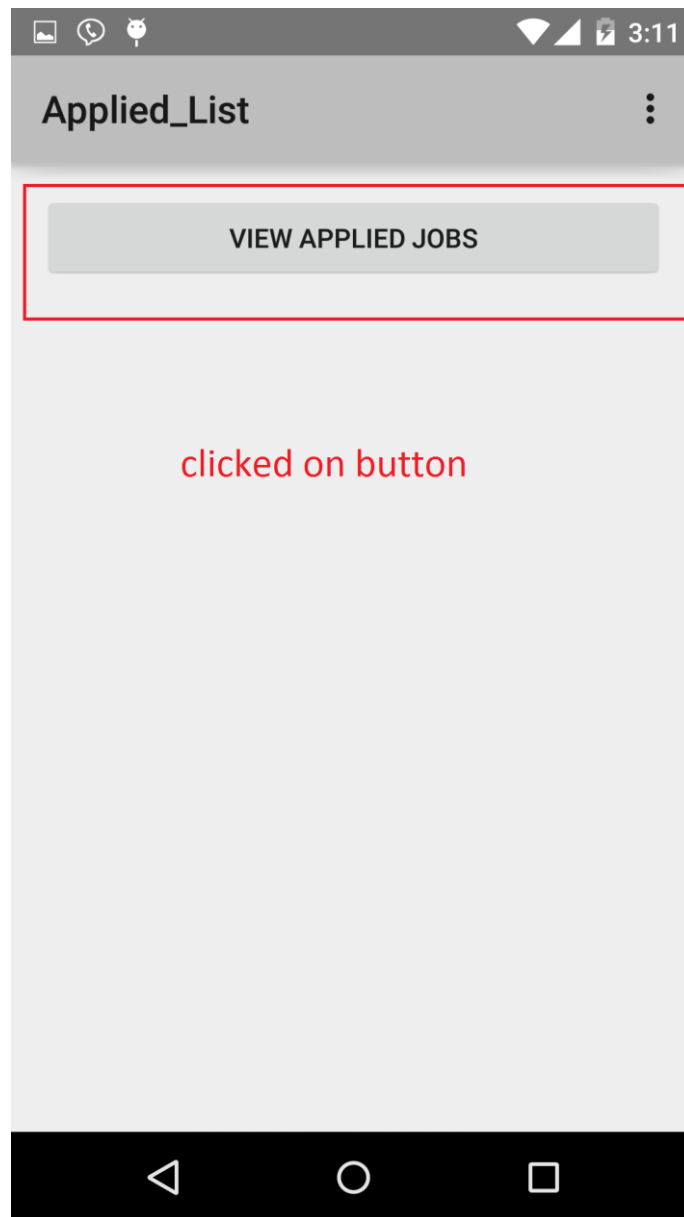




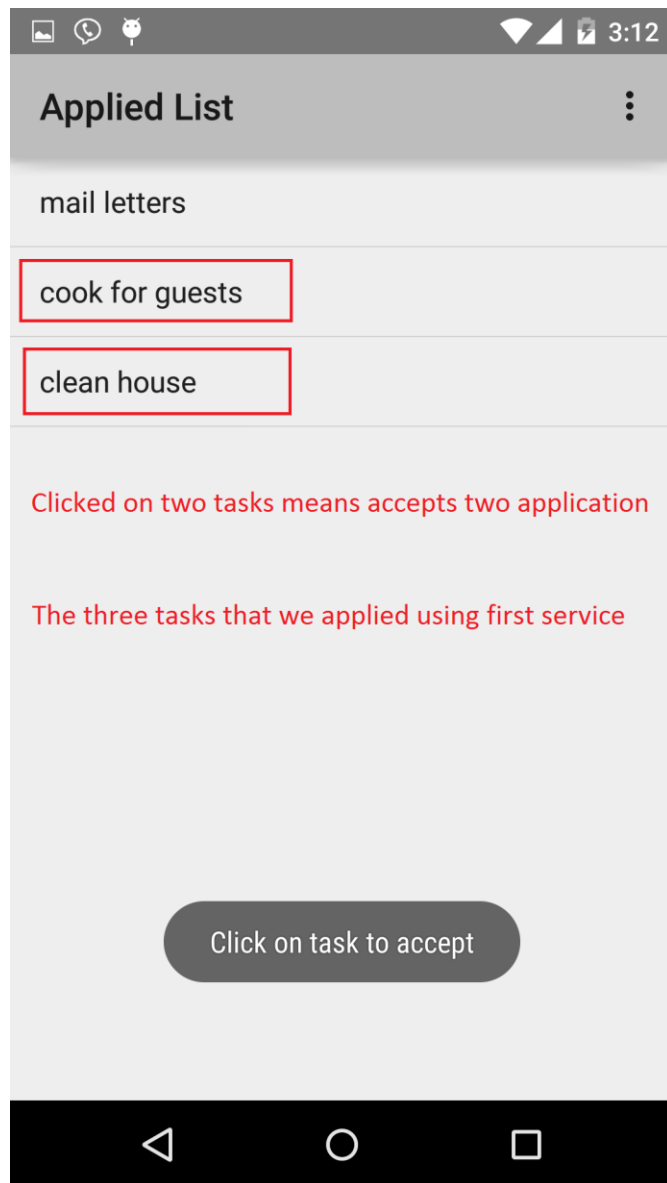
When user clicks on the task the task JSON object will be retrieved and it is stored in applied collection

## **2. Viewing applied tasks and accepting the tasks:**

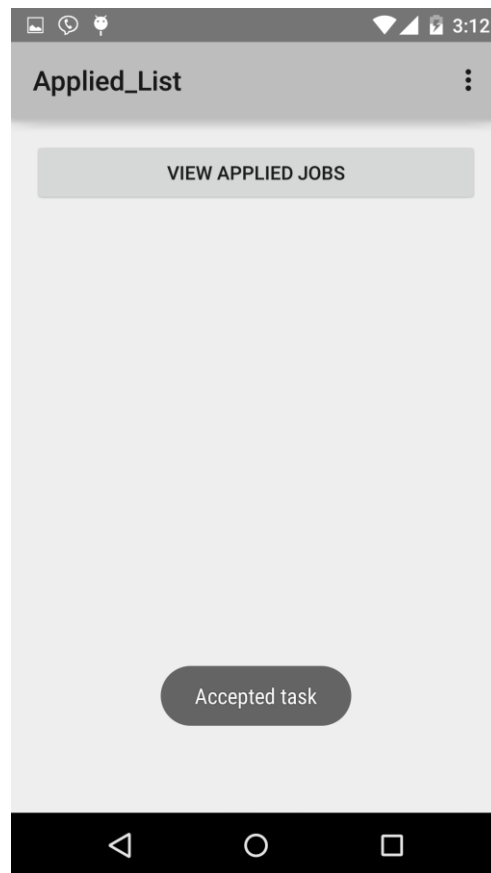
This service will have two activities. In first activity we will request the applied list of Jobs. When user presses the 'view applied list' the connection class will retrieve the applied from 'applied' collection. After retrieving the result only connection will start another activity that contains the list of applications.



Employer can accept a task by clicking on the task.

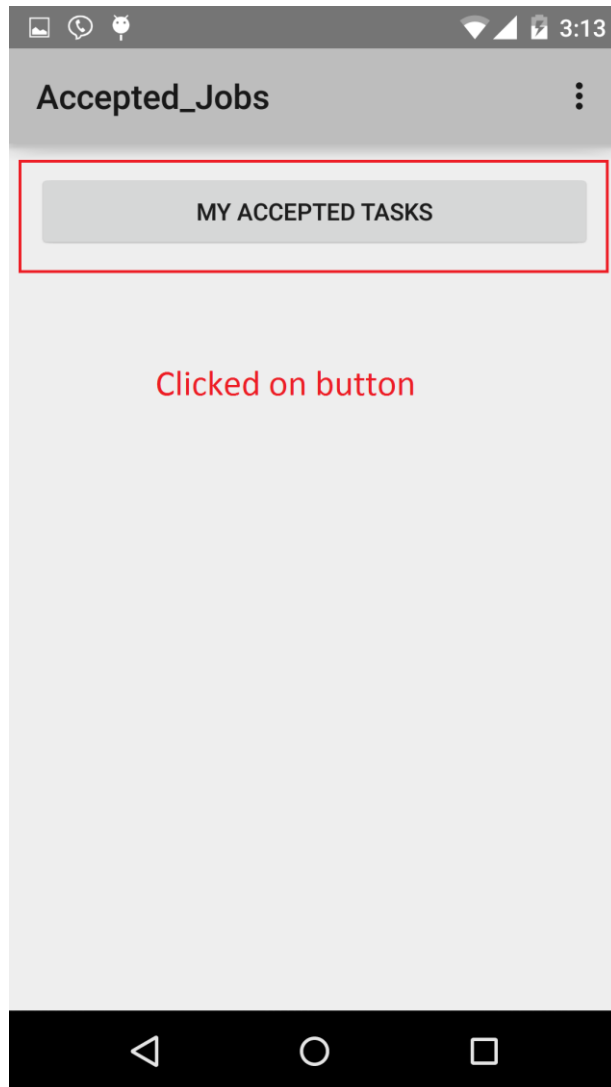


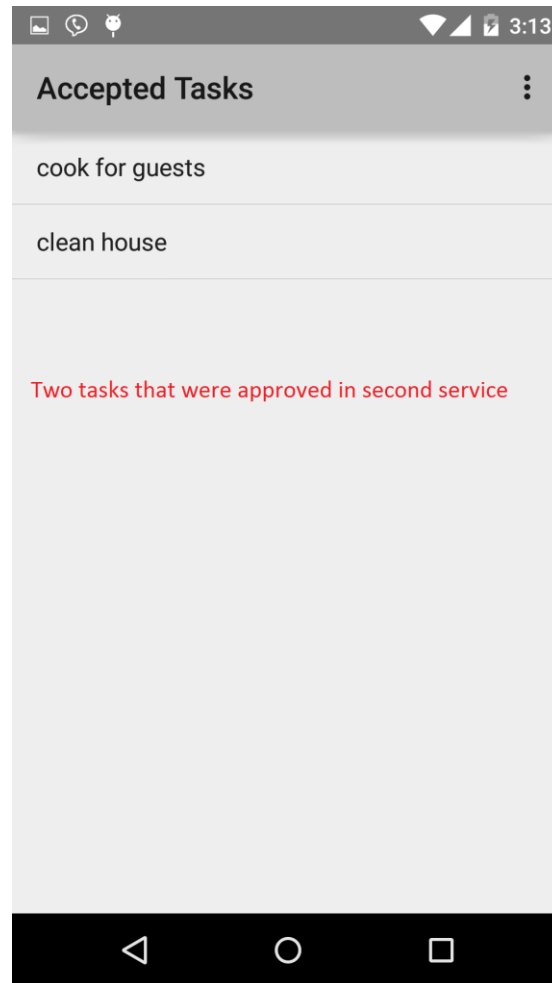
Employer will get a message that task was added. On clicking the application list the clicked item id is retrieved and its document is removed from the applied list collection and adds to accepted list.



3. **Viewing employee acceptance:**

This service contains two activities in one activity user will request the accepted list by clicking on button. Connection class will retrieve the data from accepted collection. After retrieving the data, connection class will call new activity in which we display the accepted list.



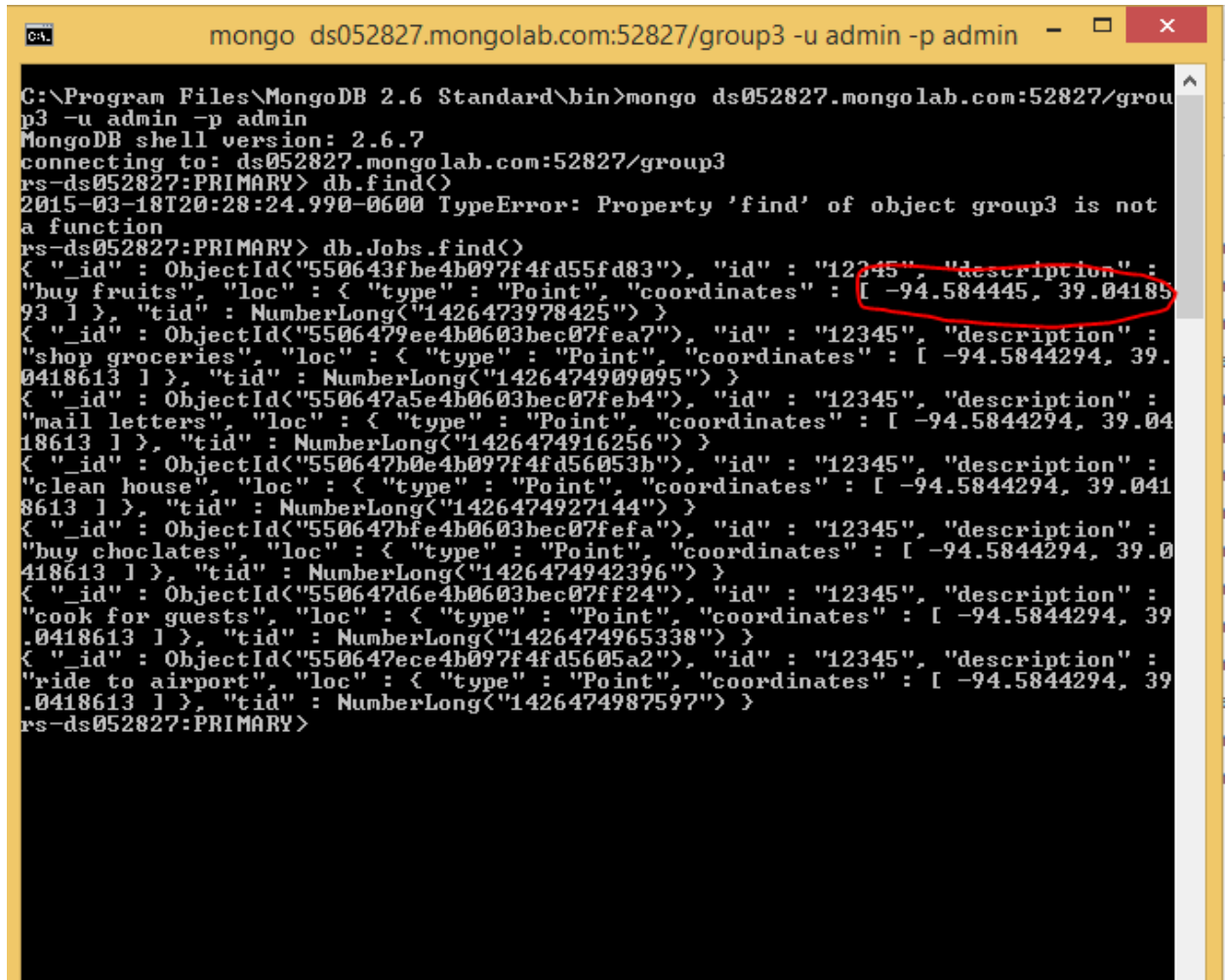


**Implementation of test cases:** REST API were tested using postman chrome extension and functional testing is done using mongoDB shell. Detailed explanation was given in testing

## Testing:

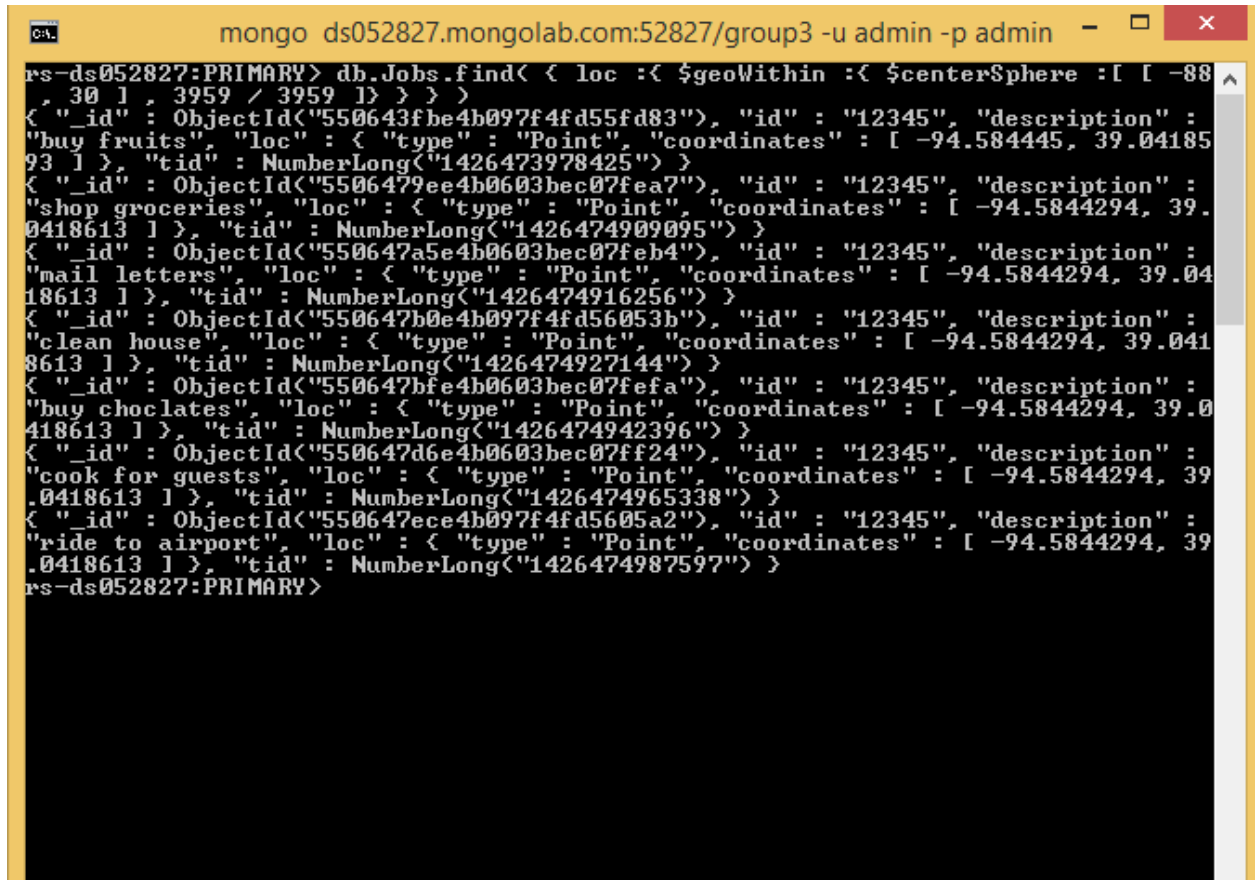
### Functionality testing:

1. Location of the task should be stored in JSONArray format in document to use the geospatial feature of mongoDB. This case is tested by using mongoDB shell. In mongoDB shell we viewed the Jobs collection in it. Location in document is stored as JSONArray.



```
C:\Program Files\MongoDB 2.6 Standard\bin>mongo ds052827.mongolab.com:52827/group3 -u admin -p admin
MongoDB shell version: 2.6.7
connecting to: ds052827.mongolab.com:52827/group3
rs-ds052827:PRIMARY> db.find()
2015-03-18T20:28:24.990-0600 TypeError: Property 'find' of object group3 is not a function
rs-ds052827:PRIMARY> db.Jobs.find()
{ "_id" : ObjectId<"550643fbe4b097f4fd55fd83">, "id" : "12345", "description" : "buy fruits", "loc" : { "type" : "Point", "coordinates" : [ -94.584445, 39.0418593 ] }, "tid" : NumberLong<"1426473978425"> }
{ "_id" : ObjectId<"5506479ee4b0603bec07fea7">, "id" : "12345", "description" : "shop groceries", "loc" : { "type" : "Point", "coordinates" : [ -94.5844294, 39.0418613 ] }, "tid" : NumberLong<"1426474909095"> }
{ "_id" : ObjectId<"550647a5e4b0603bec07feb4">, "id" : "12345", "description" : "mail letters", "loc" : { "type" : "Point", "coordinates" : [ -94.5844294, 39.0418613 ] }, "tid" : NumberLong<"1426474916256"> }
{ "_id" : ObjectId<"550647b0e4b097f4fd56053b">, "id" : "12345", "description" : "clean house", "loc" : { "type" : "Point", "coordinates" : [ -94.5844294, 39.0418613 ] }, "tid" : NumberLong<"1426474927144"> }
{ "_id" : ObjectId<"550647bfe4b0603bec07fefa">, "id" : "12345", "description" : "buy chocolates", "loc" : { "type" : "Point", "coordinates" : [ -94.5844294, 39.0418613 ] }, "tid" : NumberLong<"1426474942396"> }
{ "_id" : ObjectId<"550647d6e4b0603bec07ff24">, "id" : "12345", "description" : "cook for guests", "loc" : { "type" : "Point", "coordinates" : [ -94.5844294, 39.0418613 ] }, "tid" : NumberLong<"1426474965338"> }
{ "_id" : ObjectId<"550647ece4b097f4fd5605a2">, "id" : "12345", "description" : "ride to airport", "loc" : { "type" : "Point", "coordinates" : [ -94.5844294, 39.0418613 ] }, "tid" : NumberLong<"1426474987597"> }
rs-ds052827:PRIMARY>
```

2. 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



```
rs-ds052827:PRIMARY> db.Jobs.find( { loc : { $geoWithin : { $centerSphere : [ [ -88
, 30 ], 3959 / 3959 ] } } } )
{ "_id" : ObjectId("550643f4b097f4fd55fd83"), "id" : "12345", "description" :
"buy fruits", "loc" : { "type" : "Point", "coordinates" : [ -94.584445, 39.04185
93 ] }, "tid" : NumberLong("1426473978425") }
{ "_id" : ObjectId("5506479ee4b0603bec07fea7"), "id" : "12345", "description" :
"shop groceries", "loc" : { "type" : "Point", "coordinates" : [ -94.5844294, 39.
0418613 ] }, "tid" : NumberLong("1426474909095") }
{ "_id" : ObjectId("550647a5e4b0603bec07feb4"), "id" : "12345", "description" :
"mail letters", "loc" : { "type" : "Point", "coordinates" : [ -94.5844294, 39.04
18613 ] }, "tid" : NumberLong("1426474916256") }
{ "_id" : ObjectId("550647b0e4b097f4fd56053b"), "id" : "12345", "description" :
"clean house", "loc" : { "type" : "Point", "coordinates" : [ -94.5844294, 39.041
8613 ] }, "tid" : NumberLong("1426474927144") }
{ "_id" : ObjectId("550647bfe4b0603bec07fefa"), "id" : "12345", "description" :
"buy chocolates", "loc" : { "type" : "Point", "coordinates" : [ -94.5844294, 39.0
418613 ] }, "tid" : NumberLong("1426474942396") }
{ "_id" : ObjectId("550647d6e4b0603bec07ff24"), "id" : "12345", "description" :
"cook for guests", "loc" : { "type" : "Point", "coordinates" : [ -94.5844294, 39
.0418613 ] }, "tid" : NumberLong("1426474965338") }
{ "_id" : ObjectId("550647ece4b097f4fd5605a2"), "id" : "12345", "description" :
"ride to airport", "loc" : { "type" : "Point", "coordinates" : [ -94.5844294, 39
.0418613 ] }, "tid" : NumberLong("1426474987597") }
rs-ds052827:PRIMARY>
```

3. The order retrieving tasks should be equal to order of posting of tasks. We tested this case by using Postman chrome extension and mongoDB shell. By using mongoDB shell we inserted two Jobs 1 and 2. In postman we used the get api of mongolab. The order of retrieving and inserting are same.
4. 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. This case tested similarly to above case



## REST API testing:

We are having three collections Jobs, accepted, and applied. We tested the REST API for all collections using the postman chrome web extension.

### Getting query list from Jobs collection

The screenshot shows the Postman interface with a GET request to `https://api.mongolab.com/api/1/databases/group3/collections/Jobs?q={loc:{$geoWithin:{$centerSphere:[[-94.5844458,39.041]]}}`. The response status is 200 OK and the time taken is 1153 ms. The response body is a JSON array of three documents, each representing a job with fields like `_id`, `$oid`, `id`, `description`, `loc`, `type`, `coordinates`, and `tid`.

```
1 [
2   {
3     "_id": {
4       "$oid": "550643fbc4b097f4fd55fd83"
5     },
6     "id": "12345",
7     "description": "buy fruits",
8     "loc": {
9       "type": "Point",
10      "coordinates": [
11        [-94.584445,
12         39.0418593]
13      ]
14    },
15     "tid": 1426473978425
16   },
17   {
18     "_id": {
19       "$oid": "5506479ee4b0603bec07fea7"
20     },
21     "id": "12345",
22     "description": "shop groceries",
23     "loc": {
24       "type": "Point",
25       "coordinates": [
26         [-94.5844294,
27          39.0418613]
28       ]
29     },
30     "tid": 1426474989095
31   },
32   {
33     "_id": {
34       "$oid": "550647a5e4b0603bec07feb4"
35     },
36     "id": "12345",
37     "description": "mail letters",
38     "loc": {
39       "type": "Point",
40       "coordinates": [
41         [-94.5844458,
42          39.0418593]
43       ]
44     },
45     "tid": 1426474989095
46   }
47 ]
```

### Getting documents from applied collection

The screenshot shows the Postman interface with a GET request to `https://api.mongolab.com/api/1/databases/group3/collections/applied?q={}`. The response status is 200 OK and the time taken is 577 ms. The response body is a JSON array of two documents, each representing a document in the 'applied' collection with fields like `_id`, `$oid`, `id`, `tid`, `description`, `a_id`, and `tid`.

```
1 [
2   {
3     "_id": {
4       "$oid": "5509c484e4b0c49bcf35690c"
5     },
6     "id": "12345",
7     "tid": 1426474916256,
8     "description": "mail letters",
9     "a_id": 1426703491421
10  },
11  {
12    "_id": {
13      "$oid": "5509c485e4b0c49bcf356910"
14    },
15    "id": "12345",
16    "tid": 1426474965338,
17    "description": "cook for guests",
18    "a_id": 1426703492636
19  }
20 ]
```

## Getting list of documents from accepted collection

The screenshot shows the Postman interface with a GET request to `https://api.mongolab.com/api/1/databases/group3/collections/accepted?q={}&apiKey=5gq1g1JubzqF1gdxC`. The URL parameter `accepted?q={}` is highlighted with a red box. The response status is 200 OK, and the body is displayed in JSON format:

```
[
  {
    "_id": {
      "$oid": "55079df8e4b097f4fd5a74dc"
    },
    "id": "12345",
    "tid": 1426474927144,
    "a_id": 1426562523526,
    "description": "clean house",
    "accept_id": 1426562550885
  },
  {
    "_id": {
      "$oid": "5507a0e4e4b097f4fd5a7902"
    },
    "id": "12345",
    "tid": 1426474916256,
    "a_id": 1426562521588,
    "description": "mail letters",
    "accept_id": 1426563074749
  }
]
```

## Deleting from applied collection

The screenshot shows the Postman interface with a DELETE request to `https://api.mongolab.com/api/1/databases/group3/collections/applied/5509c484e4b0c49bcf35690c?apiKey=...`. The URL parameter `applied/5509c484e4b0c49bcf35690c` is highlighted with a red box, and the method dropdown is set to DELETE. The response status is 200 OK, and the body is displayed in JSON format:

```
{
  "_id": {
    "$oid": "5509c484e4b0c49bcf35690c"
  },
  "id": "12345",
  "tid": 1426474916256,
  "description": "mail letters",
  "a_id": 1426703491421
}
```

## Deployment:

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

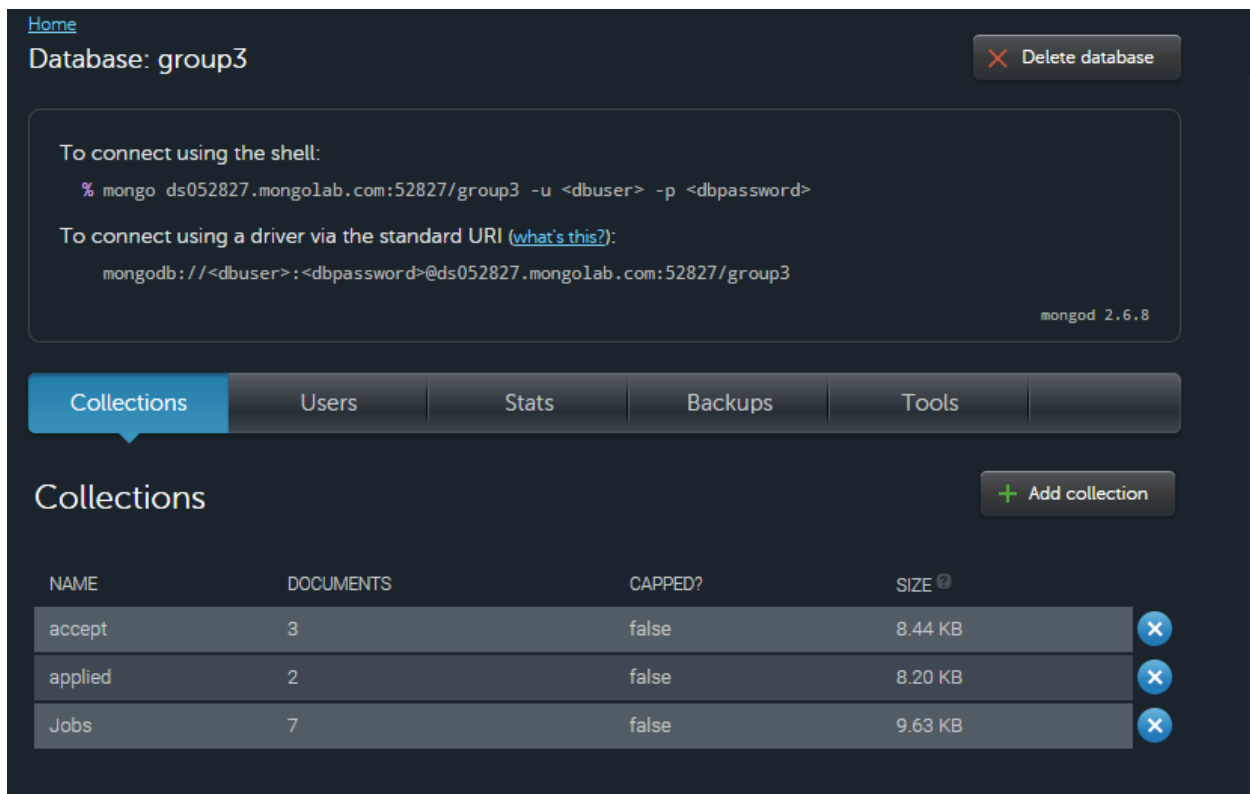
Github link: <https://github.com/ayinalakaushik/SmallTasksAtHand>

## Report:

In first increment we stored the location of employer as string with longitude and latitude.

To find the tasks in given radius we divided location in two string one is longitude and other is latitude.

We created two more collections applied, accept



Home Database: group3 ✖ Delete database

To connect using the shell:


```
% mongo ds052827.mongolab.com:52827/group3 -u <dbuser> -p <dbpassword>
```

To connect using a driver via the standard URI ([what's this?](#)):

```
mongodb://<dbuser>:<dbpassword>@ds052827.mongolab.com:52827/group3
```

mongod 2.6.8

**Collections** Users Stats Backups Tools + Add collection

| NAME    | DOCUMENTS | CAPPED? | SIZE  |                |
|---------|-----------|---------|--|----------------|
| accept  | 3         | false   | 8.44 KB  | <span>✖</span> |
| applied | 2         | false   | 8.20 KB  | <span>✖</span> |
| Jobs    | 7         | false   | 9.63 KB  | <span>✖</span> |

We retrieved all the jobs and converted the string longitude and string latitude into double and we found the tasks by using the formula  $(x*x)+(y*y) \leq r*r+2*a*x+2*b*y-a*a-b*b$

Where r is radius (a,b) are employee location

If a job location doesn't violate above condition then task in required radius.

We are retrieving the whole documents in the Jobs collection to find the require tasks. This make the method inefficient. So we read about location storing in mongoDB. We found geospatial locations storing in mongoDB.

We stored the location using JSONArray and written the query to get the list of jobs in given radius.

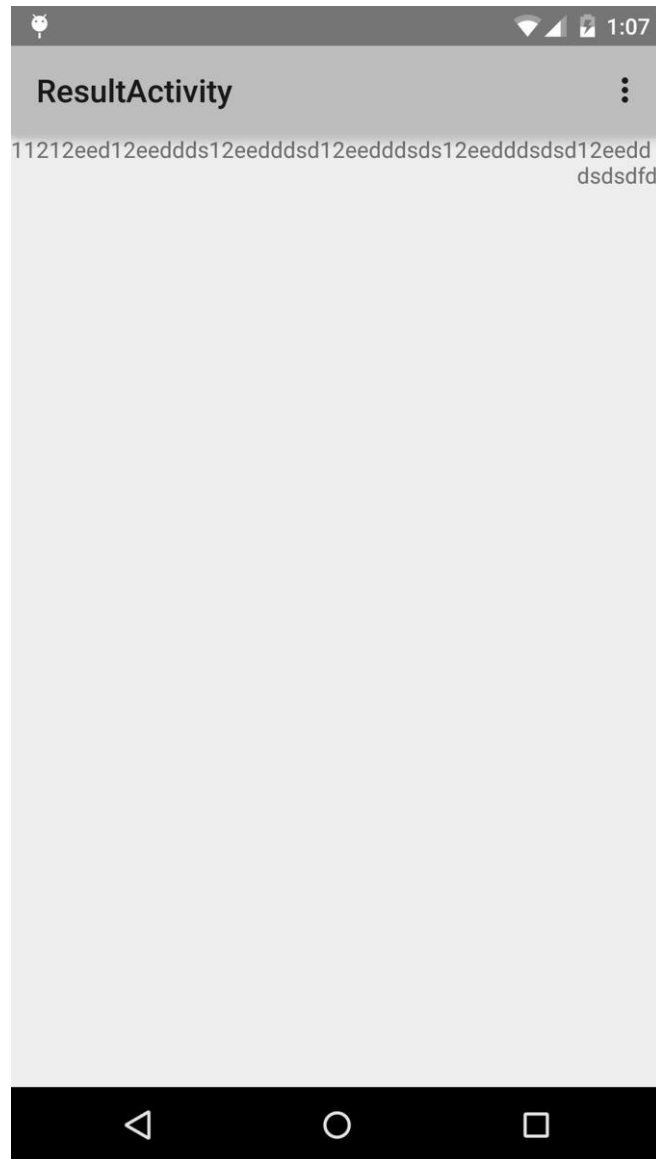
HttpGet class is not working in our code so we used the URL class to access the Jobs collection.

After retrieving the result we tried to display the result in a single textview



The get REST api gives the data in JSON result. We have written our own JSON parser which suits our requirement

We retrieved description and each description is given a textView. The output is as follows



The output has appended description. So we used ListView to display the result. After pressing on task the employee will apply for task. We stored the data in applied collection

In second service we retrieved the results from applied collection and shown to the user. User on clicking the task the particular job is deleted from applied list and moved to accepted list.

In third service we retrieved the results from accepted collection and shown to the user

## Project Management:

### Works Completed:

We completed three services in this increment.

1. Searching and applying for jobs
2. Viewing and accepting the applied jobs
3. Viewing and accepting the jobs

### Team Contributions:

**Kaushik** - 31%

**Yaswanth** - 23%

**Tharkin** - 23%

**Ravi Teja** - 23%

| First Service   | Second Service   | Third Service   |
|---|--|---|
| <b>Kaushik</b> – Implemented our first non-efficient method, Json parser, geospatial indexing, list view, query for retrieving the jobs in radius are done.<br>(Time Taken: 16 hours) |  |   |
| <b>Yaswanth</b> - Developed the front end for this service, managed the activities with respect to the asynchronous tasks.<br>(Time Taken: 9 hours)                                   | <b>Yaswanth</b> - Developed the front end for this service, managed the activities with respect to the asynchronous tasks.<br>(Time Taken: 2 hours)                              | <b>Yaswanth</b> - Developed the front end for this service, managed the activities with respect to the asynchronous tasks.<br>(Time Taken: 2 hours) |
| <b>Tharkin</b> - Developed the initial non efficient method and did testing for this service.<br>(Time Taken: 7 hours)  | <b>Tharkin</b> - Did the testing for this service.<br>(Time Taken: 2 hours)  | <b>Tharkin</b> - Did the testing for this service.<br>(Time Taken: 2 hours)   |
| <b>Ravi Teja</b> - Developed a connection class to retrieve the results using GET for jobs collection and to store the jobs applied by POST.<br>(Time Taken: 4 hours)                 | <b>Ravi Teja</b> - Developed a connection class to delete a document in collection, to get a document in applied, to post a document into the accepted.<br>(Time Taken: 8 hours) | <b>Ravi Teja</b> - Developed a connection class to retrieve the data from accept collection.<br>(Time Taken: 2 hours)                               |

**Works to be Completed:** As we still didn't implement the Login and Sign-up, we are unable to implement the bidding service.

**Issues & Concerns:** HttpGet class in android is not working in our code, so we used URL class to get data from database.