**Test App for Stress and Performance Testing**

**Introduction:**

This app is created keeping Stress Framework Web Application in mind and is aimed to have features similar to Framework. **The requirements are assumed based on the job requirements and to my knowledge.**

**APPLICATION:**

This app is aimed at Android phones for OS Android 4.4 and for devices running OS Android 4 and later. The code is written in java and data is stored in a sqlLite Database.

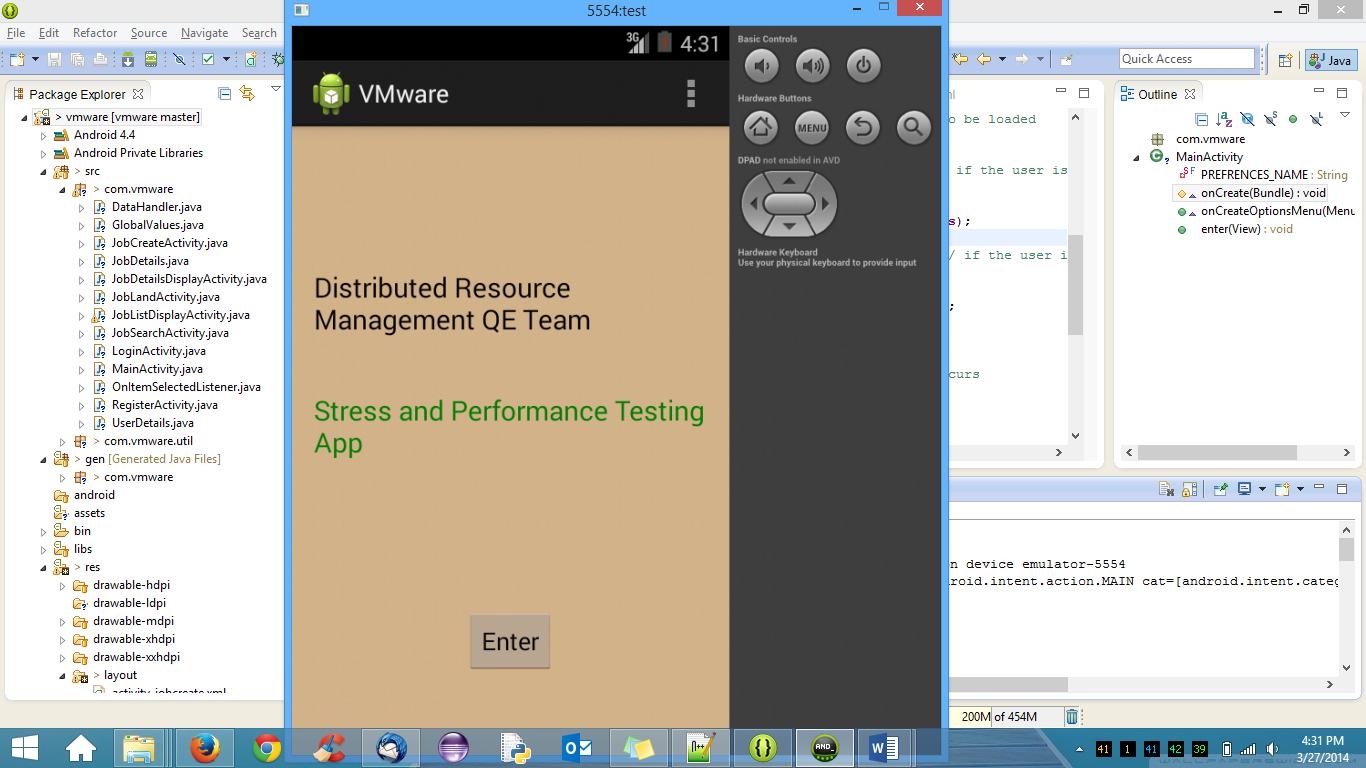
**DATABASE:**

SQLite3 as a temporary database so that data can be stored in phone. This can also be scaled such that periodically data can be pushed to cloud database and from cloud database to phone.

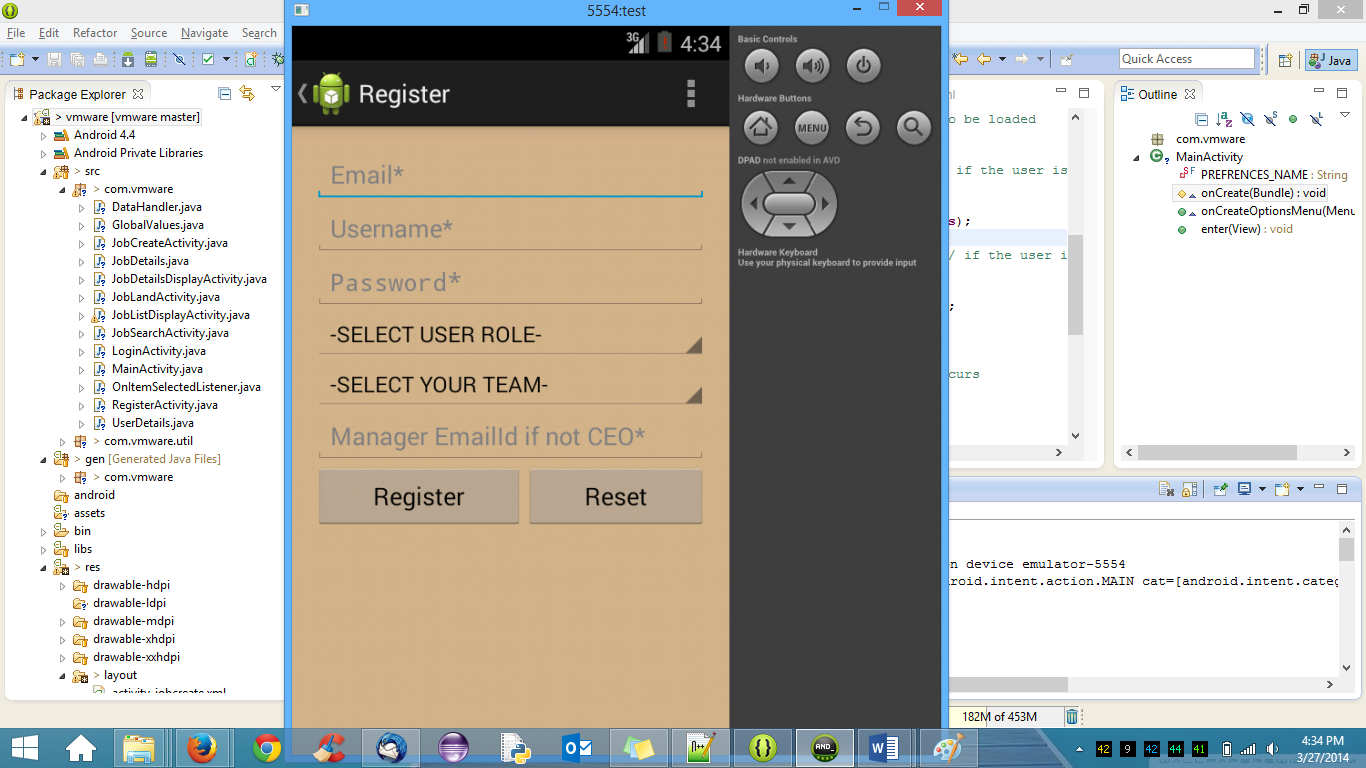
**BASIC FUNCTIONALITIES:**

1. Register the person who is entitled to use the system.
2. Check whether the user is an authenticated user or not.
3. Once loggedin , user has 2 functionalities.
   1. Create a task.
   2. Search for tasks.
   3. Edit and Save tasks.
4. When user is logged out, again user is taken through login page.

**STARTING PAGE:**

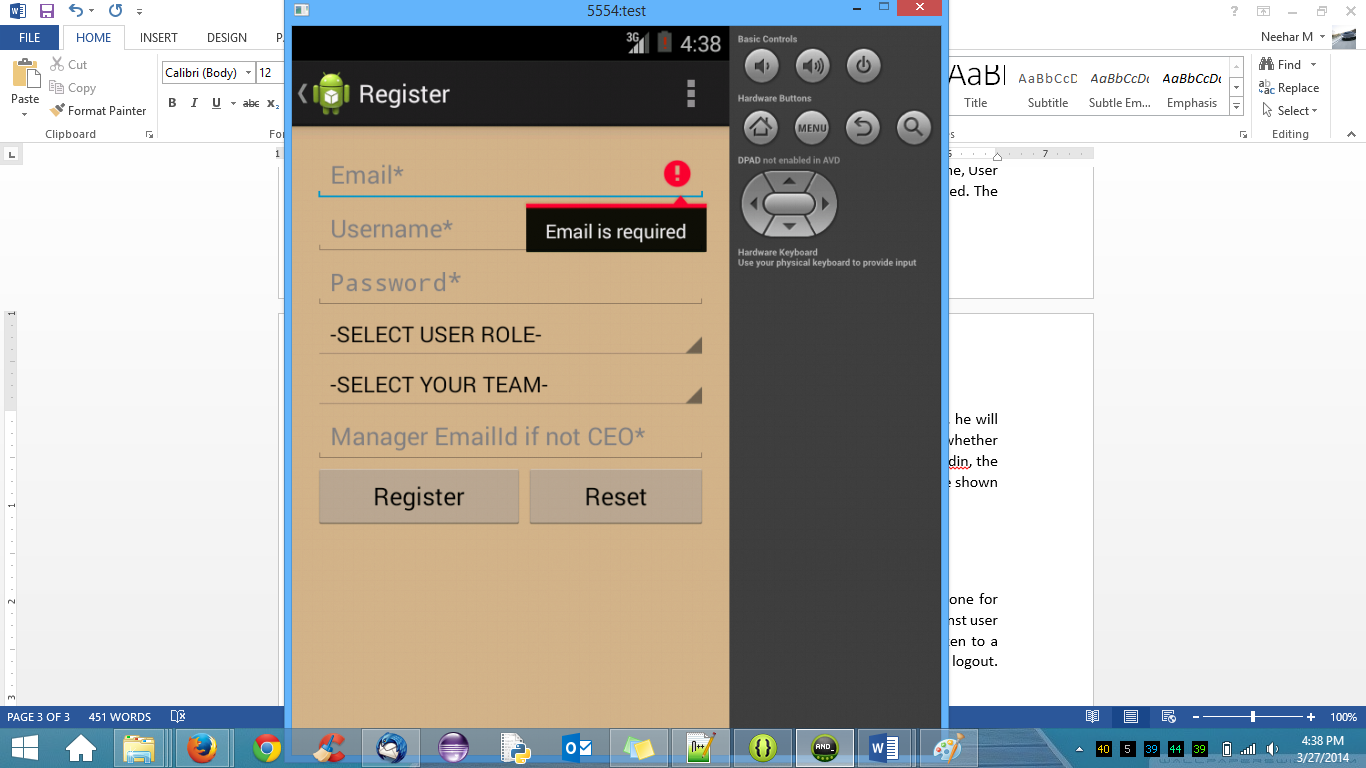


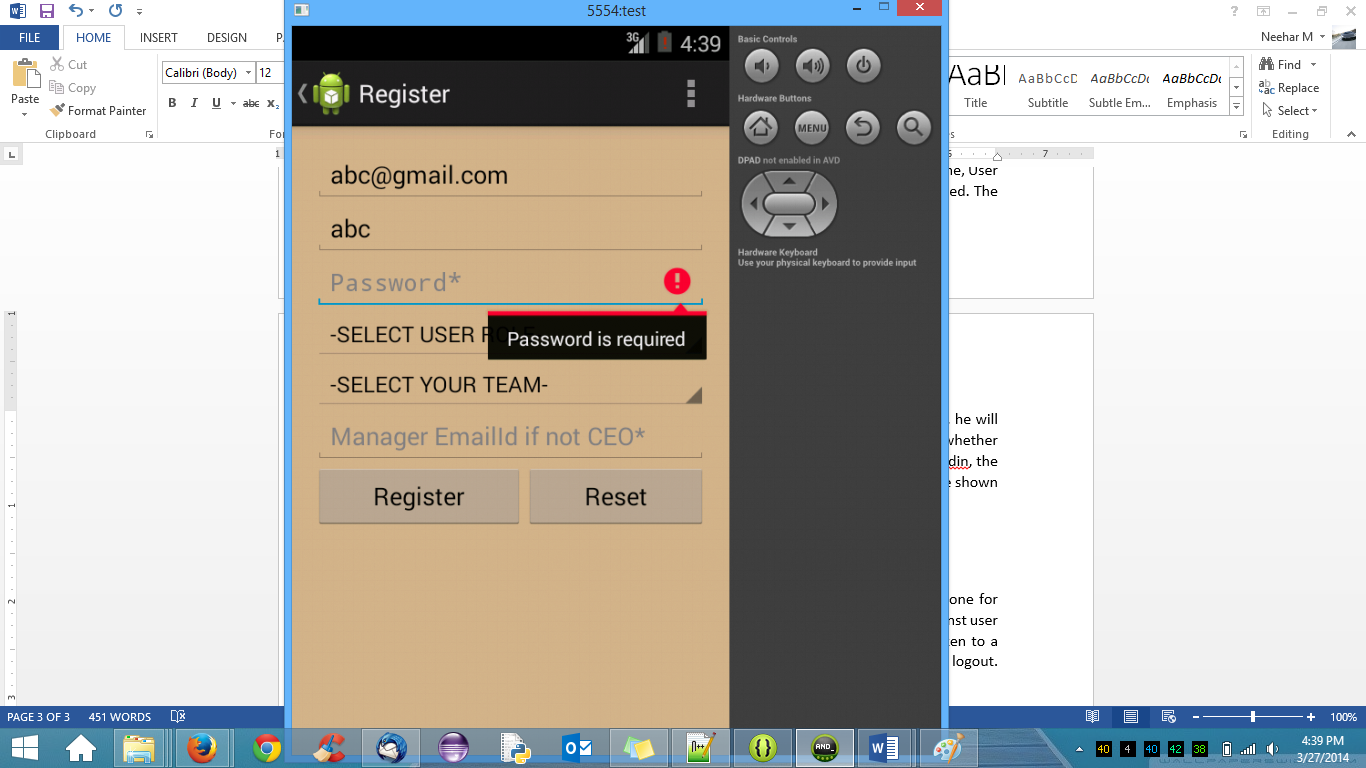
**REGISTRATION:**

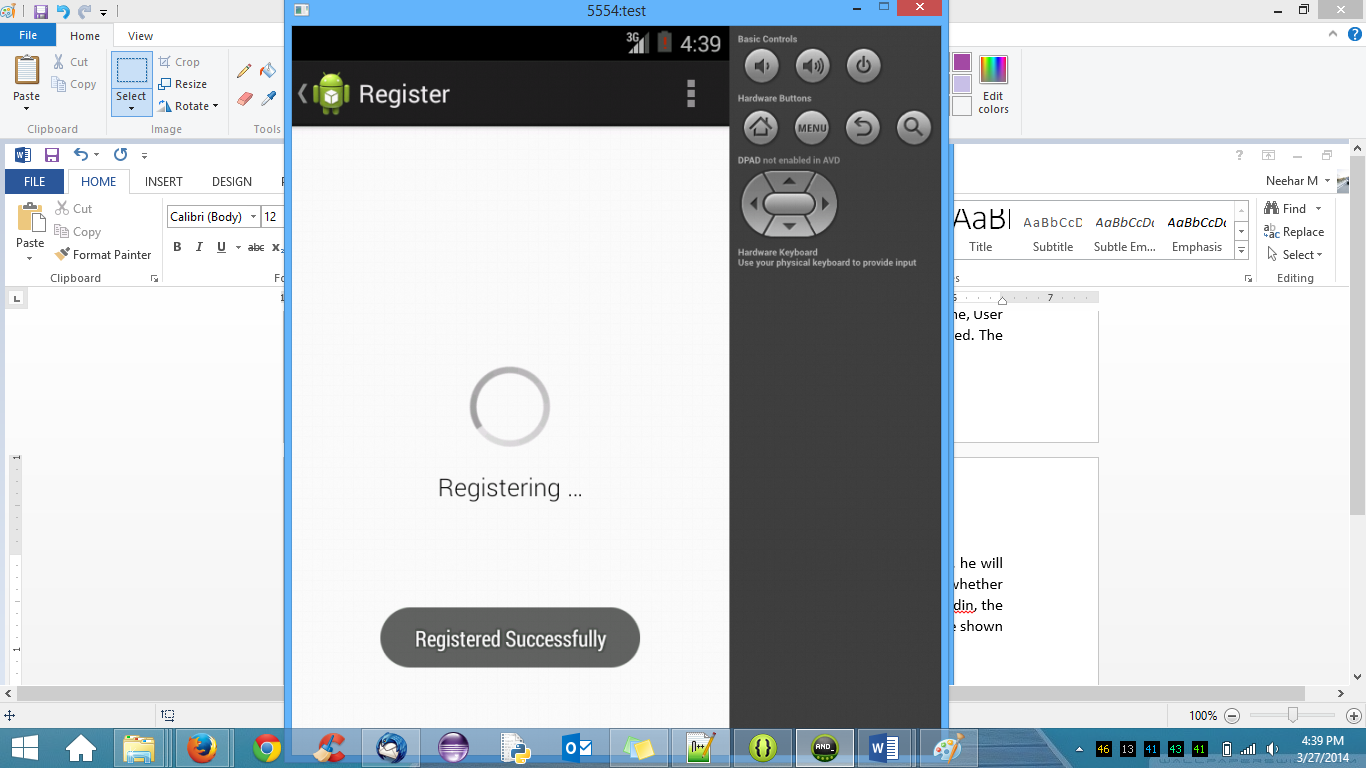


Basic details of user are taken and proper validations for email, password, username, User team, Usre group are done and if the entered passes all validations, the user is registered. The details are stored in USER\_DETAILS table. When the user is registered for the first time, he will be directly taken to the landing page and considered loggedin by default. The state of whether user is registered or not is saved in the application. If the user is registered but not loggedin, the user is shown a login page instead of landing page. If the user is not registered, he will be shown a registration page instead.

**Sample Validations:**

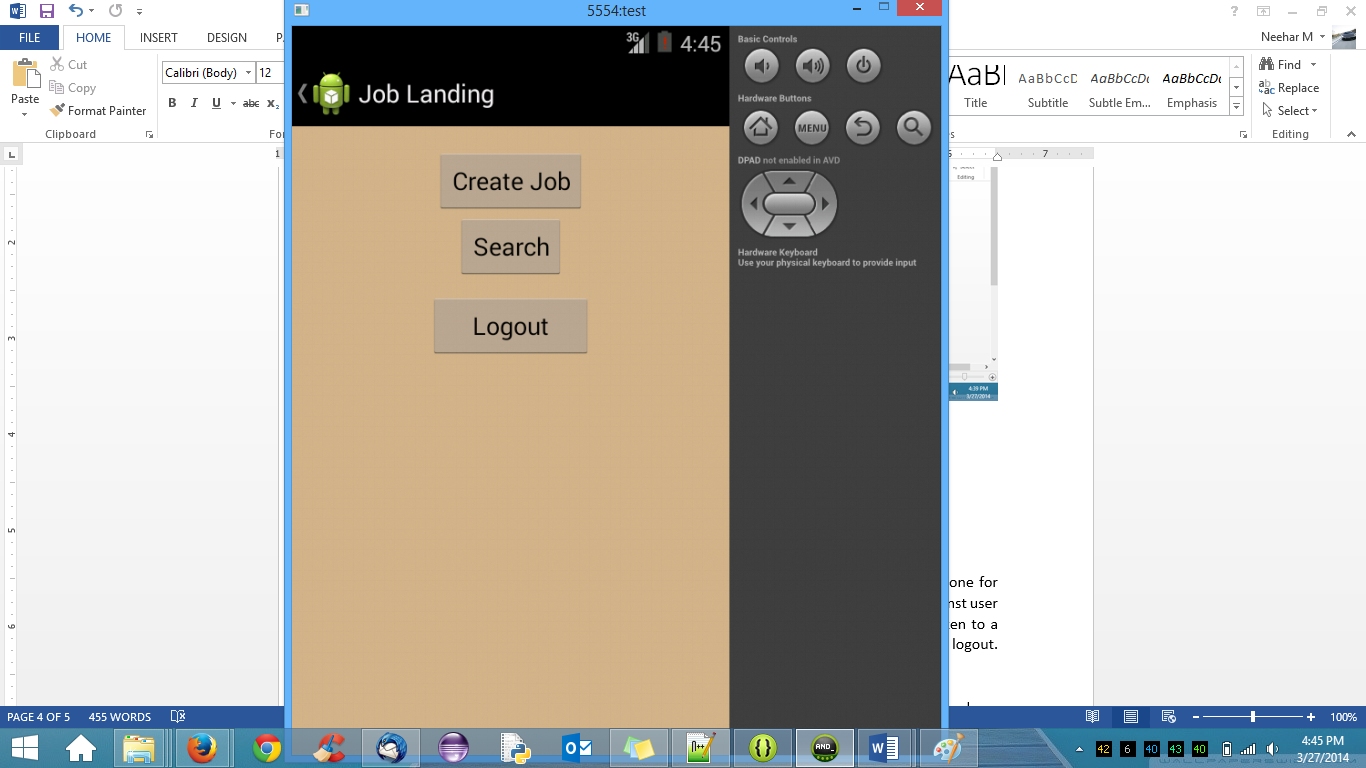






**LANDING PAGE:**

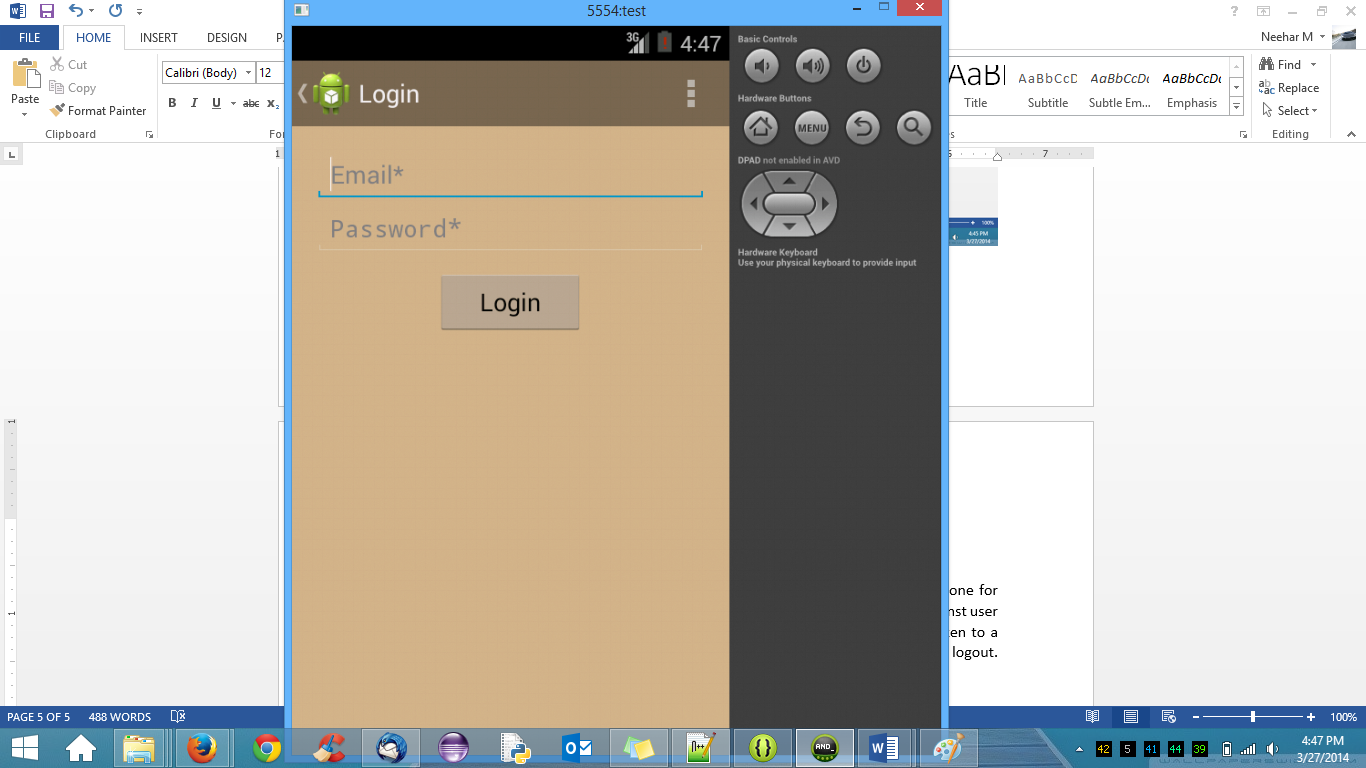
Once logged in, the user will be taken to the landing page directly. Here the user will have 3 options. To Create a job, To search for an existing job or to logout.



**USER LOGIN:**

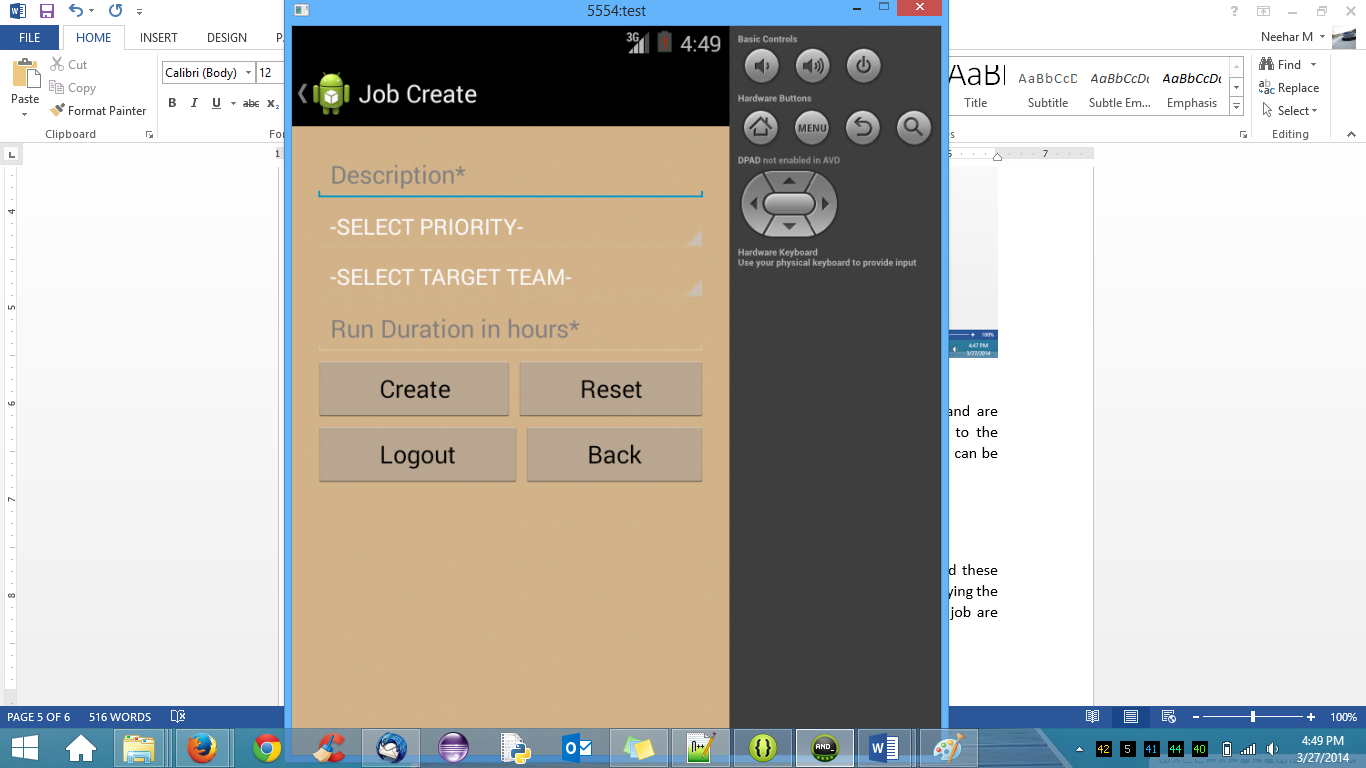
User will be prompted to go through a login page once he logs out. Until the user presses logout the user is considered to be loggedin by default.

The username, password are taken from the user and proper validations are done for these two fields and if once these fields are entered, they are checked for validations against user details present in the database for authentication. Once authenticated, the user is taken to a landing page which has the options to create a new job, search for already existing job and logout.

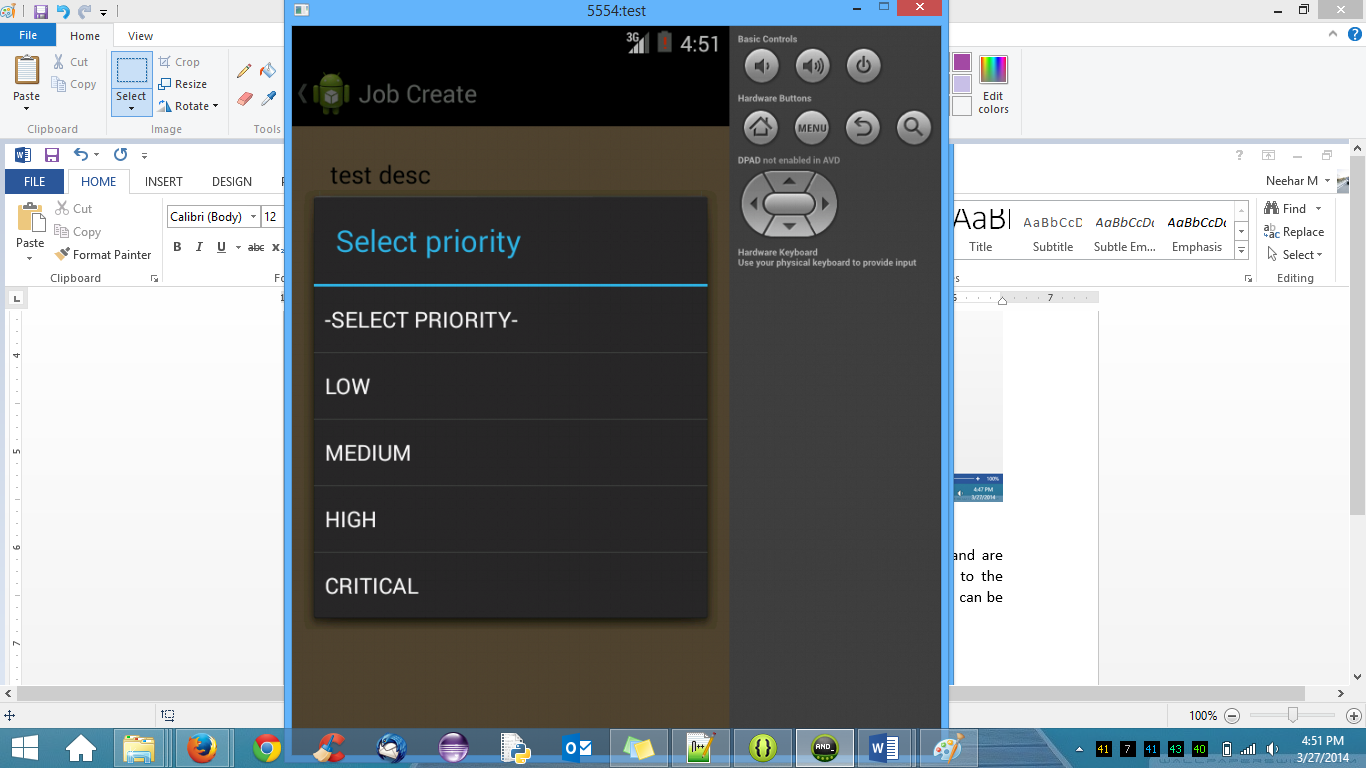


**CREATE JOB:**

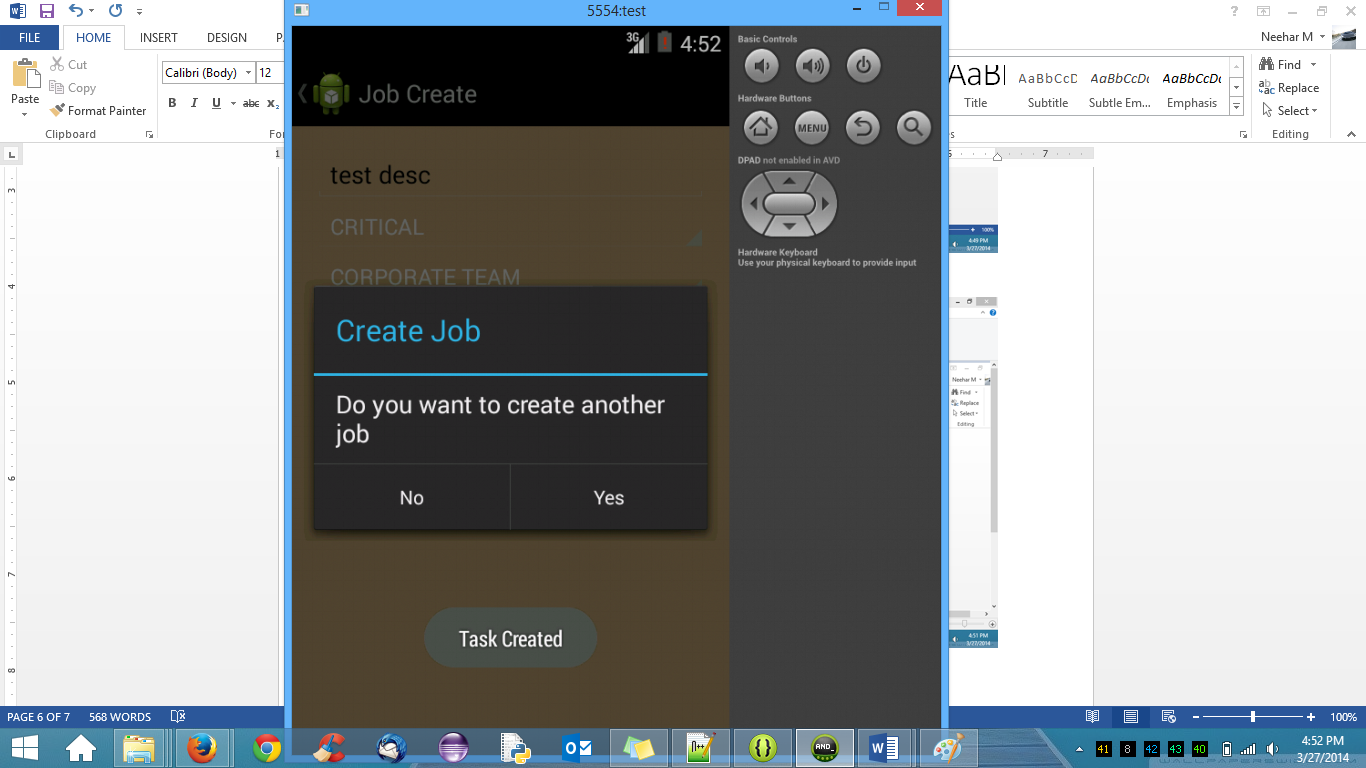
For Creating a new job, minimum necessary fields are taken from the user and are validated, Once the data passes through all the validations, the data is inserted in to the JOB\_DETAILS table in the database. Some of the fields are populated automatically and can be edited later. Once the job is created successfully, the user will be given option whether the user wants to create another job or the user wants to do nothing. This is done using an Alert box and based on the response form the user, appropriate action is taken.



One of the drop down values.

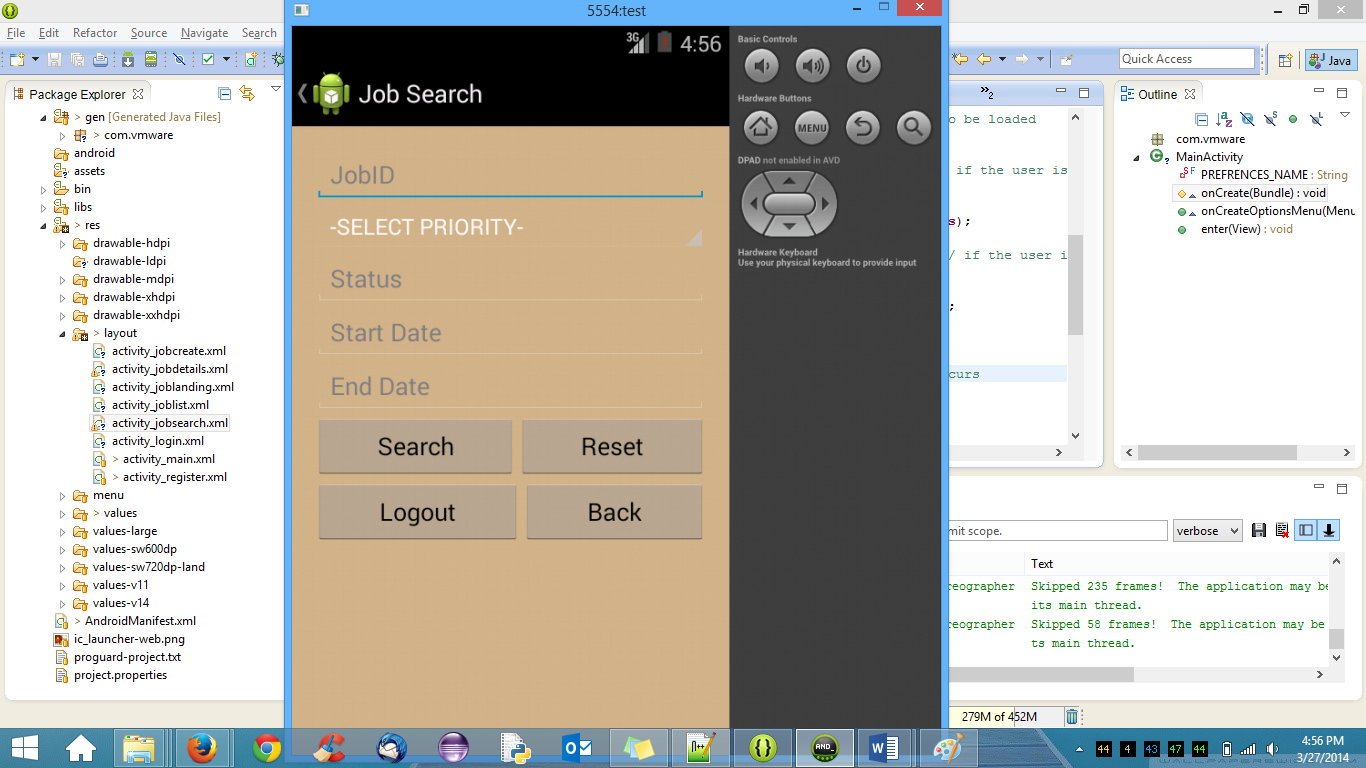


Once job is created successfully, alert box is thrown asking for user preferences



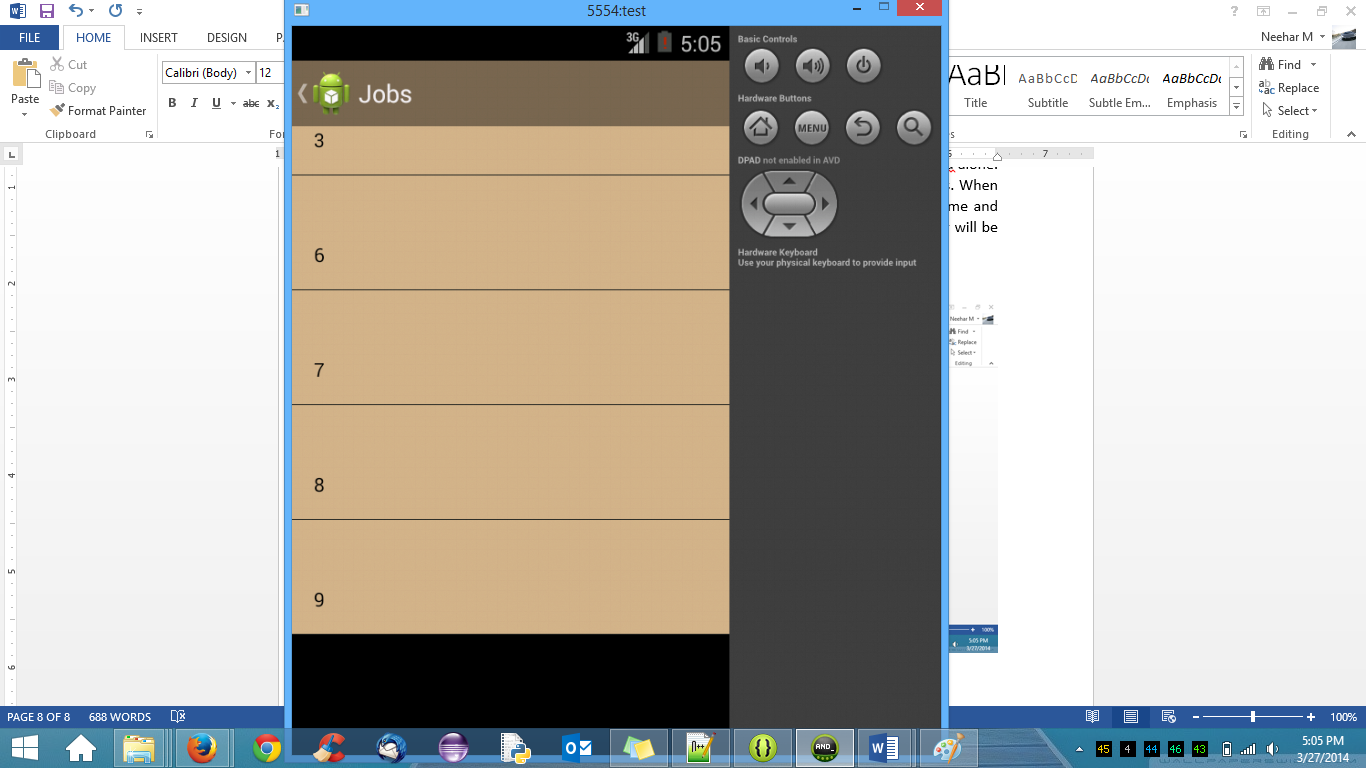
If the user selects no, the user is taken to old page. If the user selects to create another job, he is again shown the old page to create another job.

**Search Job:**



From the landing page, the user can search for a job based on some fields and these entries are validated and then searched against database to fetch data and the jobs satisfying the search criteria are listed. Once clicked on a particular job, details corresponding to a job are shown in the next view. There are different options to search. The user can search based on jobid alone. Another option can be priority, jobid, status. Another option can be priority and status. When the user has not given job id and given other options, the query appends the username and searches jobs which satisfy user entered conditions along with userid. By this way user will be able to see jobs for which job creator is the userid.

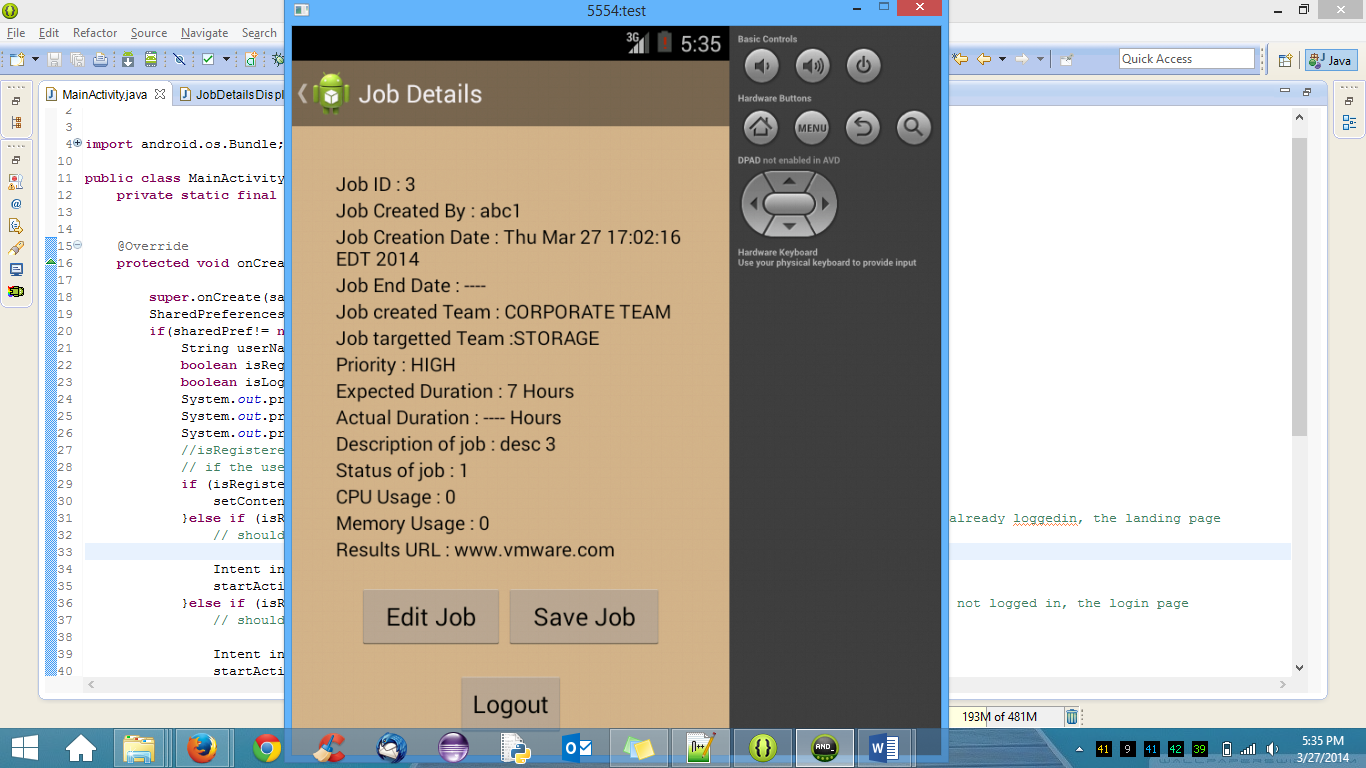
**RESULTS OF SEARCH / JOB LISTING:**



The jobids which satisfy the user defined search criteria are shown. Once the user clicks on any job id, the details corresponding to the job are shown.

**JOB DETAILS**:

The details of the job which are selected are shown in the screen.



**IMPROVEMENTS WHICH CAN BE PERFORMED.**

**LIMITATIONS:**

1. The user functionalities are limited as of just to register, login, create, search. User can have other options like edit job, save job and cancel job that user has created if the user wants to cancel the task.
2. The database is sqlLite which is useful for smartphones. This can be replaced by any standalone or cloud database so that details corresponding to other users can be shown also. Now the details corresponding to app user are only shown.
3. Few back navigations are not implemented. For eg: Back navigation in Job Listing Page and Job Details Page are not implemented. This is because of time constraints I have. Given enough time these can be implemented also.
4. Search based on Dates are not implemented now because of time constraints I have.
5. UI can be made visually appealing. At present basic UI is applied.
6. More testing and Stress Testing has to be done if the app can have any leaks.
7. Logger mechanism have to be implemented instead of sysouts.
8. The Status Table is not populated. This can be prepopulated and also Foreign keys and indexes are not created as this is kind of temporary database and doesnot hold much data. For cloud database / stand alone database, all those can be created.