SOFTWARE ENGINEERING
CS301

SOFTWARE DESIGN SPECIFICATION

MIETEN-PLACEMENT CELL ANDROID APPLICATION

GROUP MEMBERS-

Kumari Renuka(U101115FCS111)
Rishabh K Kandoi(U101115FCS283)
Shailesh Mohta(U101115FCS)305
Luvkush Singh(U101115FCS113)
Kshiteej Manoj Gilda(U101115FCS110)
Manik Garg(U101115FCS199)

TABLE OF CONTENT

The Software Design Specification

1. Introduction

The Software Design Document is a document to provide documentation which will be used to aid in software development by providing the details for how the software should be built. Within the Software Design Document are narrative and graphical documentation of the software design for the project including use case models, sequence diagrams, collaboration models, object behaviour models, and other supporting requirement information.

1.1 Purpose of this document

This document will define the design of the one runway simulator. It contains specific information about the expected input, output, classes, and functions. The interaction between the classes to meet the desired requirements are outlined in detailed figures at the end of the document.

1.2 Scope of the development project

We describe what features are in the scope of the software and what are not in the scope of the software to be developed.

In Scope:

a. Application for the job/internship by the students of the NIIT University.
b. Students can retrieve the information about the internship/jobs.
c. Preparation of the aptitude test.

Out of Scope:

a. Selection process for any internship
b. There is no communication between the NIIT University and the Industry via this application.

1.3 Definitions, acronyms, and abbreviations

IEEE: Institute of Electrical and Electronics Engineers
SDS: Software Design Specification
CRS: Campus Recruitment System
Mieten: The name of CRS, an android application.

1.4 References

1.4.1 R. S. Pressman, Software Engineering: A Practioner's Approach, 5th Ed, McGraw-Hill, 2001.

1.4.2 IEEE SDS template

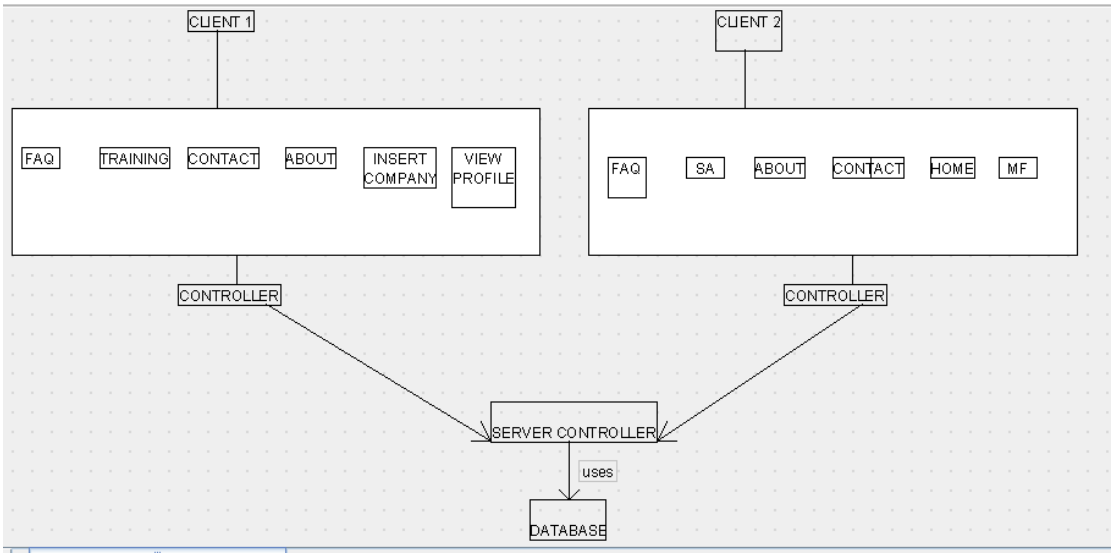
1.5 Overview of document

This SDS is divided into seven sections with various sub-sections. The sections of the Software Design Document are:

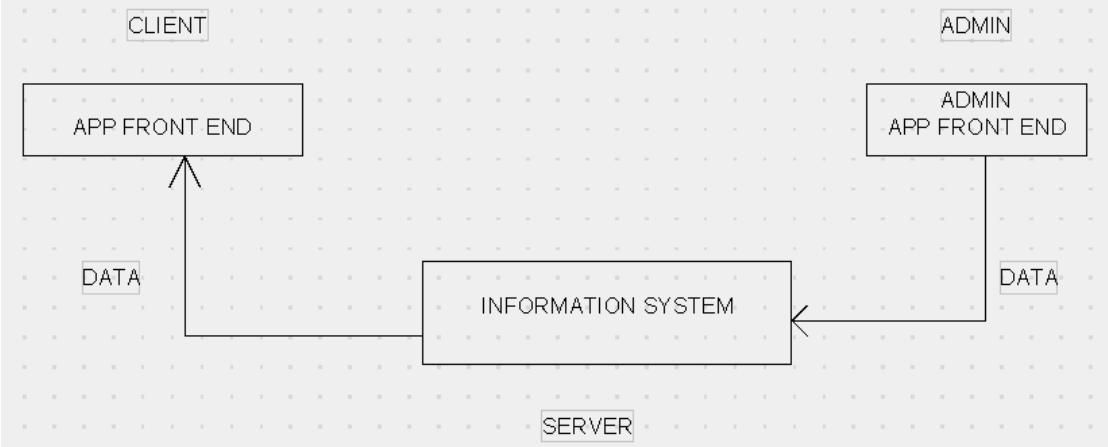
1. **Introduction**: describes about the document, purpose, scope of development project definitions and abbreviations used in the document.
2. **Conceptual Architecture/Architecture Diagram:** describes the overview of components, modules, structure and relationships and user interface issues.
3. **Logical Architecture:** describes Logical Architecture Description and Components.
4. **Execution Architecture:** defines the runtime environment, processes, deployment view.
5. **Design Decisions and Trade-offs:** describes the decisions taken along with the reason as to why they were chosen over other alternatives.
6. **Pseudocode for components:** describes pseudocode, as the name indicates.
7. **Appendices:** describes subsidiary matter if any.

2. Conceptual Architecture/Architecture Diagram

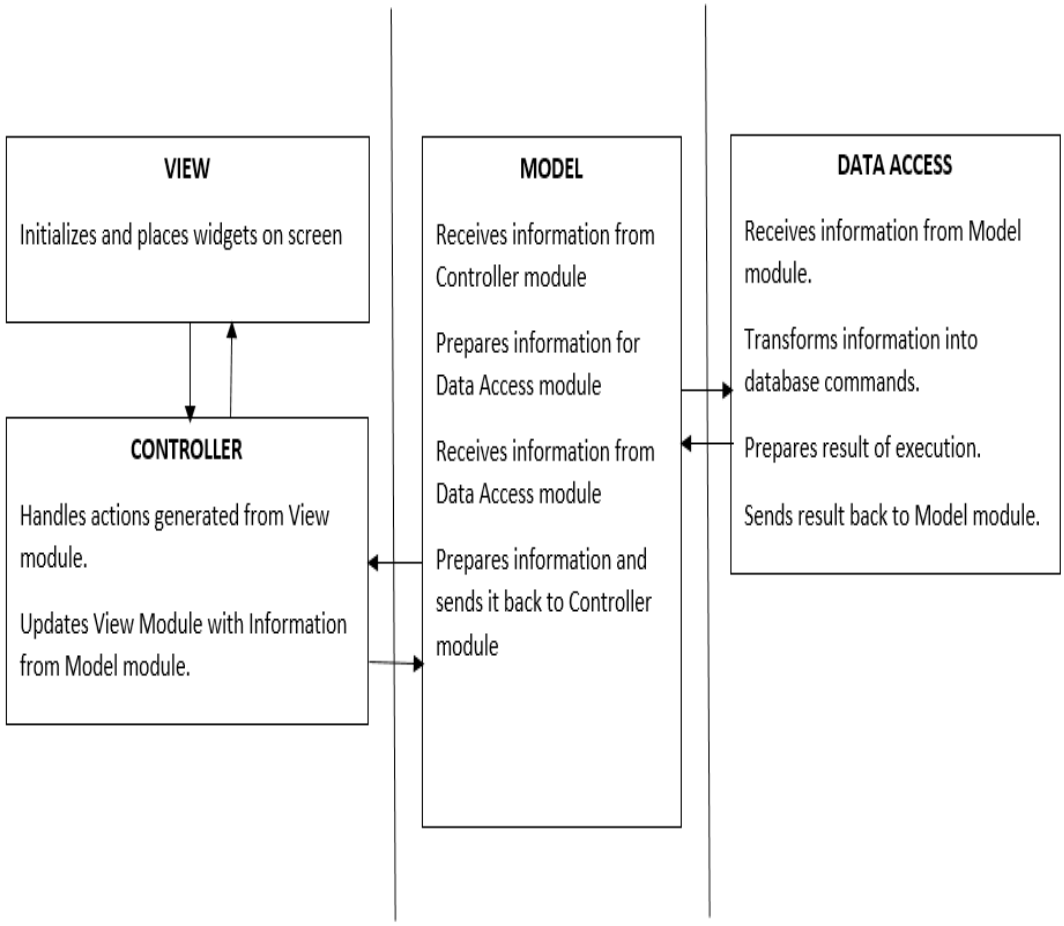
Architecture Diagram 1:



Architecture Diagram 2:



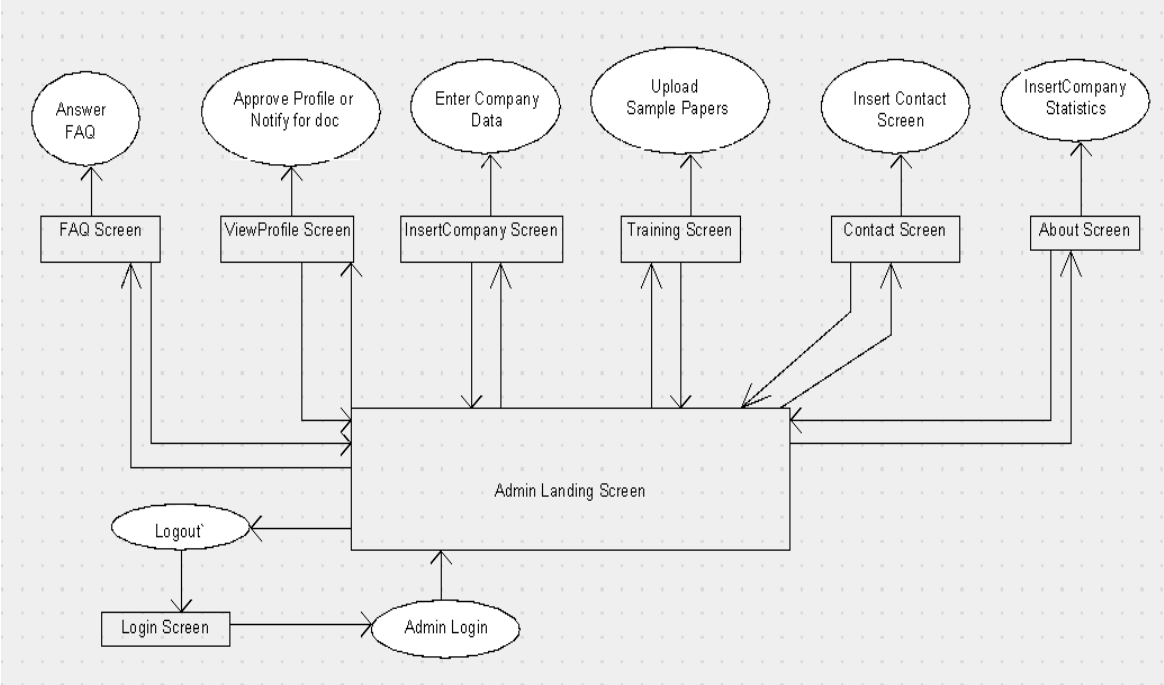
2.1 Overview of modules / components



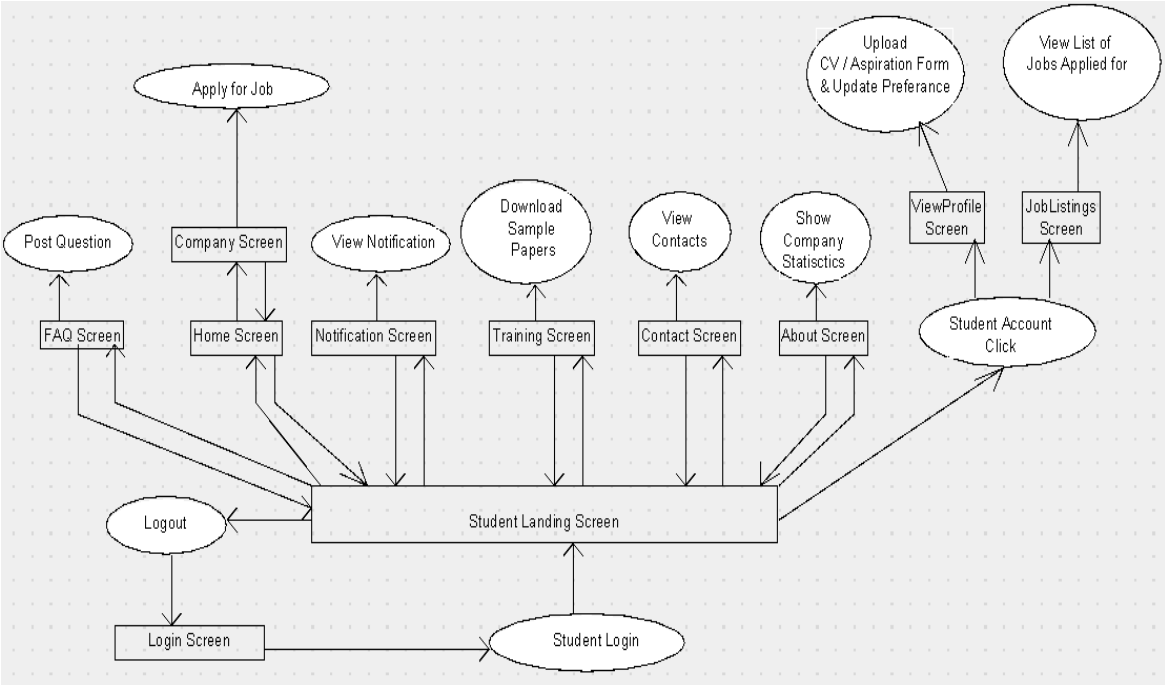
NOTE:
The horizontal lines represent the separation of modules.
More than one box within the same section represents sub-modules.

2.2 Structure and relationships

2.2.1 Admin's Side



2.2.2 Student's Side



NOTE:
The boxes represent individual screens.
The circles represent actions that do not have screens.
The arrows represent navigation between screens.

2.3 User interface issues

This section will address User Interface issues as they apply to the following hypothetical users of the Campus Recruitment System, Mieten.

- User A is a 20-year-old female, a student of NIIT University, IIIrd year, who is fairly comfortable with technology. She is proficient with using most common computer applications.

  Since User A is familiar with android applications, Mieten, will use common user interface conventions. For example, links between screens will use ordinary, easy to understand descriptions such as “Login”, ”Home”, ‘Upload CV” etc. To maintain consistency, any other links will also appear in the bottom half of the screen.

- User B is a 35-year-old male, A staff member of Placement Cell of NIIT University. He is although comfortable using android applications, he has never used any portal which helps applying for jobs/internships, like Mieten. He may or may not acknowledge if this android application is really helpful.

  Since User B is not fond of such a technology oriented(applying for jobs/internships via an android application), it is imperative that he be given clear on-screen directions. Consequently, the user interface has been tried to be such that grabs the attention of the user. It has been taken care that user should not consider this application as a burden/difficult to use. On-clicks task implementation has been done in Mieten, so that the admin (User B, in this case),

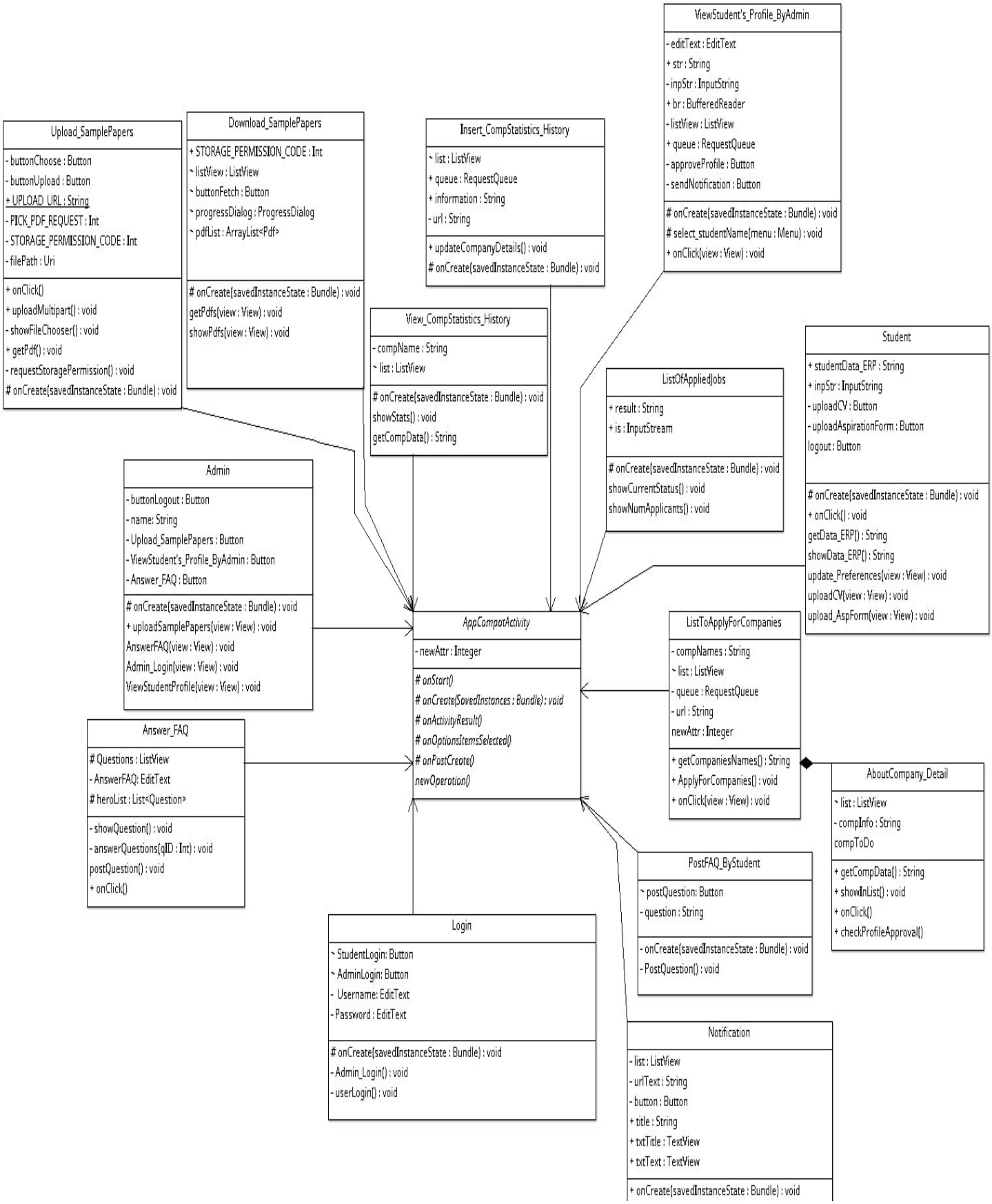
have an ease in communication with students, updating of new Industry Collaboration with the university etc. via Mieten. Color combination has been so chosen, that allows the user to read all of the text on the screen in direct sunlight. Text size is reasonably larger and, therefore, more readable.

- User C is a 29-year-old female, A faculty member of NIIT University. She is well verse with using of android apps and therefore, Mieten would be easily operated by her.

  As we are clear that Mieten is exclusive to the Placement cell and the students of NIIT University, the faculty members still can use this application in order to know the status of placements every year, to be updated about the recent/new Industry Collaborations and can help in spreading the network of NIIT University across. Again, information fetching would be a few clicks away with minimal/no extra efforts.

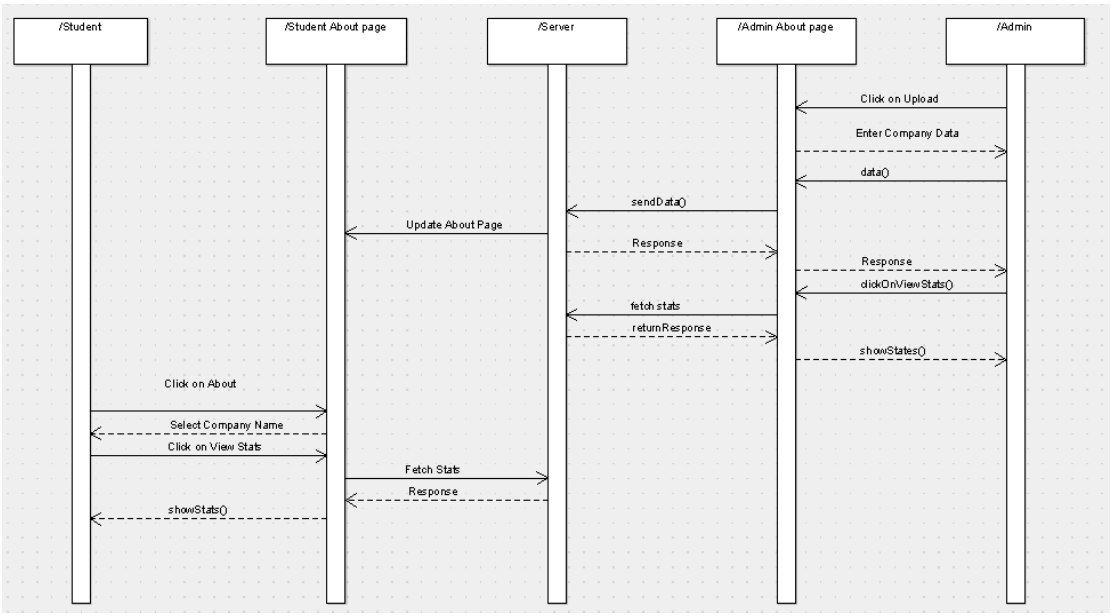
3. Logical Architecture (Class Diagram, Sequence Diagram, State Diagram)

Class Diagram:

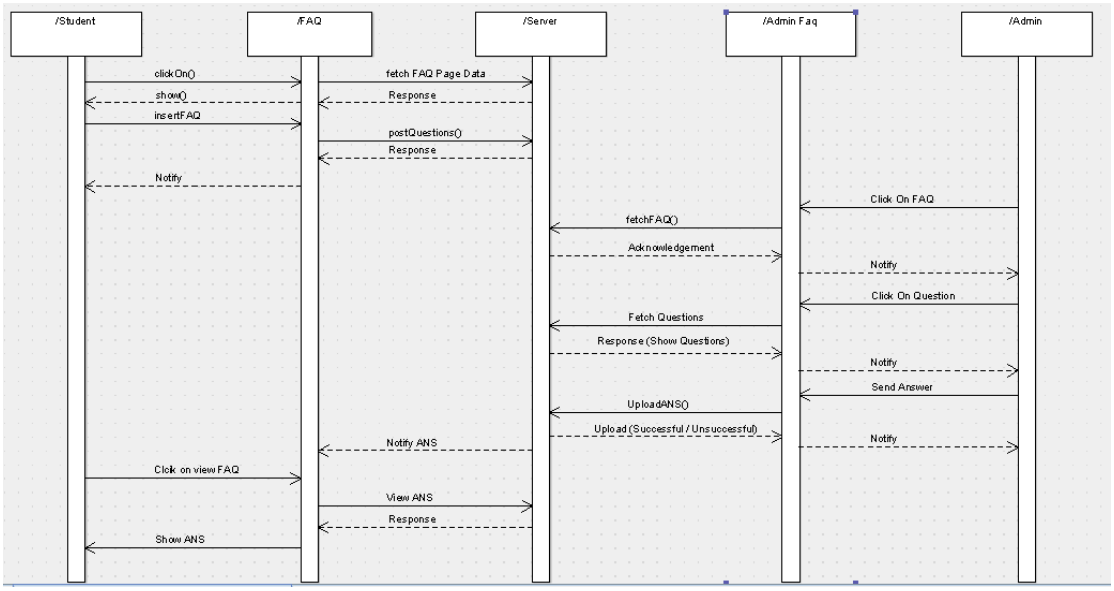

ViewStudent's_Profile_ByAdmin
- editText : EditText
+ str : String
- inpStr : InputString
+ br : BufferedReader
- listView : ListView
+ queue : RequestQueue
- approveProfile : Button
- sendNotification : Button
onCreate(savedInstanceState : Bundle) : void
select_studentName(menu : Menu) : void
+ onClick(view : View) : void
Upload_SamplePapers
- buttonChoose : Button
- buttonUpload : Button
+ UPLOAD_URL : String
- PICK_PDF_REQUEST : Int
- STORAGE_PERMISSION_CODE : Int
- filePath : Uri
+ onClick()
+ uploadMultipart() : void
- showFileChooser() : void
+ getPdf() : void
- requestStoragePermission() : void
onCreate(savedInstanceState : Bundle) : void
Download_SamplePapers
+ STORAGE_PERMISSION_CODE : Int
~ listView : ListView
~ buttonFetch : Button
~ progressDialog : ProgressDialog
~ pdfList : ArrayList<Pdf>
# onCreate(savedInstanceState : Bundle) : void
getPdfs(view : View) : void
showPdfs(view : View) : void
Insert_CompStatistics_History
~ list : ListView
+ queue : RequestQueue
+ information : String
- url : String
+ updateCompanyDetails() : void
# onCreate(savedInstanceState : Bundle) : void
View_CompStatistics_History
- compName : String
~ list : ListView
# onCreate(savedInstanceState : Bundle) : void
showStats() : void
getCompData() : String
ListOfAppliedJobs
+ result : String
+ is : InputStream
# onCreate(savedInstanceState : Bundle) : void
showCurrentStatus() : void
showNumApplicants() : void
Student
+ studentData_ERP : String
+ inpStr : InputString
- uploadCV : Button
- uploadAspirationForm : Button
logout : Button
onCreate(savedInstanceState : Bundle) : void
+ onClick() : void
getData_ERP() : String
showData_ERP() : String
update_Preferences(view : View) : void
uploadCV(view : View) : void
upload_AspForm(view : View) : void
Admin
- buttonLogout : Button
- name: String
- Upload_SamplePapers : Button
- ViewStudent's_Profile_ByAdmin : Button
- Answer_FAQ : Button
onCreate(savedInstanceState : Bundle) : void
+ uploadSamplePapers(view : View) : void
AnswerFAQ(view : View) : void
Admin_Login(view : View) : void
ViewStudentProfile(view : View) : void
AppCompatActivity
- newAttr : Integer
# onStart()
# onCreate(SavedInstances : Bundle) : void
# onActivityResult()
# onOptionsItemsSelected()
# onPostCreate()
newOperation()
ListToApplyForCompanies
- compNames : String
~ list : ListView
- queue : RequestQueue
- url : String
newAttr : Integer
+ getCompaniesNames() : String
+ ApplyForCompanies() : void
+ onClick(view : View) : void
Answer_FAQ
Questions : ListView
- AnswerFAQ: EditText
heroList : List<Question>
- showQuestion() : void
- answerQuestions(qID : Int) : void
postQuestion() : void
+ onClick()
AboutCompany_Detail
~ list : ListView
- compInfo : String
compToDo
+ getCompData() : String
+ showInList() : void
+ onClick()
+ checkProfileApproval()
PostFAQ_ByStudent
~ postQuestion: Button
- question : String
- onCreate(savedInstanceState : Bundle) : void
- PostQuestion() : void
Login
~ StudentLogin: Button
~ AdminLogin: Button
- Username: EditText
- Password : EditText
onCreate(savedInstanceState : Bundle) : void
- Admin_Login() : void
- userLogin() : void
Notification
- list : ListView
- urlText : String
- button : Button
+ title : String
+ txtTitle : TextView
+ txtText : TextView
+ onCreate(savedInstanceState : Bundle) : void

Sequence Diagrams:

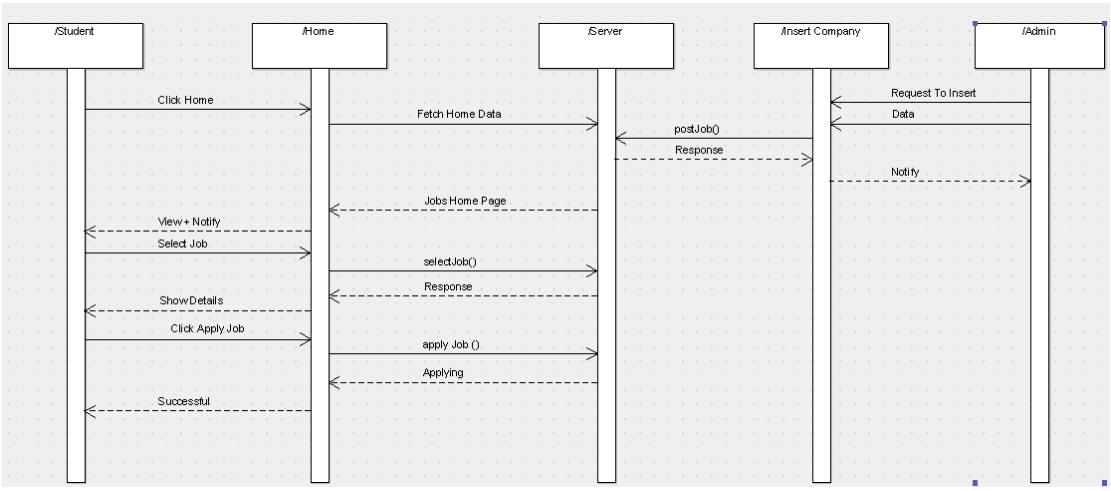
Sequence Diagram: About Page



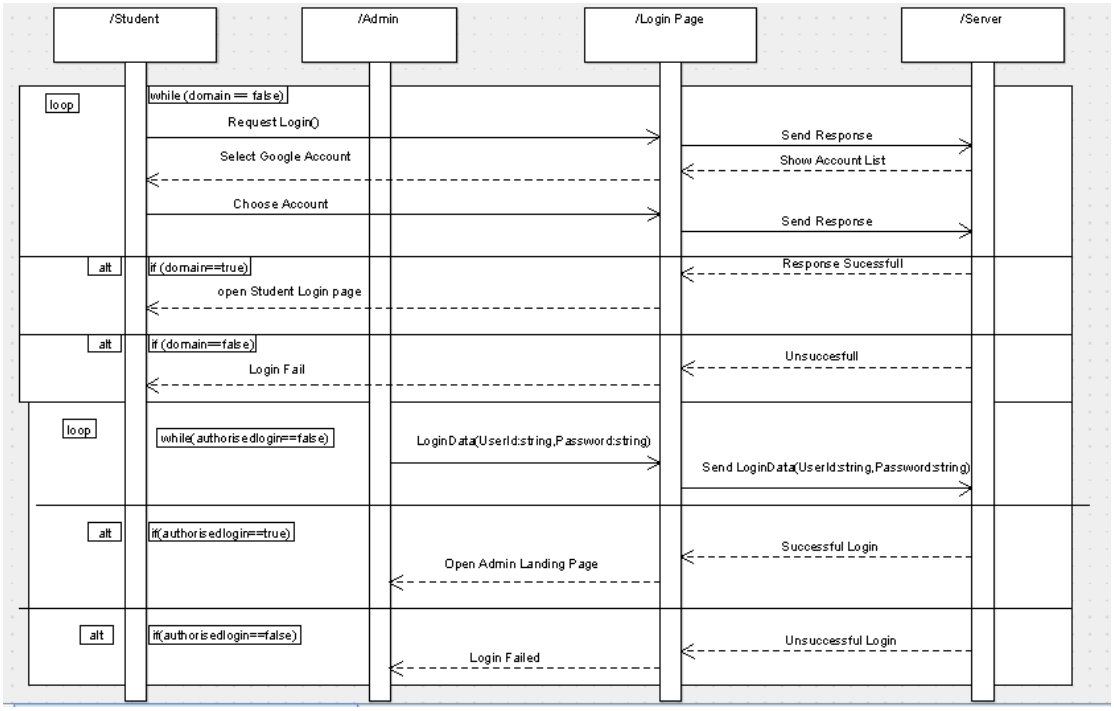
Sequence Diagram: FAQ



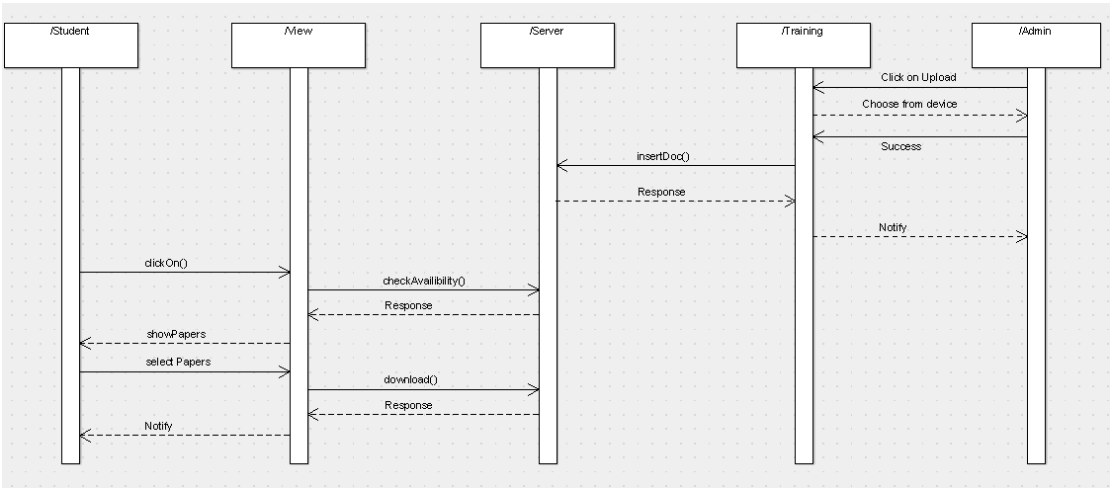
Sequence Diagram: Home



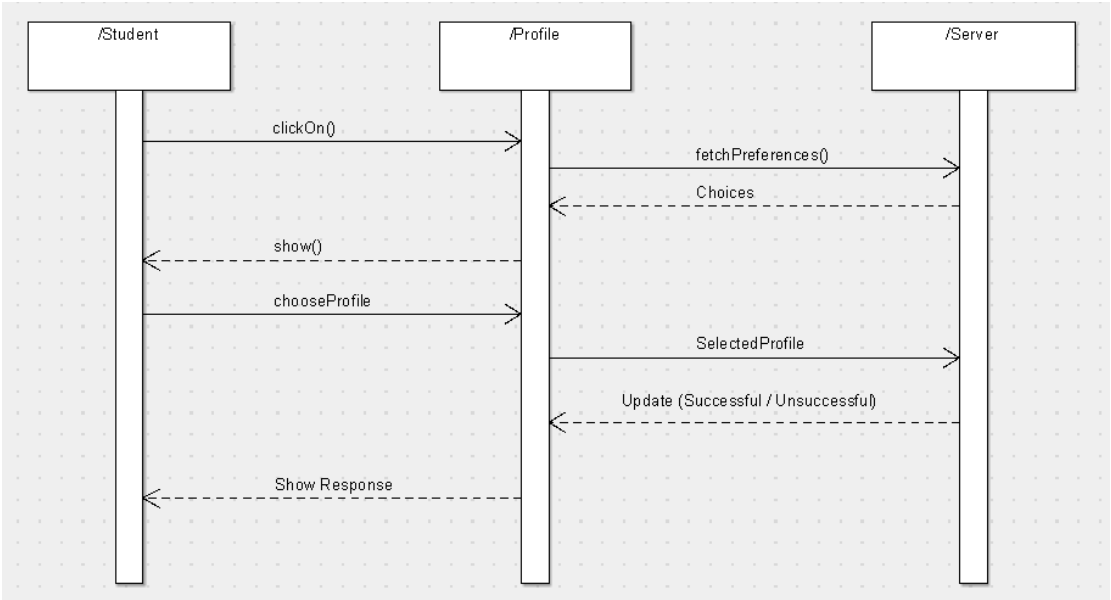
Sequence Diagram: Login Page



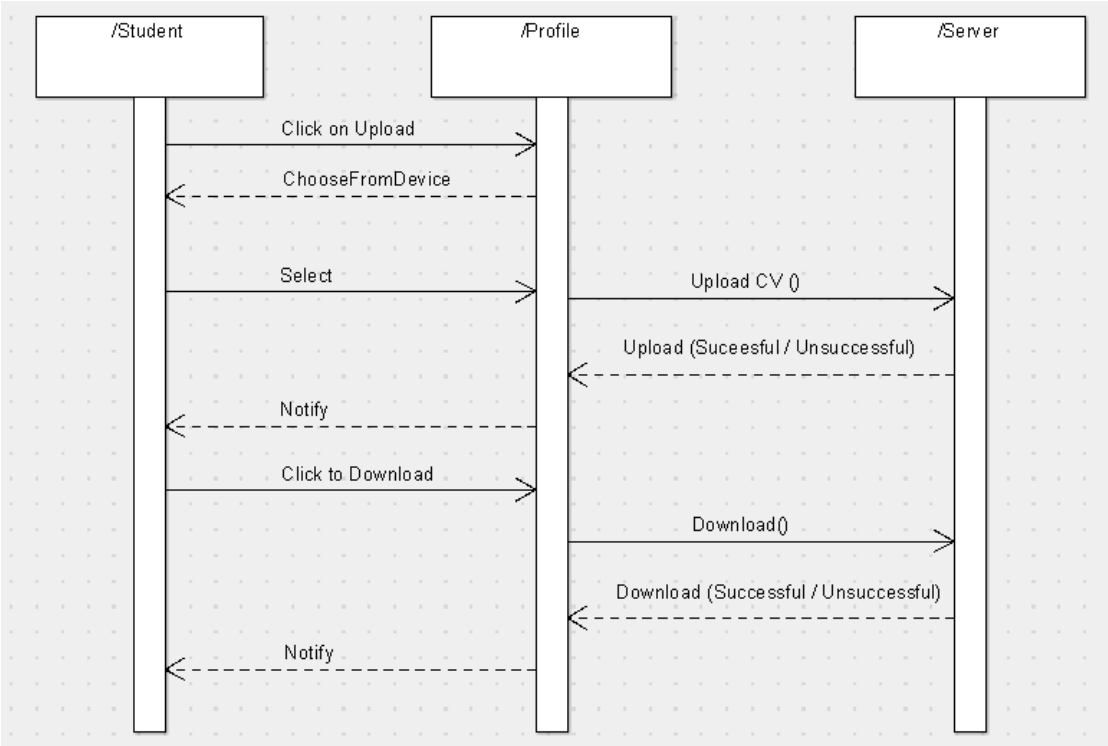
Sequence Diagram: Sample Papers



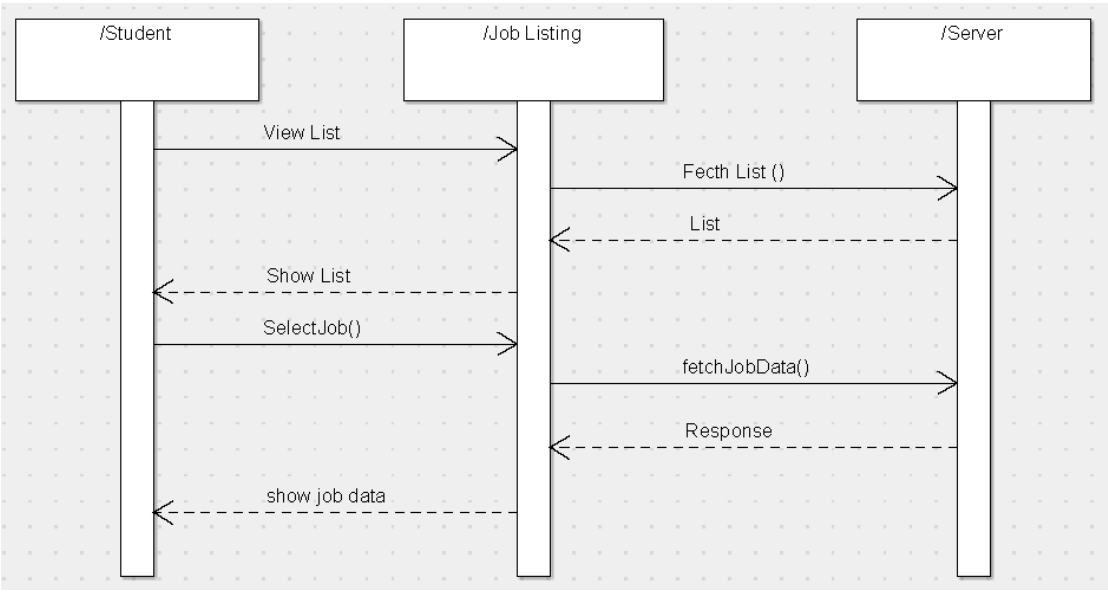
Sequence Diagram: Student Update



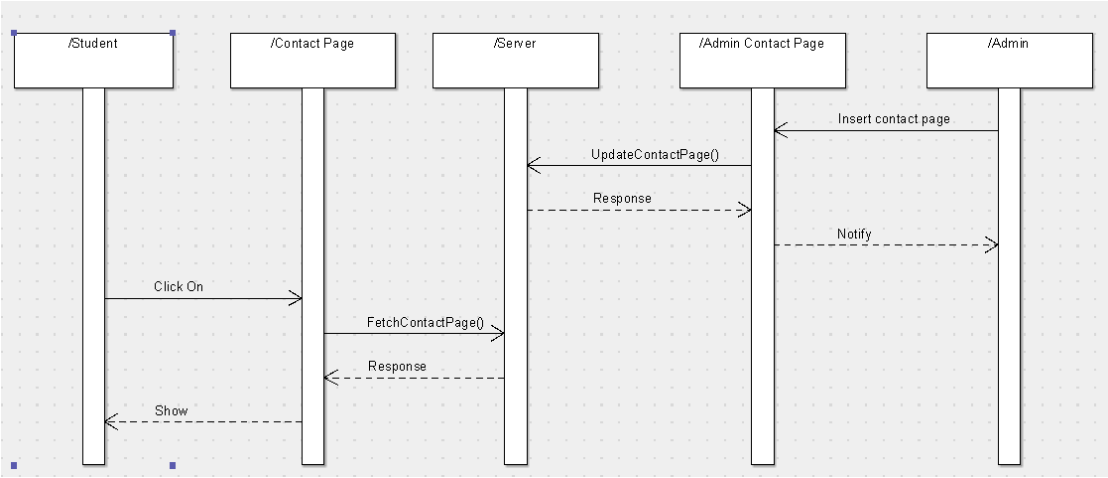
Sequence Diagram: Student Upload



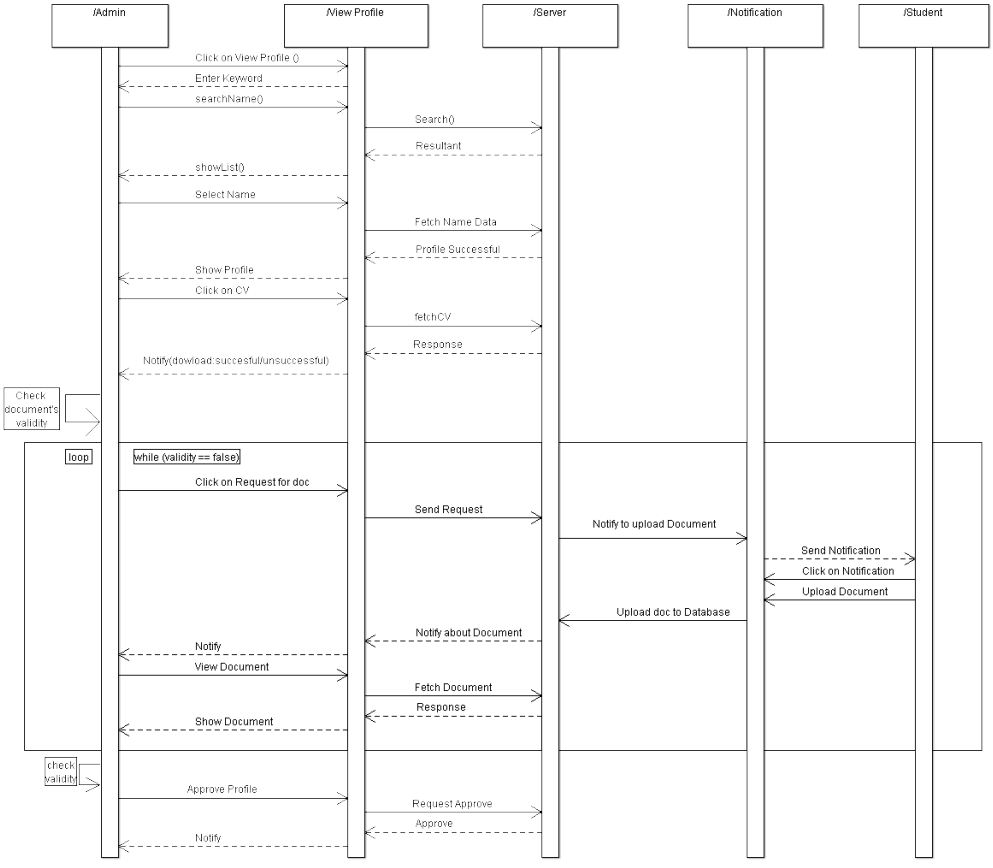
Sequence Diagram: View Job Listing



Sequence Diagram: View Update Contact Page

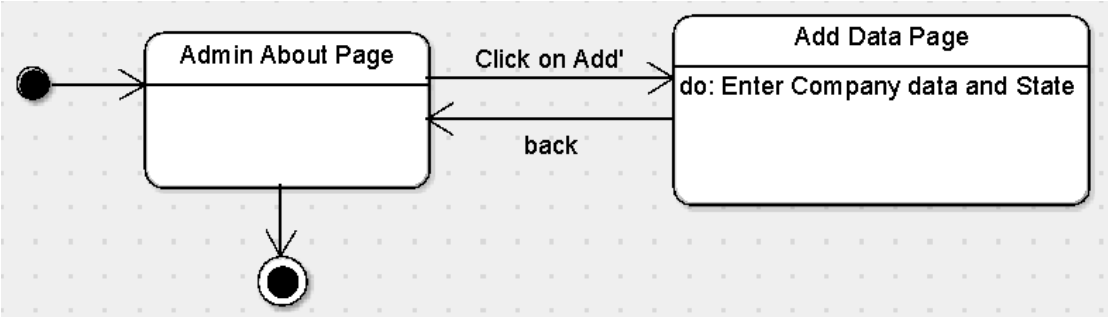


Sequence Diagram: ViewProfile

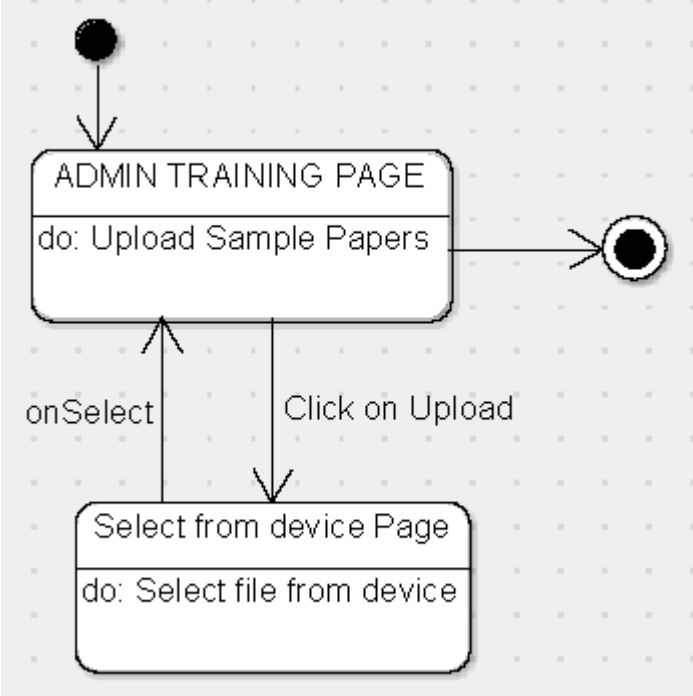


State Diagrams:

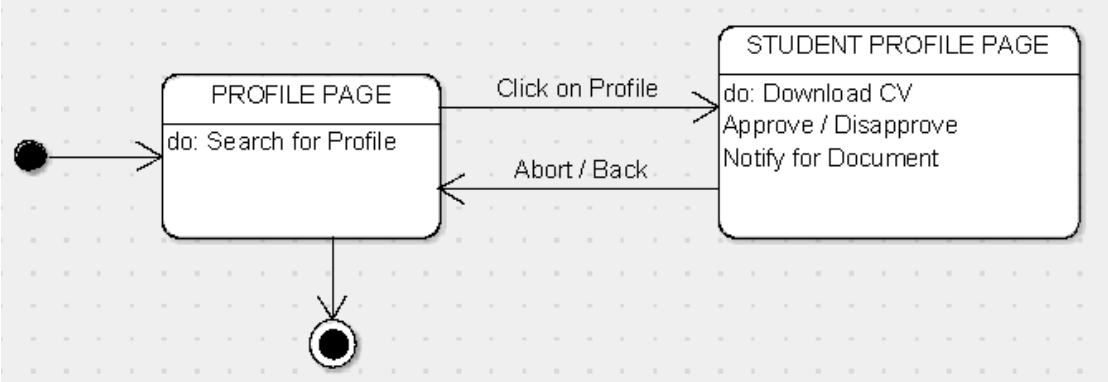
State Diagram: Admin About Page



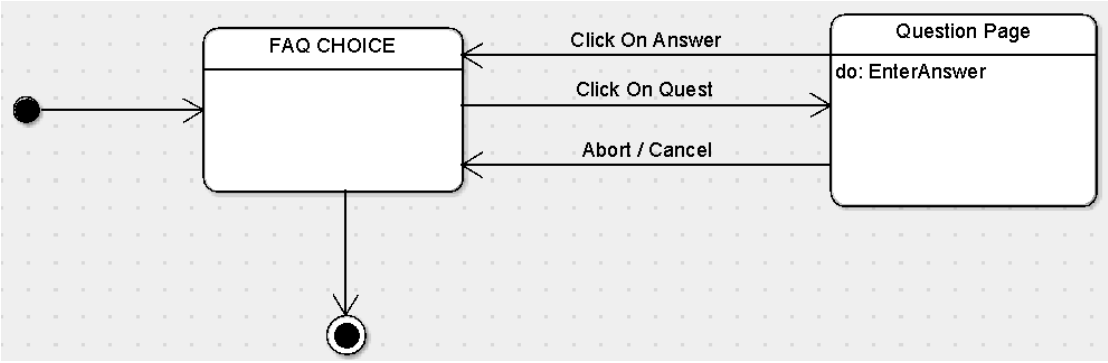
State Diagram: Admin Side Training



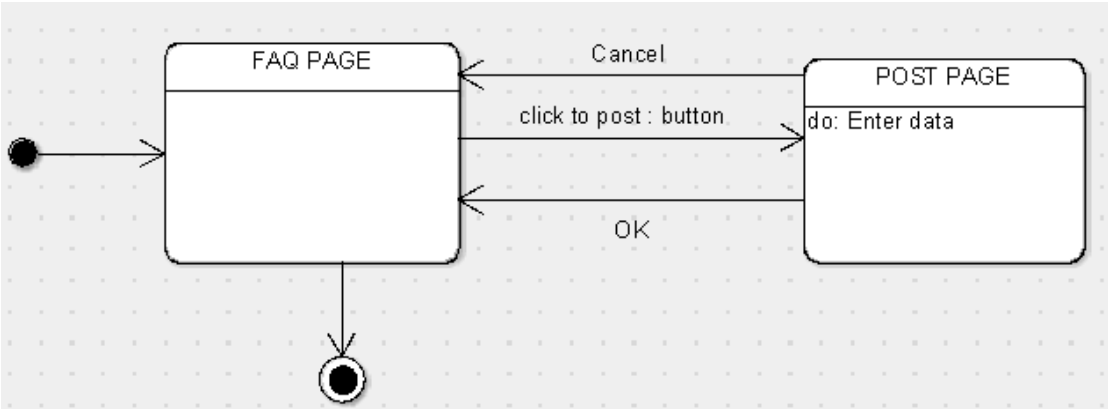
State Diagram: Admin View Profile



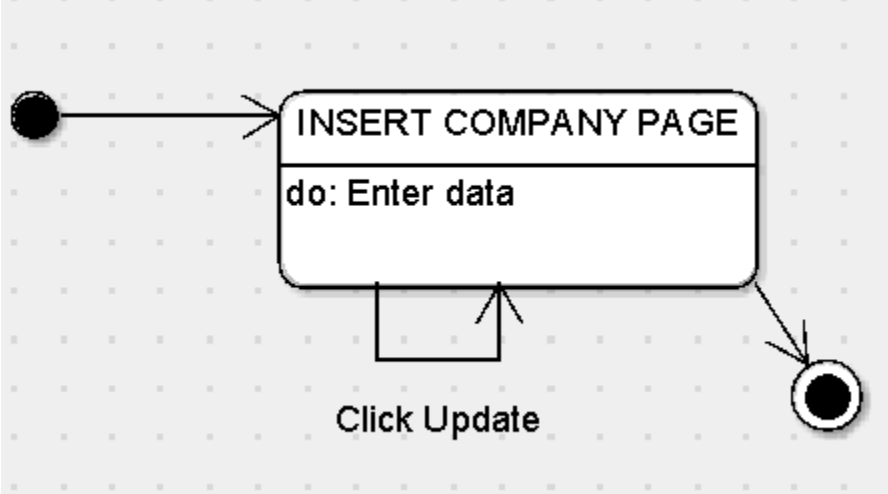
State Diagram: FAQ Admin



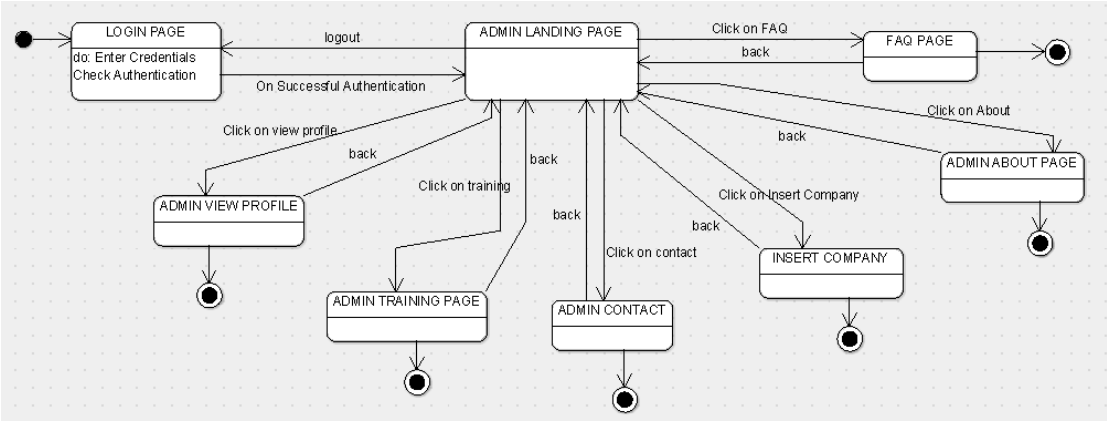
State Diagram: FAQ



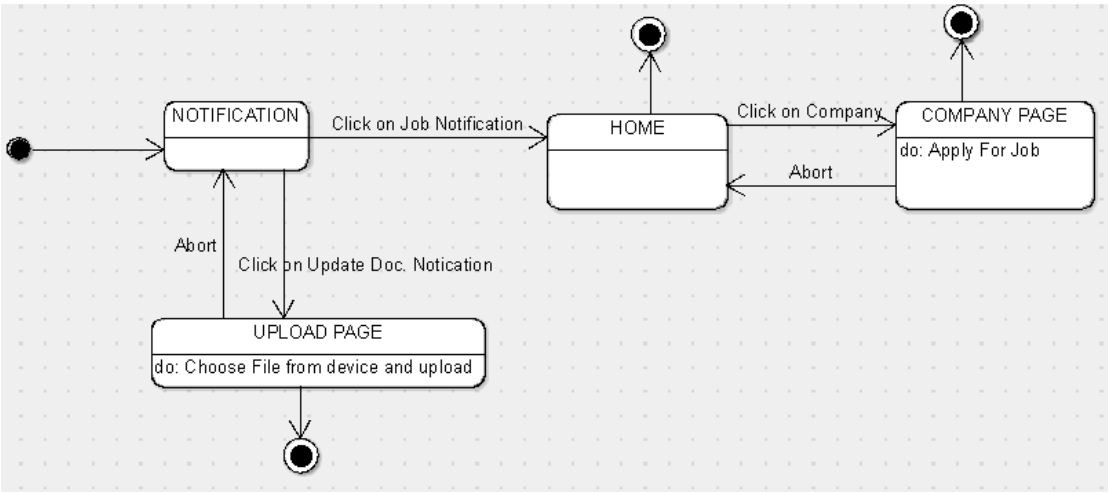
State Diagram: Insert Company



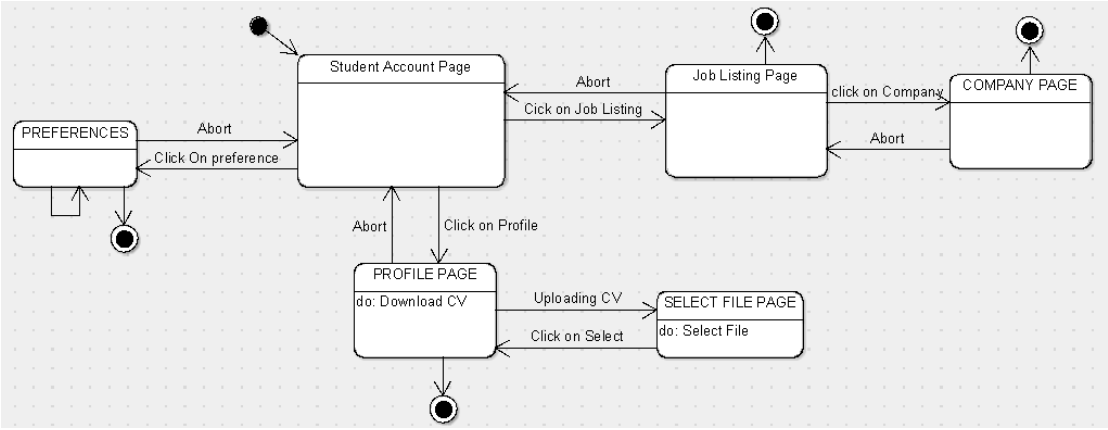
State Diagram: Login Admin



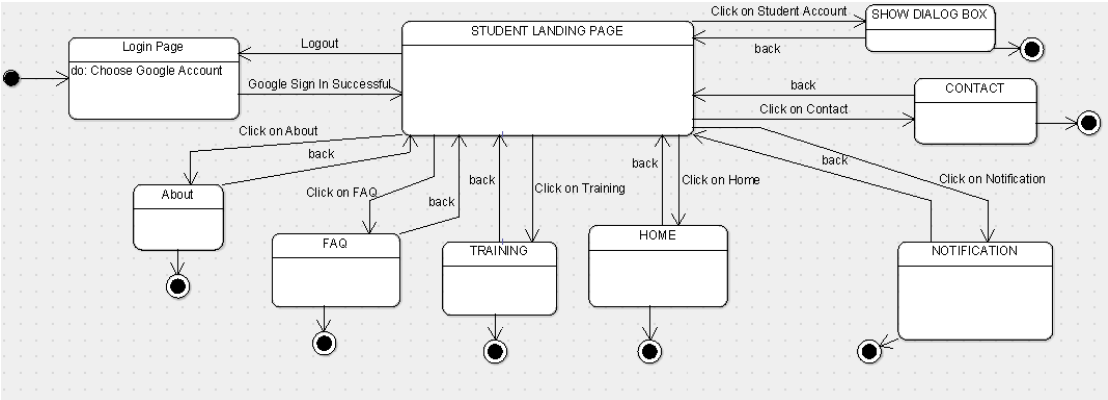
State Diagram: Notification and Home



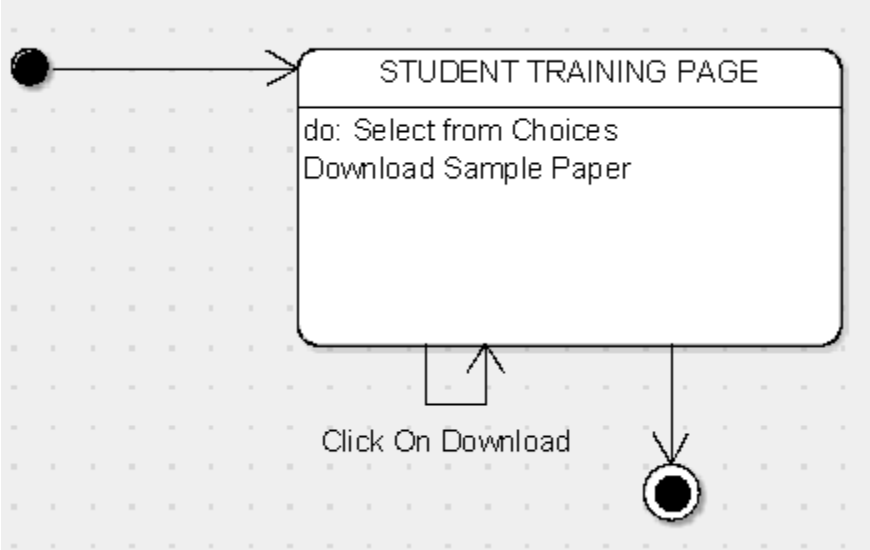
State Diagram: Student Account Page



State Diagram: Student Login



State Diagram: Student Side training



3.1 Logical Architecture Description

3.1.1 Class Diagram explanation:

Most of the classes extends AppCompatActivity class which is being shown by **association** linkage. It is being shown by black-coloured diamond. The classes which extends AppCompatActivity are: Login, StudentLanding, AdminLanding, Home, Notifications, ViewProfile_Student, JobListings, Training_Student, StudentAbout, StudentFAQ, AdminFAQ, Training_Admin, ViewProfile_Admin and AdminAbout.

There are also some **composition** linkage, shown by lines with arrow, which signifies that if the parent class is removed, the child class also loses it's existence. These are: PdfAdapter depends on ArrayAdapter<Pdf>, ViewProfile_Student and JobListings depends on StudentAccount, AboutJob depends on Home, and FilePath depends on Training_Admin.

The lines without arrow shows that the connection is **bi-direction**. It signifies that one class just makes instance of other class, but not dependent on each other in any way. These are: Pdf with Training_Student and ViewProfile_Admin, and PdfAdapter with Training_Student and ViewProfile_Admin.

3.1.2 Sequence Diagram:

Arrow line signifies there is a send message taken place. Response is being shown by dotted arrows.

3.1.2.1 AboutPage: Admin puts data on about page regarding company statistics, which later can be viewed by student on their StudentAbout page.

3.1.2.2 FAQ: Student posts question on this page, which later can be viewed by all other students and can be answered by the Admin. On being answered by Admin, the student also gets notification for the same.

3.1.2.3 Home: Admin posts company data through their InsertCompany Page, which later is visible on the Home Page of the students. It also allows students to apply for any job after their profile being approved.

3.1.2.4 Login Page: It allows students to login with their niitUniversity mail domain and Admin to login with the username and password that are being registered in the database already. It loops being on same page until the correct information is not given.

3.1.2.5 Sample Papers: Admin uploads a sample paper through their Training_Admin page, which are later available for download on the student's Training_Student page.

3.1.2.6 Student Update Profile: Student can update their preference choices.

3.1.2.7 Student Upload CV: Student can upload CV by choosing the appropriate file from their device, which then can be viewed by Admin also.

3.1.2.8 View Job Listing: Student views the list of jobs that he/she had applied for, by the data being fetched from the server.

3.1.2.9 View Profile: Admin can search for the name of the student, by the data being searched on the database and shown in response. Then the Admin can also download the student's CV and either approves that profile or sends notification to student for requirement of any document. Approval is given only when no extra requirement of document is there.

3.1.2.10 View/Update Contact: Admin puts up the contact information on the Admin_Contact page, which later can be viewed by students on their Student_Contact page.

3.1.3 State Diagram:

Initial state is being shown by starting with a black dot. Final State is being shown by the black dot surrounded by an empty circle.

3.1.3.1 FAQ: On clicking the post button on FAQ page, it lands on Post page, which requires to input data from the user. The user can either click on Ok, or Cancel button, which lands up the user to the FAQ page again.

3.1.3.2 FAQ Admin: On clicking on any question, it lands up in the Question page, on which the answer can be typed for it. On clicking Answer button, it would again land up the user on FAQ page.

3.1.3.3 Student Account: Three buttons available to be clicked are Job Listing, Preference, and View Profile, clicking on which it lands up the user on their respective pages. The Job listing page allows the user to click on company to land up on company page and return to Job listing on clicking Back. On further clicking back, moves the user to parent page. Preference page allows the user to choose preference and click on Back to return to parent page. Profile page allows user to download CV, and click on uploadCV button to open select File dialog. On clicking Back, the user can return to the parent page.

3.1.3.4Notification+Home: On clicking Job notification, user lands up on Home Page, on which user can click on company to view the company page, or click on back to return to the parent page. From the parent page, on clicking UploadDoc notification, it lands up on Upload page, which allows user to select file from device and upload the file. On clicking Back, it returns to Home Page, and on further clicking on Back, it returns to parent page.

3.1.3.5 Admin View Profile: Allows user to search for the profile from this page, and if clicked on any profile being shown after search, then it lands on the profile page of the student, on which the user can download CV, approve profile, notify for doc, or click on Back to return to parent page.

3.1.3.6 Training: Student training page allows user to choose from files being shown and download any of them by clicking on download button. Admin training page allows user to upload sample paper by clicking on upload button, which then shows up the select File dialog to choose file from the device or press Back to return to parent page.

3.1.3.7 Insert Company: Allows user to enter company data by clicking on update button.

3.1.3.8 Login Admin: Allows user to enter credentials, which are being checked for authentication in the back-end. After being authenticated successfully, it lands up on AdminLanding page. It has various buttons like FAQ, ViewProfile, Training, InsertCompany, and About to land up on their respective pages, and click on Back to return to AdminLanding page. On clicking on Logout, it returns to the parent page.

3.1.3.9 Student Login: Allows user to choose google account, which is being checked for authentication in the back-end. On being authenticated successfully, it lands up the user on the StudentLanding page. It has various buttons like About, FAQ, Home, Notification, Contact and StudentAccount to land up the user to their respective pages, and click on Back to return to StudentLanding page. On clicking Logout, it returns to the parent page.

3.1.3.10 Admin About page: On clicking Add button, it lands up the user to the Add Data page, which takes input the company data and statistics. On clicking Back, it returns to the parent page.

3.2 Class name: Login

Description: This class allows the user to enter the system by authenticating the entered credentials.

3.2.1 Method 1: onCreate()

Input: savedInstanceState , Email, Password

Output: Launch the Activity, SignIn

Method Description:

When an Activity first call or launched then onCreate(Bundle savedInstanceState) method is responsible to create the activity. When ever orientation(i.e. from horizontal to vertical or vertical to horizontal) of activity gets changed the object of Bundle class will save the state of an Activity. Basically Bundle class is used to stored the data of activity whenever above condition occur in app. SetContentView is used to fill the window with the UI provided from layout file. This method takes input as email id and password and as a result opens the landing page if login successful.

3.2.2 Method 2: Admin_Login()

Input: view email ,password

Output: Admin landing page of login successful

Method Description:

This method takes input as email and password from the user and check whether it is authorized login or not and if the result is successful it leads to another activity page.

3.2.3 Method 3: user_Login()

Input: SignIn through Google

Output: Student landing page of login successful

Method Description:

Intents are asynchronous messages which allow application components to request functionality from other Android components. This method allows student to login through their Google account. SignIn Intent stores the resultant value by checking whether the sign from Google is authorized login or not . If the user is not a Google client then the login will fail and if it is a authorized login then it will lead to student landing page.

3.3 Class Name: Admin
Description

This class enables the admin to enter into the subsystem (Landing page) after authenticating the entered credentials.

3.3.1 Method 1: onClick(View view)

Input: view- object of view

Output: Landing to Main Activity after successful logout

Method Description:

This method takes input as object of class view when Logout button is pressed, it sign out takes place closing the current activity. It finishes the current task and leads to the Main Activity Page.

3.3.2 Method 2: OnCreate(Bundle savedinstancestate)

Input: bundle object

Output: opens a layout

Description: Calls method ‘onCreate (saveInstanceState)’ of ‘super’ class and method ‘setContentView (layout_name)’.

3.3.3 Method 3: FAQ()

Input: savedInstanceState , OnClickListener on FAQ button

Output: Launch the Activity

Description-

1) When a user click on faq button, It will call a method from button class setOnClickListener.
2) Inside this method(setOnClickListener) It will create a object of Intent class.
3) Intent class object opens a new activity, which will get its content from “faq.java” file.
4) Then a new page is opened using object of Intent class like - startActivityForResult(intent, 0);

3.3.4 Method 4: View profile()

Input: savedInstanceState , OnClickListener on View profile button

Output: Launch the Activity

Description-

1) When a user click on View profile button, It will call a method from button class setOnClickListener.
2) Inside this method(setOnClickListener) It will create a object of Intent class.
3) Intent class object opens a new activity, which will get its content from "Viewprofile.java" file.
4) Then a new page is opened using object of Intent class like - startActivityForResult(intent, 0);

3.3.5 Method 5: Training data

Input: savedInstanceState , OnClickListener on Training data button

Output: Launch the Activity

Description-

1) When a user click on Training data button, It will call a method from button class setOnClickListener.
2) Inside this method(setOnClickListener) It will create a object of Intent class.
3) Intent class object opens a new activity, which will get its content from "Trainingdata.java" file.
4) Then a new page is opened using object of Intent class like - startActivityForResult(intent, 0);

3.3.6 Method 6: Insert company

Input: savedInstanceState , OnClickListener on Insert company button

Output: Launch the Activity

Description-

1) When a user click on Insert company button, It will call a method from button class setOnClickListener.
2) Inside this method(setOnClickListener) It will create a object of Intent class.
3) Intent class object opens a new activity, which will get its content from "Insertcompany.java" file.
4) Then a new page is opened using object of Intent class like - startActivityForResult(intent, 0);

3.3.7 Method 7: Approve profile

Input: savedInstanceState , OnClickListener on Approve profile button

Output: Launch the Activity

Description-

1) When a user click on Approve profile button, It will call a method from button class setOnClickListener.
2) Inside this method(setOnClickListener) It will create a object of Intent class.
3) Intent class object opens a new activity, which will get its content from "Approveprofile.java" file.
4) Then a new page is opened using object of Intent class like - startActivityForResult(intent, 0);

3.3.8 Method 8: About

Input: savedInstanceState , OnClickListener on About button

Output: Launch the Activity

Description-

1) When a user click on About button, It will call a method from button class setOnClickListener.
2) Inside this method(setOnClickListener) It will create a object of Intent class.
3) Intent class object opens a new activity, which will get its content from "About.java" file.
4) Then a new page is opened using object of Intent class like - startActivityForResult(intent, 0);

3.3.9 Method 9: Contact

Input: savedInstanceState , OnClickListener on Contact button

Output: Launch the Activity

Description-

1) When a user click on Contact button, It will call a method from button class setOnClickListener.
2) Inside this method(setOnClickListener) It will create a object of Intent class.

3) Intent class object opens a new activity, which will get its content from “Contact.java” file.
4) Then a new page is opened using object of Intent class like - startActivityForResult(intent, 0);

3.4 Class Name: Notification
Description:

This class basically handles notification. When the student logs in to the app, the notification tab will display the new activity reports in the form of notification badges.

This class also takes care of the Notifications push from the Admins side, i.e. in case Admin require for any document from the student or any important communication needs to be conveyed to the student, admin can push notification and this will notify the concerned person.

3.4.1 Method 1: Notification()

Input: mAuthListener

Output: handle authentication state changes

Method Description:

When activity start getting visible to user then onStart() will is called. Inside this method parent onStart constructor and addAuthSateListener is invoked. Since it is a Login activity so several state may occur like

1. Right after the listener has been registered
2. When a user is signed in
3. When the current user is signed out
4. When the current user changes

Hence this method listen to the listener and handle authentication state changes

3.4.2 Method: getNotification(Bundle SaveInstanceState)

Input: savedInstanceState – object of bundle

Output: Displaying the notification

Method Description-

This method is used to display the notification. It take input as the object of bundle class. Whenever the notification from Main Activity

3.5 Class Name: Upload_SamplePapers
Class Description:
This class consists of a method named as uploadMultipart() which carries out the upload task. It also consists of storage persmission which is required for this project as it involves reading Android storage.

3.5.1 Method 1: uploadMultipart ()

Input: Name and path of the file name.

Output: Handles uploading pdf

Method Description:
This method is responsible for pdf upload. We need the full pdf path and the name for the pdf in this method. We are also using Filepath class to get the path of file chosen from device.

3.5.2 Method 2: **showFileChooser ()**

Input: Option to choose from location of file to upload from device.

Output: Helps to choose file from device.

Method Description:
This method shows file chooser using intent.

3.5.3 Method 3: **onClick (View view)**

Input: view

Output: Perform action after button being clicked.

Method Description:
This is an overriden method from View.OnClickListener interface.This is to link the student page for training to the admin page, in order to view the uploaded documents and to be able to download them also.

3.5.4 Method 4: **requestStoragePermission ()**

Input: NULL

Output: Ask for permission.

Method Description:
This method is for requesting permission to external storage of the device.

3.5.5 Method: onCreate (Bundle savedInstanceState)

Input: Bundle object and button clicks.

Output: Sets onClickListener on buttons.

Method Description:
This is an overriden method from AppCompatActivity class. It opens up the layout, checks if permission for storage is given or not, and then attaches click event on the buttons.

3.6 Class Name: Download_SamplePapers

Class Description:
This class includes ListView, ArrayList of PDFs, ProgressDialog, PdfAdapter, and their initializations. After fetching pdfs from the server, list of uploaded pdfs are shown here, clicking on which they can also be downloaded to the external storage of the device.

3.6.1 Method: onCreate (Bundle savedInstanceState)

Input: Bundle object and button clicks.

Output: Downloads pdf.

Method Description:
This method fetches data from server and show them in ListView which are clickable in order to perform click event for downloading. It also uses a class ‘Pdf’, which has only ‘get’ and ‘set’ methods for variables as pdf name and pdf url name.

3.7 Class Name: ListOfAppliedJobs

Class Description:
This class fetches data from the server about the companies, the student had applied for, in ListView format.

3.7.1 Method: onCreate (Bundle savedInstanceState)

Input: Bundle object.

Output: Shows companies the student had applied for.

Method Description:

This class uses a php url to fetch data from, and converts the fetched data to string using BufferedReader and StringBuilder classes. Then Json data is parsed and shown in listView format.

3.8 Class Name: ViewStudent's_Profile_ByAdmin

Class Description:
This class fetches data from ERP server to fill the student's details automatically and show them. It also allows student to upload CV and Aspiration Form, and choose preferences within this page itself.

3.8.1 Method: onCreate (Bundle savedInstanceState)

Input: Bundle object, data fetched from ERP server, and uploading some documents.

Output: Shows complete profile of the student.

Method Description:
This method allows the user (Students of NIIT university using this application), to view their profile which is fetched from NU ERP. This also lets them to upload their resume and aspiration form.

3.9 Class Name: View_CompStatistics_History

3.9.1 Method 1: **getCompData()**

Input: None
Output: Display Company names coming in college.
Description: Will enable to fetch the data of the company queried for.

3.9.2 Method 2: **showStats()**

Input: Student profile info/CV .
Output: The user's cv/ information is passed to the server(For Admin Approval).
Description: Will enable to fetch the data of the user's details queried for.

3.10 Class Name: AboutCompany_Detail

Class Description:
This class allows admin to search for student's profile and then view complete profile of the student. Admin can also download the CV and Aspiration Form uploaded by student and thus approve the profile if he finds everything at place.

3.10.1 Method: onCreate (Bundle savedInstanceState)

Input: Bundle object, data fetched from server.

Output: Shows complete profile of the student.

Method Description:
This method allows the admin to search for the student profile and download their CV and aspiration form to send approval or notify for any required document.

3.11 Class Name: PostFAQ_ByStudent

Class Description :

This class defines all the methods from the student side to be used to **view the questions and answers** and **post a question**.
This class consists of 2 methods :

1. private void onCreate()
2. private void askQuestion()

3.11.1 Method 1: private void onCreate()

Input: None

Output: This method will display the questions asked till now and the answers in a List View.

Method Description : Method responsible for displaying the questions and answers.

3.11.2 Method 2: private void askQuestion()

Input: The question string asked by the student.

Output: The question asked is sent to the server.

Method Description: to send a question asked by the student to the server at the backend.

3.12 Class Name: ListToApplyForCompanies

Class Description – This home class will let you see status of different companies coming to your college, like how many students are applying for this company. You can even apply/ enroll for companies from this home class . This home tab can be accessed easily after students login.

3.12.1 Method 1: GetCompaniesNames()

Input: None

Output: Display Company names coming in college.

Description- This method will give you full details of the companies coming in college and what are their requirements for applying for job in particular company.

3.12.2 Method 2: ApplyForCompanies()

Input: Student profile info/CV .

Output: The user's cv/ information is passed to the server(For Admin Approval).

Description- It will show all available companies , in which students can apply.

1) First it will ask for a user input to choose a company from the list available.
2) Then it will take user profile data like CV, and other useful information and stores it in the servers, so that applied company/ Placement Team can approve their registration done.

3.13 Class: Insert_CompStatistics_History

3.13.1 Method 1: updateCompanyDetails()

Input: None
Output: All the questions being asked by students.
Description: Will enable the functionality to update the company details in the database.

3.13.2 Method 2: onCreate(Bundle savedInstanceState)

Input: The question id of the question asked.
Output: The answer to the question is passed to the server.
Description: Overridden method. The main method of the class.

3.14 Class name: Student

Description: This class consists of all the methods and initialization of the Navigation drawer present in all the activities of the application. The navigation drawer is basically the menu which slides from the left towards the right upon clicking the action button situated on the top left corner of the application or by just providing sliding input on the screen from left to right.

3.14.1 Method 1: addDrawerItems()

Input: None

Output: Initializes the various options in Navigation Drawer.

Description: This method basically initializes the various options seen in the navigation drawer.

3.14.2 Method 2: setupDrawer()

Input: None

Output: Sets up the Action Bar title whenever the Navigation drawer is opened or closed.

Description: This method basically modifies the activity when the drawer is opened and when the drawer is closed.

3.14.3 Method 3: onOptionsItemsSelected(MenuItem item)

Input: MenuItem item (Object item of class MenuItem)

Output: The activity to be launched or the method to be executed upon clicking upon any of the Navigation Drawer options.

Description: This method will handle all the action bar item clicks.

3.14.4 Method 4: onPostCreate(Bundle savedInstanceState)

Input: Bundle savedInstanceState

Output: Call the sync state method of the ActionBarDrawerToggle object.

Description: Overriden method. To basically call the syncState() method.

3.14.5 Method 5: onConfigurationChanged(Configuration newConfig)

Input: Configuration newConfig

Output: Calls the onConfigurationChanged method of the ActionBarDrawerToggle object.

Description: Overridden method. To basically call the onConfigurationChanged() method.

3.14.6 Method 6: onCreate(Bundle savedInstanceState)

Input: Bundle savedInstanceState

Output: Sets the activity in order, basically the main function of this class.

Description: Overridden method. The main method of the class.

3.15 Class name: Answer_FAQ

Description: This class consists of all the methods to be used by the admin to view the FAQ section, show the questions and answer the questions.

3.15.1 Method 1: showQuestions()

Input: None

Output: All the questions being asked by students.

Description: Method to display all the questions being asked by students using the volley library.

3.15.2 Method 2: answerQuestions(int qID)

Input: The question id of the question asked.

Output: The answer to the question is passed to the server.

Description: Method to answer the user asked question.

4.0 Execution Architecture

Runtime environment required is any device supporting Android Operating System with the minimum version of Ice-cream Sandwich, Android Studio as a deployment platform.

4.1 Reuse and relationships to other products

NIL

5.0 Design decisions and tradeoffs

The design decision to use two screens separately for admin and student is to provide encapsulation. It may have been possible to get all the information on one screen. However, using two screens will keep the data of admin separate from the data being accessed by students.

A possible tradeoff when considering links is to use buttons instead of items in the menu. This design decision - to use buttons for navigating between screens - is to enhance visibility. Text links in the menu bar located at the bottom of the PDA's screen can be hard to see. The tradeoff for buttons with descriptive labels rather than text links in the menu bar will be that navigation from screen to screen will be easier. Descriptive labels will let the user know where he is navigating. Buttons are larger

than the text links located in the menu bar of the PDA. Therefore, it is easier for the user to locate the mechanisms needed to navigate from screen to screen.

6.0 Pseudocode for components

6.0.1 Class Name: Login

Method 1: **onCreate()**

Pseudo-code:

Input: savedInstanceState , Email, Password

Output: Launch the Activity, SignIn

1. super.onCreate(savedInstanceState);
2. setContentView(R.layout.activity_main);
3. button = stores the ID of Google sign-in button
4. mAuth = get FirebaseAuth instance
5. editTextEmail = stores the ID of Admin editTextEmail
6. editTextPassword = stores the ID of Admin editTextPassword
7. buttonSignIn = stores the ID of Admin SignInButton
8. progressDialog = create new progressDialog object
9. button.setOnClickListener(new View.OnClickListener()
10. public void onClick(View v)
11. signIn()) // Admin sign in
10. button.setOnClickListener(new View.OnClickListener()) // Google sign in
11. mAuthListener = new FirebaseAuth.AuthStateListener()
12. public void onAuthStateChanged(FirebaseAuth firebaseAuth)
13. if firebaseAuth.getCurrentUser() != null then
14. start another Activity
15. end if
12. if firebaseAuth.getCurrentUser() != null then
13. finish();
14. start another Activity
15. end if

Method 2: **Admin_Login()**

Pseudo-code:

Input: view, email, password

Output: Admin landing page of login successful

1. String email = get email from the user
2. String password = get password from the user
3. if string email!=null then
4. print the email of the user
5. return
6. end if
3. if string password !=null then
4. print the password of the user
5. return
6. end if
7. Shows progress print “Login in please wait”
8. verify (email, password) by sending it to firebaseAuth
9. result= get result from firebaseAuth
10. public void onComplete()
11. Dismiss the progress dialog
12. if result is Successful then
13. finish();
14. start another Activity leading to Admin Landing page
15. end if

Method 3: **user_Login()**

Pseudo-code:

Input: SignIn through Google

Output: Student landing page of login successful

1. Intent signInIntent = get signIn Intent from Google for the given Google client
2. if signInIntent==Google client then
3. start another activity leading to student landing page

4. else

5. print message login unsuccessful

6. end if

6.0.2 Class Name: Notification

Method 1: **Notification()**

Pseudo-code:
Input: mAuthListener
Output: handle authentication state changes
1. String strtitle = set notification title
2. String strtext = set notification text

Method 2: **getNotification(Bundle SaveInstanceState)**

Pseudo-code:

Input: savedInstanceState – object of bundle

Output: Displaying the notification

1. super.onCreate(savedInstanceState);

2. setContentView(R.layout.activity_main);

4. NotificationManager notificationmanager = Creating new notificationmanager

5. Intent i = Retrieve data from Main Activity

6. title = i.getStringExtra("title");

7. text = i.getStringExtra("text");

8. txttitle.setText(title);

9. txttext.setText(text);

6.0.3 Class Name: Upload_SamplePapers

Method 1: **uploadMultipart ()**

Pseudo-code:
Input: Name and path of the file name.
Output: Handles uploading pdf

1. Initialize string ‘str’ to ‘uploadFileName’ and string ‘path’ to pathName given

by “FilePath.getpath ()”.
2. Notify if path is NULL.
3. ELSE:
4. TRY:
5. A) Initialize string ‘uploadid’ to ‘UUID’.
6. B) Make ‘MultipartUploadRequest’ object to start uploading.
7. CATCH: Handle ‘EXCEPTION’.

Method 2: **showFileChooser ()**

Pseudo-code:
Input: Option to choose from location of file to upload from device.
Output: Helps to choose file from device.

1. Make an ‘Intent’ object and set it’s Type to ‘application/pdf’, and set ‘Action’ to ‘ACTION_GET_CONTENT’.
2. Call method ‘startActivityForResult’ to start Intent.

Method 3: **onClick (View view)**

Pseudo-code:
Input: view
Output: Perform action after button being clicked.

1. Call method ‘showFileChooser’ if ‘view=buttonChoose’.
2. Call method ‘uploadMultipart’ if ‘view=buttonUpload’.
3. Call method ‘getPdf ()’ if ‘view=buttonFetch’.

Method 4: **requestStoragePermission ()**

Pseudo-code:
Input: NULL
Output: Ask for permission.

1. Check if ‘selfPermission’ is ‘granted’ or not.
2. Show appropriate message if user has denied permission at first attempt.
3. Ask for permission by calling method ‘requestPermission’.

Method 5: **onCreate (Bundle savedInstanceState)**

Pseudo-code:
Input: Bundle object and button clicks.
Output: Sets onClickListener on buttons.

1. Call method ‘onCreate (saveInstanceState)’ of ‘super’ class.
2. Call method ‘setContentView (layout_name)’.
3. Call method ‘requestForPermission’.

4. Initialize buttons ‘buttonUpload’, ‘buttonChoose’, and ‘buttonFetch’ by choosing appropriate ids to map to.
5. Make ‘EditText’ object.
6. Set ‘onClickListener’ on buttons created.

6.0.4 Class Name: Download_SamplePapers

Method 1: **onCreate (Bundle savedInstanceState)**

Pseudo-code:
Input: Bundle object and button clicks.
Output: Downloads pdf.

1. Call method ‘onCreate (saveInstanceState)’ of ‘super’ class.
2. Call method ‘setContentView (layout_name)’.
3. Call method ‘setOnClickListener’ from ListView object, and override the ‘onItemClick’ method as:
4. A) Initialize ‘Pdf’ object by ‘parent.getItemAtPosition’.
5. B) Create intent & call ‘setAction’, ‘addCategory’ and ‘setData’ on it.
6. C) Call method ‘startActivity (intent)’.
7. Code to fetch pdf:
8. A) Call method ‘setMessage(“”)’ by ‘ProgessDialog’ object.
9. B) Call method ‘show()’ by ‘ProgessDialog’ object.
10. C) Initialize string by Pdf url fetched, with parameter overriden method ‘onResponse(string)’. Define ‘onResponse(string)’ as:
11. i. Call method ‘dismiss’ by ‘ProgessDialog’ object.
12. ii. TRY:
13. a) Initialize JSONObject object
14. b) FOR: each jsonArray.length()
15. c) Initialize JSONArray object.
16. Store name of pdf and url in string.
17. d) Make ‘PdfAdapter’ object and set listview to show fetched pdfs.
18. CATCH: Handle ‘JSONException’.
19. iii. Call ‘Response.ErrorListener()’ and ovveride a method ‘onErrorResponse (volleyError)’.
20. D) Initialize ‘RequestQueue’ object by ‘Volley.newRequestQueue (this)’.
21. E) Call ‘request.add (stringRequest)’.

6.0.5 Class Name: ListOfAppliedJobs

Method 1: **onCreate (Bundle savedInstanceState)**

Pseudo-code:
Input: Bundle object.
Output: Shows companies the student had applied for.

1. Call method ‘onCreate(savedInstanceState)’ of ‘super’ class.

2. Call method ‘setContentView(layout_name)’.
3. Initialize string variable and ‘InputString’ object to NULL.
4. TRY:
5. A)Make ‘DefaultHttpClient()’ object and set ‘HttpPost’ object to point to a php url to fetch data from.
6. B)Make ‘HttpResponse’ object ‘response’ and ‘HttpEntity’ object.
7. CATCH: Handle ‘Exception’.
8. TRY:
9. A) Make ‘BufferedReader’ object to convert the response to string.
10. B) Use ‘StringBuilder’ object also.
11. CATCH: Handle ‘Exception’.
12. TRY:
13. A) Make ‘JSONArray’ object and pass the converted string to it.
14. B) Make ‘ArrayList<String>’ object.
15. C) FOR: each ‘i’ in ‘jArray.length()’
16. i. ‘JSONObject json_data = jArray.getJSONObject (i)’.
17. ii. ‘String str = String.valueOf (json_data.getInt (“CompanyData”))’.
18. iii. ‘arrayList.add (str)’.
19. D) ‘ArrayAdapter items = new ArrayAdapter (this, arrayList)’.
20. E) ListView listView = assign it to an ‘id’.
21. F) ‘listView.setAdapter (items)’.
22. CATCH: Handle ‘JSONException’.

6.0.6 Class Name: ViewStudent’s_Profile_ByAdmin

Method 1: **onCreate (Bundle savedInstanceState)**

Pseudo-code:
Input: Bundle object, data fetched from ERP server, and uploading some documents.
Output: Shows complete profile of the student.

1. Call method ‘onCreate(savedInstanceState)’ of ‘super’ class.
2. Call method ‘setContentView(layout_name)’.
3. Initialize string variable and ‘InputString’ object to NULL.
4. TRY:
5. A) Make ‘DefaultHttpClient()’ object and set ‘HttpPost’ object to point to the ERP database php url to fetch data from.
6. B) Make ‘HttpResponse’ object ‘response’ and ‘HttpEntity’ object.
7. CATCH: Handle ‘Exception’.
8. TRY:
9. A) Make ‘BufferedReader’ object to convert the response to string.
10. B) Use ‘StringBuilder’ object also.
11. CATCH: Handle ‘Exception’.
12. TRY:
13. A) Make ‘JSONArray’ object and pass the converted string to it.
14. B) Make ‘ArrayList<String>’ object.
15. C) FOR: each ‘i’ in ‘jArray.length()’
16. i. ‘JSONObject json_data = jArray.getJSONObject (i)’.
17. ii. ‘String str = String.valueOf (json_data.getInt (“StudentData”))’.

18. iii. ‘arrayList.add (str)’.
19 D) ‘ArrayAdapter items = new ArrayAdapter (this, arrayList)’.
20. E) ListView listView = assign it to an ‘id’.
21. F) ‘listView.setAdapter (items)’.
22. CATCH: Handle ‘JSONException’.
23. Make ‘Button’ objects ‘UploadCv’ and ‘UploadAspirationForm’.
24. Make ‘EditText’ object to read entered name for CV and Aspiration form.
25. Check if ‘selfPermission’ is ‘granted’ or not.
26. Show appropriate message if user has denied permission at first attempt.
27. Ask for permission by calling method ‘requestPermission’.
28. IF permission is granted, THEN:
29. Set ‘onClickListener’ on the buttons, which has an overriden method ‘onClick(View view)’ defined as:
30. A) Make an ‘Intent’ object and set it’s Type to ‘application/pdf’, and set ‘Action’ to ‘ACTION_GET_CONTENT’.
31. B) Call method ‘startActivityForResult’ to start Intent.
32. C) Initialize string ‘str’ to ‘uploadFileName’ and string ‘path’ to pathName given by “FilePath.getpath ()”.
33. D) Notify if path is NULL.
34. ELSE:
35 TRY:
36. i. Initialize string ‘uploadid’ to ‘UUID’.
37. ii. Make ‘MultipartUploadRequest’ object to start uploading.
38. CATCH: Handle ‘EXCEPTION’.

6.0.7 Class Name: View_CompStatistics_History

Method 1: **getCompData()**

Pseudo-Code:
Input: None
Output: Display Company names coming in college.
1. JSONObject obj = new JSONObject(response);
2. JSONArray nameArray = obj.getJSONArray("CompaniesNames");
3. for (int i = 0; i < nameArray.length(); i++)
a. JSONObject nameObject = nameObject.getJSONObject(i);
b. CompaniesNames names = new names(heroObject.getString("CompaniesNames"));
c. namesList.add(names);
4. End for
5. ListViewAdapter adapter = new ListViewAdapter(namesList, getApplicationContext());
6. listView.setAdapter(adapter);

Method 2: **showStats()**

Pseudo-Code:

Input: Student profile info/CV .
Output: The user's cv/ information is passed to the server(For Admin Approval).

```
1.      String info = infoTextBox.getText();
2.      RequestQueue queue = Volley.newRequestQueue(this);
3.      String url = String.format("Address of the server"), StringRequest request =
new StringRequest(Request.Method.POST, url,  new Response.Listener<String>()
4.      public void onResponse(String response)
5.              new ApiErrorListener(this));
6.              queue.add(request);
```

6.0.8 Class Name: AboutCompany_Detail

Method 1 : **onCreate (Bundle savedInstanceState)**

Pseudo-Code:
Input: Bundle object.
Output: Shows list of companies, their info and their statistics in reference to the number of students applied, selected.

1. Call method 'onCreate(savedInstanceState)' of 'super' class.
2. Call method 'setContentView(layout_name)'.
3. Initialize string variable and 'InputString' object to NULL.
4. TRY:

```
5.      A) Make 'DefaultHttpClient()' object and set 'HttpPost' object to point to a
php url to fetch data from.
6.      B) Make 'HttpResponse' object 'response' and 'HttpEntity' object.
7. CATCH: Handle 'Exception'.
8. TRY:
9.      A) Make 'BufferedReader' object to convert the response to string.
10.     B) Use 'StringBuilder' object also.
11. CATCH: Handle 'Exception'.
12. TRY:
13.     A) Make 'JSONArray' object and pass the converted string to it.
14.     B) Make 'ArrayList<String>' object.
15.     C) FOR: each 'i' in 'jArray.length()'
16.             i. 'JSONObject json_data = jArray.getJSONObject (i)'.
17.             ii. 'String str = String.valueOf (json_data.getInt ("CompanyData"))'.
18.             iii. 'arrayList.add (str)'.
19.     D) 'ArrayAdapter items = new ArrayAdapter (this, arrayList)'.
20.     E) ListView listView = assign it to an 'id'.
21.     F) 'listView.setAdapter (items)'.
22. CATCH: Handle 'JSONException'.
```

6.0.9 Class Name: PostFAQ_ByStudent

Method 1: **private void onCreate()**

Pseudo-Code:
Input: None
Output: This method will display the questions asked till now and the answers in a List View.

1. Call method ‘onCreate(savedInstanceState)’ of ‘super’ class.
2. Call method ‘setContentView(layout_name)’.
3. Initialize string variable and ‘InputString’ object to NULL.
4. TRY:
5. A) Make ‘DefaultHttpClient()’ object and set ‘HttpPost’ object to point to a php url to fetch data from.
6. B) Make ‘HttpResponse’ object ‘response’ and ‘HttpEntity’ object.
7. CATCH: Handle ‘Exception’.
8. TRY:
9. A) Make ‘BufferedReader’ object to convert the response to string.
10. B) Use ‘StringBuilder’ object also.
11. CATCH: Handle ‘Exception’.
12. TRY:
13. A) Make ‘JSONArray’ object and pass the converted string to it.
14. B) Make ‘ArrayList<String>’ object.
15. C) FOR: each ‘i’ in ‘jArray.length()’
16. i. ‘JSONObject json_data = jArray.getJSONObject (i)’.
17. ii. ‘String str = String.valueOf (json_data.getInt (“QuetionAnswers”))’.
18. iii. ‘arrayList.add (str)’.
19. D) ‘ArrayAdapter items = new ArrayAdapter (this, arrayList)’.
20. E) ListView listView = assign it to an ‘id’.
21. F) ‘listView.setAdapter (items)’.
22. CATCH: Handle ‘JSONException’

Method 2: **private void askQuestion()**

Pseudocode :
Input: The question string asked by the student.
Output: The question asked is sent to the server.

1. String question = Question that the student wants to post on the forum.
2. RequestQueue queue = Volley.newRequestQueue(this);
3. String url = String.format("Address of the server"),
   StringRequest request = new StringRequest(Request.Method.POST, url,
   new Response.Listener<String>()
   public void onResponse(String response)
   new ApiErrorListener(this));
4. queue.add(request);

6.0.10 Class Name: ListToApplyForCompannies

Method 1: **GetCompaniesNames()**

Pseudo-Code:
Input: None
Output: Display Company names coming in college.

```
1.      JSONObject obj = new JSONObject(response);
2.       JSONArray nameArray = obj.getJSONArray("CompaniesNames");
3.      for (int i = 0; i < nameArray.length(); i++)
            a. JSONObject nameObject = nameObject.getJSONObject(i);
            b. CompaniesNames names = new
        names(heroObject.getString("CompaniesNames"));
            c. namesList.add(names);
4.      End for
5.      ListViewAdapter adapter = new ListViewAdapter(namesList,
getApplicationContext());
6.       listView.setAdapter(adapter);
```

Method 2: **ApplyForCompanies()**

Pseudo-Code:
Input: Student profile info/CV .
Output: The user's cv/ information is passed to the server(For Admin Approval).

```
1.      String info = infoTextBox.getText();
2.      RequestQueue queue = Volley.newRequestQueue(this);
3.      String url = String.format("Address of the server"), StringRequest request =
new StringRequest(Request.Method.POST, url,  new Response.Listener<String>()
4.      public void onResponse(String response)
5.            new ApiErrorListener(this));
6.            queue.add(request);
```

6.0.11 Class: Admin

Method 1: **FAQ()**

Pseudo-Code:
Input: savedInstanceState , OnClickListener on FAQ button
Output: Launch the Activity

```
1. super.onCreate(savedInstanceState);
2. setContentView(R.layout.activity_main);
3.          button = stores the ID of FAQ button
4.                  button1.setOnClickListener(new View.OnClickListener()
5.       public void onClick(View view)
6.          Intent intent = new Intent(view.getContext(), faq.class);
7.          startActivityForResult(intent, 0);
```

Method 2: **View profile()**

Pseudo-code:
Input: savedInstanceState , OnClickListener on View profile button
Output: Launch the Activity
1. super.onCreate(savedInstanceState);
2. setContentView(R.layout.activity_main);
3. button = stores the ID of View profile button
4. button1.setOnClickListener(new View.OnClickListener()
5. public void onClick(View view)
6. Intent intent = new Intent(view.getContext(),viewprofile.class);
7. startActivityForResult(intent, 0);

Method 3: **Training data**

Pseudo-Code:
Input: savedInstanceState , OnClickListener on Training data button
Output: Launch the Activity
1. super.onCreate(savedInstanceState);
2. setContentView(R.layout.activity_main);
3. button = stores the ID of Training data button
4. button1.setOnClickListener(new View.OnClickListener()
5. public void onClick(View view) {
6. Intent intent = new Intent(view.getContext(),traningdata.class);
7. startActivityForResult(intent, 0);

Method 4: **Insert company**

Pseudo-code:
Input: savedInstanceState , OnClickListener on Insert company button
Output: Launch the Activity
1. super.onCreate(savedInstanceState);
2. setContentView(R.layout.activity_main);
3. button = stores the ID of Insert company button
4. button1.setOnClickListener(new View.OnClickListener()
5. public void onClick(View view)
6. Intent intent = new Intent(view.getContext(),insertcompany.class);
7. startActivityForResult(intent, 0);

Method 5: **Approve profile**

Pseudo-code:
Input: savedInstanceState , OnClickListener on Approve profile button
Output: Launch the Activity
1. super.onCreate(savedInstanceState);

2. setContentView(R.layout.activity_main);
3. button = stores the ID of Approve profile button
4. button1.setOnClickListener(new View.OnClickListener()
5. public void onClick(View view) {
6. Intent intent = new Intent(view.getContext(),approveprofile.class);
7. startActivityForResult(intent, 0);

Method 6: **About**

Pseudo-code:
Input: savedInstanceState , OnClickListener on About button
Output: Launch the Activity
1. super.onCreate(savedInstanceState);
2. setContentView(R.layout.activity_main);
3. button = stores the ID of About button
4. button1.setOnClickListener(new View.OnClickListener()
5. public void onClick(View view)
6. Intent intent = new Intent(view.getContext(),about.class);
7. startActivityForResult(intent, 0);

Method 7: **Contact**

Pseudo-code:
Input: savedInstanceState , OnClickListener on Contact button
Output: Launch the Activity
1. super.onCreate(savedInstanceState);
2. setContentView(R.layout.activity_main);
3. button = stores the ID of Contact button
4. button1.setOnClickListener(new View.OnClickListener()
5. public void onClick(View view)
6. Intent intent = new Intent(view.getContext(),contact.class);
7. startActivityForResult(intent, 0);

6.0.12 Class: Insert_CompStatistics_History

Method 1: **updateCompanyDetails()**

Pseudo-code:
Input: None
Output: All the questions being asked by students.

1. JSONObject obj = new JSONObject(response);
2. JSONArray questionArray = obj.getJSONArray("Questions");
3. for (int i = 0; i < questionArray.length(); i++)
    a. JSONObject questionObject = questionObject.getJSONObject(i);
    b. Question question = new question(heroObject.getString("question"));

c. questionList.add(question);
4. End for
5. ListViewAdapter adapter = new ListViewAdapter(questionLIst, getApplicationContext());
6. listView.setAdapter(adapter);

Method 2: **onCreate(Bundle savedInstanceState)**

Pseudo-code:
Input: The question id of the question asked.
Output: The answer to the question is passed to the server.

1. String answer = answerTextBox.getText();
2. RequestQueue queue = Volley.newRequestQueue(this);
3. String url = String.format("Address of the server"),
   StringRequest request = new StringRequest(Request.Method.POST, url,
   new Response.Listener<String>()
   public void onResponse(String response)
   new ApiErrorListener(this));
4. queue.add(request);

6.0.13 Class: Student

Method 1: **addDrawerItems()**

Pseudo-code:
Input: None
Output: Initializes the various options in Navigation Drawer.
String navOptions[] = {"Home","FAQ","View Jobs","About Us"};

1. mAdapter = new ArrayAdapter<String>(this,android.R.layout.*simple_list_item_1*,navOptions);
2. mDrawerList.setAdapter(mAdapter);
3. mDrawerList.setOnItemClickListener(new AdapterView.OnItemClickListener()
4. public void onItemClick(AdapterView<?> parent, View view, int position, long id)
5. Display or go to the wherever the user wants to navigate to.

Method 2: **setupDrawer()**

Psudo-code:
Input: None
Output: Sets up the Action Bar title whenever the Navigation drawer is opened or closed.

1. mDrawerToggle = new ActionBarDrawerToggle(this, mDrawerLayout, R.string.*drawer_open*, R.string.*drawer_close*)
2. public void onDrawerOpened(View drawerView)
3. super.onDrawerOpened(drawerView);
4. getSupportActionBar().setTitle("Navigation!");

```
5.                 invalidateOptionsMenu();
6.       public void onDrawerClosed(View view)
7.           super.onDrawerClosed(view);
8.           getSupportActionBar().setTitle(mActivityTitle);
9.           invalidateOptionsMenu();
10. mDrawerToggle.setDrawerIndicatorEnabled(true);
11. mDrawerLayout.setDrawerListener(mDrawerToggle);
```

Method 3: **onOptionsItemsSelected(MenuItem item)**

Pseudo-code:
Input: MenuItem item (Object item of class MenuItem)
Output: The activity to be launched or the method to be executed upon clicking upon any of the Navigation Drawer options.

```
1.  int id = item.getItemId();
2.  if (mDrawerToggle.onOptionsItemSelected(item))
3.      return true;
4.  return super.onOptionsItemSelected(item);
```

Method 4: **onPostCreate(Bundle savedInstanceState)**

Pseudo-code:
Input: Bundle savedInstanceState
Output: Call the sync state method of the ActionBarDrawerToggle object.

```
1.  super.onPostCreate(savedInstanceState);
2.  mDrawerToggle.syncState();
```

Method 5: **onConfigurationChanged(Configuration newConfig)**

Pseudo-code:
Input: Configuration newConfig
Output: Calls the onConfigurationChanged method of the ActionBarDrawerToggle object.

```
1.   super.onConfigurationChanged(newConfig);
2.  mDrawerToggle.onConfigurationChanged(newConfig);
```

Method 6: **onCreate(Bundle savedInstanceState)**

Pseudo-code:
Input: Bundle savedInstanceState
Output: Sets the activity in order, basically the main function of this class.

```
1.  super.onCreate(savedInstanceState);
2.  setContentView(R.layout.activity_main);
3.  mDrawerList = (ListView)findViewById(R.id.navList);
4.  mDrawerLayout = (DrawerLayout)findViewById(R.id.drawyer_layout);
5.  mActivityTitle = getTitle().toString();
```

6. addDrawerItems();
7. setupDrawer();
8. getSupportActionBar().setDisplayHomeAsUpEnabled(true);
9. getSupportActionBar().setHomeButtonEnabled(true);

6.0.14 Class: Answer_FAQ

Method 1: **showQuestions()**

Pseudo-code:
Input: None
Output: All the questions being asked by students.

7. JSONObject obj = new JSONObject(response);
8. JSONArray questionArray = obj.getJSONArray("Questions");
9. for (int i = 0; i < questionArray.length(); i++)
    a. JSONObject questionObject = questionObject.getJSONObject(i);
    b. Question question = new question(heroObject.getString("question"));
    c. questionList.add(question);
10. End for
11. ListViewAdapter adapter = new ListViewAdapter(questionLIst, getApplicationContext());
12. listView.setAdapter(adapter);

Method 2: **answerQuestions(int qID)**

Pseudo-code:
Input: The question id of the question asked.
Output: The answer to the question is passed to the server.

5. String answer = answerTextBox.getText();
6. RequestQueue queue = Volley.newRequestQueue(this);
7. String url = String.format("Address of the server"),
   StringRequest request = new StringRequest(Request.Method.POST, url,
   new Response.Listener<String>()
   public void onResponse(String response)
   new ApiErrorListener(this));
8. queue.add(request);

7.0 Appendices (if any)