**Software Design Specification**

**Table of Contents**

**1.Introduction.......................................................Error! Bookmark not defined.**

1.1 Purpose of this document .................................................................Error! Bookmark not defined.

1.2 Scope of the development project ....................................................Error! Bookmark not defined.

1.3 Definitions, acronyms, and abbreviations ........................................Error! Bookmark not defined.

1.4 References........................................................................................Error! Bookmark not defined.

1.5 Overview of document .....................................................................Error! Bookmark not defined.

**2. Conceptual Architecture/Architecture Diagram ......................Error! Bookmark not defined.**

2.2 Structure and relationships ...............................................................Error! Bookmark not defined.

2.3 User interface issues.........................................................................Error! Bookmark not defined.

**3. Logical Architecture (Class Diagram, Sequence Diagram, State Diagram)Error! Bookmark not defined**.

**3.1 Class Diagram**

**3.2. Sequence Diagram**

**3.3. State diagram**

**4.0. Execution architecture**

**4.1. Reuse and relationships to the other products**

**5.0. Design decisions and trade-offs**

**6.0. pseudo-code for components**

**1.Introduction**

Software design document shows how the software system will be structured to satisfy the requirements. Software design document contains overall system architecture,data architecture, use case models, sequence diagrams, collaboration models, object behaviour models, and other supporting requirement information,these shows the outline of software and how it will work.

* 1. **Purpose of this document**

In this software designer gives a overall guidance to architecture of the software project. 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**

TPC is an online application that can be accessed throughout the organization and outside as well with proper login provided. Students logging will be able to upload their information in the form of a CV. Visitors or company representatives logging in may also access/Search any information put up by the Students

**1.3Definitions, acronyms, and abbreviations**

* TPC – Training and placement cell
* CV – Curriculum vitae
* HTTP – Hyper Text Transfer Protocol
* JDBC – Java Database Connectivity
* IEEE: Institute of Electrical and Electronics Engineers
* SDS: Software Design Specification

**1.4 References**

[1] The Complete Reference “Crystal Reports 2008” by George peck.

[2]Training and placement website of IIT,Patna.

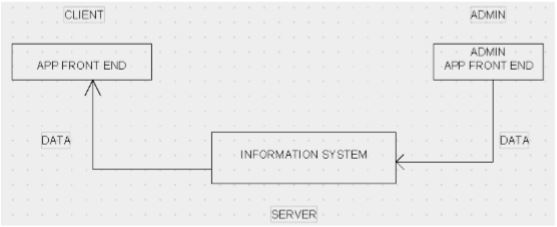
[3] “Microsoft Visual Basic 2010” by Michael Halvorson.

[4] Tynjälä, P., Perspective into learning at the workplace, Educational Research Review, 3, 2008, pp.130-154.

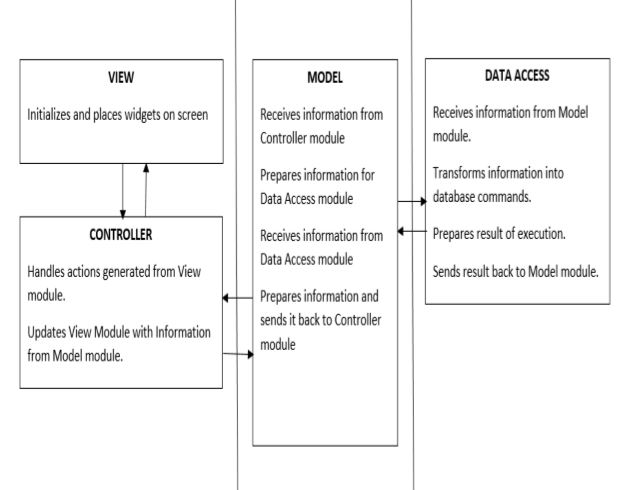
[5] “The Complete Reference ASP.NET” book by Robert Standefer III

**2. Conceptual Architecture/Architecture Diagram**

**Architecture Diagram:**

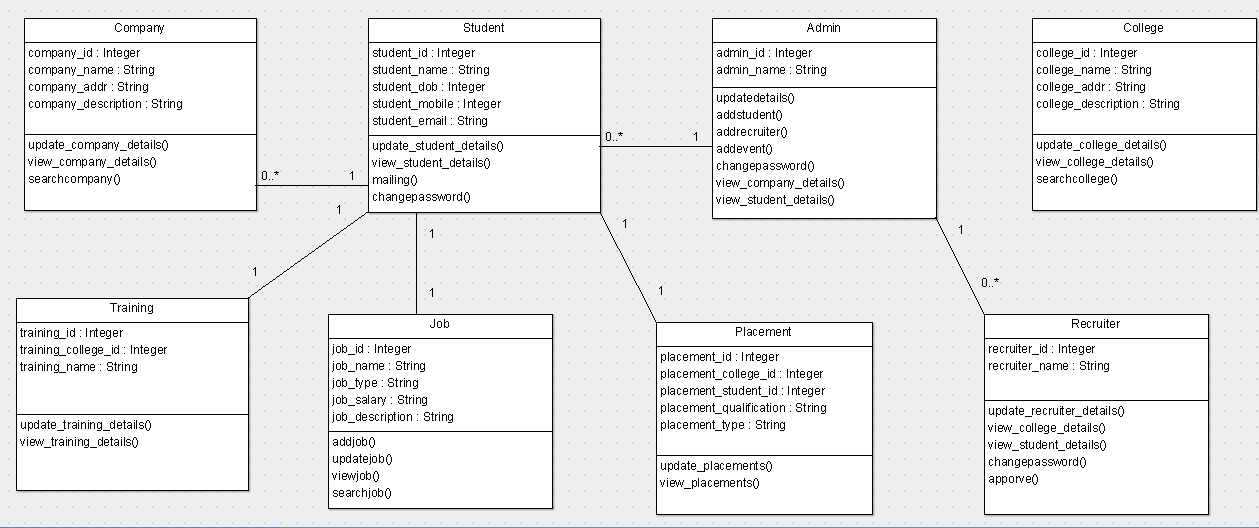


**2.1 Overview of modules / components**



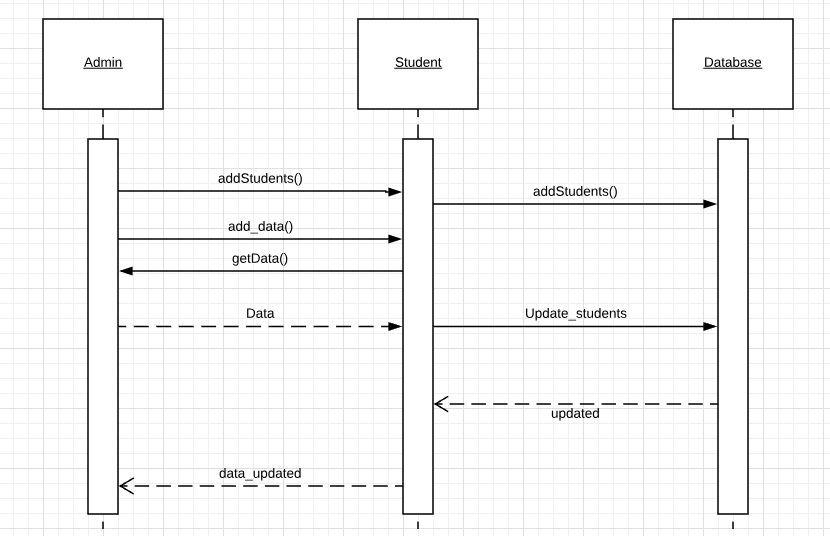
**3. Logical Architecture (Class Diagram, Sequence Diagram, State Diagram)**

**Class Diagram:**

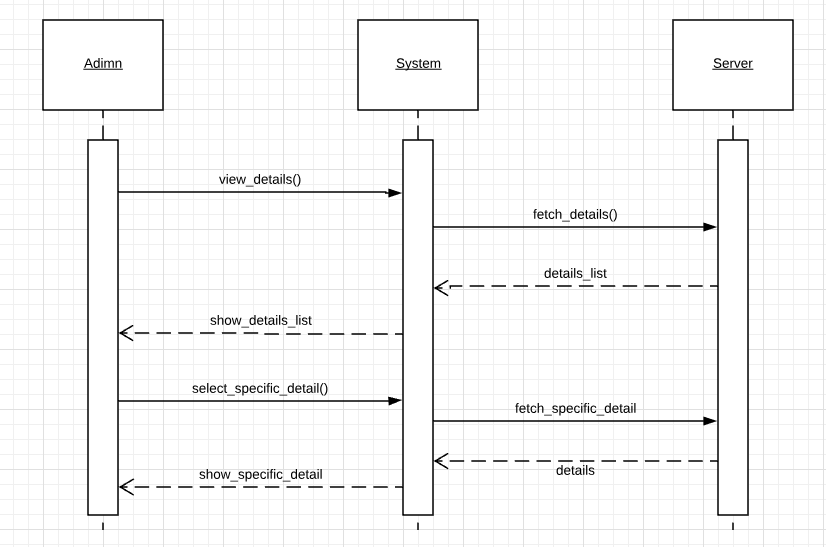


**Sequence Diagram**

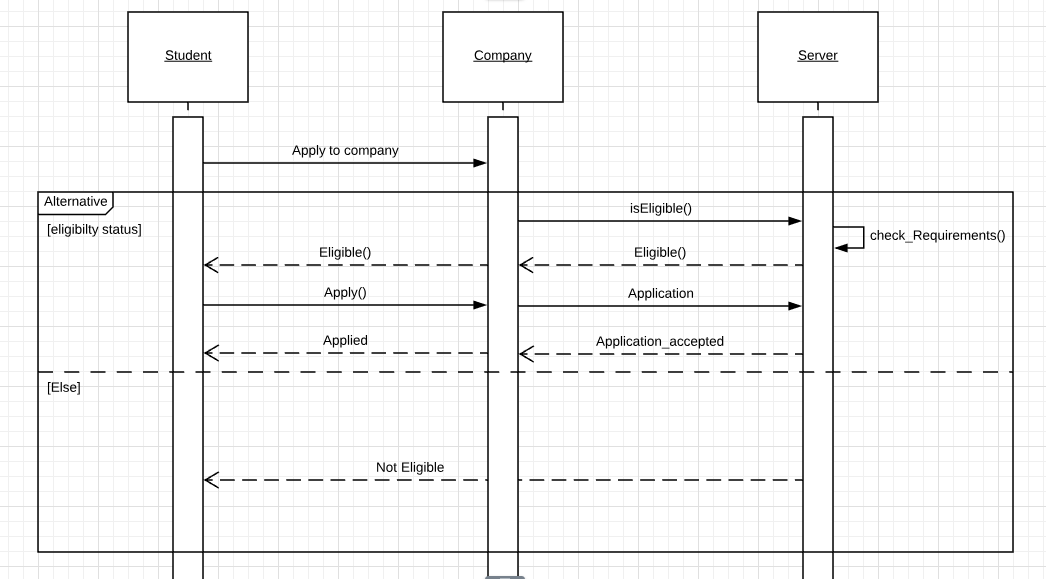
**Add Students**



**Sequence diagram:View Details**

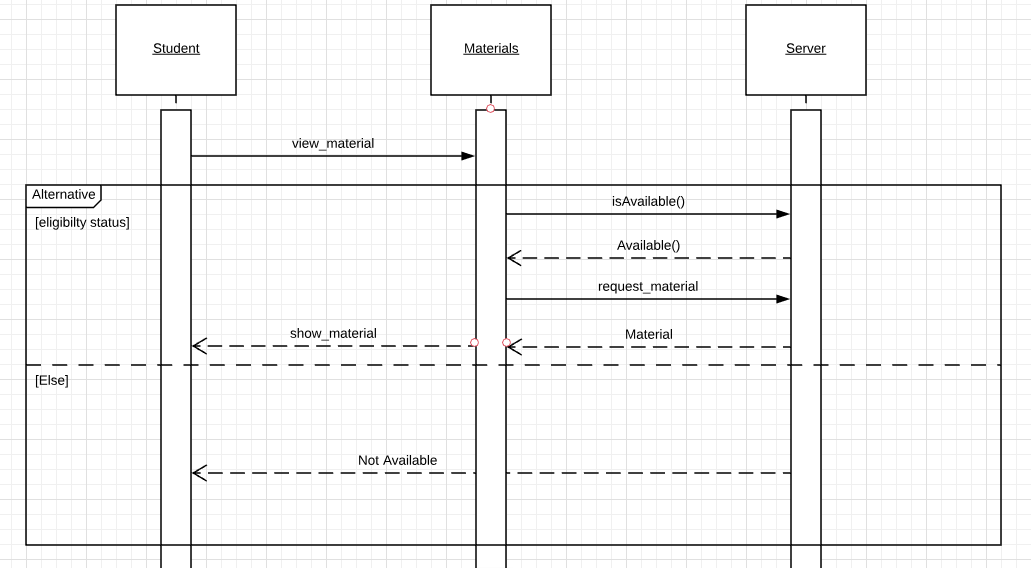


**Student apply\_to\_company:**

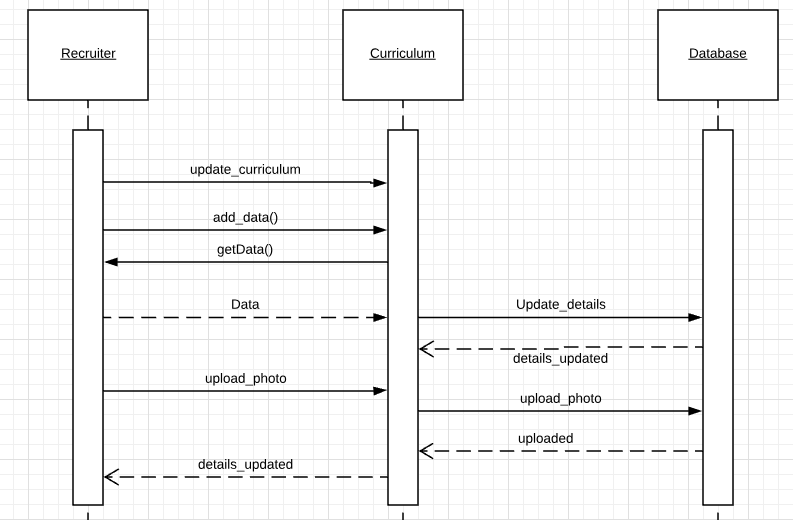


**Student material:**

.

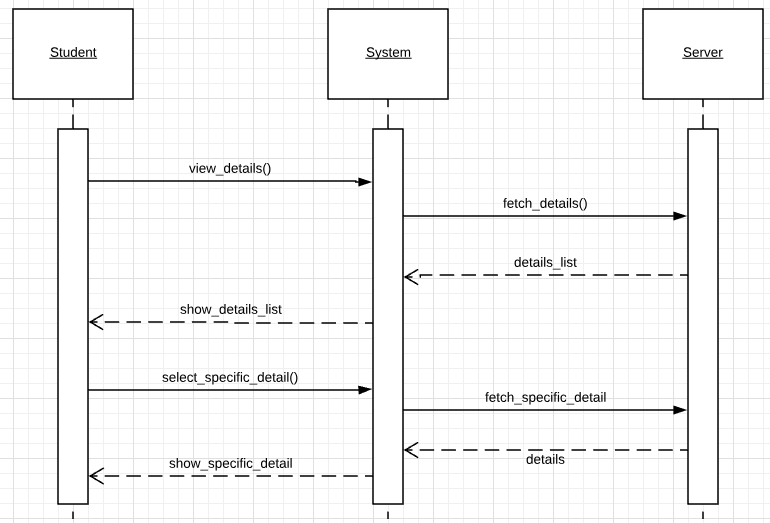


**Student update\_cv:**

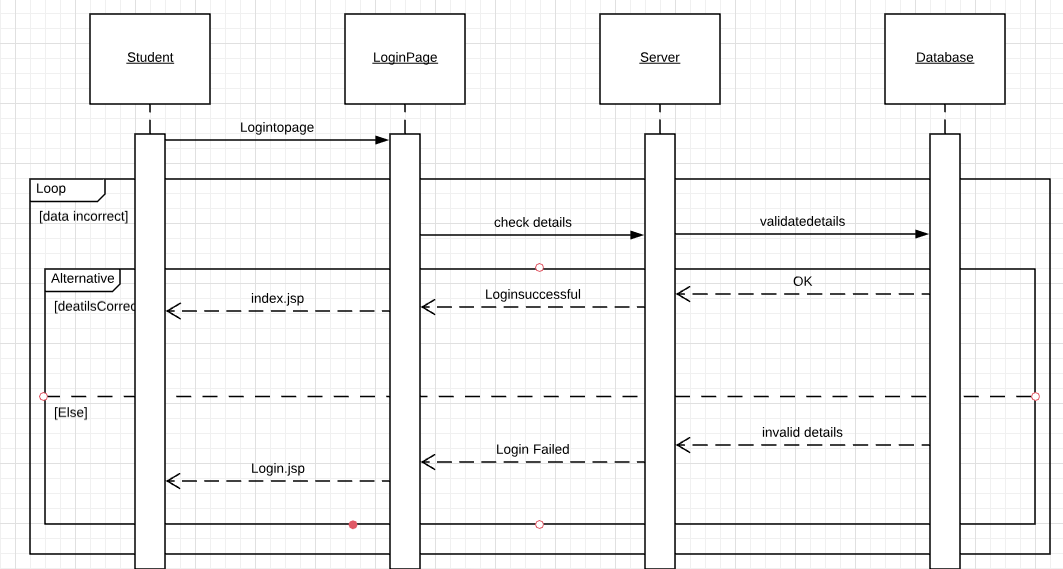


**Student view\_details:**

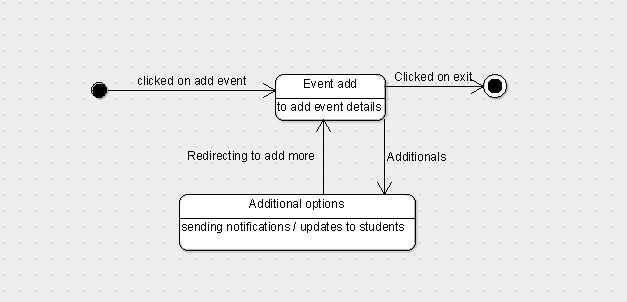
.



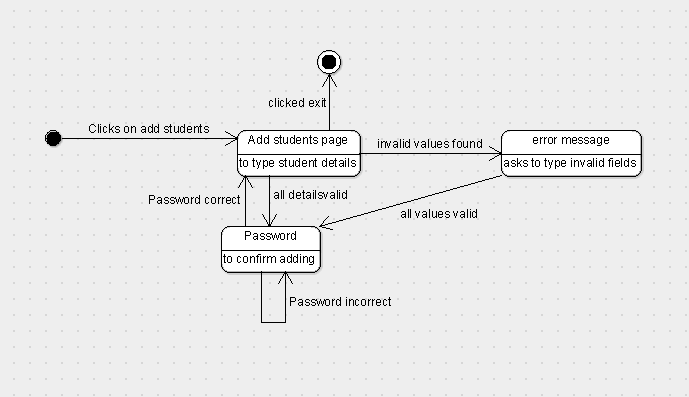
**Student\_login:**



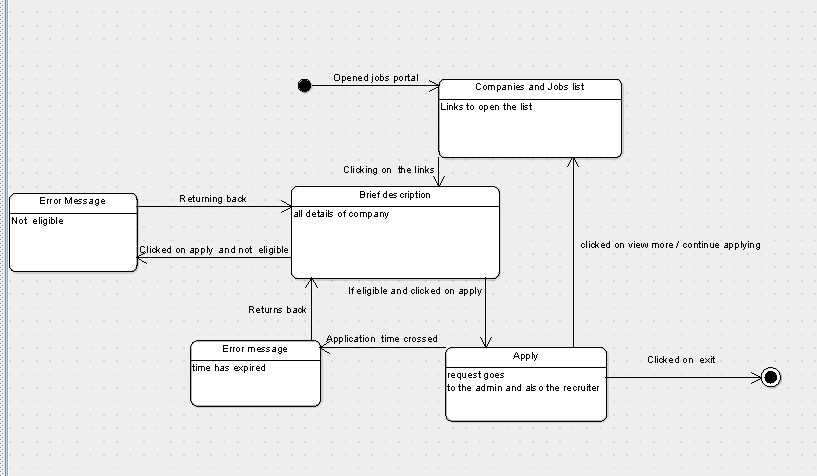
**State diagram: Adding events (by admin)**

****

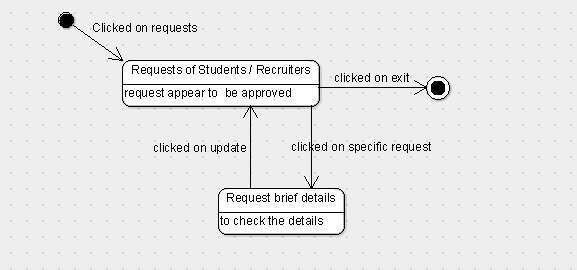
**State diagram: adding students (by admin)**

****

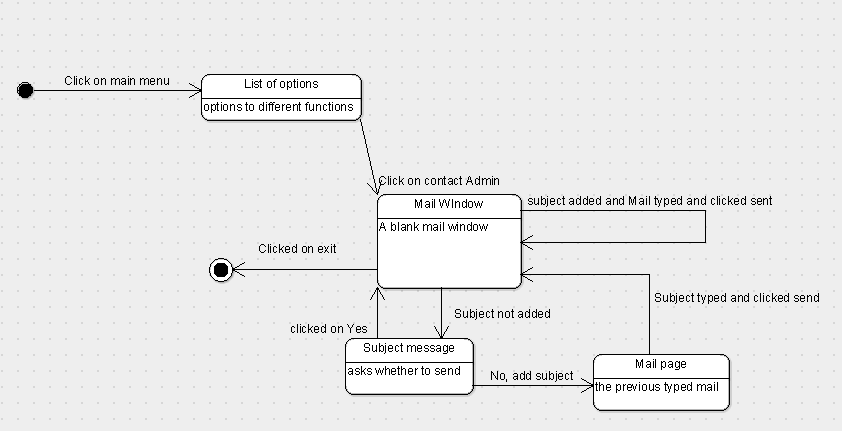
**State diagram: student applying for a company**

****

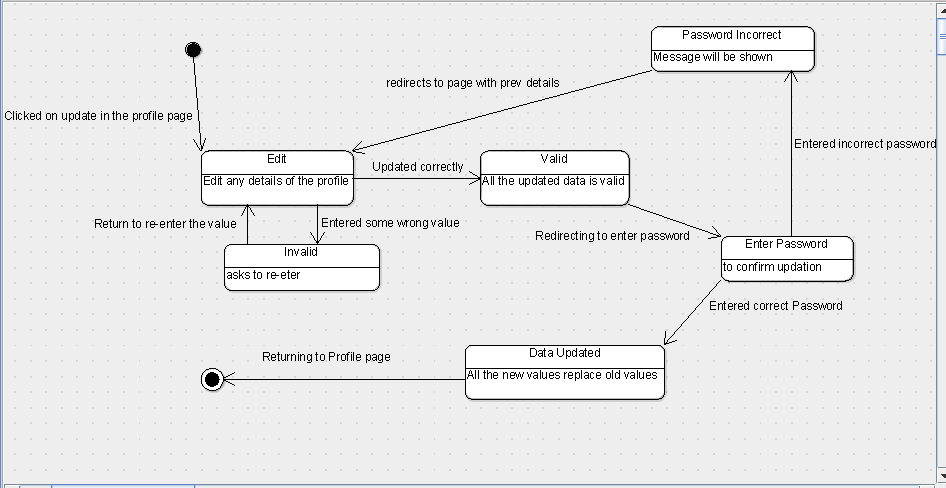
**State diagram: approve requests (by admin)**

****

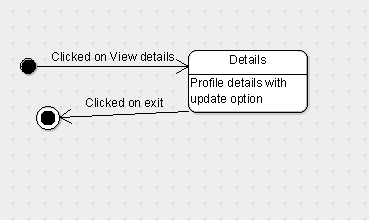
**State diagram: mailing the admin (by student)**

****

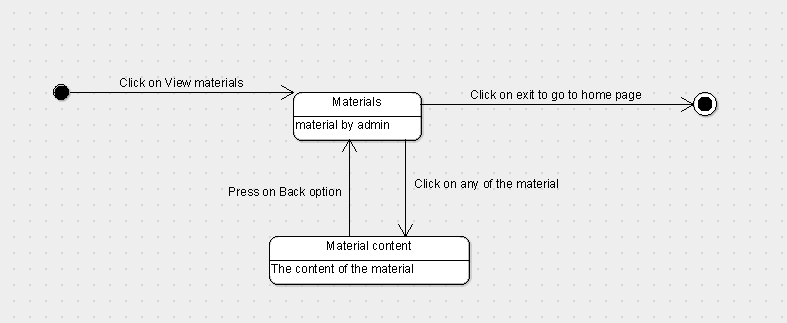
**State diagram: update student details**

****

**State diagram: view details (student)**



**State diagram: view material (students)**



**3.1. Logical Architecture Description**

**3.1.1 Class Diagram explanation:**

The class diagram has company class, student class, admin class, college class, training class, job class, placement class, recruiter class. The lines without arrow show 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

**3.1.2 Sequence Diagram**

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

**3.1.2.2 Add Students :**

Admin adds the Students and the data/information regarding them. Students can even update their details .

**3.1.2.3 View Details:**

Admin can view details of all the recruiters and students of the university present on the system and can even fetch for specific details regarding a particular student or a recruiter. The recruiter can even view or fetch details of all students present on the system.

**3.1.2.6 Student apply\_to\_company:**

When student apply to a company,server checks whether the student is eligible as per the company requirements or not.If he/she is eligible then the application is accepted ,if not the application will be declined

**3.1.2.7Student material:**

Student can practice for aptitude tests through some materials provided in this section which includes previous year question papers ,some important aptitude questions collected from books.

**3.1.2.8 Student update\_cv:**

In this student can update his CV and picture.

**3.1.2.9Student view\_details:**

This allows the student to view his profile

**3.1.2.10 Student\_login:**

Here Student can login into the system by providing correct information of user id and password which is checked by the database .A successful login provides access to other pages.

**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.** **Adding Events (by admin):**

By clicking on add event, the add event page opens and redirects the admin to add more events and If does not want to add any more event he can click on exit.

3.1.3.2.Adding Students(by admin):

By clicking on add students on the admin’s home page, the add student page is opened, If any invalid value is typed an error message is generated and shown. If all the details are valid, the page asks for password to confirm adding, if the password is correct, the admin can add details about more students or exit and if the password is incorrect the page again asks for valid password.

**3.1.3.3. Student applying for a company:**

On clicking apply for a company, several links to open the list are shown. By clicking on a specific link, brief description page is opened with all the details of the company. By clicking apply on the brief description page, the student can check if he/she is eligible for the company or not. If eligible, he can click on apply button and the request goes to admin and exit. If not a error message “not eligible” is generated.

**3.1.3.4. Approve requests (by admin):**  
On clicking the requests in the home button in the nav bar of admin’s page, the admin can see the requests of the students. To approve any request, the admin needs to click on a specific request and request brief details to check details and then click on update and then exit.

**3.1.3.5. Mailing the admin (by student):**After clicking main menu, different set of options can be seen. On clicking contact admin, Mail window page is opened. If subject is not added, it shows a message asking whether to send without a subject or add subject, if clicked on yes, the mail is sent, if not the page is redirected to the mail window page/previous typed mail and clicked send and then exit.

**3.1.3.6. Update student details:**

On clicking edit on the profile page, if all the details are correctly entered then it asks for password to confirm updation, If password is wrond it shows an error message “invalid password” and redirects to the previous edit page, if the password is correct the details are updated and redirects to the main profile page.

**3.1.3.7. View Student details:**

On clicking Profile, the student can see his profile details with an update option.

**3.1.3.8. view material:**On clicking on view materials in the main menu, the material page is opened with different materials are available, On clicking a specific material, the material content page is opened, on clicking back page is redirected to materials page. On clicking exit present on the materials page, it redirects the student to the home page.

**3.2. Class name**: Company

**Description:** Here the admin can add, update and view details of the company

**3.2.1. Method 1:** update\_company\_details()

**Input:** editing the previous details

**Output:** Updated company details on the view company profile page

**3.2.2. Method 2:**view company details()

**Input:** clicking on view company profile

**Output:** ‘view company profile‘ page appears

**3.3 Class name:** Student

**Description:** Here the student can view or update his/ her profile, CV and can also view materials provided, company profile, news and events uploaded by the admin.

**3.3.1 Method 1:** update\_student\_details()

**Input:** clicking on edit button on student profile page

**Output:** Page to update the details

**3.3.2. Method 2:** view\_student\_details()

**Input:** clicking on profile

**Output:** profile page with all details is opened

**3.3.3. Method 3:** upload\_cv()

**Input :** clicking upload CV button in the menu bar

**Output:** upload CV page is opened to upload file

**3.3.4. Method 4:** view\_news\_events()

**Input:** Login into the application

**Output:** Home page is opened with all the news and events

**3.3.5: Method 5:** view\_company\_profile()

**Input:** Clicking view company profile in the main menu bar

**Output:** Company profile page is opened

**3.4. Class name:** Admin

**Description:** Here the admin can add/update the details of company and students. He can also add news, events, company requirement details and materials required for aptitude test preparation and he can view requests made by the students for job application

**3.4.1. Method 1:** add\_students()

**Input:** clicking add students in the menu bar of admin page

**Output:** page to add student details is opened

**3.4.2. Method 2:** add\_news\_events()

**Input:** clicking on add events/news in the menu bar

**Output:** page to upload/ write events/news is opened

**3.4.3.** **Method 3:** add\_company\_details()

**Input:** Cicking on add companies in the menu bar

**Output:** page to add company details is opened

**3.4.4 Method 4:** view\_student\_details()

Input: Clicking on Student details in the menu bar

Output: Page with all the student details is opened

3.4.5. Method 5: add\_requirement\_details()

Input: Clicking on requirement details in the menu bar

Output: Page to add requirement details to apply for a specific company is opened

3.4.6. Method 6: view\_applied\_candidates()

Input: Clicking on the requests in the menu bar

Output: Page with all the requests made by the students is opened.

3.5. Class name: Placement

Description: Here the admin and student can see the students who got placed in a specific company.

3.5.1. Method 1: view\_placements()

Input: Clicking on Placements in the menu bar of admin/student

Output: Placement page is opened.

**4.0 Execution Architecture**

Runtime environment required is a web browser like chrome, Mozilla fire fox etc and eclipse J2EE IDE 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:** Company

Method1: update\_company\_details()

Pseudo-code:

**Input:** editing the previous details

**Output:** Updated company details on the view company profile page

1. doPost(HttpServletRequest request, HttpServletResponse response)
2. doGet(request, response);
3. try
4. String button = “submit”
5. If(button.equals(“select”)
6. String company name = gets the company name
7. String url = stores the url of tpc
8. String user = stores the name of the user
9. String pass = stores the password
10. Connection con=DriverManager.getConnection(url,user,pass); :gets the connection
11. PreparedStatement ps=con.prepareStatement("update company\_register set company\_profile =? where company\_name=?");
12. ps.setString(1,company name); :views the particular company
13. response.sendRedirect("addcompanydetails.jsp"); :redirects to the companydetails page

Method 2 : view\_company\_details()

Pseudo-code:

**Input:** clicking on view company profile

**Output:** ‘view company profile‘ page appears

1.doPost(HttpServletRequest request, HttpServletResponse response)

2.doGet(request, response);

3.try

4. String button = “submit”

5. If(button.equals(“select”))

6. String company name = gets the company name

7. String url = stores the url of tpc

8. String user = stores the name of the user

9. String pass = stores the password

10.Connection con=DriverManager.getConnection(url,user,pass); :gets the connection

11.PreparedStatement ps=con.prepareStatement("update company\_register set company\_profile =? where company\_name=?");

12.ps.setString(1,company name); :views the particular company

13.response.sendRedirect("addcompanydetails.jsp"); :redirects to the companydetails page

6.0.2. Class name: Student

Method 1: update\_student\_details()

Pseudocode:

**Input:** clicking on edit button on student profile page

**Output:** Page to update the details

1. doPost(HttpServletRequest request, HttpServletResponse response)
2. doGet(request, response);
3. try
4. String button = request.getParameter("update")
5. if (button.equals("update"))
6. String mobile =gets mobile number
7. String address = gets address
8. String cgpa =gets cgpa
9. String emailid =stores email id
10. try{
11. Class.forName("com.mysql.cj.jdbc.Driver"); :loading the driver
12. String url= stores the url of tpc
13. String user=stores the user name;
14. String pass=stores the password
15. Connection con=DriverManager.getConne"ction(url,user,pass); :gets connection
16. PreparedStatement ps=con.prepareStatement("update student\_register set mobile =?,address =?,cgpa =? where std\_emailid=?"); :sql query to update student details
17. ps.setString(1,mobile ); :updates mobile number
18. ps.setString(2,address); :updates address
19. ps.setString(3,cgpa) :updates cgpa
20. ps.setString(4,emailid); :updates email id
21. ps.executeUpdate();
22. response.sendRedirect("studentprofile.jsp"); :redirects to student profile page

Method 2: view\_student\_details()

Pseudo-code:

**Input:** clicking on profile

**Output:** profile page with all details is opened

1. doPost(HttpServletRequest request, HttpServletResponse response)

2. doGet(request, response);

3.try{

4.Class.forName("com.mysql.cj.jdbc.Driver");

5.String url= stores url of tpc

6.String user= stores the username;

7.String pass=stores the password

8.Connection con=DriverManager.getConnection(url,user,pass); :gets connection

9.PreparedStatement ps=con.prepareStatement("select \* from student\_register"); :sql query to view details of the student

10.ps.executeUpdate();

11.response.sendRedirect("Viewstudentdetails.jsp"); :redirects to the Student profile page

Method 3: view\_news\_events()

Pseudocode:

**Input:** Login into the application

**Output:** Home page is opened with all the news and events

1. doPost(HttpServletRequest request, HttpServletResponse response
2. doGet(request, response);
3. try {
4. try{
5. Class.forName("com.mysql.cj.jdbc.Driver");
6. String url=stores the url of tpc
7. String user=stores the username
8. String pass=stores the password
9. Connection con=DriverManager.getConnection(url,user,pass); :gets connection
10. PreparedStatement ps=con.prepareStatement("select \* from news\_events"); :sql query to view news and events
11. ps.executeUpdate();

Method 4: upload\_cv()

Pseudocode:

**Input :** clicking upload CV button in the menu bar

**Output:** upload CV page is opened to upload file

1. Click on the upload cv option in the menu bar
2. Redirects to the upload cv page
3. On clicking add file, uploads the cv file on to the page
4. Click back- redirects to the home page

Method 5: view\_company\_details():

Pseudocode:

**Input:** Clicking view company profile in the main menu bar

**Output:** Company profile page is opened

1. doPost(HttpServletRequest request, HttpServletResponse response)
2. doGet(request, response);
3. try {
4. String button = request.getParameter("submit");
5. if (button.equals("select")) {
6. String company name = request.getParameter("company name");
7. Try{
8. Class.forName("com.mysql.cj.jdbc.Driver")
9. String url=gets url of tpc
10. String user=stores username
11. String pass=stores password
12. Connection con=DriverManager.getConnection(url,user,pass); :gets connection
13. PreparedStatement ps=con.prepareStatement("select \* from company\_register where company\_name=? "); :sql query to view company profile
14. ps.setString(1,company name);
15. ps.executeUpdate();
16. response.sendRedirect("Viewcompanydetails.jsp"); :redirects o company profile page

6.0.3. Class name: Admin

Method 1 : view\_student\_details()

Pseudo-code:

Input: Clicking on Student details in the menu bar

Output: Page with all the student details is opened

1. doPost(HttpServletRequest request, HttpServletResponse response)

2. doGet(request, response);

3.try{

4.Class.forName("com.mysql.cj.jdbc.Driver");

5.String url= stores url of tpc

6.String user= stores the username;

7.String pass=stores the password

8.Connection con=DriverManager.getConnection(url,user,pass); :gets connection

9.PreparedStatement ps=con.prepareStatement("select \* from student\_register"); :sql query to view details of the student

10.ps.executeUpdate();

11.response.sendRedirect("Viewstudentdetails.jsp"); :redirects to the Student profile page

Method 2: add\_news\_events()

Pseudo-code:

**Input:** clicking on add events/news in the menu bar

**Output:** page to upload/ write events/news is opened

1. doPost(HttpServletRequest request, HttpServletResponse response

2. doGet(request, response);

3.try {

4.String button = request.getParameter("submit");

5.if (button.equals("submit")) {

6.String newsevents = request.getParameter("newsevents");

7.try{

8.Class.forName("com.mysql.cj.jdbc.Driver");

9.String url="jdbc:mysql://localhost:3306/tpc";

10.String user=stores user name

11.String pass=stores password

12.Connection con=DriverManager.getConnection(url,user,pass); :gets connection

13.PreparedStatement ps=con.prepareStatement("insert into news\_events(newsandevents) values (?)"); :sql query to add events and news

14.ps.setString(1,newsevents);

15.ps.executeUpdate();

16.response.sendRedirect("addnewsevents.jsp"); :redirects to the add news and events page

**Method : Login(Type):**

**Input :**Login Type , Email , Password

**Output** : Launch the activity , login

1. doPost(HttpServletRequest request, HttpServletResponse response)

2. String type = Stores the type of user (Client or Admin)

3.  String name = Stores user name

4.  String pass = Stores user password

5. Submit Log In = When clicked, authentication process starts

6. ResultSet rs = ps.executeQuery() : Stores the submitted values in result set.

7. rs.next() = Tells the user is valid or not.

8. HttpSession session = request.getSession()  : Creates a session for the user if he/she is a valid user

9.   session.setAttribute("username",name) : Username submitted is set in the session

10.   session.setAttribute("utype",type) = Usertype submitted is set in the session

11. response.sendRedirect("index.jsp") = Redirects the user to homepage if logged in.

12.  response.sendRedirect("log.jsp") = Redirects the user to login page if an invalid login occurs.