**ACKNOWLEDGEMENT**

The satisfaction and joy that accompanies the successful completion of this project would be incomplete and short-lived, without the mention of the people who made it possible. We wish to express our sincere gratitude **Dr.S.B Vanakudure** and to the Head of the CSE department **Prof. J V Vadavi** for granting the permission to carry out this project and for providing us the resources. We express our indebtedness to our project guide **Prof. Mayur patil** and our course instructors **Dr.S.B.Kulkarni** who helped us in every possible aspect for the successful completion of the project. We express our sincere thanks to staff of the department of CSE for having helped us in the successful completion of the project. Lastly we express thanks to our beloved parents and friends who gave us the support and motivated us to emerge successful of this project.

Anand

Mantesh Hosamani

Nagaraj G.Kandoor

Prakash Sindhe

CHAPTER 1

**Introduction**

**1.1 Problem scenario**

In today’s world rating plays an important role in our life, like Standard & Poor and Moody’s give credit rating to all the countries, based on that the investor will choose which country to invest his money, in the same way the Student Placement Rating System is proposed to give the rating to the students based on their subjects, subject’s relevance in industry and users’ digital foot prints, where as the current rating scheme takes only academic examination scores into consideration.

**1.2 Overview of the report: what material will you be covering and how is it arranged?**

Student Placement Rating System is proposed to give the rating to the students based on their subjects, subject’s relevance in industry and users’ digital foot prints, where as the current rating scheme takes only academic examination scores into consideration.

**Materials we are covering such as:-**

1. UML diagrams- Architecture diagram, Use case diagrams, Sequence diagram, Activity diagram, Class diagram ,State diagram.
2. Theoretical models – Javascript , HTML,XML.
3. Algorithms –
4. Tasks and Scheduling of the tasks
5. Specification of front end specification of backend and the middle layer

**1.3 List of criteria that define a successful project:**

**Expected outcomes.**

* **Skills in using tools.**
* **Report writing skills.**
* **Writing industry standard code.**
* **Learning new technologies.**

**1.4 Required System Behavior:**

* System need to be information on service offered to student with specific needs.
* System has to check the status of the student activities details.
* System has to check the status of the Rating of the students details.
  1. **Hardware Requirements**
     + Pentium IV & above
     + RAM & above
     + HDD 80 GB & above

**1.6 Software Requirements.**

* Rational rose-7 (Design Tool).
* Dreemweaver-8(Front End Design).
* Eclipse-Kepler-64bit(Google-App-Engine).
* Codenvy (Cloud-tool)

**Database:** Google app engine database (Big-Table).

**CHAPTER 2**

**Theoretical Foundations: The Engineering Model**

**2.1 JAVA History**

Java is a [computer programming language](http://en.wikipedia.org/wiki/Computer_programming_language) that is [concurrent](http://en.wikipedia.org/wiki/Concurrent_computing), [class-based](http://en.wikipedia.org/wiki/Class-based), [object-oriented](http://en.wikipedia.org/wiki/Object-oriented_programming), and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that code that runs on one platform does not need to be recompiled to run on another. Java applications are typically [compiled](http://en.wikipedia.org/wiki/Compiler) to [byte code](http://en.wikipedia.org/wiki/Java_bytecode) ([class file](http://en.wikipedia.org/wiki/Class_(file_format))) that can run on any [Java virtual machine](http://en.wikipedia.org/wiki/Java_virtual_machine) (JVM) regardless of [computer architecture](http://en.wikipedia.org/wiki/Computer_architecture). Java is, as of 2012, one of the most popular programming languages in use, particularly for client-server web applications, with a reported 9 million developers. Java was originally developed by [James Gosling](http://en.wikipedia.org/wiki/James_Gosling) at [Sun Microsystems](http://en.wikipedia.org/wiki/Sun_Microsystems)(which has since [merged into Oracle Corporation](http://en.wikipedia.org/wiki/Sun_acquisition_by_Oracle)) and released in 1995 as a core component of Sun Microsystems' [Java platform](http://en.wikipedia.org/wiki/Java_(software_platform)). The language derives much of its [syntax](http://en.wikipedia.org/wiki/Syntax_(programming_languages)) from [C](http://en.wikipedia.org/wiki/C_(programming_language)) and [C++](http://en.wikipedia.org/wiki/C%2B%2B), but it has fewer [low-level](http://en.wikipedia.org/wiki/Low-level_programming_language) facilities than either of them.

The original and [reference implementation](http://en.wikipedia.org/wiki/Reference_implementation_(computing)) Java [compilers](http://en.wikipedia.org/wiki/Compiler), virtual machines, and [class libraries](http://en.wikipedia.org/wiki/Library_(computing)) were developed by Sun from 1991 and first released in 1995. As of May 2007, in compliance with the specifications of the [Java Community Process](http://en.wikipedia.org/wiki/Java_Community_Process), Sun relicensed most of its Java technologies under the GNU. Others have also developed alternative implementations of these Sun technologies, such as the [GNU Compiler forJava](http://en.wikipedia.org/wiki/GNU_Compiler_for_Java) (byte code compiler), [GNU Class path](http://en.wikipedia.org/wiki/GNU_Classpath) (standard libraries), and Iced Tea-Web (browser plugin for applets).

**Principles**

There were five primary goals in the creation of the Java language:

1. It should be "simple, object-oriented and familiar"
2. It should be "robust and secure"
3. It should be "architecture-neutral and portable"
4. It should execute with "high performance"
5. It should be "interpreted, threaded, and dynamic"

Java is the foundation for virtually every type of networked application and is the global standard for developing and delivering embedded and mobile applications, games, Web-based content, and enterprise software. With more than 9 million developers worldwide, Java enables you to efficiently develop, deploy and use exciting applications and services.

From laptops to datacenters, game consoles to scientific supercomputers, cell phones to the Internet, Java is everywhere!



* 97% of Enterprise Desktops Run Java
* 89% of Desktops (or Computers) in the U.S. Run Java
* 9 Million Java Developers Worldwide
* #1 Choice for Developers
* #1 Development Platform
* 3 Billion Mobile Phones Run Java
* 100% of Blu-ray Disc Players Ship with Java
* 5 Billion Java Cards in Use
* 125 million TV devices run Java
* 5 of the Top 5 Original Equipment Manufacturers Ship Java ME

**Why Software Developers Choose Java**

Java has been tested, refined, extended, and proven by a dedicated community of Java developers, architects and enthusiasts. Java is designed to enable development of portable, high-performance applications for the widest range of computing platforms possible. By making applications available across heterogeneous environments, businesses can provide more services and boost end-user productivity, communication, and collaboration—and dramatically reduce the cost of ownership of both enterprise and consumer applications. Java has become invaluable to developers by enabling them to:

* Write software on one platform and run it on virtually any other platform
* Create programs that can run within a web browser and access available web services
* Develop server-side applications for online forums, stores, polls, HTML forms processing, and more
* Combine applications or services using the Java language to create highly customized applications or services
* Write powerful and efficient applications for mobile phones, remote processors, microcontrollers, wireless modules, sensors, gateways, consumer products, and practically any other electronic device

2.2 **GOOGLE APP-ENGINE**:

Google App Engine is a Platform as a Service (PaaS) offering that lets you build and run applications on Google’s infrastructure. App Engine applications are easy to build, easy to maintain, and easy to scale as your traffic and data storage needs change. With App Engine, there are no servers for you to maintain. You simply deploy your application and it’s ready to go.

Google App Engine makes it easy to build and deploy an application that runs reliably even under heavy load and with large amounts of data. It includes the following features:

* Persistent storage with queries, sorting, and transactions.
* Automatic scaling and load balancing.
* Asynchronous task queues for performing work outside the scope of a request.
* Scheduled tasks for triggering events at specified times or regular intervals.

**2.3 Facebook API:**

The core Facebook Platform API is the Graph API that allows you to read and write data to and from Facebook. Facebook also has what is called the Old Rest API. The newer Graph API changes the API paradigm from a method-oriented way of reading and writing data to and from Facebook to a new way that uses objects (think user profiles, friends, posts, photos, likes, and so on) and their relationships or connections with each other. This approach simplifies the Facebook API and makes it more consistent when working with objects. Note that while the Graph API is the preferred Facebook API, the Old REST API is still active and supported. Both the Graph and the REST APIs are applicable to mobile applications, both native and mobile web applications, including mobile web content within native applications through the use of WebViews.

Graph API objects are assigned a unique ID and are easily addressable using a URL that can be further qualified to address a specific object/connection.

The general structure of an object URL is as follows

https://graph.facebook.com/OBJECT\_ID/CONNECTION\_TYPEwhere OBJECT\_ID is the object's unique ID and CONNECTION\_TYPE is one of the connection types supported by the object. For example, a page supports the following connections: feed/wall, photos, notes, posts, members, and so on.

With the Graph API, you can retrieve an object, delete an object, and publish objects. You can search, update objects, filter results, and even dynamically discover the connections/relationships of an object.

**2.4 Gmail API:**

Designed to let you easily deliver Gmail-enabled features, this new API is a standard Google API, which gives RESTful access to a user’s mailbox under OAuth 2.0 authorization. It supports CRUD operations on true Gmail datatypes such as messages, threads, labels and drafts.As a standard Google API, you make simple HTTPS calls and get your responses in JSON, XML or Google Protobuf formats. You can also make these calls from standard web languages like Java and Python without using a TCP socket, which means the API is accessible from many cloud environments that couldn’t support IMAP.In contrast to IMAP, which requires access to all of a user’s messages for all operations, the new API gives fine-grained control to a user’s mailbox. For example, if your app only needs to send mail on behalf of a user and does not need to read mail, you can limit your permission request to send-only.To keep in sync, the API allows you to query the inbox change history, thereby avoiding the need to do “archaeology” to figure out what changed.Finally, a huge benefit is speed. feedback from pre-release developers suggest that the new Gmail.

**2.5 Yahoo Sock Price API:**

Yahoo Finance can be used to retrieve quotes and some extra information on stocks. There is also the possibility to include historical quotes on the requested stocks. When trying to get information on multiple stocks at once, please use the provided methods that accept a String of symbols to get the best performance. To retrieve the basic quote, statistics and dividend data a single request can be sent to Yahoo Finance for multiple stocks at once. For the historical data however, a separate request has to be sent to Yahoo Finance for each of the requested stocks. The provided methods will retrieve all of the required information in the least amount of requests possible towards Yahoo Finance. Please be aware that the data received from Yahoo Finance is not always complete for every single stock.

Stocks on the American stock exchanges usually have a lot more data available than stocks on other exchanges.

**CHAPTER 3**

**System Specification**

**3.1 Functionality Provided By the System**

When user logs in into the system with facebook or gmail. Suppose the user doesn’t have both accoutnts then next option will be gives the registration. once the user log into our application the user enter his cat marks as well as academic details. Student Placement Rating System is proposed to give the rating to the students based on their subjects, subject’s relevance in industry and users’ digital foot prints, where as the current rating scheme takes only academic examination scores into consideration.

* 1. **System interfaces, inputs, and outputs**

**3.2.1 Client Side Interfaces:**

**Login Tab**

**Inputs:** Username, Password.

**Output:** A message indication “login successful”.

* + 1. **Server Side Interfaces:**

**Login**

**Inputs:** Admin-id, Password.

**Output:** Login Successful.

CHAPTER 4

**System Modeling and Design**

**4.1 Class-Diagram:**

****

**4.2 Use Case Diagrams.**



**4.3 Sequence Diagram of Successful Login:**

****

**4.4 Unsuccessful login**



**4.5 Registration Successful:**



**4.6 Registration failed:**



**4.7 Successful Enter Cat-Marks**:

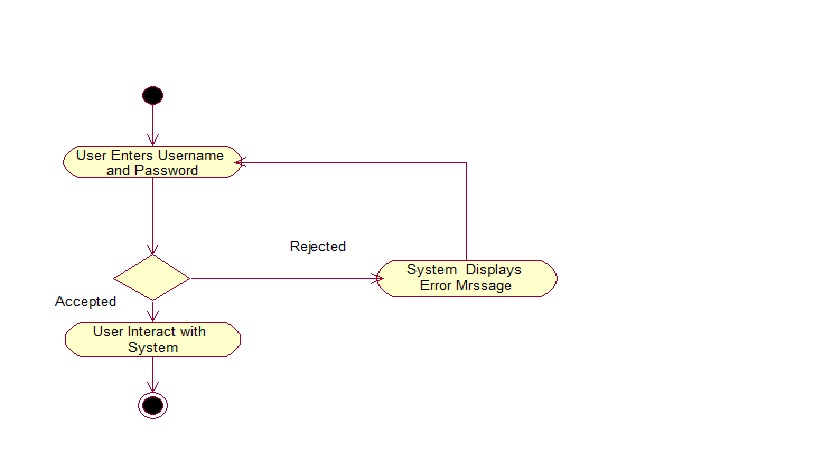


**4.8 Getting Rating:**



**4.4 Activity Diagram:**

**4.9 Activity Login.**



**4.10 Successful Registration:**



**4.11 Rating Activity:**



CHAPTER 5

**Google App Engine Architecture:**

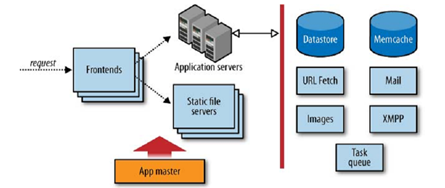


Figure 5.1 Google App Engine Architecture

**How it is works:**

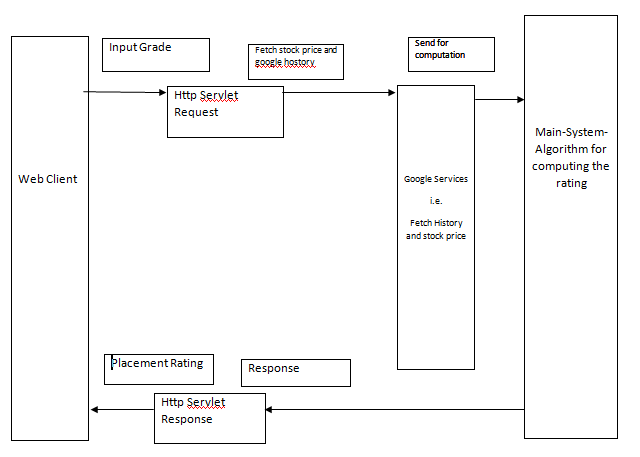
****

Figure 5.2 How it is works

**Single image shows details about project**

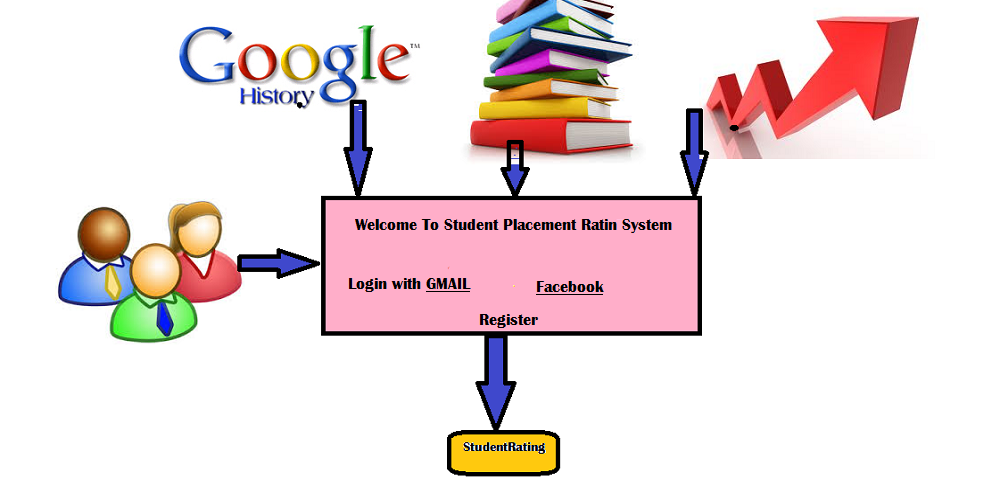
****

Figure 5.3Single image shows details working of application.

**CHAPTER 6**

**Software Requirement Specification**

***Abstract***

This shows the abstract system specification

|  |  |
| --- | --- |
| **System Specification-Abstract** | |
| R1 | Access security through “login and Password” and “role based privilege” of accessing information in administrative, user. |
| R2 | Provide user to register, the system, using with Facebook & Gmail. |
| R3 | Provide the service to the user 24/7. |
| R4 | System should provide user to enter cat-marks, final-marks. |
| R5 | System should provide how it works page to user. |
| R6 | System should provide the page(html),which companies are visiting to our college and they are using which technology |
| R7 | Fetch live stock price of companies. |
| R8 | User should have get rating option. |
| R9 | System should have the about-us page |
| R10 | Web enabled application |

**Software Requirement Specification Details.**

|  |  |
| --- | --- |
| **System Specification-Details:** | |
| R1 | Access Security through “login and Password” and “role based privilege” of accessing information in administrative, user.  **1.Login:**   * 1. **Brief Description:**   This describes how an administrator, user login to the system.  **1.2Flow Of events.**  **1.2.1Basic Flow:**  This use case starts when the user wishes to log into the System.   1. The system request that the actor to enter his\her username and password. 2. The actor enters his\her username and password.   The system validates the entered username and password and logs the actor into the system.  **1.2.2Alternative Flows:**  **1.2.2.1: Invalid username/password:** If, in the basic flow actor enter the  Wrong username/password the system displays error message. |
| R2 | **2.Register System using Facebook &Gmail:**  **2.1 Brief Description:**  This describes how to register to the system using with Facebook and Gmail.  **2.2Flow of events:**  **2.2.1Basic Flow:**  This use case starts when the actor want to register/login to the system.  1.The system should have the option of Gmail and  Face book to login into system.  2.If the user does not have Gmail and Face book  Account , than system should provide register  Option to user.    **2.2.2Alternative Flows:**  **2.2.2.1: Server Down:** If, the server down than system show the error message  To user registration/login failed. |
| R3 | **3. System Provide service to user 24/7:**  **3.1Brief Description:**  This describes how system provides the service to the user.  **3.2Flow of events:**  **3.2.1Basic Flow:**     1. System should provide the service 24/7 to user.   **3.2.2Alternative Flows:**  **3.2.2.1: Server Down:** If, the server down system give error message to  User, server is down application cannot available.  . |
| R4 | **4.System provide enter cat-marks, final-marks:**  **4.1Brief Description:**  This describes how to enter the cat-marks, final-marks to the system.  **4.2Flow of events:**  **4.2.1Basic Flow:**  This use case starts when the actor want to enter their cat-marks, final-  Marks to the system.   1. The user enters his\her user-name, password to login   To the system.  2. The system validates the given details and save to  Database.  3. System should how the option to user enter their cat-  Marks, final-marks.  **4.2.2Alternative Flows:**  **4.2.2.1: Invalid username/password:** If, in the basic flow user enters the  Information wrong, the systems give error message. |
| R5 | **5.System provide page how it works:**  **5.1Brief Description:**  This describes detail working of system.  **5.2Flow of events:**  **5.2.1Basic Flow:**     1. System should have the page how-it works. 2. The users see the detail working of the system.   **5.2.2Alternative Flows:**  **5.2.2.1**: **Server Down:** If, the server down than system show the error message to  User, the page temporally unavailable. |
| R6 | **6.System provide a link Google-web history:**  **6.1Brief Description:**  This describes how the system provides the link to Google-web history.  **6.2Flow of events:**  **6.2.1Basic Flow:**  1.The user enters his/her user-name password to login  The system.  2.The system validates the details and logged-in to the  System and save detail to the database.  3. System extracts the Google history of the user.  4. And fetch the Google history into data store.  **6.2.2Alternative Flows**  **6.2.2.1**: **Server Down:** If, the server is down system show the error message,  Server is down operation is failed. |
| R7 | **7.System provide list companies to visiting our college:**  **7.1 Brief description:**  This describes how many companies are visiting to our college.  **7.2.1Basic flow:**      1. The system allows the user to see the information  how many various companies are visited to our  College etc.  **7.2.2 Alternative flow:**  **7.2.2.1 Server Down:** If, the server down system show the error message to user  Server is down, cannot display the page.  . |
| R8 | **8 Fetch live stock price of company:**  **8.1 Brief Description:**  Here extracting the live stock price of the companies.  **8.2 Basic flow:**  This use case starts when the actor wishes to login to the system and  Want to see the live stock price of company.   1. The system request that the actor to enter his\her   Username and password.  2. The system validates the entered user name and  Password, and logged-in, into the system.  3. System internally extract the live stock price of  various companies.      **8.2.1 Alternative flow:**  **8.2.2 Google/Yahoo service unavailable:** If, Google API’s or Yahoo API’s does  not work properly than operation is failed. |
| R9 | **9. User should have rating option:**  **9.1 Brief Description:**  Here actor will get their rating.  **9.2. Basic flow:**  This use case starts when the actor wishes to login to the system  And want to get their rating.  1.The system request that the actor to enter his\her  Username and password.  2.The system validates the entered username and  Password and logs the actor into the system.  3.The system allows the actor to enter the cat-marks,  And final-marks to get the rating.  4.System gives the user final rating.  **9.2.1 Alternative flow:**  **9.2.2 Invalid username/password:** If, in the basic flow actor enter the wrong  Username/password the system displays error message**.** |
| R10 | **10.System have the about-us page:**  **10.1 Brief Description:**  This page describes information of developer this application  And their basic detail.  **10.2 Basic flow**  1. The system should have the about-us page.  2.System shows the developer information (basic detail)  **10.2.1 Alternative flow:**  **10.2.2 Server Down:** If, the server down system show the error message to user  Server is down, cannot display the page.  . |
| R11 | **13.web enable application:**  **13.1 Brief Description:**  The system should be of web enabled and all  Validation should be performed on the system. |

**Implementation**

Rating Calculation JSP code:

<%!

String usn="";

Double finalstock=0.0;

public void fetch(HttpServletRequest request){

usn=request.getParameter("txtusn").toString().trim();

}

%>

<table align="center" border="1" cellspacing="12"><tr><td>Your details</td></tr>

<tr><td>CAT1</td><td>CAT2</td><td>CAT3</td><td>SUBJECT</td><td>Grade</td></tr>

<%if(request.getParameter("btnsubmit")!=null){

Double cat1marks=0.0,cat2marks=0.0,cat3marks=0.0,gradeTemp=0.0;

Double finalmarks;

Double rating=0.0;

fetch(request);

//LiveStock();

DatastoreService datastore = DatastoreServiceFactory.getDatastoreService();

Query q2=new Query("AcademicDetails");

PreparedQuery pq= datastore.prepare(q2);

for(Entity Details:pq.asIterable()){

String usnnew=Details.getProperty("USN").toString();

if(usnnew.equalsIgnoreCase(usn)){

String cat1=Details.getProperty("CAT1").toString();

String cat2=Details.getProperty("CAT2").toString();

String cat3=Details.getProperty("CAT3").toString();

String grade=Details.getProperty("Grade").toString();

String subject=Details.getProperty("Subject").toString();

if(subject.equalsIgnoreCase("dms")){

cat1marks=cat1marks+Double.parseDouble(cat1.trim());

cat2marks=cat2marks+Double.parseDouble(cat2.trim());

cat3marks=cat3marks+Double.parseDouble(cat3.trim());

finalmarks=cat1marks+cat2marks+cat3marks;

gradeTemp=Double.parseDouble(grade);

finalmarks=finalmarks+gradeTemp;

if(finalmarks == 160 && finalmarks<=150){

rating=2.0;

}

else if( finalmarks >=140 && finalmarks<= 150){

rating=rating+1.9;

}

else if (finalmarks>=130 && finalmarks <=140){

rating=rating+1.8;

}

else if (finalmarks>=120 && finalmarks <=130){

rating=rating+1.7;

}

else if (finalmarks>=100 && finalmarks <=120){

rating=rating+1.5;

}

else if (finalmarks>=55 && finalmarks <=100){

rating=rating+1.25;

}

}

else if(subject.equalsIgnoreCase("oomd")){

cat1marks=cat1marks+Double.parseDouble(cat1.trim());

cat2marks=cat2marks+Double.parseDouble(cat2.trim());

cat3marks=cat3marks+Double.parseDouble(cat3.trim());

finalmarks=cat1marks+cat2marks+cat3marks;

gradeTemp=Double.parseDouble(grade);

finalmarks=finalmarks+gradeTemp;

if(finalmarks == 160 && finalmarks<=150){

rating=1.0;

}

else if( finalmarks >=140 && finalmarks<= 150){

rating=rating+0.9;

}

else if (finalmarks>=130 && finalmarks <=140){

rating=rating+0.8;

}

else if (finalmarks>=120 && finalmarks <=130){

rating=rating+0.7;

}

else if (finalmarks>=100 && finalmarks <=120){

rating=rating+0.5;

}

else if (finalmarks>=55 && finalmarks <=100){

rating=rating+0.25;

}

}

else if(subject.equalsIgnoreCase("se")){

cat1marks=cat1marks+Double.parseDouble(cat1.trim());

cat2marks=cat2marks+Double.parseDouble(cat2.trim());

cat3marks=cat3marks+Double.parseDouble(cat3.trim());

finalmarks=cat1marks+cat2marks+cat3marks;

gradeTemp=Double.parseDouble(grade);

finalmarks=finalmarks+gradeTemp;

if(finalmarks == 160 && finalmarks<=150){

rating=1.0;

}

else if( finalmarks >=140 && finalmarks<= 150){

rating=rating+0.9;

}

else if (finalmarks>=130 && finalmarks <=140){

rating=rating+0.8;

}

else if (finalmarks>=120 && finalmarks <=130){

rating=rating+0.7;

}

else if (finalmarks>=100 && finalmarks <=120){

rating=rating+0.5;

}

else if (finalmarks>=55 && finalmarks <=100){

rating=rating+1.0;

}

}

else if(subject.equalsIgnoreCase("ds")){

cat1marks=cat1marks+Double.parseDouble(cat1.trim());

cat2marks=cat2marks+Double.parseDouble(cat2.trim());

cat3marks=cat3marks+Double.parseDouble(cat3.trim());

finalmarks=cat1marks+cat2marks+cat3marks;

gradeTemp=Double.parseDouble(grade);

finalmarks=finalmarks+gradeTemp;

if(finalmarks == 160 && finalmarks<=150){

rating=1.0;

}

else if( finalmarks >=140 && finalmarks<= 150){

rating=rating+0.9;

}

else if (finalmarks>=130 && finalmarks <=140){

rating=rating+0.8;

}

else if (finalmarks>=120 && finalmarks <=130){

rating=rating+0.7;

}

else if (finalmarks>=100 && finalmarks <=120){

rating=rating+0.5;

}

else if (finalmarks>=55 && finalmarks <=100){

rating=rating+1.0;

}

}

else if(subject.equalsIgnoreCase("ada")){

cat1marks=cat1marks+Double.parseDouble(cat1.trim());

cat2marks=cat2marks+Double.parseDouble(cat2.trim());

cat3marks=cat3marks+Double.parseDouble(cat3.trim());

finalmarks=cat1marks+cat2marks+cat3marks;

gradeTemp=Double.parseDouble(grade);

finalmarks=finalmarks+gradeTemp;

if(finalmarks == 160 && finalmarks<=150){

rating=1.0;

}

else if( finalmarks >=140 && finalmarks<= 150){

rating=rating+0.9;

}

else if (finalmarks>=130 && finalmarks <=140){

rating=rating+0.8;

}

else if (finalmarks>=120 && finalmarks <=130){

rating=rating+0.7;

}

else if (finalmarks>=100 && finalmarks <=120){

rating=rating+0.5;

}

else if (finalmarks>=55 && finalmarks <=100){

rating=rating+1.0;

}

}

else if(subject.equalsIgnoreCase("cn")){

cat1marks=cat1marks+Double.parseDouble(cat1.trim());

cat2marks=cat2marks+Double.parseDouble(cat2.trim());

cat3marks=cat3marks+Double.parseDouble(cat3.trim());

finalmarks=cat1marks+cat2marks+cat3marks;

gradeTemp=Double.parseDouble(grade);

finalmarks=finalmarks+gradeTemp;

if(finalmarks == 160 && finalmarks<=150){

rating=1.0;

}

else if( finalmarks >=140 && finalmarks<= 150){

rating=rating+0.9;

}

else if (finalmarks>=130 && finalmarks <=140){

rating=rating+0.8;

}

else if (finalmarks>=120 && finalmarks <=130){

rating=rating+0.7;

}

else if (finalmarks>=100 && finalmarks <=120){

rating=rating+0.5;

}

else if (finalmarks>=55 && finalmarks <=100){

rating=rating+1.0;

}

}

else if(subject.equalsIgnoreCase("dbms")){

cat1marks=cat1marks+Double.parseDouble(cat1.trim());

cat2marks=cat2marks+Double.parseDouble(cat2.trim());

cat3marks=cat3marks+Double.parseDouble(cat3.trim());

finalmarks=cat1marks+cat2marks+cat3marks;

gradeTemp=Double.parseDouble(grade);

finalmarks=finalmarks+gradeTemp;

if(finalmarks == 160 && finalmarks<=150){

rating=1.0;

}

else if( finalmarks >=140 && finalmarks<= 150){

rating=rating+0.9;

}

else if (finalmarks>=130 && finalmarks <=140){

rating=rating+0.8;

}

else if (finalmarks>=120 && finalmarks <=130){

rating=rating+0.7;

}

else if (finalmarks>=100 && finalmarks <=120){

rating=rating+0.5;

}

else if (finalmarks>=55 && finalmarks <=100){

rating=rating+1.0;

}

}

else if(subject.equalsIgnoreCase("cd")){

cat1marks=cat1marks+Double.parseDouble(cat1.trim());

cat2marks=cat2marks+Double.parseDouble(cat2.trim());

cat3marks=cat3marks+Double.parseDouble(cat3.trim());

finalmarks=cat1marks+cat2marks+cat3marks;

gradeTemp=Double.parseDouble(grade);

finalmarks=finalmarks+gradeTemp;

if(finalmarks == 160 && finalmarks<=150){

rating=1.0;

}

else if( finalmarks >=140 && finalmarks<= 150){

rating=rating+0.9;

}

else if (finalmarks>=130 && finalmarks <=140){

rating=rating+0.8;

}

else if (finalmarks>=120 && finalmarks <=130){

rating=rating+0.7;

}

else if (finalmarks>=100 && finalmarks <=120){

rating=rating+0.5;

}

else if (finalmarks>=55 && finalmarks <=100){

rating=rating+1.0;

}

}

else if(subject.equalsIgnoreCase("ads")){

cat1marks=cat1marks+Double.parseDouble(cat1.trim());

cat2marks=cat2marks+Double.parseDouble(cat2.trim());

cat3marks=cat3marks+Double.parseDouble(cat3.trim());

finalmarks=cat1marks+cat2marks+cat3marks;

gradeTemp=Double.parseDouble(grade);

finalmarks=finalmarks+gradeTemp;

if(finalmarks == 160 && finalmarks<=150){

rating=1.0;

}

else if( finalmarks >=140 && finalmarks<= 150){

rating=rating+0.9;

}

else if (finalmarks>=130 && finalmarks <=140){

rating=rating+0.8;

}

else if (finalmarks>=120 && finalmarks <=130){

rating=rating+0.7;

}

else if (finalmarks>=100 && finalmarks <=120){

rating=rating+0.5;

}

else if (finalmarks>=55 && finalmarks <=100){

rating=rating+1.0;

}

}

else if(subject.equalsIgnoreCase("fafl")){

cat1marks=cat1marks+Double.parseDouble(cat1.trim());

cat2marks=cat2marks+Double.parseDouble(cat2.trim());

cat3marks=cat3marks+Double.parseDouble(cat3.trim());

finalmarks=cat1marks+cat2marks+cat3marks;

gradeTemp=Double.parseDouble(grade);

finalmarks=finalmarks+gradeTemp;

if(finalmarks == 160 && finalmarks<=150){

rating=1.0;

}

else if( finalmarks >=140 && finalmarks<= 150){

rating=rating+0.9;

}

else if (finalmarks>=130 && finalmarks <=140){

rating=rating+0.8;

}

else if (finalmarks>=120 && finalmarks <=130){

rating=rating+0.7;

}

else if (finalmarks>=100 && finalmarks <=120){

rating=rating+0.5;

}

else if (finalmarks>=55 && finalmarks <=100){

rating=rating+1.0;

}

}

else if(subject.equalsIgnoreCase("aoop")){

cat1marks=cat1marks+Double.parseDouble(cat1.trim());

cat2marks=cat2marks+Double.parseDouble(cat2.trim());

cat3marks=cat3marks+Double.parseDouble(cat3.trim());

finalmarks=cat1marks+cat2marks+cat3marks;

gradeTemp=Double.parseDouble(grade);

finalmarks=finalmarks+gradeTemp;

if(finalmarks == 160 && finalmarks<=150){

rating=1.0;

}

else if( finalmarks >=140 && finalmarks<= 150){

rating=rating+0.9;

}

else if (finalmarks>=130 && finalmarks <=140){

rating=rating+0.8;

}

else if (finalmarks>=120 && finalmarks <=130){

rating=rating+0.7;

}

else if (finalmarks>=100 && finalmarks <=120){

rating=rating+0.5;

}

else if (finalmarks>=55 && finalmarks <=100){

rating=rating+1.0;

}

}

else if(subject.equalsIgnoreCase("os")){

cat1marks=cat1marks+Double.parseDouble(cat1.trim());

cat2marks=cat2marks+Double.parseDouble(cat2.trim());

cat3marks=cat3marks+Double.parseDouble(cat3.trim());

finalmarks=cat1marks+cat2marks+cat3marks;

gradeTemp=Double.parseDouble(grade);

finalmarks=finalmarks+gradeTemp;

if(finalmarks == 160 && finalmarks<=150){

rating=1.0;

}

else if( finalmarks >=140 && finalmarks<= 150){

rating=rating+0.9;

}

else if (finalmarks>=130 && finalmarks <=140){

rating=rating+0.8; }

else if (finalmarks>=120 && finalmarks <=130){

rating=rating+0.7;

}

else if (finalmarks>=100 && finalmarks <=120){

rating=rating+0.5;

}

else if (finalmarks>=55 && finalmarks <=100){

rating=rating+1.0;

}

}

%>

Client side validation javascript code:

<script type="text/javascript">

function bsnum(e) {

var k;

document.all ? k = e.keyCode : k = e.which;

if( k === 8 || k === 32 || k===9 || (k >= 48 && k <= 57))

{

return true;

}

else

{

alert("Enter vaild no");

return false;

}

}

function csnum(e) {

var k;

document.all ? k = e.keyCode : k = e.which;

if( ((k => 64 && k < 91) || (k > 96 && k < 123) || k===9 || k===44 || k===46 || k === 8 || k === 32 || (k >= 48 && k <= 57)))

{

return true;

}

else

{

alert(" special charcater not allowed");

return false;

}

}

function onlnum(e) {

var k;

document.all ? k = e.keyCode : k = e.which;

if( k >= 48 && k <= 57)

{

return true;

}

else

{

alert("Enter only Numbers");

return false;

}

}

function onlchar(e) {

var k;

document.all ? k = e.keyCode : k = e.which;

if( k >= 65 && k <= 90 || k > 96 && k < 123)

{

return true;

}

else

{

alert("Enter only charecters");

return false;

}

}

</script>

Registration from jsp code:

<%!

String login,password,name,email,cellphone;

String msg;

public void fetch(HttpServletRequest request){

try{

login=request.getParameter("txt\_login");

password=request.getParameter("txt\_passwprd");

name=request.getParameter("txt\_name");

email=request.getParameter("txt\_email");

cellphone=request.getParameter("txt\_cellphone");

}catch(Exception e){

msg=e.getMessage();

}

}

public void insert(){

try{

Entity user=new Entity("StudentDeatils");

user.setProperty("Login\_id", login);

user.setProperty("Password", password);

user.setProperty("Name", name);

user.setProperty("Email", email);

user.setProperty("Cellphone", cellphone);

DatastoreService datastore=DatastoreServiceFactory.getDatastoreService();

datastore.put(user);

msg="Your Registration done successfully";

}catch(Exception e){

msg=e.getMessage();

}

}

%>

<% if(request.getParameter("btn\_submit")!=null){

fetch(request);

insert();

// response.getWriter().println(msg);

}

%>

Student Academic details entry:

<%!

int i;

int sem=0,cat1,cat2,cat3;

String semester="";

String subject\_name="";

int finalgrade;

String usn="",subject="";

String msg="";

String subname="";

public void input(HttpServletRequest request){

usn=request.getParameter("txtusn").toString().trim();

subject=request.getParameter("txtsubject").toString().trim();

cat1=Integer.parseInt(request.getParameter("txtcat1").toString().trim());

cat2=Integer.parseInt(request.getParameter("txtcat2").toString().trim());

cat3=Integer.parseInt(request.getParameter("txtcat3").toString().trim());

finalgrade=Integer.parseInt(request.getParameter("txtgrade").trim());

}

public void insert(){

try{

Entity studdetails=new Entity("AcademicDetails");

studdetails.setProperty("USN", usn);

studdetails.setProperty("Subject", subject);

studdetails.setProperty("CAT1", cat1);

studdetails.setProperty("CAT2", cat2);

studdetails.setProperty("CAT3", cat3);

studdetails.setProperty("Grade", finalgrade);

DatastoreService datastore=DatastoreServiceFactory.getDatastoreService();

datastore.put(studdetails);

msg="Your Insertion done successfully";

}catch(Exception e){

msg=e.getMessage();

}

}

%>

**CHAPTER 7**

**Testing**

**7.1 Testing objective**

Testing is a set of activities that can be planned in advance and conducted systematically. For this reason a template for software testing, a set of steps into which we can place specific test case design techniques and testing methods should be defined for software process. Testing often accounts for more effort than any other software engineering activity. If it is conducted haphazardly, time is wasted, unnecessary effort is expanded, and even worse, errors sneak through undetected. It would therefore seem reasonable to establish a systematic strategy for testing software.

**7.2 Integration testing**

Testing is done for each module. After testing all the modules, the modules are integrated and testing of the final system is done with the test data, specially designed to show that the system will operate successfully in all its aspects conditions. Thus the system testing is a confirmation that all is correct and an opportunity to show the user that the system works

**7.3 White box testing**

It is just the vice versa of the Black Box testing. There we do not watch the internal variables during testing. This gives clear idea about what is going on during execution of the system. The point at which the bug occurs were all clear and were removed.

**7.4 Black box testing**

In this testing we give input to the system and test the output. Here we do not go for watching the internal file in the system and what are the changes made on them for the required output

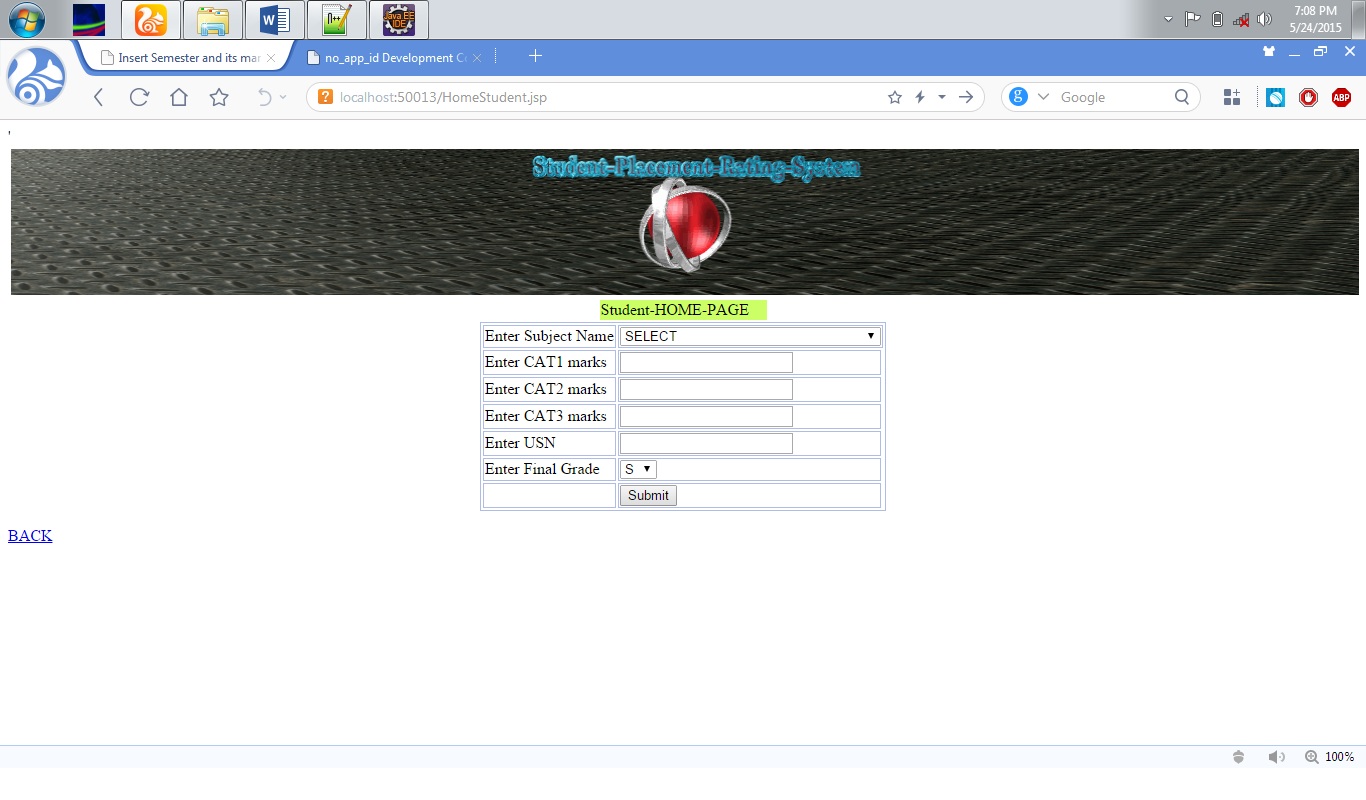
**Design Of Test Cases (Validation and Verification)**

|  |  |  |
| --- | --- | --- |
| User id | Password | Login |
|  |  | Successful login |
|  |  | Not successful |
|  |  | Not successful |
|  |  | Not successful |

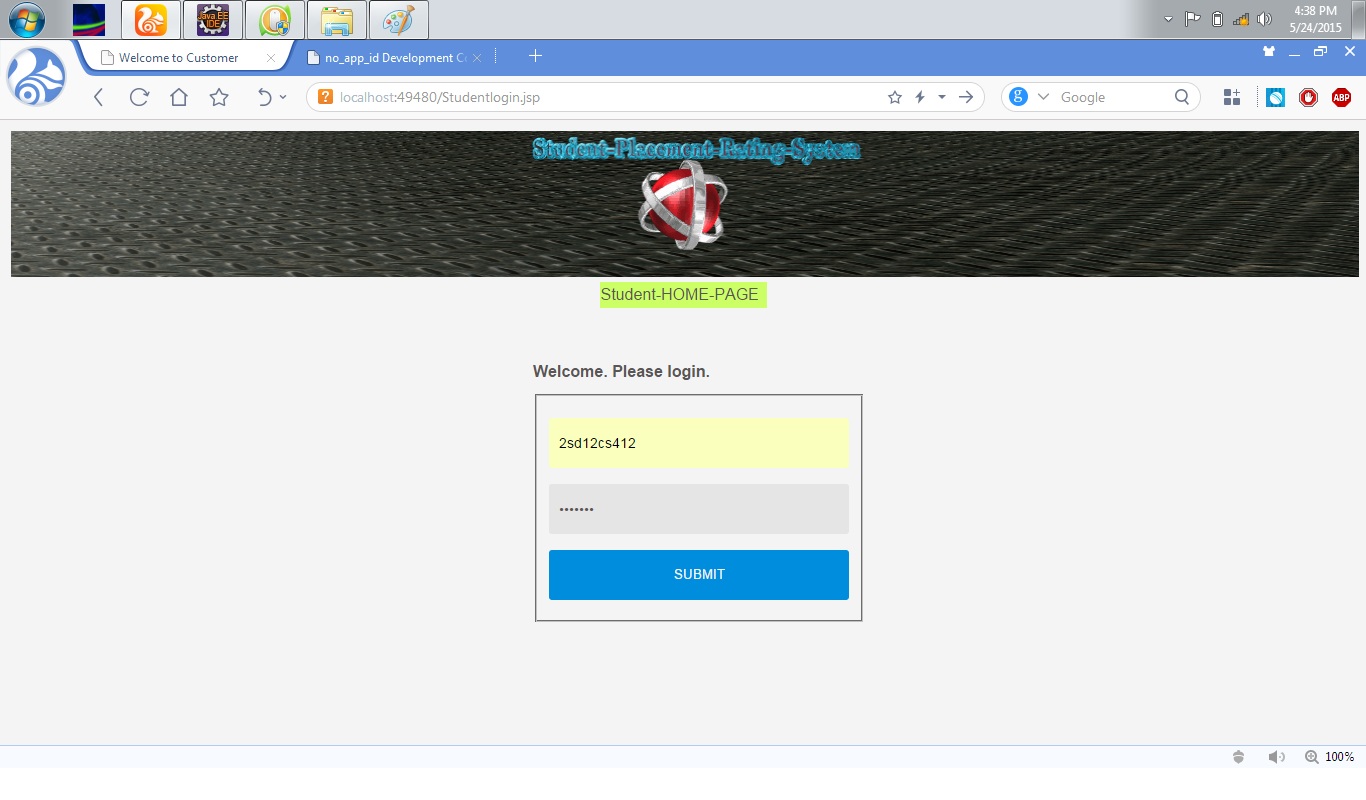
**Specification of user interface (Design of Front end)**

****

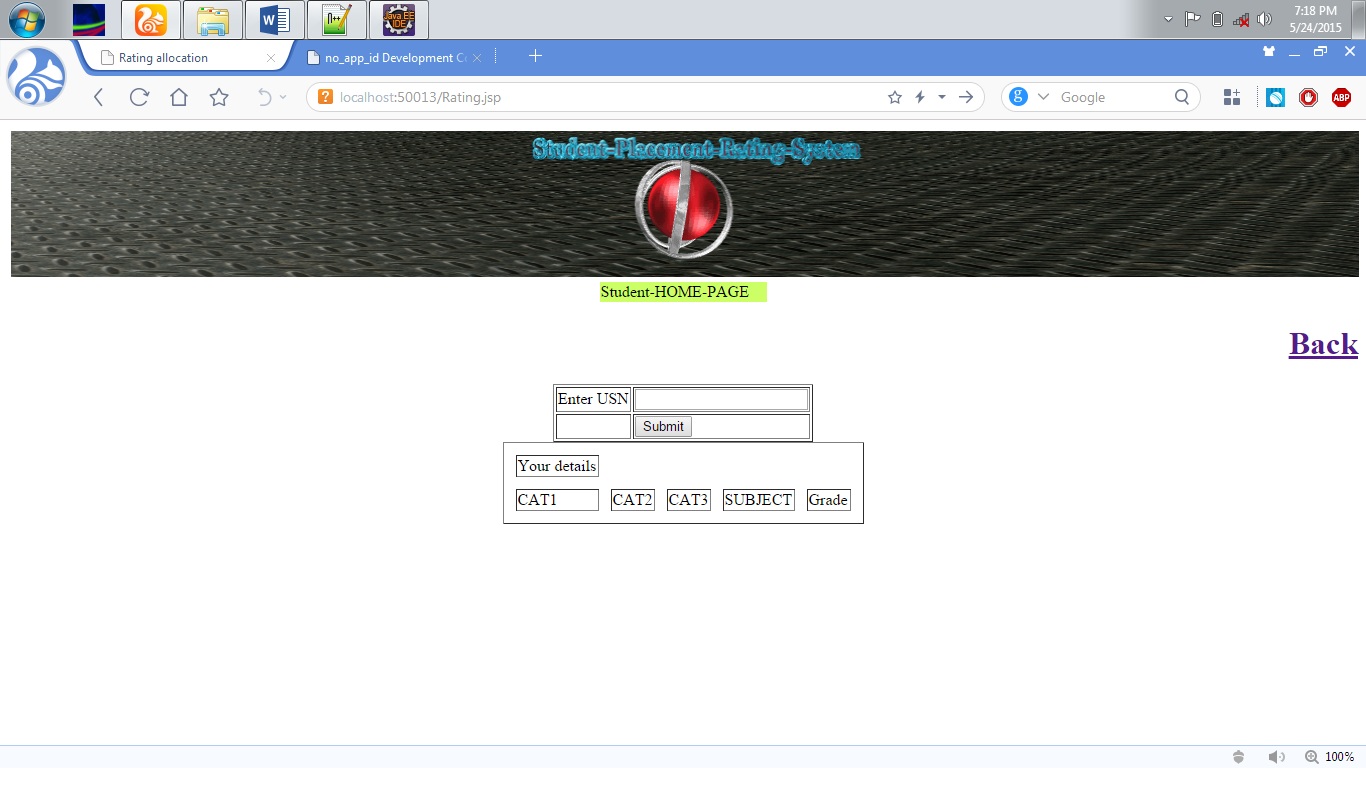
**Data Insert Page:**

****

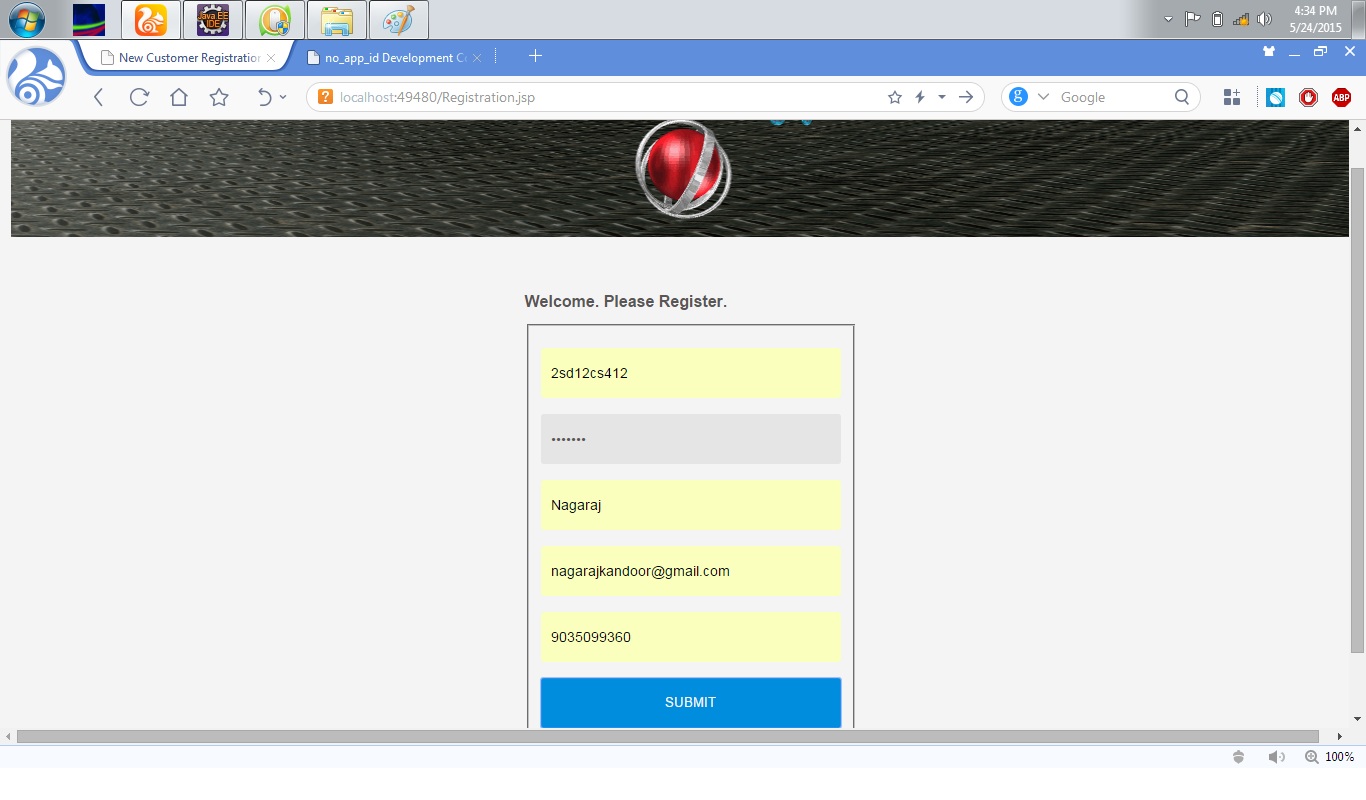
**Login Page:**

****

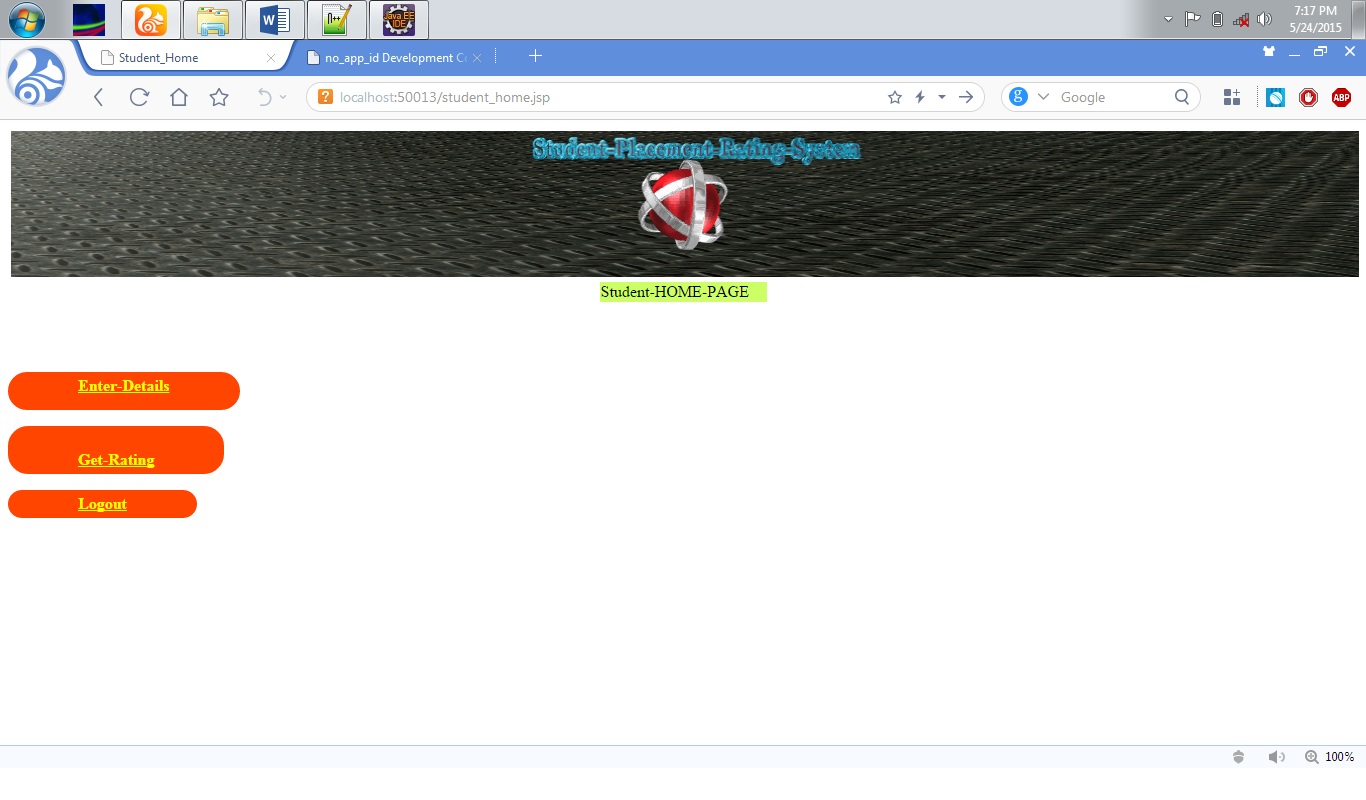
**Rating Page:**



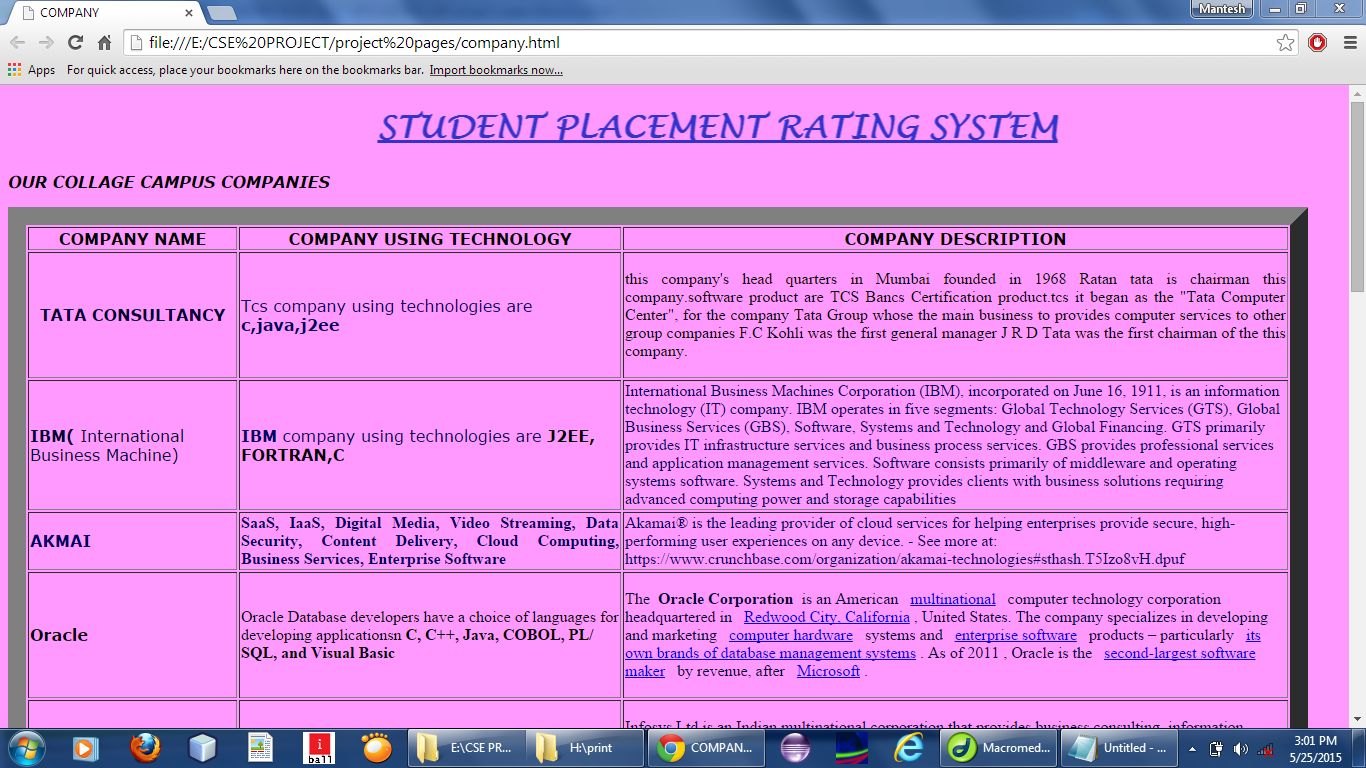
**Register Page:**

****

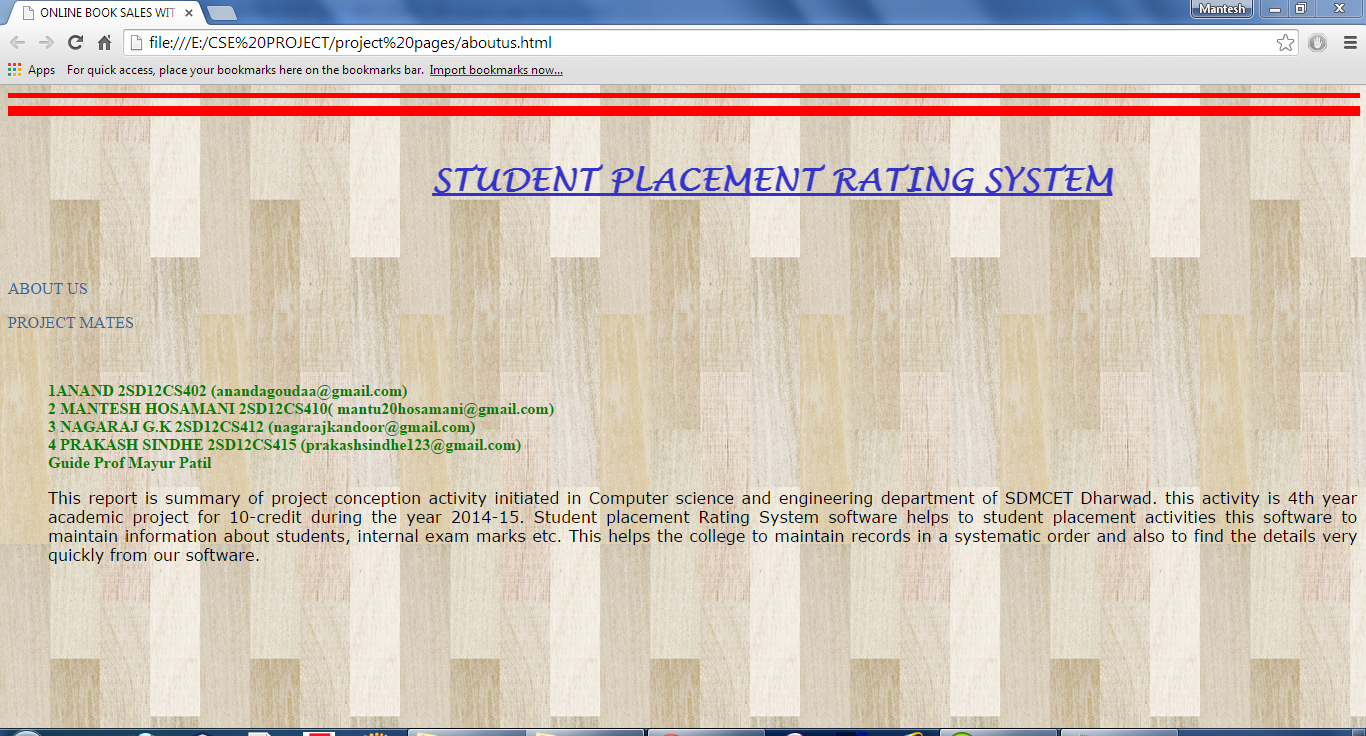
**Home Page (student):**

****

**Company Information Page.**

****

**About Us Page.**

****

**Conclusion**

**Conclusion and Future Enhancement.**

The main aim of developing “Student placement Rating System” software is to helps student placement activities. This software give rating to student based on his academic detail and google activity. This application is very helpful for student to know there ability of problem solving on subject and other activity of google. And this application also help company to hire the student.

In the future, we have the aim to add more and more futures to the website, like

1. Fetching Google History of student.
2. Fetching student Facebook likes, post and Gmail post etc….
3. Aim to add more and more future to enhancement application and to make application more interactive and useful etc…

|  |
| --- |
| **References.** |
| [1]Programming Google App Engine by Dan Sanderson.  [2]Using Google App Engine by Charles Severance.  [3] software engineering Ian Somerville 8th edtion,person eduction Ltd  [4]Stock overflow website.//http:// Stock overflow.com  [5]Yahoo Finance. //http:// Yahoo Finance.com. |