Automatic Question Paper Generator

(AQPG)

Project Report submitted in partial fulfillment of the
requirement for the degree of

Bachelor of Technology.

in

Computer Science Engineering and Information Technology

under the Supervision of

Dr. NITIN RAKESH

By

MONIL SOOD (091443)
NAINIKA THAKUR (091410)

to

Jaypee University of Information and Technology
Waknaghat, Solan – 173234, Himachal Pradesh

Certificate

This is to certify that work titled "**AUTOMATIC QUESTION PAPER GENERATOR**" submitted by "**Monil Sood (091443)** and **Nainika Thakur (091410)**" in partial fulfillment for the award of degree of Bachelor of Technology in Information Technology of Jaypee University of Information Technology, Waknaghat has been carried out under my supervision.

This work has not been submitted partially or wholly to any other university or institute for the award of this or any other degree or diploma.

Dr. Nitin Rakesh

Assistant Professor,

Department of Computer Science Engineering and Information Technology,

Jaypee University of Information Technology,

Waknaghat,

Acknowledgement

We express our heartfelt gratitude to all those who have contributed directly or indirectly towards obtaining our baccalaureate degree and at the same time, we cherish the years spent in the department of Computer Science Engineering and Information Technology. We are highly indebted to our esteemed supervisor, Dr. Nitin Rakesh, who has guided us through thick and thin. This project would not have been possible without his guidance and active support. His positive attitude towards research and zest for high quality research work has prompted us for its timely completion. We deem it a privilege to be working under Dr. Nitin Rakesh who has endeared himself to his students and scholars.

We are indebted to Prof. Ravi Prakash (Vice Chancellor, JUIT), Brig. (Retd.) Balbir Singh (Director, JUIT) and Prof. Dr. S.P. Ghrera (Head of the Department) for providing all necessary facilities to carry out our project. The help rendered by all our teachers, in one way or the other, is thankfully acknowledged.

We would also like to thank members of the lab and colleagues from other labs for their constant support. It was a pleasure to work with them.

As is usual we adore our parents and family and thank them for their best wishes and moral support. Computer Science is a fast developing subject and new dimensions are being added from time to time. During the last decade many new dimensions have been introduced and old ones redefined. In light of new developments and recent findings, we devote the task that was a part of our final year curriculum at Jaypee University of Information Technology to Automatic Question Paper Generator (AQPG), computer science and the society. Needless to say, errors and omissions are ours.

Nainika Thakur	(091410)
Monil Sood	(091443)

Dated: …………

Table of Content

List of Figures

Title **Page No.**

List of Tables

Abstract

Testing is one of the major aspects of Indian education system which requires setting of question papers related to disciplines and of varying difficulty levels. This process is time consuming and requires high level of confidentiality. An automated system for the process is the need of the hour which can provide both ease to the user and can save time. "Automatic Question Paper Generator" is a solution to this problem.

In this project we aim to develop one such automated system for our university. This system will allow the faculty and students to login to their respective departments whereby the faculty can upload questions and rate them and further when required can get any number of question papers generated from the data bank for exams. Also the faculty can put up assignments for the students. On the other hand, students can take mock papers and assignments.

This is a java based project which adheres to J2EE platform along with MySQL which will be used for database operations. It is a generalized project which can have its applications in schools, colleges and various other educational institutions.

Chapter -1
INTRODUCTION

Till date institutions have been solely relying on a panel of faculty for setting up the question papers for examinations. The process required additional faculty time as well as effort to solve redundancy and complexity issues. Also they had to set up additional quizzes and assignments for the students which again was hectic task.

1.1 Problem Statement

In this ever changing world technology has influenced our lives in a big way. As such the education community wanting to make use of this tremendous resource is rightly justified.

As discussed earlier the present paper setting system is manual. Generating confidential and redundancy free paper is a time taking and tedious process.

1.2 Purpose of AQPG

We have conceptualized this portal as a solution to the problem stated above which will facilitate easy and automatic generation of question papers for test taking. It integrates the benefits of experience of faculty with the convenience of an automated procedure for the generation of question papers. It will provide ease to the faculty and collaborative environment for the students.

1.3 Scope of AQPG

AQPG is a system currently being developed for JUIT, but it can be further be extended over the entire educational sector since test taking is a necessary criteria for student evaluation in every educational institution and thus the need for setting question papers arises where AQPG will have its advantages. It can also be used in places other than educational institutions where timely testing needs to be done like police services, companies etc.

1.4 Motivation

This project aims at integrating question paper setting, quizzes and assignment taking procedures in single web portal for an educational institution. This will mainly be used by faculty for generating exam papers and uploading assignments and by students to take quizzes. A streaming mode of operation which will be live or online will be backed by a rich GUI that will integrate all these tasks.

CHAPTER -2
LITERATURE SURVEY

We have used HTML, CSS and Java for making of the simplest prototype of Automatic Question Paper Generator. We have stored the text in session and have also used the MySQL database wherever required.

2.1 J2EE

Short for Java 2 Platform Enterprise Edition, J2EE is a platform-independent, Java-centric environment from Sun for developing, building and deploying Web-based enterprise applications online. The J2EE platform consists of a set of services, API's and protocols that provide the functionality for developing multitier, Web-based applications.

2.1.1 J2EE vs. .NET

The effort required to fully vet our technology choice paid off very well. Virtually we found it fairly easy to select J2EE for the overall development architecture. As a beginner, platform independence and scalability were the key issues. By selecting J2EE, we ensured that its applications would be deployable on a variety of platforms, enabling their customers to support our application on the platform that they were most familiar and comfortable with. The tools and technologies available in J2EE will make our applications easy to deploy, integrate, and manage. The availability of UNIX operating systems, if any, will provide increased uptime and reliability for the users and the applications are built to scale in order to meet the needs of our larger and rapidly growing enterprise class customers. .Net however is catching up but the growth is comparatively slower.

2.2 MySQL

Databases in general are useful, arguably the most consistently used family of software product. It is a systematically arranged collection of computer data, structured so that it can be automatically retrieved or manipulated. MySQL is a free database product. Java supports many database operations. MySQL is an open source, SQL Relational Database Management System (RDBMS) that is free for many uses. Early in its history, MySQL occasionally faced opposition due to its lack of support for some core SQL constructs such as sub-selects and foreign keys. Ultimately however, MySQL found a broad, enthusiastic user base for its liberal licensing terms, perky performance, and ease of use. Its acceptance was aided in part by the wide variety of other technologies such as Java, PHP, Perl, and Python have encouraged its use through stable, well documented modules and extensions. Like many competing products, both free and commercial, MySQL isn't a database until you give it some structure and form.

2.3 Benefits of Java and MySQL

Benefits of Java and MySQL are enlisted in tabular form as follows:

Item	ASP/SQL Server	ColdFusion MX/SQL Server	JSP/Oracle	Java/MySQL
Development tool	$0-2,499	$599	$0-~2,000	$0-249
Server	$999	$2,298	$0-~35,000	$0
RDBMS	$4,999	$4,999	$15,000	$0-220

TABLE 1: Benefits of Java and MySQL

2.4 Study of Similar Software

In order to undertake this project, we have studied the similar type of software already present in the market. Their features were taken into account while developing this project.

2.4.1 APG – by Harsh Eduserve ltd.

This was designed to generate automated question papers for Maharashtra state board, CBSE, ICSE and Colleges.

The key features are:

1. Any number of Question papers can be generated
2. Intuitive interface

2.4.2 Smart Question Paper Generator Software

Developed by knowledge ware India, the smart question paper generation software takes over the tedious task and automates the entire laborious work on school level.

The key features are:

1. Question papers can be generated for any class 1 to 12.
2. Question papers can be generated in any medium(English/Hindi/Guajarati)
3. Question paper can be generated for any subject.
4. Any type of question can be generated.

2.4.3 AIEPIC automatic question paper generator project

This was a project undertaken in AIEPIC by a group of students. Their project focused on developing automatic question paper generation software for college authorities.

The key features are:

1. Process different unique sets of papers.
2. Change question bank according to subject.
3. Generates class tests, unit tests, final tests.

2.4.4 Quick Quest

Quick Quest is software that is developed to generate question papers in any educational institution in the world having uniformly distributed complexity.

The key features are:

1. Single software solution for all types of institutions in world.
2. Stores unlimited questions.
3. Unlimited user accounts with three levels of security.

2.4.5 g-net QPG

This is designed to generate question papers based on degree, semester, subjects etc.

The key features are:

1. Web based package

2. Question paper with random selection of questions.
3. Department wise admin and users.

S.NO.	*FEATURES*	*APG*	*Smart-QPG*	*AIEPIC-AQPG*	*Quick Quest*	*g-net QPG*	*AQPG*
1.	*Time Saving*	✓	✓	✓	✓	✓	✓
2.	*Secure*	✓	✓	✓	✓	✓	✓
3.	*Unlimited papers*	✓	✓	✓	✓	✓	✓
4.	*Exam papers*	✓	✓	✓	✓	✓	✓
5.	*Online assignments*	-	-	-	-	-	✓
6.	*Difficulty level*	-	-	-	✓	✓	✓
7.	*Modify questions*	-	✓	-	✓	✓	✓
8.	*MCQs*	-	✓	-	✓	✓	✓
9.	*Java technology*	✓	-	-	✓	✓	✓

TABLE 1: COMPARISON WITH EXISTING SOFTWARES

2.5 Proposed System

2.5.1 Users

The portal will consist of three main users:

Admin: Admin is the main user of the portal. The job of admin can be explained as follows:

- The admin will get him authenticated by registering into the portal. Admin will be the first user to get registered to the portal failing which the portal will not work.
- Admin will also enter the subject details according to the semester, faculty ids and student ids in their respective tables and update them from time to time.

Faculty: Faculty is second type of possible user . The duties of the faculty are:

- Register to the portal with valid faculty-id
- Update the question bank
- Generate question paper of desired difficulty level
- Upload regular class assignments

Student

- Generate mock paper for practice
- Download class assignments
- Take quizzes

2.5.2 User Characteristics

The user is expected to be registered with the University for using AQPG. There will be access rights provided to faculty and students separately. However certain features of the system will be exclusive only to the faculty, in order to ensure the security and confidentiality of the system. Every user must be comfortable in using the computer. He /she must also be familiar with the internet and its usage and must be well versed in English.

2.5.3 Features of AQPG

Based on the study of the existing question paper generators and our own innovative ideas we have formulated the following features for AQPG:

- Time saving
- Robust
- Redundancy free
- Secure
- Effective
- Efficient

2.5.4 Functions

- User can generate unlimited number of question papers
- Faculty can update the question bank at their ease.
- Faculty can upload daily assignment for the students
- Students can get help by using mock test generator
- The difficulty level of question paper can be decided before hand.

2.6 Advantages

- The portal will make paper setting automated and time saving over manual procedure.
- Stimulate the productivity of test taking procedure.
- Enable faculty to adjust difficulty and number of questions as per their requirement.
- Improve student performance by regularly notifying about deadlines of assignments.
- Ensure security and privacy.

2.7 Limitations

- There can be more of technological limitations as compared to personal and traditional issues.
- Non feasible in the areas where internet is not accessible.
- There can be limited bandwidth problem such as traffic problems and internet connectivity.
- Difficulty in ensuring academic honesty in an online environment despite of various problems.
- Concern about the reputation and accreditation status of an automated paper generating mechanism.
- Users may still have a negative perception of automatic paper generating system.

CHAPTER -3
SPECIFIC REQUIREMENTS

3.1 External Interface Requirements

3.1.1 Product Perspective

- The web page (HTML/Servlets) is present to provide the user interface on customer client side. Customer and server communicate through HTTP/HTTPS protocols
- At the server , the java enabled web server is for the business logic and database server is for storing and furnishing the information
- The system is designed to automate the voluminous task of generating question papers

3.1.2 Software Interface

- Client on internet: web browser with any operating system
- Client on intranet: client software or web browser with any operating system
- Web server: glassfish v3.1 or above
- Development Tools: NetBeans (for J2EE,JSP,HTML)
- Database Server: MySQL server 5.5
- Operating system: Windows XP/Vista/7/8

3.1.3 Hardware Interface

	Processor	RAM	Disk Space
Client Side Chrome, Mozilla, IE 6.0 or above, etc.	Pentium III at 500 MHz(min)	512 MB (min)	40 GB(min)
Server Side Glassfish server	Pentium III at 1 GHz(min)	512 MB (min)	2 GB(min)
Database	Pentium III at 1	512 MB (min)	1 GB

MySQL server 5.5	GHz(min)		(excluding datasize)

TABLE 3: Hardware Interface Specifications of AQPG

3.2 Functional Requirements

- Automatically generated question papers of specified difficulty.
- Online quizzes and assignments for students.
- Faculty response to student queries.
- Faculty and students can register only if their record is present in university database.
- Only registered members can access the system.
- Essential steps to be taken to prevent authenticity infringement.
- Interface to be user friendly to provide maximum output in minimum time possible.
- Hardware independent system; should be able to run in any environment.

3.3 Non Functional Requirements

Nonfunctional (supplementary) requirements pertain to other information needed to produce the correct system and are detailed separately.

- **Performance**

The system to perform under all environmental conditions and should comply with maximum change possible.

- **Reliability**

Data produced to be reliable enough for using in the most important part of the educational system.

- **Availability**

24 X 7 availability of the system to the users.

- **Security**

Access rights to prevent the leakage of important information to irrelevant sources.

- **Maintainability**

Regular system checks for maintenance.

- **Portability**

The system to be portable and should work properly in all external environmental conditions.

CHAPTER 4
SOFTWARE DESIGN

Software design is the activity where software requirements are analyzed in order to produce a description of the internal structure and organization of the system that will serve as the basis for its construction. There are two activities:

- Software architectural design: the top level structure and organization of the system is described and various components are identified (how the system is decomposed and organized into components) and describe the interfaces between these components.

- Software implementation design: each component is precisely described to allow for its coding

The software design objectives can be listed as follows:

- To produce various models that can be analyzed and evaluated to determine if they will allow the various requirements to be fulfilled
- To examine and evaluate various alternative solutions and trade-offs
- To plan the subsequent development activities

Software system design results in the following product:

- A list of design goals derived from the non functional requirements
- Software architecture

Sub-system decomposition in terms of

- Responsibilities
- Dependencies
- Mapping to hardware

Major policy decision such as:

- Control flow
- Access control
- Data storage

4.1 Architectural Design

4.1.1 Data Flow Diagram

A data flow diagram (DFD) represents the flow of data through an information system modeling its process aspects in a graphical manner. Often they act as a preliminary step used to create an overview of the system which can be later elaborated. DFD's can also be used to visualize the process of data processing (structured design)

A DFD shows various kinds of data input and output from the system, where the data will come from and go to, and where the data will be stored. It contains no information about the timing of the processes or information about whether the processes will operate sequentially or in parallel. DFD is a graphical representation that depicts information flow and the transforms that are applied as data moves from input to output.

The basic form of DFD is also known as data flow graph or bubble chart. The DFDs are used to represent a system or software at any level of abstraction (Level 0, Level 1, etc.). The DFDs are mainly classified into two modules. They are:

- Level 0 DFD
- Level 1 DFD

4.1.1.1 Level 0 DFD

The level 0 DFD is also called as fundamental system model or a context model that represents the entire software element as a single bubble with input and output data indicated by incoming and outgoing arrows respectively. All the additional processes and information flow parts are represented in this form of DFD. The LEVEL 0 DFD of AQPG is given below:

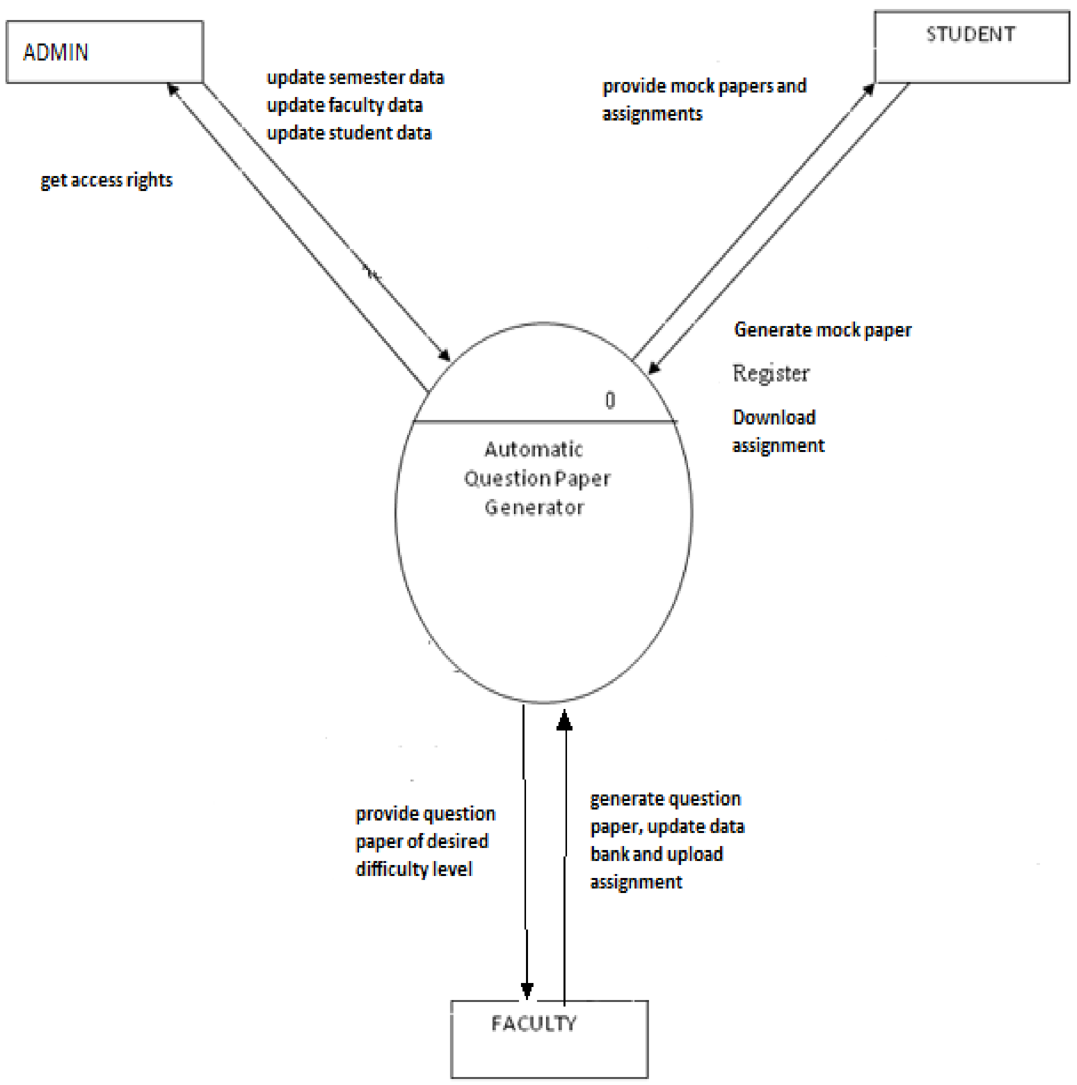

ADMIN
STUDENT
update semester data
update faculty data
update student data
provide mock papers and assignments
get access rights
Generate mock paper
Register
Download assignment
0
Automatic Question Paper Generator
provide question paper of desired difficulty level
generate question paper, update data bank and upload assignment
FACULTY

FIGURE 1: LEVEL 0 DFD FOR AQPG

4.1.1.2 Level 1 DFD

The next stage is to create a level 1 DFD. This level shows the whole system. The purpose of level 1 DFD is to describe the working of the system by placing identifiable processes of the level 1 DFD. It is more specific than context or) Level DFD. Generally it contains sub-processes of the general system processes. Given below is the Level 1 DFD of our system:

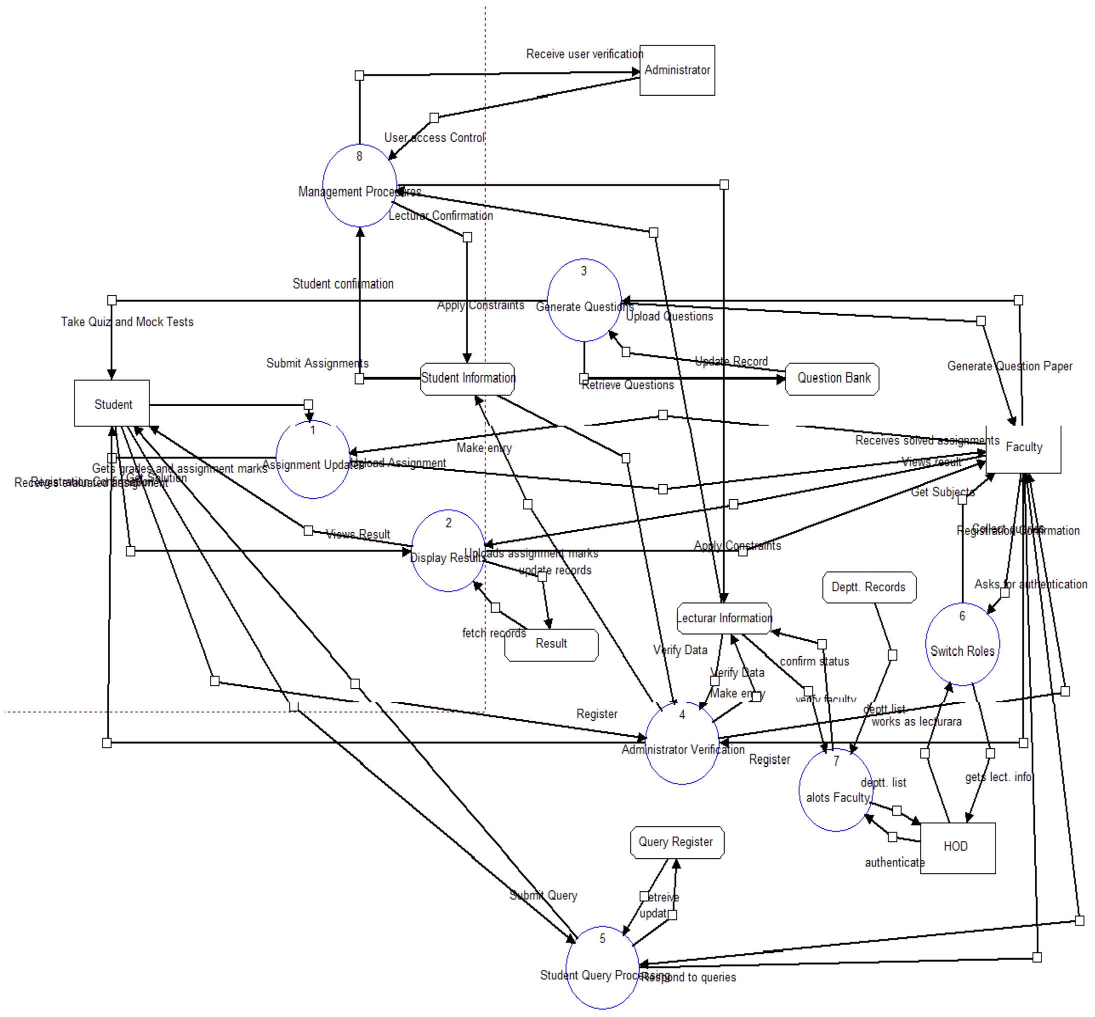


FIGURE 2: LEVEL 1 DFD FOR AQPG

4.1.2 Entity- Relationship Diagram

In software engineering, an entity – relationship(ER) model represents an abstract and conceptual form of data.ER modeling is a database modeling method, used to produce a type of conceptual schema or semantic data model of a system, often a relational database, and its requirements in a top-down fashion.

This ER diagram is complete blueprint of a project in which each and every operation performed at a time is explained briefly and fluently.

There are three basic elements in ER models:

- Entities are the things about which we seek information. In the area of the organization being modeled, entities are objects of interest.
- Attributes are the data which we gather about the entities. Attributes give us detailed information about an entity type.
- A relationship is basically an association between two entities and is represented as a straight line between two entities connecting them.

The ER diagram for AQPG is given below:

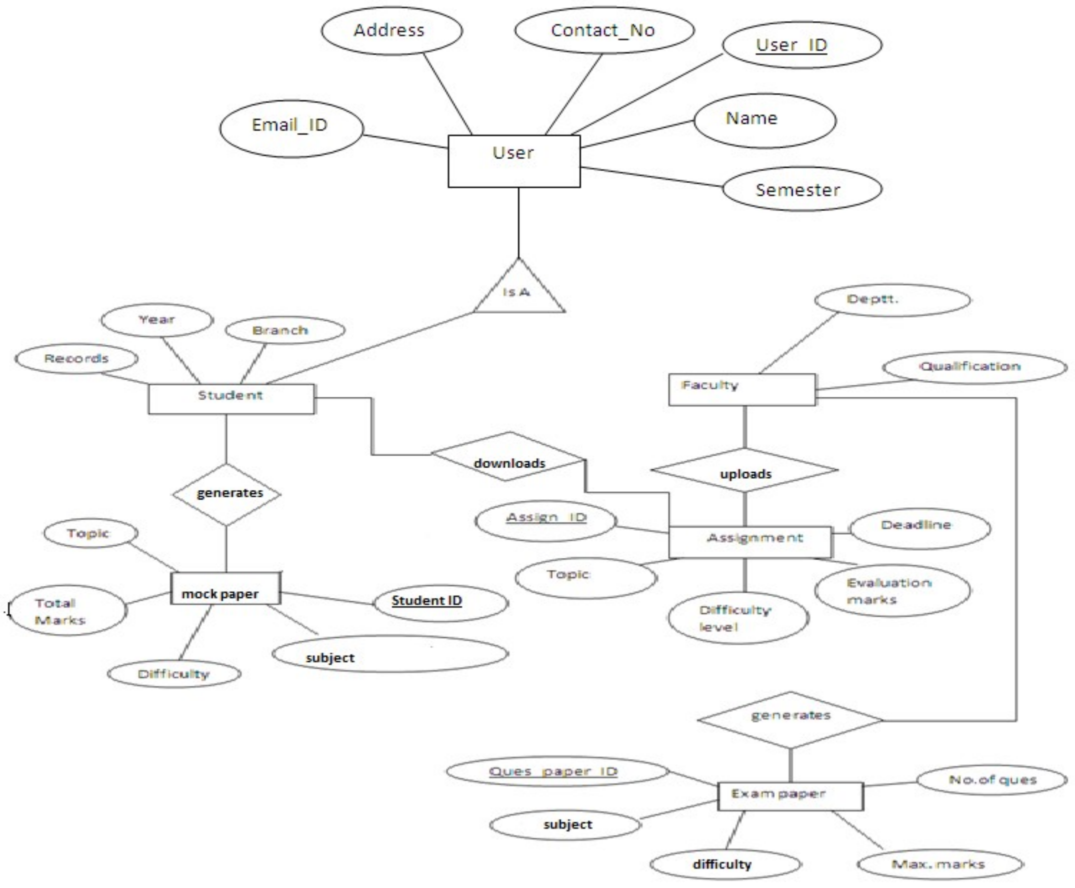


FIGURE 3 : ER DIAGRAM FOR AQPG

4.1.3 Use Case Model Description

In software and system engineering, a use case is a list of interactions between a role (known in UML as an actor) and a system, to achieve a goal. The actor can either be a human or an external system depending upon the problem statement.

Use cases are used at a higher level in systems engineering than within software engineering, representing missions or stakeholder goals.
A use case defines the interactions between external actors and the corresponding system under consideration to accomplish a particular goal. Actors must be able to make decisions, but need not be human: “an actor might be a person , company or organization, a program running a computer, or a computer system- hardware, software or middleware” Actors are always stakeholders, but not all stakeholders are actors, since they never interact directly with the system.

- **FUNCTIONS OF STUDENT**

Register- First of all students has to register so that he is able to access the facilities of “AUTOMATIC QUESTION PAPER GENERATOR”.

Download assignment- Students can download their assignments online.

Mock test- Students can take mock test any time they want.

Edit personal information- Students can update their profile date to date.

- **FUNCTIONS OF FACULTY**

Register- First of all faculty members has to register so that he is able to access the facilities of “AUTOMATIC QUESTION PAPER GENERATOR”.

Upload question- Faculty can upload any number of questions any time of various of their respective discipline.

Rate question- Faculty has to rate the questions as easy, medium, and hard at the time of uploading.

Upload assignment- Faculty uploads assignment providing deadline for submission.

Generate question paper- Faculty can generate any number of question paper of his/her respective subject/s.

- **FUNCTIONS OF ADMINISTRATOR**

Manage Subject Database- Admin has to update the database of subjects in particular semesterat regular intervals of time.

Access control- Admin looks that no one without required permissions can access the portal or database by authorizing faculty and student to the portal.

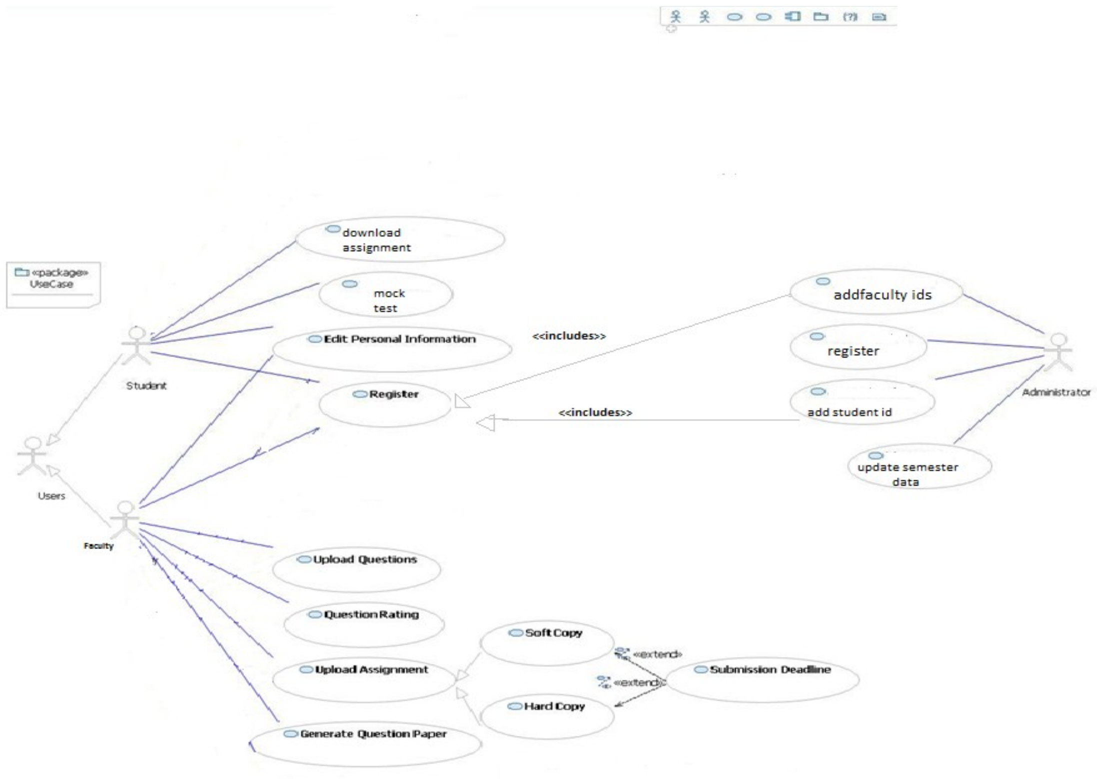


FIGURE 4: USE CASE DIAGRAM FOR AQPG

4.1.4 Schema Design

- LOGIN DATABASE

ATTRIBUTES	**DATATYPE**
USERNAME(PRIMARY KEY)	VARCHAR2(20)
PASSWORD	VARCHAR2(15)
USER_TYPE	VARCHAR2(15)

- FACULTY INFORMATION DATABASE

ATTRIBUTES	**DATATYPE**

NAME	VARCHAR2(15)
EMPLOYEE ID(PRIMARY KEY)	VARCHAR2(15)
ADDRESS	VARCHAR2(25)
CONTACT NUMBER	INTEGER(10)
EMAIL ID	VARCHAR2(20)
DEPARTMENT	VARCHAR2(15)
QUALIFICATIONS	VARCHAR2(15)

- STUDENT INFORMATION DATABASE

ATTRIBUTES	**DATATYPE**
STUDENT NAME	VARCHAR(15)
STUDENT ID(PRIMARY KEY)	VARCHAR(15)
ADDRESS	VARCHAR(25)
CONTACT NUMBER	VARCHAR(12)
EMAIL ID	VARCHAR(20)
DEPARTMENT	VARCHAR(15)
DOB	VARCHAR(15)

- ADMINISTRATOR DETAILS

ATTRIBUTES	**DATATYPE**
EMPLOEE ID	VARCHAR(15)
EMAIL ID	VARCHAR(15)
NAME	VARCHAR(20)
CONTACT NUMBER	VARCHAR(12)

- 1st SUBJECT QUESTION

ATTRIBUTES	**DATATYPE**

QUESTION ID(PRIMARY KEY)	VARCHAR(10)
DESCRIPTION	TEXT
RATING	VARCHAR(1)

- 2ND SUBJECT QUESTION

ATTRIBUTES	**DATATYPE**
QUESTION ID(PRIMARY KEY)	VARCHAR(10)
DESCRIPTION	TEXT
RATING	VARCHAR(1)

CHAPTER 5
IMPLEMENTATION

We have divided the whole project into 2 sessions and each session is further divided into respectively.

This whole segment is divided as follows:

- SESSION I

A) Making of a simple prototype

B) Database connectivity

C) Database Description

- SESSION II

Random Generation and Shuffling Algorithm

Revising existing modules

In the pages to follow each module consists of an involution into the topic and also our implementation and contribution to make it a success. Further, for each module, this report discusses the development of the module, the formative evolution design and results, and the revisions that were made in the modules.

5.1 SESSION I

A) Making of a Simple Prototype

AIM: To be able to understand the structure of prototype of **AQPG.**

We have used HTML, CSS and SERVLETS for making of the simple prototype of AQPG so that each one of us is well acquainted to proceed further.

Also, we stored the text in BLOB making the use of the databases for the storage of questions.

We began with the interface design using simple HTML and CSS for the static pages.

Result: The module was completed successfully within 10 days from the commencement of the work.

5.1.1 Interface Coding

- **Home Page- index.html**

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Home</title>
<meta charset="utf-8">
<link rel="stylesheet" href="css/reset.css" type="text/css" media="all">
<link rel="stylesheet" href="css/layout.css" type="text/css" media="all">
<link rel="stylesheet" href="css/style.css" type="text/css" media="all">
<script type="text/javascript" src="js/jquery-1.4.2.js" ></script>
<script type="text/javascript" src="js/cufon-yui.js"></script>
<script type="text/javascript" src="js/cufon-replace.js"></script>
<script type="text/javascript" src="js/Avenir_900.font.js"></script>
<script type="text/javascript" src="js/Avenir_300.font.js"></script>
<script type="text/javascript" src="js/Avenir_500.font.js"></script>
</head>
<body id="page1">
<div class="main">
<div class="body1"></div>
<!-- header -->
<header>
<div class="wrapper">
<h1><a href="index.html" id="logo">A.Q.P.G.</a></h1>
<nav>
<ul id="menu">
<li id="menu_active"><a href="index.html">Home</a></li>
<li ><a href="About.html">About Us</a></li>
<li ><a href="login1.html">LOG IN</a></li>
<li ><a href="redirect.html">REGISTER</a></li>
<li ><a href="Contacts.html">Contact Us</a></li>
</ul>
</nav>
</div>
<div id="banner">
<div class="text">
<h1>Automatic Question Paper Generator<span></br></br>Fast and
Secure</span></h1>
</div>
</div>
 </header>
<!-- / header -->
<!-- content --><div class="ic">More Website Templates at
TemplateMonster.com!</div>
<section id="content">
<div class="box1">
<div class="wrapper">
<h2>Services</h2>
<ul class="list1 cols  pad_left2">
<li><a href="Generation of papers.html">Generate Question Papers</a></li>
<li><a href="onlinequizzes.html">Online Quizzes</a></li>
<a href="http://www.abc.com"></a></li>
```

```
</ul>
<ul class="list1 cols  pad_left2">
<li><a href="onlineassignments.html">Online Assignments</a></li>
<li><a href="mocktests.html">Mock Tests</a></li>
</ul>
</div>
</div>
<div class="wrapper">
<article class="col1 pad_left1">
<h2>Submitted By:</h2>
<p><span class="color1">Nainika Thakur (091410)</span> <br>
</p>
<p><span class="color1">Monil Sood (091443)</span> <br>
</p>
</article>
</div>
</section>
<!-- content end-->
<!-- footer -->
<footer></footer>
<!-- footer end-->
</div>
<script type="text/javascript"> Cufon.now(); </script>
</body>
</html>
```

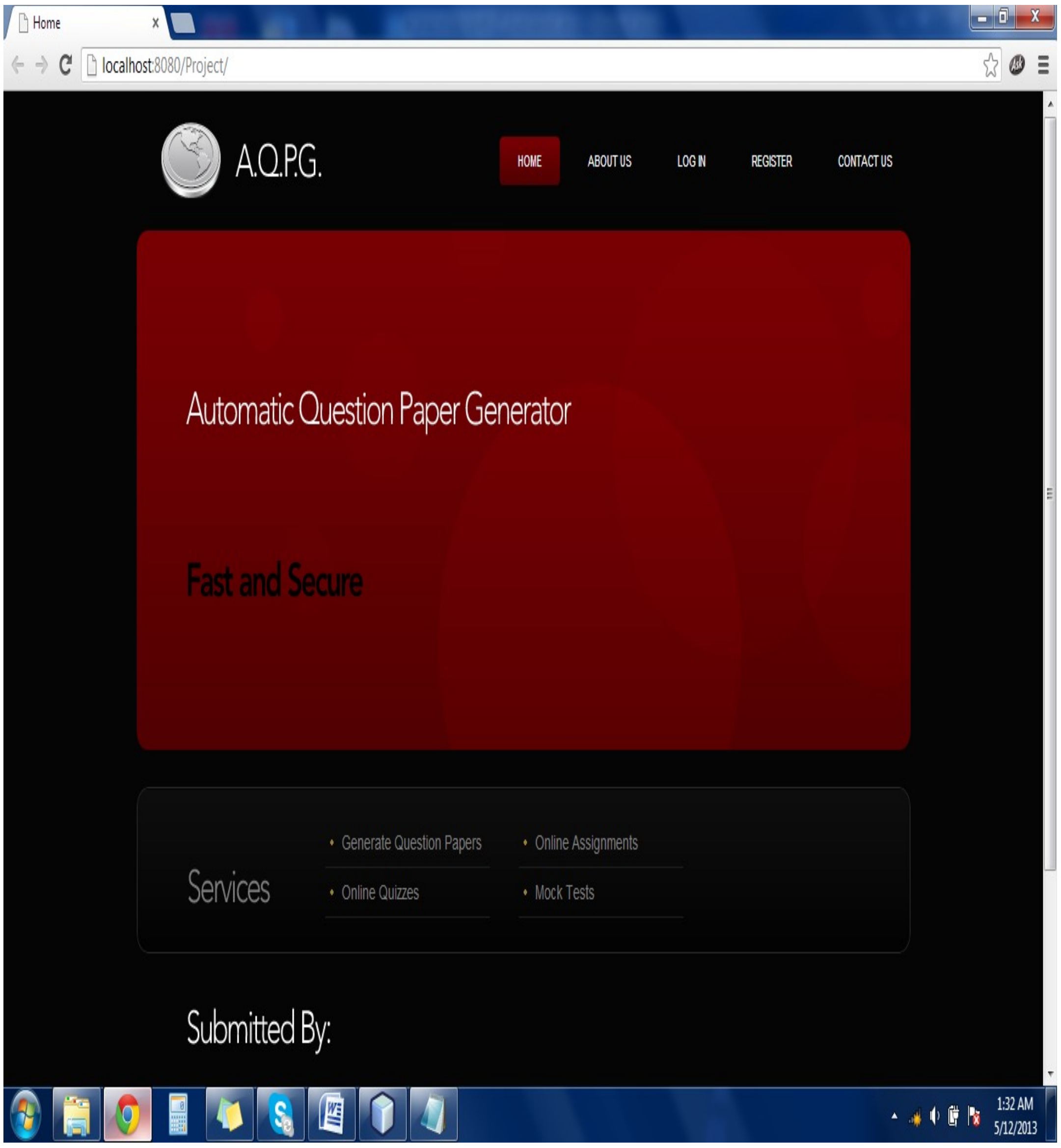


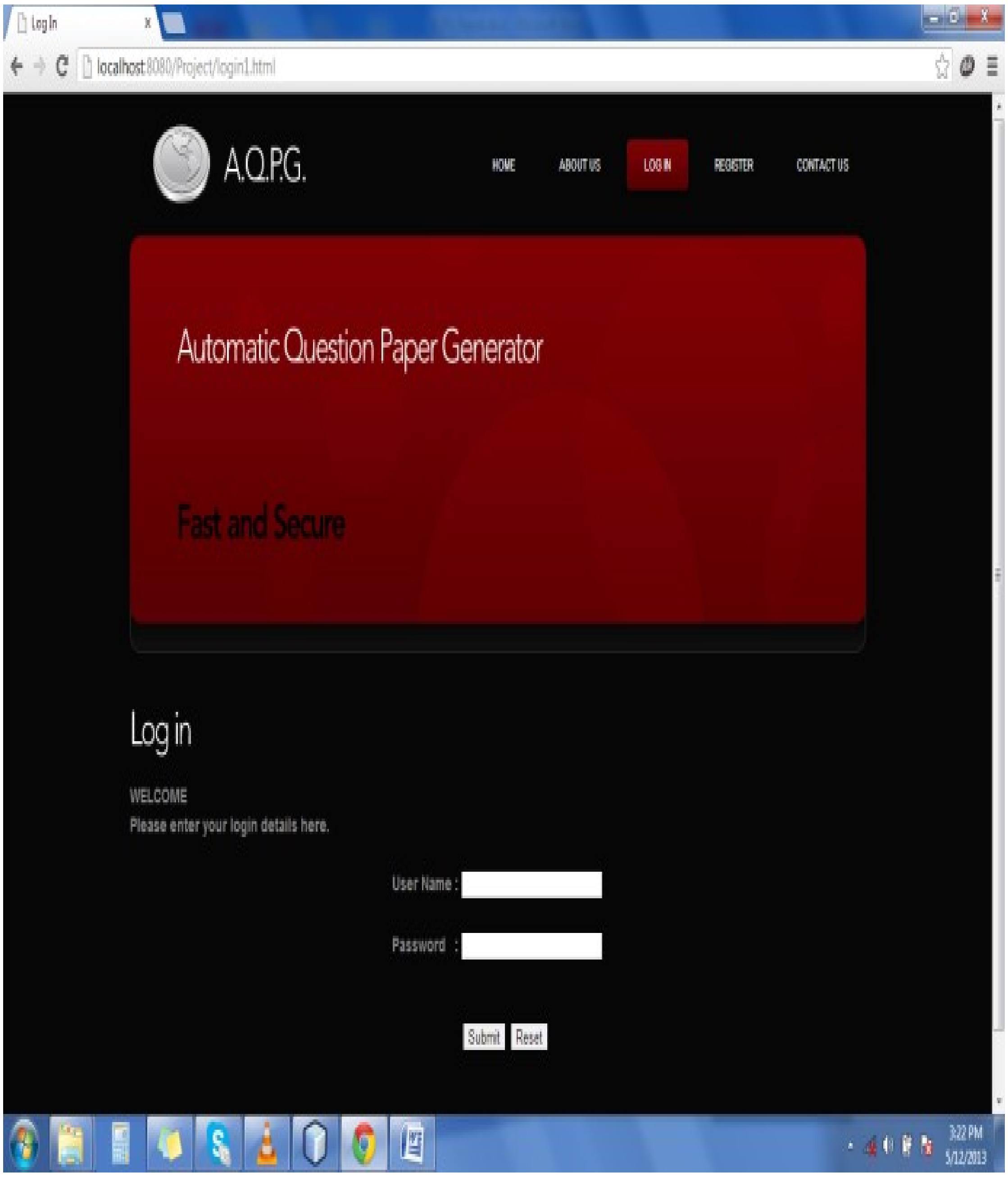
FIGURE 5: SNAPSHOT OF HOME PAGE

- **Login Page- login1.html**

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Log In</title>
<meta charset="utf-8">
<link rel="stylesheet" href="css/reset.css" type="text/css" media="all">
<link rel="stylesheet" href="css/layout.css" type="text/css" media="all">
<link rel="stylesheet" href="css/style.css" type="text/css" media="all">
<script type="text/javascript" src="js/jquery-1.4.2.js" ></script>
<script type="text/javascript" src="js/cufon-yui.js"></script>
<script type="text/javascript" src="js/cufon-replace.js"></script>
<script type="text/javascript" src="js/Avenir_900.font.js"></script>
<script type="text/javascript" src="js/Avenir_300.font.js"></script>
<script type="text/javascript" src="js/Avenir_500.font.js"></script>

</head>
<body>
 <div class="main">
<div class="body1"></div>
 <header>
<div class="wrapper">
<h1><a href="index.html" id="logo">A.Q.P.G.</a></h1>
<nav>
<ul id="menu">
<li ><a href="index.html">Home</a></li>
<li ><a href="About.html">About Us</a></li>
<li id="menu_active"><a href="login1.html">LOG IN</a></li>
<li ><a href="redirect.html">REGISTER</a></li>
<li ><a href="Contacts.html">Contact Us</a></li>
</ul>
</nav>
</div>
<div id="banner">
<div class="text">
<h1>Automatic Question Paper Generator<span></br></br>Fast and
Secure</span></h1>
</div>
</div>
</header>
<div id="content">
<div class="content_item">
<h2>Log in </h2>
<h3>WELCOME</h3> <b><p> Please enter your login details here.</p>
<center>
<form name="emailf" action="login" method="post">
<p>User Name : <input type="text" name="username" size=25>
</p>
<p>Password &nbsp : <input type="password" name="password" size=25>
</p>
```

```
<br>
&nbsp&nbsp&nbsp&nbsp&nbsp<input type="submit" value="Submit"></a>
<input type="reset" value="Reset"></p>
</form>
</center>
</div>
</div>
<div class="sidebar_container">
<div class="sidebar">
<div id="footer">
<p>Thanks for Visiting.</p>
</div>
</div>
</body>
</html>
```



Log in

WELCOME

Please enter your login details here.

User Name :

Password :

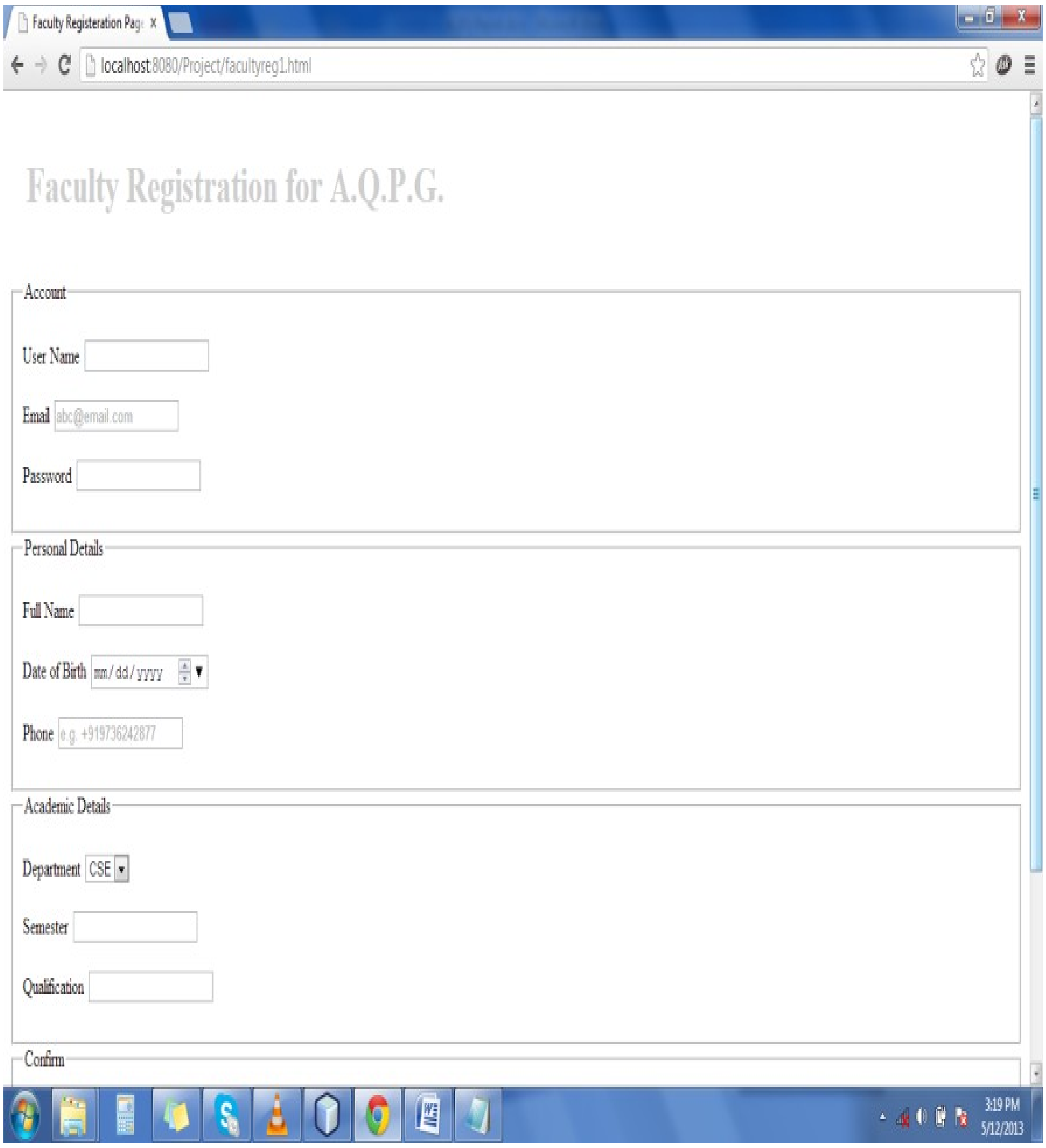
FIGURE 6: SNAPSHOT OF LOGIN PAGE

- **Registration Page – facultyreg1.html**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title> Faculty Registeration Page </title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
<meta name="description" content="Fancy Sliding Form with jQuery" />
<meta name="keywords" content="jquery, form, sliding, usability, css3,
validation, javascript"/>
<link rel="stylesheet" href="css/style1.css" type="text/css" media="screen"/>
<script type="text/javascript" src="jquery.min.js"></script>
<script type="text/javascript" src="sliding.form.js"></script>
</head>
<style>
span.reference{
position:fixed;
left:5px;
top:5px;
font-size:10px;
text-shadow:1px 1px 1px #fff;
}
span.reference a{
color:#555;
text-decoration:none;
text-transform:uppercase;
}
span.reference a:hover{
color:#000;
}
h1{
color:#ccc;
font-size:36px;
text-shadow:1px 1px 1px #fff;
padding:20px;
}
</style>
<body>
<div id="content">
<h1>Faculty Registration for A.Q.P.G.</h1>
<div id="wrapper">
<div id="steps">
<form id="formElem" action="facultyreg" name="formElem" method="post">
<fieldset class="step">
<legend>Account</legend>
<p>
<label for="username">User Name</label>
<input id="username" name="username" />
</p>
<p>
```

```
<label for="email">Email</label>
<input id="email" name="email" placeholder="abc@email.com" type="email"
AUTOCOMPLETE=OFF />
</p>
<p>
<label for="password">Password</label>
<input id="password" name="password" type="password"
AUTOCOMPLETE=OFF />
</p>
</fieldset>
<fieldset class="step">
<legend>Personal Details</legend>
<p>
<label for="name">Full Name</label>
<input id="name" name="name" type="text" AUTOCOMPLETE=OFF />
</p>
<p>
<label for="dob">Date of Birth</label>
<input id="dob" name="dob" type="date" AUTOCOMPLETE=OFF />
</p>
<p>
<label for="phone">Phone</label>
<input id="phone" name="phone" placeholder="e.g. +919736242877" type="tel"
AUTOCOMPLETE=OFF />
</p>
</fieldset>
<fieldset class="step">
<legend>Academic Details</legend>
<p>
<label for="deptt">Department</label>
<select>
<option value = "cse">CSE</option>
<option value = "ict">ICT</option>
<option value = "ece">ECE</option>
</select>
</p>
<p>
<label for="semester">Semester</label>
<input id="semester" name="semester" type="text" AUTOCOMPLETE=OFF />
</p>
<p>
<label for="name">Qualification</label>
<input id="qual" name="qual" type="text" AUTOCOMPLETE=OFF />
</p>
</fieldset>
<fieldset class="step">
<legend>Confirm</legend>
<p>
Everything in the form was correctly filled
if all the steps have a green checkmark icon.
```

```
A red checkmark icon indicates that some field
is missing or filled out with invalid data. In this
last step the user can confirm the submission of
the form.
</p>
<p class="submit">
<button id="registerButton" type="submit">Register</button>
</p>
</fieldset>
</form>
</div>
<div id="navigation" style="display:none;">
<ul>
<li class="selected">
<a href="#">Account</a>
</li>
<li>
<a href="#">Personal Details</a>
</li>
<li>
<a href="#">Confirm</a>
</li>
</ul>
</div>
</div>
</div>
<p align="center"><a href="index.html">Back to HOME</a>
</p>
</body>
</html>
```



Faculty Registeration Page

localhost:8080/Project/facultyreg1.html

Faculty Registration for A.Q.P.G.

Account

User Name

Email abc@email.com

Password

Personal Details

Full Name

Date of Birth mm/dd/yyyy

Phone e.g. +919736242877

Academic Details

Department CSE

Semester

Qualification

Confirm

3:19 PM
5/12/2013

FIGURE 7: SNAPSHOT OF REGISTRATION PAGE

- **Update Question Bank Interface 1 – firstuploadpage.html**

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Log In</title>
<meta charset="utf-8">
<link rel="stylesheet" href="css/reset.css" type="text/css" media="all">
<link rel="stylesheet" href="css/layout.css" type="text/css" media="all">
<link rel="stylesheet" href="css/style.css" type="text/css" media="all">
<script type="text/javascript" src="js/jquery-1.4.2.js" ></script>
<script type="text/javascript" src="js/cufon-yui.js"></script>
<script type="text/javascript" src="js/cufon-replace.js"></script>
<script type="text/javascript" src="js/Avenir_900.font.js"></script>
<script type="text/javascript" src="js/Avenir_300.font.js"></script>
<script type="text/javascript" src="js/Avenir_500.font.js"></script>
</head>
<body>
 <div class="main">
<div class="body1"></div>
<header>
<div class="wrapper">
<h1><a href="index.html" id="logo">A.Q.P.G.</a></h1>
<nav>
<ul id="menu">
<li ><a href="index.html">Home</a></li>
<li ><a href="About.html">About Us</a></li>
<li id="menu_active"><a href="login1.html">LOG IN</a></li>
<li ><a href="redirect.html">REGISTER</a></li>
<li ><a href="Contacts.html">Contact Us</a></li>
</ul>
</nav>
</div>
<div id="banner">
<div class="text">
<h1>Automatic Question Paper Generator<span></br></br>Fast and
Secure</span></h1>
</div>
</div>
</header>
<div id="content">
<div class="content_item">
<h2>WELCOME</h2> <b><p> Please enter details here.</p>
<center>
<form action="firstuploadpage">
branch: <input type="text" name="branch"><br>
semester: <input type="text" name="semester">
<br>
<input type="submit">
</form>
</center>
```

```
</div>
</div>
<div class="sidebar_container">
<div class="sidebar">
<div id="footer">
<p>Thanks for Visiting.</p>
</div>
</div>
</body>
</html>
```

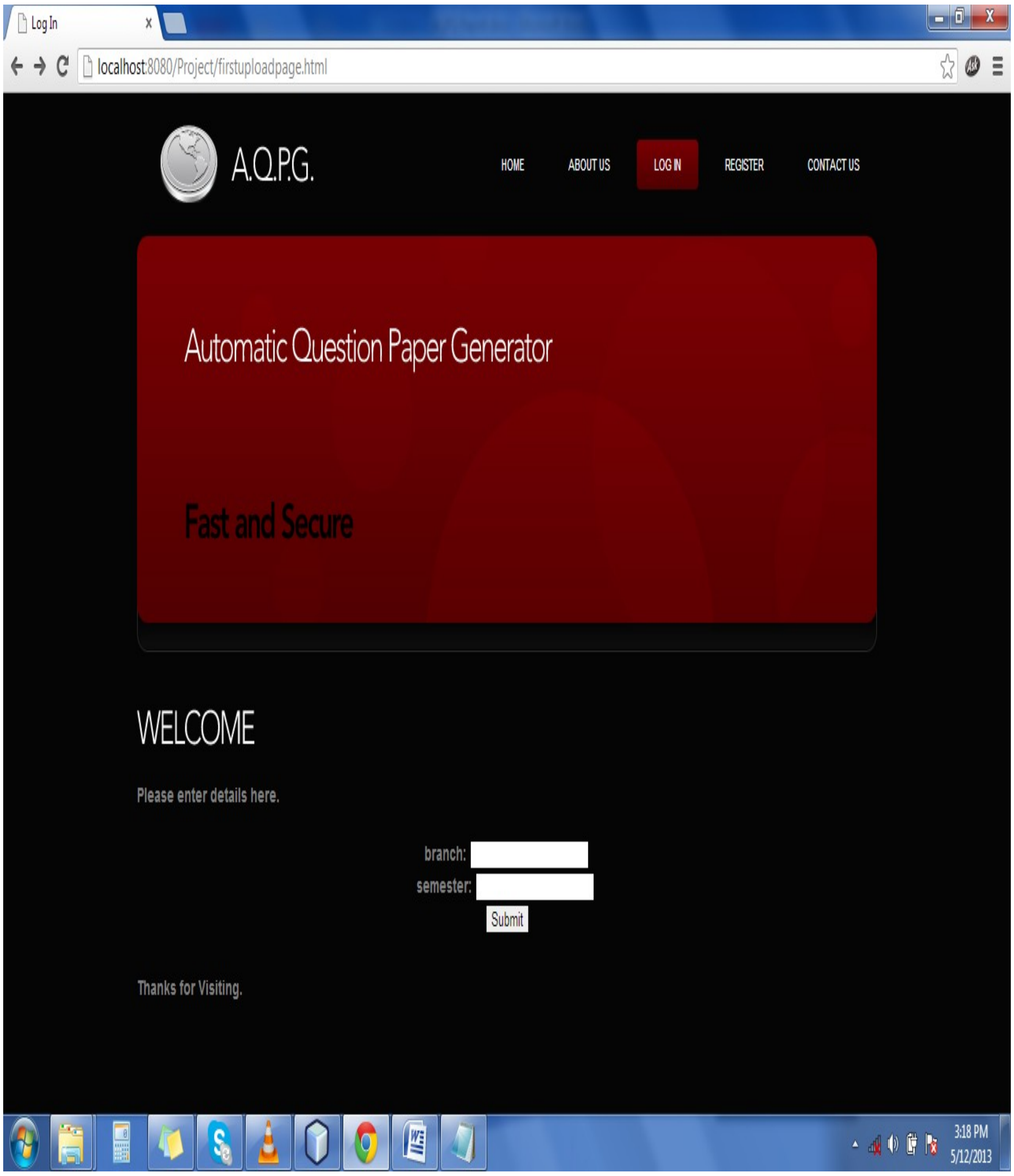
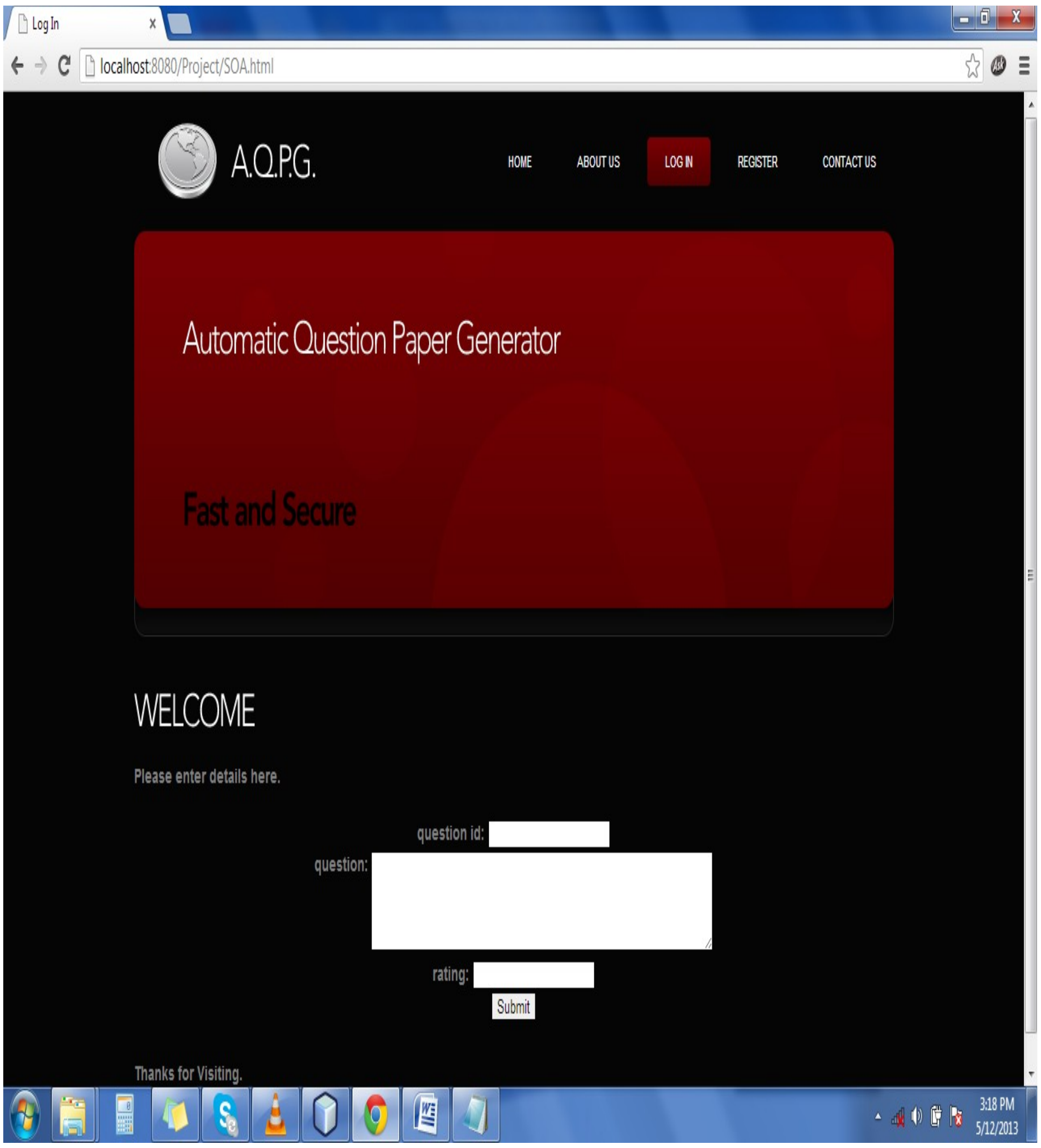


FIGURE 8: SNAPSHOT OF UPLOAD QUESTION BANK INTERFACE 1

- **Update Question Bank Interface 2 – subjectname.html**

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Log In</title>
<meta charset="utf-8">
<link rel="stylesheet" href="css/reset.css" type="text/css" media="all">
<link rel="stylesheet" href="css/layout.css" type="text/css" media="all">
<link rel="stylesheet" href="css/style.css" type="text/css" media="all">
<script type="text/javascript" src="js/jquery-1.4.2.js" ></script>
<script type="text/javascript" src="js/cufon-yui.js"></script>
<script type="text/javascript" src="js/cufon-replace.js"></script>
<script type="text/javascript" src="js/Avenir_900.font.js"></script>
<script type="text/javascript" src="js/Avenir_300.font.js"></script>
<script type="text/javascript" src="js/Avenir_500.font.js"></script>
</head>
<body>
<div class="main">
<div class="body1"></div>
<header>
<div class="wrapper">
<h1><a href="index.html" id="logo">A.Q.P.G.</a></h1>
<nav>
<ul id="menu">
<li ><a href="index.html">Home</a></li>
<li ><a href="About.html">About Us</a></li>
<li id="menu_active"><a href="login1.html">LOG IN</a></li>
<li ><a href="redirect.html">REGISTER</a></li>
<li ><a href="Contacts.html">Contact Us</a></li>
</ul>
</nav>
</div>
<div id="banner">
<div class="text">
<h1>Automatic Question Paper Generator<span></br></br>Fast and
Secure</span></h1>
</div>
</div>
</header>
<div id="content">
<div class="content_item">
<h2>WELCOME</h2> <b><p> Please enter details here.</p>
<center>
<form action="uploadsoa">
question id: <input type="text" name="qid"><br>
question: <textarea rows="4" cols="50" name="question">
</textarea><br>
rating: <input type="text" name="rating"><br>
<input type="submit">
</form>
```

```
</center>
</div>
</div>
<div class="sidebar_container">
<div class="sidebar">
<div id="footer">
<p>Thanks for Visiting.</p>
</div>
</div>
</body>
</html>
```



Log In

localhost:8080/Project/SOA.html

A.Q.P.G.

HOME ABOUT US LOG IN REGISTER CONTACT US

Automatic Question Paper Generator

Fast and Secure

WELCOME

Please enter details here.

question id:	
question:	
rating:	

Submit

Thanks for Visiting.

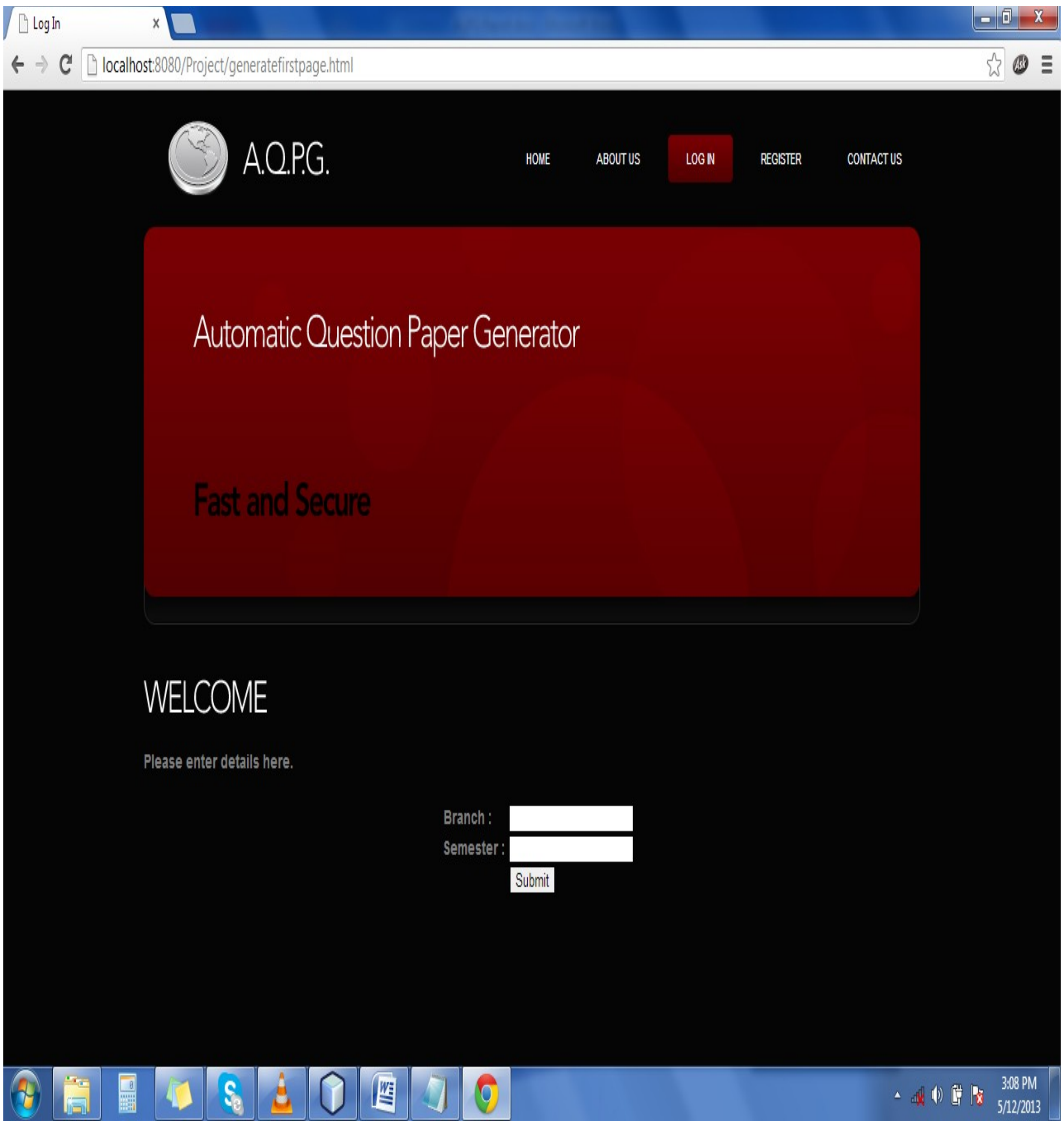
3:18 PM 5/12/2013

FIGUR 9: SNAPSHOT OF QUESTION UPLOAD INTERFACE 2

- **Generate Paper Interface 1 – generatefirstpage.html**

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Log In</title>
<meta charset="utf-8">
<link rel="stylesheet" href="css/reset.css" type="text/css" media="all">
<link rel="stylesheet" href="css/layout.css" type="text/css" media="all">
<link rel="stylesheet" href="css/style.css" type="text/css" media="all">
<script type="text/javascript" src="js/jquery-1.4.2.js" ></script>
<script type="text/javascript" src="js/cufon-yui.js"></script>
<script type="text/javascript" src="js/cufon-replace.js"></script>
<script type="text/javascript" src="js/Avenir_900.font.js"></script>
<script type="text/javascript" src="js/Avenir_300.font.js"></script>
<script type="text/javascript" src="js/Avenir_500.font.js"></script>
</head>
<body>
<div class="main">
<div class="body1"></div>
<header>
<div class="wrapper">
<h1><a href="index.html" id="logo">A.Q.P.G.</a></h1>
<nav>
<ul id="menu">
<li ><a href="index.html">Home</a></li>
<li ><a href="About.html">About Us</a></li>
<li id="menu_active"><a href="login1.html">LOG IN</a></li>
<li ><a href="redirect.html">REGISTER</a></li>
<li ><a href="Contacts.html">Contact Us</a></li>
</ul>
</nav>
</div>
<div id="banner">
<div class="text">
<h1>Automatic Question Paper Generator<span></br></br>Fast and
Secure</span></h1>
</div>
</div>
</header>
<div id="content">
<div class="content_item">
<h2>WELCOME</h2> <b><p> Please enter details here.</p>
<center>
<form action="firstgeneratepage">
&nbsp;&nbsp;&nbsp; Branch : &nbsp;&nbsp;&nbsp; <input type="text"
name="branch"><br>
&nbsp;&nbsp;&nbsp;&nbsp; Semester : <input type="text" name="semester">
<br>
<input type="submit">
</form>
```

```
</center>
</div>
</div>
<div class="sidebar_container">
<div class="sidebar">
<div id="footer">
</div>
</div>
</body>
</html>
```



WELCOME

Please enter details here.

Branch :

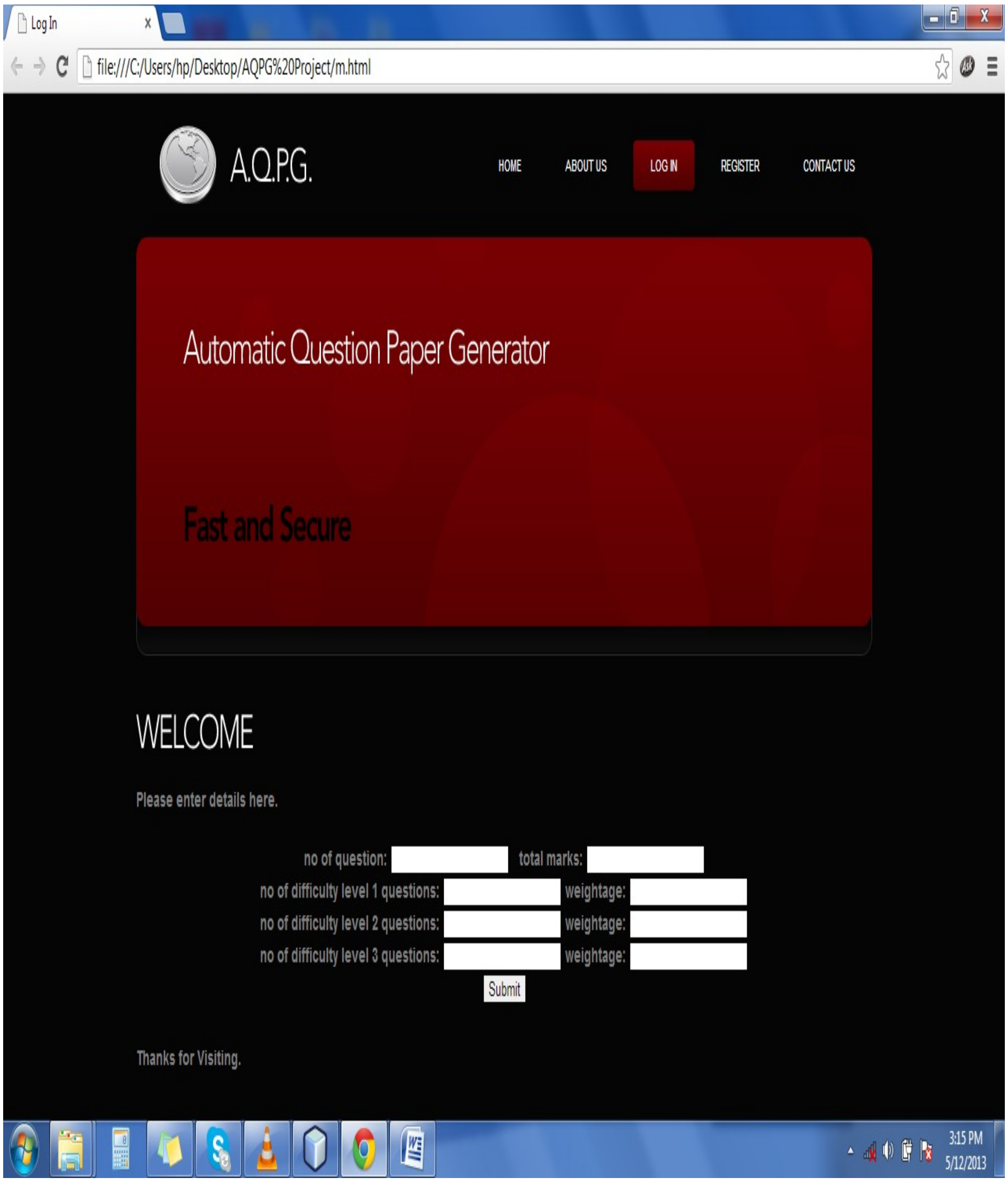
Semester :

FIGURE 10: SNAPSHOT OF PAPER GENERATION INTERFACE 1

- **Generate Paper Interface 2 – gensubname.html**

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Log In</title>
<meta charset="utf-8">
<link rel="stylesheet" href="css/reset.css" type="text/css" media="all">
<link rel="stylesheet" href="css/layout.css" type="text/css" media="all">
<link rel="stylesheet" href="css/style.css" type="text/css" media="all">
<script type="text/javascript" src="js/jquery-1.4.2.js" ></script>
<script type="text/javascript" src="js/cufon-yui.js"></script>
<script type="text/javascript" src="js/cufon-replace.js"></script>
<script type="text/javascript" src="js/Avenir_900.font.js"></script>
<script type="text/javascript" src="js/Avenir_300.font.js"></script>
<script type="text/javascript" src="js/Avenir_500.font.js"></script>
</head>
<body>
<div class="main">
<div class="body1"></div>
<header>
<div class="wrapper">
<h1><a href="index.html" id="logo">A.Q.P.G.</a></h1>
<nav>
<ul id="menu">
<li ><a href="index.html">Home</a></li>
<li ><a href="About.html">About Us</a></li>
<li id="menu_active"><a href="login1.html">LOG IN</a></li>
<li ><a href="redirect.html">REGISTER</a></li>
<li ><a href="Contacts.html">Contact Us</a></li>
</ul>
</nav>
</div>
<div id="banner">
<div class="text">
<h1>Automatic Question Paper Generator<span></br></br>Fast and
Secure</span></h1>
</div>
</div>
</header>
<div id="content">
<div class="content_item">
<h2>WELCOME</h2> <b><p> Please enter details here.</p>
<center>
<form action="gensoa" method="post">
no of question: <input type="text" name="no"> &nbsp;&nbsp;total marks: <input
type="text" name=" marks"><br>
no of difficulty level 1 questions: <input type="text" name="eques"> weightage:
<input type="text" name="weques"><br>
no of difficulty level 2 questions: <input type="text" name="mques">
weightage: <input type="text" name="wmques"><br>
```

```
no of difficulty level 3 questions: <input type="text" name="hques">   weightage:
<input type="text" name="whques"><br>
<input type="submit">
</form>
</center>
</div>
</div>
<div class="sidebar_container">
<div class="sidebar">
<div id="footer">
<p>Thanks for Visiting.</p>
</div>
</div>
</body>
</html>
```



WELCOME

Please enter details here.

no of question: total marks:
no of difficulty level 1 questions: weightage:
no of difficulty level 2 questions: weightage:
no of difficulty level 3 questions: weightage:

Thanks for Visiting.

FIGURE 11: SNAPSHOT OF PAPER GENERATION INTERFACE 2

5.1.2 CSS Files

- **Reset.css**

```
a, abbr, acronym, address, applet, article, aside, audio, b, blockquote, big, body, center, canvas, caption, cite, code, command, datalist, dd, del, details, dfn, dl, div, dt, em, embed, fieldset, figcaption, figure, font, footer, form, h1, h2, h3, h4, h5, h6, header, hgroup, html, i, iframe, img, ins, kbd, keygen, label, legend, li, meter, nav, object, ol, output, p, pre, progress, q, s, samp, section, small, span, source, strike, strong, sub, sup, table, tbody, tfoot, thead, th, tr, tdvideo, tt, u, ul, var {background:transparent;border:0 none;font-size:100%;margin:0;padding:0;border:0;outline:0;vertical-align:top}
ol, ul {list-style:none}
blockquote, q {quotes:none}
table, table td {padding:0;border:none;border-collapse:collapse}
img {vertical-align:top}
embed {vertical-align:top}
* {border:none}
```

- **Layout.css**

```
.col1, .col2, .col3, .cols {float:left}
.col1 {width:260px}
.col2 {width:500px}
.cols {width:200px}
/* index.html */
#page1 .box1 h2 {color:#808080;float:left;width:133px;padding:26px 0 0 30px}
#page1 .col2 .cols {width:226px}
#page1 .body1 {
background:url() center 0 no-repeat;
height:342px;
width:385px;
right:126px;
top:137px
}
#page1 #banner {background:url(../images/bg_banner.jpg) 0 0 no-repeat;height:365px}
#page1 .text {padding:103px 0 0 60px}
#page1 #content {padding-top:26px}
/* index-1.html */
#page2 .cols {width:220px}
/* index-2.html */
#page3 .cols {width:190px}
#page3 .line1 {background:url(../images/line_ver1.gif) 220px 0 repeat-y}
#page3 .box1 {margin-bottom:20px}
/* index-3.html */
/* index-4.html */
#page5 .cols {width:80px}
/* index-5.html */
/* index-6.html */
```

- **Style.css**

```
/* Getting the new tags to behave */
article, aside, audio, canvas, command, datalist, details, embed, figcaption, figure, footer, header, hgroup, keygen, meter, nav, output, progress, section, source, video {display:block}
mark, rp, rt, ruby, summary, time {display:inline}
.left {float:left}
.right {float:right}
.wrapper {width:100%;overflow:hidden}
/* Global properties */
body {background:#050505;border:0;font:14px Arial, Helvetica, sans-serif;color:#808080;line-height:20px}
.css3{border-radius:8px;-moz-border-radius:8px;-webkit-border-radius:8px;box-shadow:0 0 4px rgba(0, 0, 0, .4);-moz-box-shadow:0 0 4px rgba(0, 0, 0, .4);-webkit-box-shadow:0 0 4px rgba(0, 0, 0, .4);behavior:url(js/PIE.htc);position:relative}
.main {margin:0 auto;width:940px;position:relative;z-index:1}
.body1 {background:url(../images/bg_img2.png) center 0 no-repeat;height:318px;width:515px;right:0px;top:120px;position:absolute;z-index:2}
.ic, .ic a {border:0;float:right;background:#fff;color:#f00;width:50%;line-height:10px;font-size:10px;margin:-220% 0 0 0;overflow:hidden;padding:0}
a {color:#a1802b;text-decoration:underline;outline:none}
a:hover {text-decoration:none}
h1 {float:left;padding:19px 0 0 30px}
h2 {font-size:32px;line-height:40px;padding:30px 0 18px 0;color:#fff;font-weight:300;letter-spacing:-2px}
h2 span {font-size:17px;display:block;line-height:24px;font-weight:500;color:#b4b4b4;letter-spacing:-1px;margin-top:-6px}
p {padding-bottom:20px}
/* header */
header {width:100%;overflow:hidden}
#logo {display:block;background:url(../images/logo.jpg) 0 3px no-repeat;padding:0 0 0 90px;font-size:32px;color:#fff;font-weight:300;text-decoration:none;line-height:56px;letter-spacing:-2px}
#menu {float:right;padding:32px 0 0 0}
#menu li {float:left;padding-left:12px}
#menu li a {display:block;font-size:11px;color:#fff;text-transform:uppercase;text-decoration:none;line-height:34px;padding:0 22px;behavior:url(js/PIE.htc);position:relative;border-radius:4px;-moz-border-radius:4px;-webkit-border-radius:4px}
#menu li a:hover, #menu #menu_active a {background:url(../images/bg_menu_active.gif) top repeat-x #520001}
#banner {background:url(../images/bg_banner_2.jpg) 0 0 no-repeat;height:285px;width:100%;overflow:hidden;margin-top:23px}
.text {padding:53px 0 0 60px}
.text h1 {float:none;font-size:32px;line-height:40px;font-weight:300;color:#fff;padding:0 0 33px 0;letter-spacing:-2px}
.text h1 span {display:block;color:#000;font-weight:900}
```

```
.text .button {display:inline-block;background:url(../images/bg_button1.gif) top
repeat-x #b3b3b3;font-size:11px;color:#000;text-decoration:none;text-
transform:uppercase;line-height:34px;padding:0
20px;behavior:url(js/PIE.htc);position:relative;border-radius:4px;-moz-border-
radius:4px;-webkit-border-radius:4px}
.text .button:hover {background:#000;color:#fff}
/* content */
#content {width:100%;overflow:hidden;padding-top:0;padding-bottom:26px}
.pad_left1 {padding-left:60px}
.pad_left2 {padding-left:35px}
.pad_bot1 {padding-bottom:15px}
.pad_bot2 {padding-bottom:35px}
.marg_right1 {margin-right:28px}
.box1 {background:url(../images/bg_box1.gif) top repeat-x #050505;border:1px
solid #2a2a2a;behavior:url(js/PIE.htc);position:relative;border-radius:14px;-moz-
border-radius:14px;-webkit-border-radius:14px;padding:21px 30px 24px 30px}
.list1 li {line-height:34px;border-bottom:1px solid #1e1e1e}
.list1 .bg_none {border:none}
.list1 li a {color:#808080;text-decoration:none;padding-
left:20px;background:url(../images/marker_1.gif) 6px 6px no-repeat}
.list1 li a:hover {color:#a1802b}
.list2 {margin:-5px 0}
.list2 li {line-height:30px}
.list2 li a {color:#808080;padding-
left:12px;background:url(../images/marker_1.gif) 0 5px no-repeat}
.list2 li a:hover{color:#a1802b}
.list3 {margin:-12px 0}
.list3 li {line-height:35px}
.list3 li a{color:#808080;padding-
left:12px;background:url(../images/marker_1.gif) 0 5px no-repeat;text-
decoration:none}
.list3 li a:hover{color:#a1802b}
.color1 {color:#fff}
.line1 {background:url(../images/line_ver1.gif) 250px 0 repeat-y;padding:10px 0}
.link1 {color:#808080;text-decoration:none}
.link1:hover {text-decoration:underline}
/* footer */
footer {padding:35px 0 38px 0;text-align:center;font-size:12px;line-
height:18px;background:#0d0d0d}
footer a{}
footer a:hover{}
/* forms */
#ContactForm {line-height:24px}
#ContactForm a {margin-left:40px;float:right}
#ContactForm .input {float:right;width:386px;height:16px;border:1px solid
#373737;background:none;padding:3px 5px;margin-
bottom:6px;color:#808081;font:14px Arial, Helvetica, sans-serif}
#ContactForm textarea
{overflow:auto;float:right;width:386px;height:224px;border:1px solid
```

#373737;background:none;padding:3px 5px;margin-bottom:10px;color:#808081;font:14px Arial, Helvetica, sans-serif}

B) Database Connectivity

Here we make connection to the database by loading drivers and establishing connections by making use of various servlet commands which we studied and applied as under:

5.1.3 Database Connectivity Using Servlets

- **Code Snippet 1 – Register User**

```
/*
* To change this template, choose Tools | Templates
* and open the template in the editor.
*/
import java.sql.*;
import java.io.*;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
/**
*
* @author dell
*/
public class facultyreg extends HttpServlet {
/**
* Processes requests for both HTTP <code>GET</code> and
<code>POST</code> methods.
* @param request servlet request
* @param response servlet response
* @throws ServletException if a servlet-specific error occurs
* @throws IOException if an I/O error occurs
*/
protected void processRequest(HttpServletRequest request, HttpServletResponse
response)
throws ServletException, IOException
{
PrintWriter out = response.getWriter();
response.setContentType("text/html");
try
{
String uname=request.getParameter("username");
String upass=request.getParameter("password");
String utype="Faculty";
Class.forName("com.mysql.jdbc.Driver");
Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/project1","root",
"angrybird");
```

```
Statement st= con.createStatement();
String match="select fid from facultyid where fid='"+uname+"'";
ResultSet rs1=st.executeQuery(match);
if(rs1.next())
{
if(match.equals(uname))
{
String check="select employeeid from faculty where employeeid='"+uname+"'";
Statement stat=con.createStatement();
ResultSet rs=stat.executeQuery(check);
if(rs.next())
{
response.sendRedirect("facultyreg1.html");
}
String login="insert into login values(?,?,?);";
String faculty="insert into faculty values(?,?,?,?,?,?,?);";
PreparedStatement pst=con.prepareStatement(login);
PreparedStatement pst1=con.prepareStatement(faculty);
pst.setString(1,uname);
pst.setString(2,upass);
pst.setString(3,utype);
pst.executeUpdate();
String umail=request.getParameter("email");
String ufull=request.getParameter("name");
String uphone=request.getParameter("phone");
String qual=request.getParameter("qual");
pst1.setString(1,uname);
pst1.setString(2,umail);
pst1.setString(3,ufull);
pst1.setString(4,uphone);
pst1.setString(5,qual);
pst1.setString(6,qual);
pst1.setString(7,qual);
pst1.executeUpdate();
response.sendRedirect("index.html");
}
else
{
response.sendRedirect("facultyreg1.html");
}
else
{
response.sendRedirect("index.html");
}
}
catch(ClassNotFoundException e)
{
e.printStackTrace();
}
catch(SQLException e)
```

```
{
e.printStackTrace();
}
finally
{
out.close();
}
}
// <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the +
sign on the left to edit the code.">
/**
* Handles the HTTP <code>GET</code> method.
* @param request servlet request
* @param response servlet response
* @throws ServletException if a servlet-specific error occurs
* @throws IOException if an I/O error occurs
*/
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
processRequest(request, response);
}
/**
* Handles the HTTP <code>POST</code> method.
* @param request servlet request
* @param response servlet response
* @throws ServletException if a servlet-specific error occurs
* @throws IOException if an I/O error occurs
*/
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
processRequest(request, response);
}
/**
* Returns a short description of the servlet.
* @return a String containing servlet description
*/
@Override
public String getServletInfo() {
return "Short description";
}// </editor-fold>
}
```

- **Code Snippet 2 – Login User**

```
/*
* To change this template, choose Tools | Templates
* and open the template in the editor.
*/
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.sql.*;
/**
*
* @author dell
*/
public class login extends HttpServlet {
/**
* Processes requests for both HTTP <code>GET</code> and
<code>POST</code> methods.
* @param request servlet request
* @param response servlet response
* @throws ServletException if a servlet-specific error occurs
* @throws IOException if an I/O error occurs
*/
protected void processRequest(HttpServletRequest request, HttpServletResponse
response)
throws ServletException, IOException {
response.setContentType("text/html;charset=UTF-8");
PrintWriter out = response.getWriter();
try
{
Class.forName("com.mysql.jdbc.Driver");
Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/project1","root",
"angrybird");
Statement st= con.createStatement();
String username=request.getParameter("username");
String password= request.getParameter("password");
String login="select usertype from login where username='"+username+"' "+"and
password='"+password+"'";
ResultSet rs=st.executeQuery(login);
if(rs.next())
{
String usertype=rs.getString(1);
if(usertype.equals("Student"))
{
response.sendRedirect("studentwelcomepage.html");
}
else
if(usertype.equals("Faculty"))
{
response.sendRedirect("facultywelcomepage.html");
}
else if(usertype.equals("Admin"))
{
response.sendRedirect("adminwelcomepage.html");
}
}
else
{
response.sendRedirect("login1.html");
}
}
catch(ClassNotFoundException e)
{
}
catch(SQLException e)
{
}
finally {
out.close();
}
}
// <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the +
sign on the left to edit the code.">
/**
* Handles the HTTP <code>GET</code> method.
* @param request servlet request
* @param response servlet response
* @throws ServletException if a servlet-specific error occurs
* @throws IOException if an I/O error occurs
*/
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
processRequest(request, response);
}
/**
* Handles the HTTP <code>POST</code> method.
* @param request servlet request
* @param response servlet response
* @throws ServletException if a servlet-specific error occurs
* @throws IOException if an I/O error occurs
*/
@Override
```

```
protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
processRequest(request, response);
}
/**
* Returns a short description of the servlet.
* @return a String containing servlet description
*/
@Override
public String getServletInfo() {
return "Short description";
}// </editor-fold>
}
```

- **Code Snippet 3 – Upload Question Bank**

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.sql.*;
/**
*
* @author dell
*/
public class uploadsoa extends HttpServlet {
/**
* Processes requests for both HTTP <code>GET</code> and
<code>POST</code> methods.
* @param request servlet request
* @param response servlet response
* @throws ServletException if a servlet-specific error occurs
* @throws IOException if an I/O error occurs
*/
protected void processRequest(HttpServletRequest request, HttpServletResponse
response)
throws ServletException, IOException {
response.setContentType("text/html;charset=UTF-8");
PrintWriter out = response.getWriter();
try {
Class.forName("com.mysql.jdbc.Driver");
Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/project1","root",
"angrybird");
String uid=request.getParameter("qid");
String urate=request.getParameter("rating");
String uques=request.getParameter("question");
String check="select questionid from soa where questionid='"+uid+"'";
Statement stat=con.createStatement();
ResultSet rs=stat.executeQuery(check);
if(rs.next())
{
response.sendRedirect("SOA.html");
}
String upload="insert into soa values(?,?,?);";
PreparedStatement pst=con.prepareStatement(upload);
pst.setString(1,uid);
pst.setString(2,uques);
pst.setString(3,urate);
pst.executeUpdate();
response.sendRedirect("SOA.html");
}
catch(ClassNotFoundException e)
{
e.printStackTrace();
}
catch(SQLException e)
{
e.printStackTrace();
}
finally {
out.close();
}
}
// <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the +
sign on the left to edit the code.">
/**
* Handles the HTTP <code>GET</code> method.
* @param request servlet request
* @param response servlet response
* @throws ServletException if a servlet-specific error occurs
* @throws IOException if an I/O error occurs
*/
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
processRequest(request, response);
}
/**
* Handles the HTTP <code>POST</code> method.
* @param request servlet request
* @param response servlet response
* @throws ServletException if a servlet-specific error occurs
* @throws IOException if an I/O error occurs
*/
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
processRequest(request, response);
}
/**
* Returns a short description of the servlet.
* @return a String containing servlet description
*/
@Override
```

```
public String getServletInfo() {
return "Short description";
}// </editor-fold>
}
```

- **Code Snippet 4 – Generate Question Paper**

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.sql.*;
/**
*
* @author dell
*/
public class genhpca extends HttpServlet {
/**
* Processes requests for both HTTP <code>GET</code> and
<code>POST</code> methods.
* @param request servlet request
* @param response servlet response
* @throws ServletException if a servlet-specific error occurs
* @throws IOException if an I/O error occurs
*/
protected void processRequest(HttpServletRequest request, HttpServletResponse
response)
throws ServletException, IOException {
response.setContentType("text/html;charset=UTF-8");
PrintWriter out = response.getWriter();
out.println("<html><body>");
try {
int i=1;
Class.forName("com.mysql.jdbc.Driver");
Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/project1","root",
"angrybird");
String subject= "hpca";
String uno=request.getParameter("no");
Integer a= Integer.parseInt(uno);
String check="select description from hpca order by rand() limit "+a;
Statement stat=con.createStatement();
ResultSet rs=stat.executeQuery(check);
out.println("<p align=center><b>Jaypee University of Information
Technology</b></p>");
out.println("<p align=center><b>HPCA</b></p>");
out.println();
out.println("<table cellpadding=12>");
while(rs.next())
{
String n = rs.getString(1);
out.println("<tr><td>"+(i++)+"</td><td>"+n+"</td></tr>");
}
out.println("</table>");
out.println("</body></html>");
}
catch(ClassNotFoundException e)
{
}
catch(SQLException e)
{
}
finally {
out.close();
}
}
// <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the +
sign on the left to edit the code.">
/**
* Handles the HTTP <code>GET</code> method.
* @param request servlet request
* @param response servlet response
* @throws ServletException if a servlet-specific error occurs
* @throws IOException if an I/O error occurs
*/
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
processRequest(request, response);
}
/**
* Handles the HTTP <code>POST</code> method.
* @param request servlet request
* @param response servlet response
* @throws ServletException if a servlet-specific error occurs
* @throws IOException if an I/O error occurs
*/
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
processRequest(request, response);
}
/**
* Returns a short description of the servlet.
* @return a String containing servlet description
*/
```

```
@Override
public String getServletInfo() {
return "Short description";
}// </editor-fold>}
```

C) Schema Design

Here we explain the detailed formation of our project's database.

5.1.4 Schema Description

The schema **project1** is designed comprising of various tables to accommodate the user and related data. Detailed description is given as under:

- Admin
  Primary Key: emploeeid
  Foreign Key: FK_admin_1

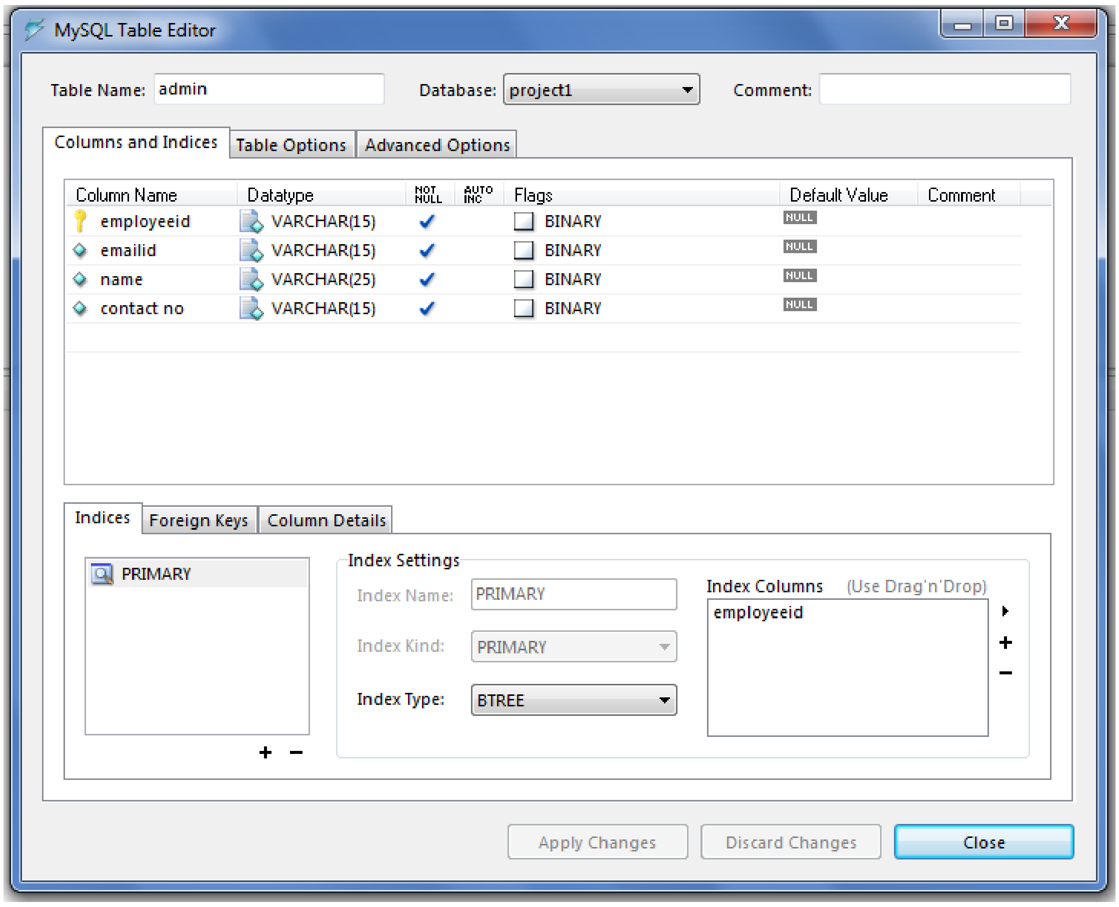


Table Name: admin Database: project1 Comment:

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
employeeid	VARCHAR(15)	✓		BINARY	NULL	
emailid	VARCHAR(15)	✓		BINARY	NULL	
name	VARCHAR(25)	✓		BINARY	NULL	
contact no	VARCHAR(15)	✓		BINARY	NULL	

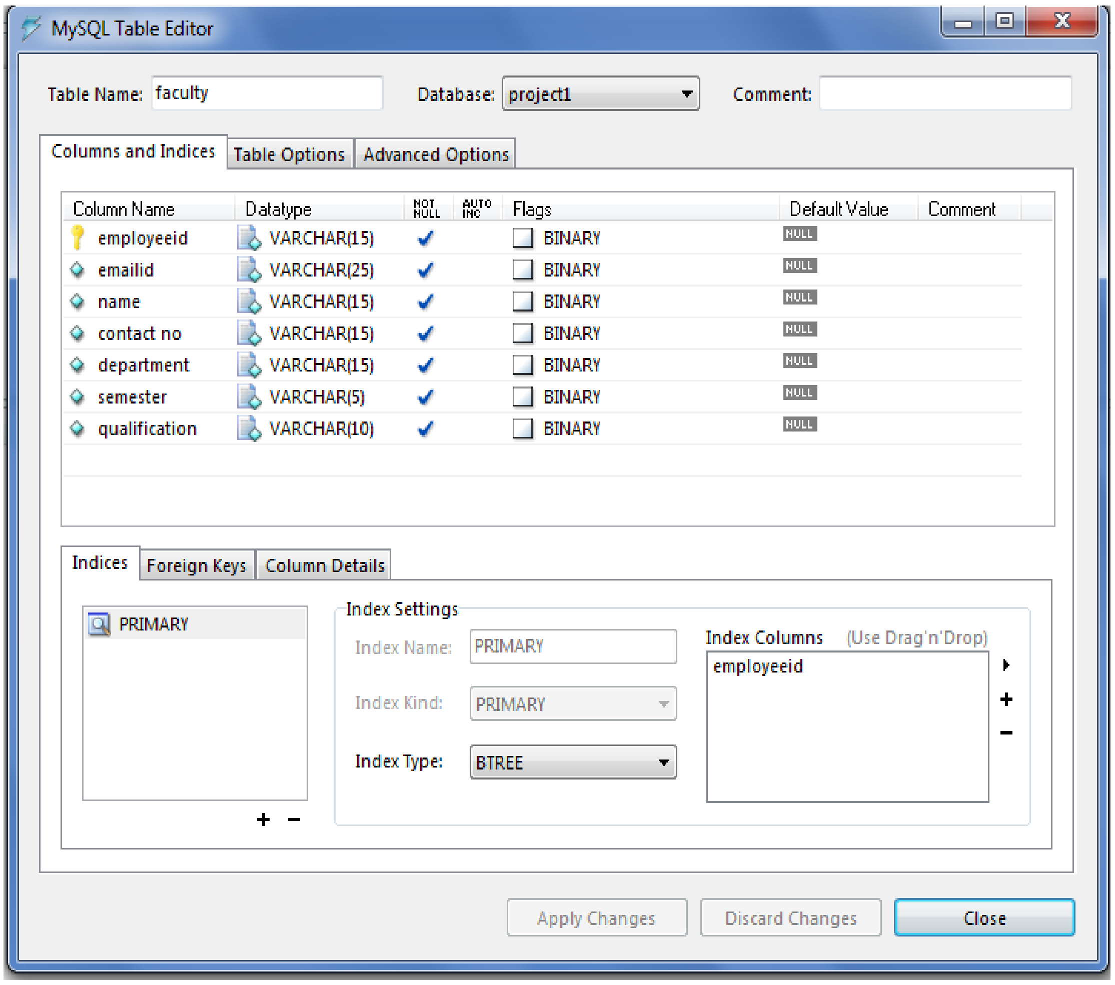
Indices: PRIMARY

Index Settings — Index Name: PRIMARY; Index Kind: PRIMARY; Index Type: BTREE

Index Columns (Use Drag'n'Drop): employeeid

FIGURE 12: TABLE OF ADMIN DETAILS

- Faculty
  Primary Key: employeeid
  Foreign Key: FK_faculty_1



MySQL Table Editor

Table Name: faculty   Database: project1   Comment:

Columns and Indices | Table Options | Advanced Options

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
employeeid	VARCHAR(15)	✓		☐ BINARY	NULL	
emailid	VARCHAR(25)	✓		☐ BINARY	NULL	
name	VARCHAR(15)	✓		☐ BINARY	NULL	
contact no	VARCHAR(15)	✓		☐ BINARY	NULL	
department	VARCHAR(15)	✓		☐ BINARY	NULL	
semester	VARCHAR(5)	✓		☐ BINARY	NULL	
qualification	VARCHAR(10)	✓		☐ BINARY	NULL	

Indices | Foreign Keys | Column Details

PRIMARY

Index Settings
Index Name: PRIMARY
Index Kind: PRIMARY
Index Type: BTREE

Index Columns (Use Drag'n'Drop)
employeeid

FIGURE 13: TABLE OF FACULTY DETAILS

- Student

Primary Key: studentid
Foreign Key: FK_tudent_1

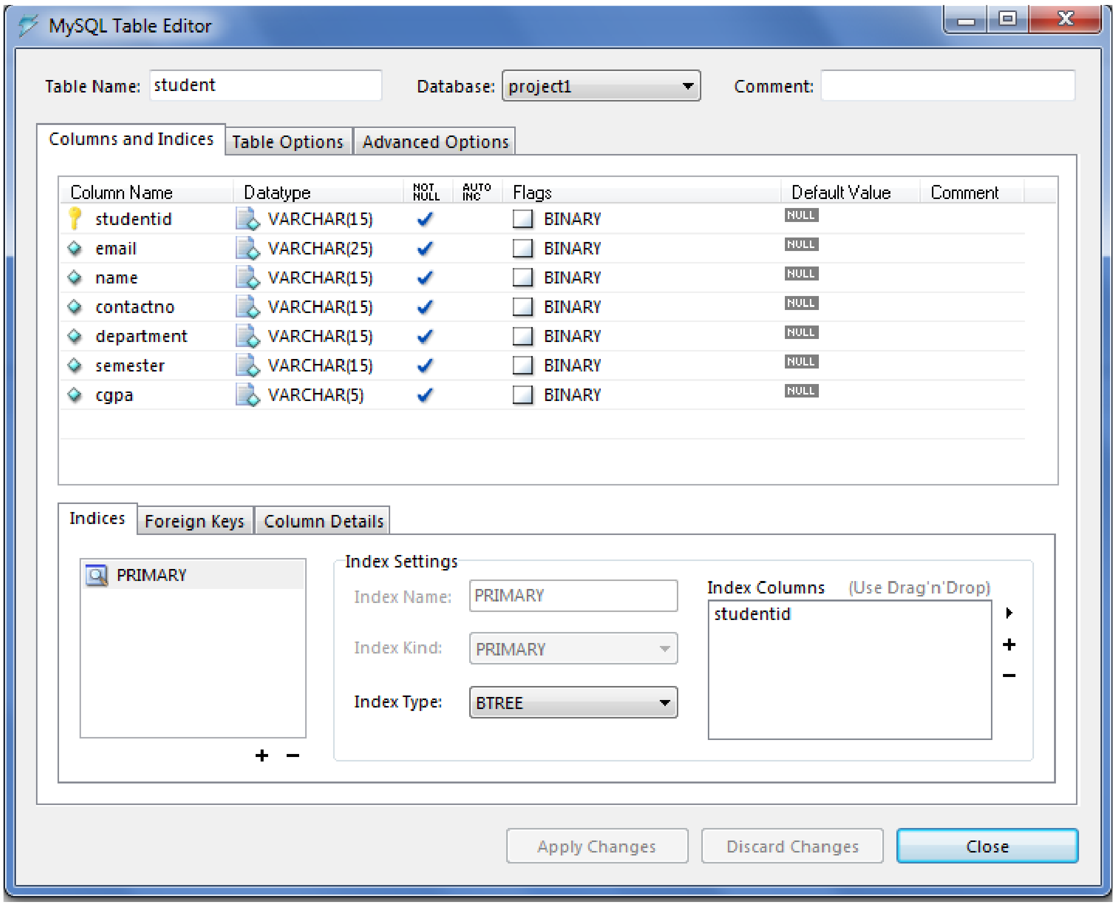


Table Name: student   Database: project1   Comment:

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
studentid	VARCHAR(15)	✓		☐ BINARY	NULL	
email	VARCHAR(25)	✓		☐ BINARY	NULL	
name	VARCHAR(15)	✓		☐ BINARY	NULL	
contactno	VARCHAR(15)	✓		☐ BINARY	NULL	
department	VARCHAR(15)	✓		☐ BINARY	NULL	
semester	VARCHAR(15)	✓		☐ BINARY	NULL	
cgpa	VARCHAR(5)	✓		☐ BINARY	NULL	

PRIMARY

Index Settings
Index Name: PRIMARY
Index Kind: PRIMARY
Index Type: BTREE

Index Columns (Use Drag'n'Drop)
studentid

FIGURE 14: TABLE OF STUDENT DETAILS

- Login
  Primary Key: username

Foreign Key: none

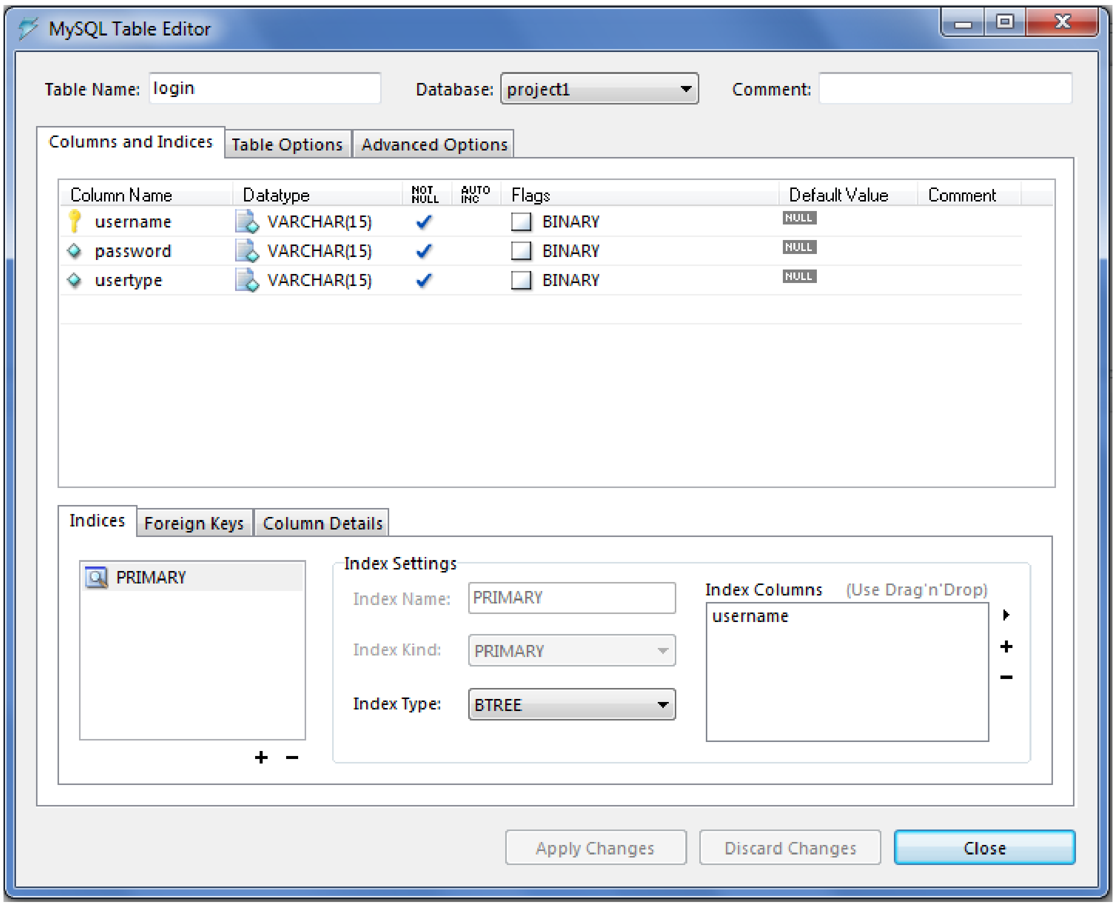

MySQL Table Editor

Table Name: login Database: project1 Comment:

Columns and Indices | Table Options | Advanced Options

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
username	VARCHAR(15)	✓		☐ BINARY	NULL	
password	VARCHAR(15)	✓		☐ BINARY	NULL	
usertype	VARCHAR(15)	✓		☐ BINARY	NULL	

Indices | Foreign Keys | Column Details

PRIMARY

Index Settings

Index Name: PRIMARY

Index Kind: PRIMARY

Index Type: BTREE

Index Columns (Use Drag'n'Drop)

username

Apply Changes | Discard Changes | Close

FIGURE 15: TABLE OF LOGIN DETAILS OF USERS

- Subject data

  Primary Key: subid

Foreign Key: None

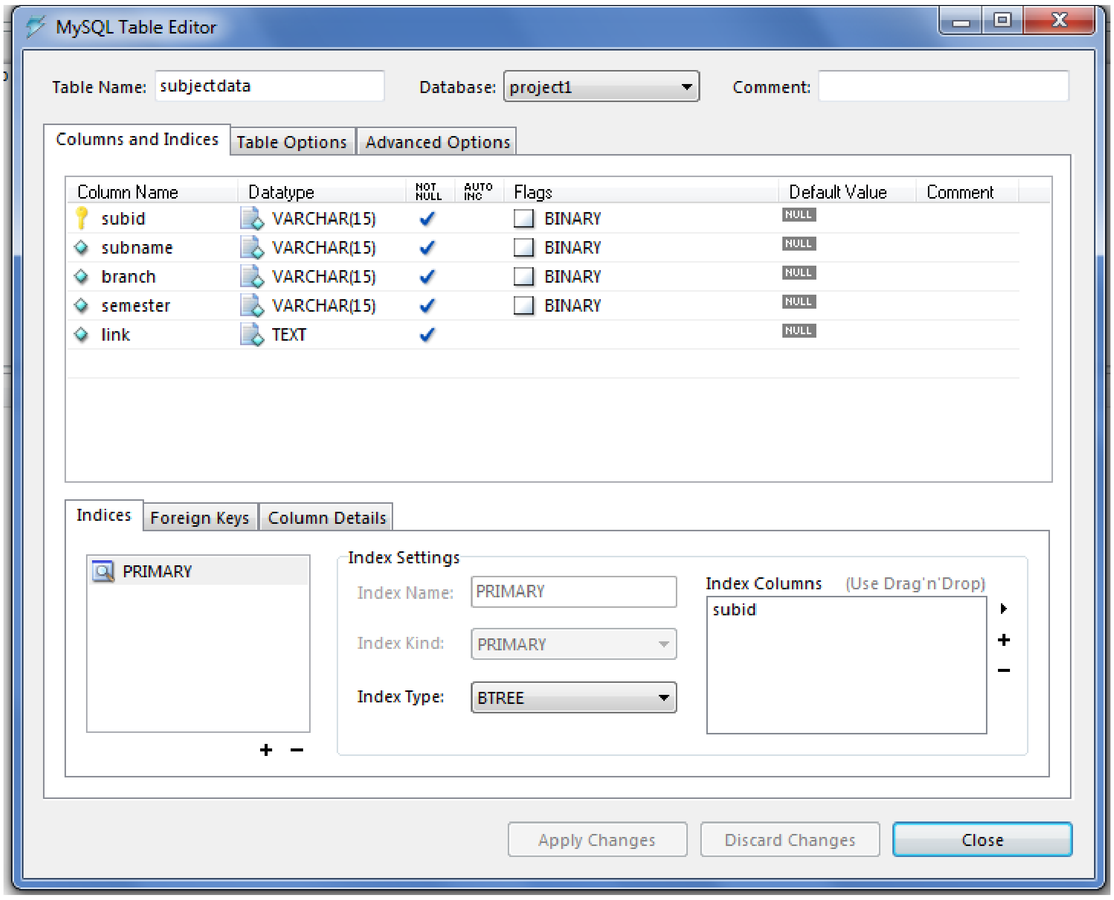

MySQL Table Editor

Table Name: subjectdata Database: project1 Comment:

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
subid	VARCHAR(15)	✓		☐ BINARY	NULL	
subname	VARCHAR(15)	✓		☐ BINARY	NULL	
branch	VARCHAR(15)	✓		☐ BINARY	NULL	
semester	VARCHAR(15)	✓		☐ BINARY	NULL	
link	TEXT	✓			NULL	

PRIMARY

Index Settings

Index Name: PRIMARY

Index Kind: PRIMARY

Index Type: BTREE

Index Columns (Use Drag'n'Drop)

subid

FIGURE 16: TABLE OF SUBJECT DATA

- Subject
  Primary Key: questioned
  Foreign Key: None

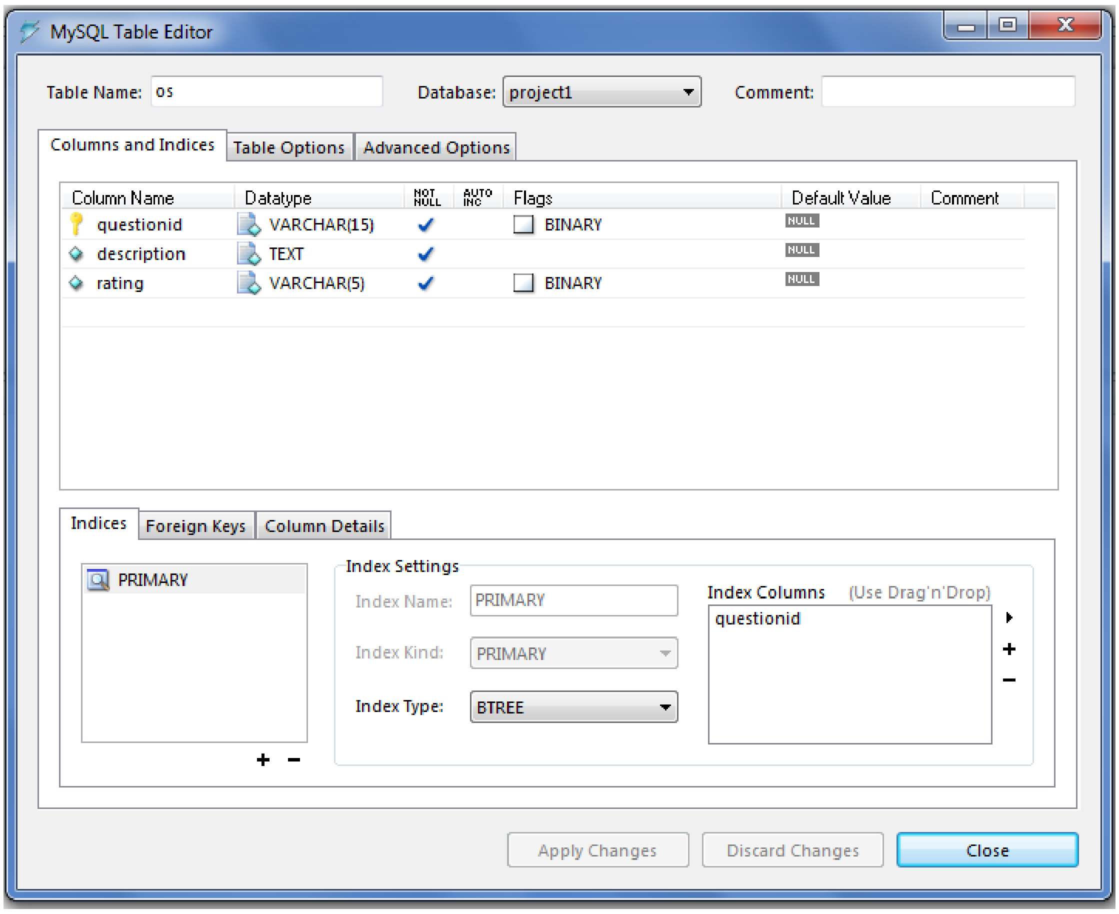

MySQL Table Editor

Table Name: os   Database: project1   Comment:

Columns and Indices | Table Options | Advanced Options

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
questionid	VARCHAR(15)	✓		☐ BINARY	NULL	
description	TEXT	✓			NULL	
rating	VARCHAR(5)	✓		☐ BINARY	NULL	

Indices | Foreign Keys | Column Details

PRIMARY

Index Settings

Index Name: PRIMARY

Index Kind: PRIMARY

Index Type: BTREE

Index Columns (Use Drag'n'Drop): questionid

FIGURE 17: TABLE OF QUESTION RECORD OF PARTICULAR SUBJECT

5.2 SESSION II

A) Random Generation Algorithm with SHUFFLING

Producing a quality graduate is one of the main objective in any educational institution. The higher acceptance of their graduates in work market indicates the quality of the institutions. The quality of graduates produced by any institutions is determined by many factors. One of the factors is the quality of the evaluation system. Evaluation system could exist in many forms. Conventional evaluation system is normally based on the exam system. Before the exam could be given to the student, the instructors or lecturers must prepare the questions according to the topics covered for each of the subject. Preparing exam questions is challenges, tedious and time consuming for the instructors. Usually the instructors keeping their own test bank in some form to help them prepare future exams. Current technologies help the instructors to store the questions in computer databases. The issue arise is how the current technologies would also help the instructors automatically generate the different sets of questions from time to time without concern about repetition and duplication from the pass exam while the exam bank growing.

This paper introduced the usage of shuffling algorithm in Automatic Question Paper Generator to overcome the issue stated. The main role of the shuffling algorithms is to provide randomization technique in AQPG thus different sets of question could be generated without repetition and duplication. Randomization technique is a method that has grown enormously over the past 20 years.

The functions in GQS are embedded with learning outcomes measure that would help lecturers produced quality exam question according to learning outcome objective for each course. The questions are stored in huge Question bank. Lecturers can add new questions, delete the old questions and update the existing question in the Question bank at anytime. Lecturers could also generate different sets of question papers from the same database with just one click by selecting all the requirements needed. This system will prevent duplication of the questions by using the shuffling algorithms. The option to choose shuffling algorithm for randomization is because of simplicity. Apart of shuffling algorithm features are to prevent duplication and repetition of generated sets of question paper. All the questions are picks randomly from the database.

5.2.1 Shuffling Algorithm Implementation

Shuffling algorithms is very suitable and effective ways to implement for randomization of generated question. This algorithms check for duplication and repetition of the randomize question. The behavior of the algorithm is as followed, for a set of N (the total number of question in the database) elements for generating a random permutation of the numbers 1–*N* goes as follows

1) Select the numbers from one to *N* in the database.

2) Pick a random number *k* between one and the number of unstuck numbers remaining (inclusive).

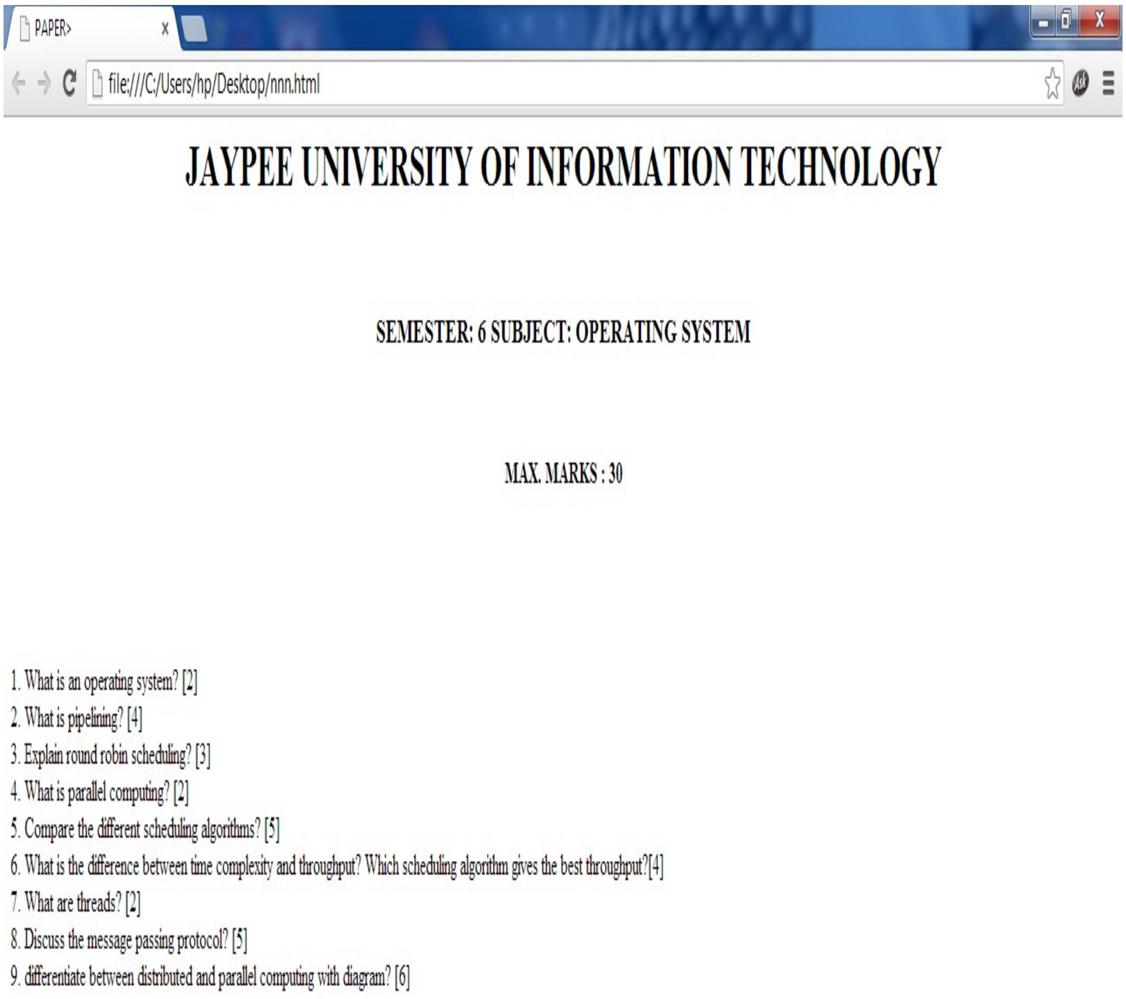
3) Counting from the low end, strike out the *k*th number not yet struck out, and mark it.

4) Repeat from step 2 until all the numbers have been struck out.

5) The sequence of numbers written down in step 3 is now a random permutation of the original numbers.

5.2.2 Final Output

a


PAPER>

file:///C:/Users/hp/Desktop/nnn.html

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY

SEMESTER: 6 SUBJECT: OPERATING SYSTEM

MAX. MARKS : 30

1. What is an operating system? [2]
2. What is pipelining? [4]
3. Explain round robin scheduling? [3]
4. What is parallel computing? [2]
5. Compare the different scheduling algorithms? [5]
6. What is the difference between time complexity and throughput? Which scheduling algorithm gives the best throughput?[4]
7. What are threads? [2]
8. Discuss the message passing protocol? [5]
9. differentiate between distributed and parallel computing with diagram? [6]

FIGURE 18: SNAPSHOT OF FINAL RESULT

CHAPTER – 6
TESTING

6.1 Testing

Testing is the process of exercising the program with the specific intent of finding errors prior to delivery to the end use. We test in order to have an error free system and assure the quality of software. The step to take is to review the specification, design and the code generation of the software before it reaches the customers.

6.2 Types of Testing

6.2.1 White Box Testing

Also known as glass box, structural, clear box and open box testing. A software testing technique whereby, explicit knowledge of internal workings of the item being tested are used to select the test data. Unlike black box testing, white box testing uses specific knowledge of programming code to examine outputs. The test is accurate only if the tester knows what the program is supposed to do. He or she can then see if the program diverges from its goal. White box testing does not account for errors caused by omission, and all visible code must also be readable.

The **advantages** of this type of testing include:

- As the knowledge of internal coding structure is prerequisite, it becomes very easy to find out which type of input/data can help in testing the applications effectively.
- It helps in optimizing the code.
- It helps in removing extra lines of code, which can bring in hidden effects.
- Forces test developer to reason carefully about implementation.

The **disadvantages** of this type of testing include:

- As knowledge of the code and internal structure is prerequisite, a skilled tester is needed to carry out this type of testing, which increases the cost.
- And it is nearly impossible to look in to every bit of code to find out hidden errors, which may create problems, resulting in failure of the application.
- Not looking at the code in run time environment. That is important for a number of reasons. Exploitation of vulnerability is dependent upon all aspects of the platform being targeted and source code is just of those components. The underlying operating system, the backend database being used, third party security tools, dependent libraries etc must all be taken into account when determining exploitability. A source code review is not able to these factors into account.
- Very few white box tests can be done without modifying the program, changing values to force different execution paths, or to generate a full range of inputs to test a particular function.
- Miss cases omitted in the code.

6.2.2 Black Box Testing

It is also known as functional testing. A software testing technique whereby, the internal workings of the items being tested are not known by the tester. For example, in a black box test on software design the tester only knows the inputs and what the expected outcome should be and not how the program arrives at those outputs. The tester does not ever examine the programming code and does not need any further knowledge of program other than its specifications.

The **advantages** of this type of testing include:

- The test is unbiased because the designer and the tester are independent of each other.
- The tester does not need any knowledge of specific programming language.
- The test is done from point of view of user not the designer.
- Test cases can be designed as soon as the specifications are complete.

The **disadvantages** of this type of testing include:

- The test can be redundant if the designer has already run a test case.
- The test cases are difficult to design.
- Testing every possible input stream is unrealistic because it would take an inordinate amount of time, therefore many program paths will go untested.

For a complete software examination both white box and black box testing are required.

6.3 Testing level

Testing can be done on following levels:

- Unit testing tests the minimal software component, or module. Each unit(basic component) of the software is tested to verify that the detailed design for the unit has been correctly implemented. In an object oriented environment this is usually at the class level, and the minimal unit tests include constructors and destructors.
- Integration testing exposes defects in the interfaces and interaction between integrated components. Progressively larger groups of tested software components corresponding to elements of architectural design are integrated and tested until the software works as a system.
- System testing tests a completely integrated system to verify that it meets its requirements.
- System integration testing verifies that a system is integrated to any external or third party systems defined in the system requirements.

6.4_Regression Testing

Regression testing is a commonly used activity whose purpose is to determine whether the modifications made to a software system have introduced new faults. Regression testing is the process of testing changes to computer programs to make sure that older programming still works with the new changes.

Testing and debugging of the system is one of the main steps in the process of any project development.

For making the AQPG user friendly and removing the chance of any error/flaws, we tested our AQPG with various techniques.

-Testing while development:

- In the process of development of any question paper generator the main emphasis was laid on the fact that it is time saving and it should be secure enough to be used by both faculty and student. So we checked our AQPG for any such vulnerability and implemented various methods login time checks and admin authentication etc.

-Testing after implementation:

- After making AQPG, we generated various papers to check if there is any redundancy in consecutively generated papers and the paper is being generated for all possible values entered in the form provided.
- We tried maximum possible cases to break it but it always came out to be true.

-**Testing in future:**

- After implementing AQPG, we intent to provide feedback, suggestion and query forms to users who will be the actual user of AQPG because their response is what mainly matters.

CHAPTER 7
CONCLUSION

This project has been interesting right from the beginning. The project required the use of research to find information regarding the already proposed question paper generators along with the same, looking at various options available for making automatic question paper generator more efficient and increasing their user-friendliness was a major task.

The initiation of the project was aimed at getting to know what all things an automatic question paper generator can do that was not possible by traditional methods and also to know what its limitations are. Thus we successfully incorporated major aspects of the same and designed a simple **Automatic Question Paper Generator.**

After the implementation of the simple version of automatic question paper generator, we modified it to set desired difficulty level of paper and solved security related issues. Next we raised the bar higher and targeted at removing the redundancy issues so that minimum number of repetition can take place between consecutive papers.

We started with making user friendly interfaces, establishing database connectivity and finally moved on to make a working model of AQPG for our own university.

Hopefully AQPG will be a great success and will be of use not only to our university but also to other educational institutions. Further, this portal which we have designed so far will be available to use at our central server which when published can be used by users/institutions to make use of its facilities.

The project has been a great success till date as we were able to implement all those proposed modules which were different from already present question paper generators in the market.

7.1 Contribution of the Project

The contribution of the project so far has been great; both to our learning and to the educational sector. The project made us learn how to provide signup and login for portals, access controls and handle databases.

The project has made fairly good contribution to the field of education and web where new domains were introduced and implemented by us. Also the main benefiters of AQPG i.e. the faculty can now save a lot of time and effort by using our portal without worrying for confidentiality and security issues. Our user-friendly and

interactive interface allows for easy access and handling for a layman. A novice user need not to be aware of the intricate details of programming to use the portal.
To sum up, the AQPG which we have designed is user-friendly, efficient and innovative and will prove to be of great use to the industry.

7.2 Future of AQPG

We have now reached the stage where almost everything in this world is computerized and everyone wants to get their work done efficiently and in minimum time. More and more researchers have concentrated their efforts in spreading technologies to maximum sectors in the society.
So, what can we expect from AQPG in the future? Is AQPG even worth implementing if it is so easily worked around? How can AQPG fight back? Will it be slowly replaced with other technologies and methods?
Certainly, the future of AQPG is not short spanned. Many new domains or modules can be integrated within AQPG-student teacher interaction forum, biometric, online examinations being a few. Thus we see AQPG ever growing with time and reaching every corner of the society.

APPENDIX

Facultyreg.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */


import java.sql.*;
import java.io.*;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 *
 * @author dell
 */
public class facultyreg extends HttpServlet {

   /**
              * Processes requests for both HTTP <code>GET</code> and
<code>POST</code> methods.
    * @param request servlet request
    * @param response servlet response
    * @throws ServletException if a servlet-specific error occurs
    * @throws IOException if an I/O error occurs
    */
     protected void processRequest(HttpServletRequest request, HttpServletResponse
response)
   throws ServletException, IOException
   {
      PrintWriter out = response.getWriter();

      response.setContentType("text/html");

      try
      {
         String uname=request.getParameter("username");
          String upass=request.getParameter("password");
          String utype="Faculty";

         Class.forName("com.mysql.jdbc.Driver");
                                                                    Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/project1","root","an
grybird");

                              String check="select employeeid from faculty where
employeeid='"+uname+"'";
         Statement stat=con.createStatement();
```

```
            ResultSet rs=stat.executeQuery(check);
            if(rs.next())
            {
               response.sendRedirect("facultyreg1.html");
            }


            String login="insert into login values(?,?,?);";
            String faculty="insert into faculty values(?,?,?,?,?,?,?);";
            PreparedStatement pst=con.prepareStatement(login);
            PreparedStatement pst1=con.prepareStatement(faculty);


            pst.setString(1,uname);
            pst.setString(2,upass);
            pst.setString(3,utype);
            pst.executeUpdate();

            String umail=request.getParameter("email");
            String ufull=request.getParameter("name");
            String uphone=request.getParameter("phone");
            String qual=request.getParameter("qual");
            pst1.setString(1,uname);
            pst1.setString(2,umail);
            pst1.setString(3,ufull);
            pst1.setString(4,uphone);
            pst1.setString(5,qual);
            pst1.setString(6,qual);
            pst1.setString(7,qual);
            pst1.executeUpdate();
            response.sendRedirect("index.html");

        }
        catch(ClassNotFoundException e)
        {
            e.printStackTrace();
        }
        catch(SQLException e)
        {
            e.printStackTrace();
        }
        finally
        {
            out.close();

        }
    }

    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">
    /**
     * Handles the HTTP <code>GET</code> method.
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Handles the HTTP <code>POST</code> method.
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Returns a short description of the servlet.
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>

}
```

Uploadsoa.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */


import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.sql.*;

/**
 *
 * @author dell
 */
public class uploadsoa extends HttpServlet {

   /**
     * Processes requests for both HTTP <code>GET</code> and
<code>POST</code> methods.
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    protected void processRequest(HttpServletRequest request, HttpServletResponse
response)
   throws ServletException, IOException {
      response.setContentType("text/html;charset=UTF-8");
      PrintWriter out = response.getWriter();
      try {

       Class.forName("com.mysql.jdbc.Driver");
                                                                    Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/project1","root","an
grybird");


         String uid=request.getParameter("qid");
         String urate=request.getParameter("rating");
         String uques=request.getParameter("question");
         String check="select questionid from soa where questionid='"+uid+"'";
         Statement stat=con.createStatement();
         ResultSet rs=stat.executeQuery(check);
         if(rs.next())
         {
            response.sendRedirect("SOA.html");
         }
```

```
            String upload="insert into soa values(?,?,?);";

            PreparedStatement pst=con.prepareStatement(upload);

            pst.setString(1,uid);
            pst.setString(2,uques);
            pst.setString(3,urate);
            pst.executeUpdate();

            response.sendRedirect("SOA.html");

        }
        catch(ClassNotFoundException e)
        {
            e.printStackTrace();
        }
        catch(SQLException e)
        {
            e.printStackTrace();
        }
            finally {
            out.close();
        }
    }

    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the +
sign on the left to edit the code.">
    /**
     * Handles the HTTP <code>GET</code> method.
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Handles the HTTP <code>POST</code> method.
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Returns a short description of the servlet.
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>

}
```

Uploadpd.java

```
/*
```

```
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */


import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.sql.*;

/**
 *
 * @author dell
 */
public class uploadpd extends HttpServlet {

   /**
          * Processes requests for both HTTP <code>GET</code> and
<code>POST</code> methods.
    * @param request servlet request
    * @param response servlet response
    * @throws ServletException if a servlet-specific error occurs
    * @throws IOException if an I/O error occurs
    */
    protected void processRequest(HttpServletRequest request, HttpServletResponse
response)
   throws ServletException, IOException {
      response.setContentType("text/html;charset=UTF-8");
      PrintWriter out = response.getWriter();
      try {

       Class.forName("com.mysql.jdbc.Driver");
                                                                  Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/project1","root","an
grybird");


         String uid=request.getParameter("qid");
         String urate=request.getParameter("rating");
         String uques=request.getParameter("question");
         String check="select questionid from pd where questionid='"+uid+"'";
         Statement stat=con.createStatement();
         ResultSet rs=stat.executeQuery(check);
         if(rs.next())
         {
            response.sendRedirect("PD.html");
         }
```

```
        String upload="insert into pd values(?,?,?);";

        PreparedStatement pst=con.prepareStatement(upload);

        pst.setString(1,uid);
        pst.setString(2,uques);
        pst.setString(3,urate);
        pst.executeUpdate();

        response.sendRedirect("facultywelcomepage.html");

      }
      catch(ClassNotFoundException e)
      {
        e.printStackTrace();
      }
      catch(SQLException e)
      {
        e.printStackTrace();
      }
        finally {
        out.close();
      }
    }

    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the +
sign on the left to edit the code.">
    /**
     * Handles the HTTP <code>GET</code> method.
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
      processRequest(request, response);
    }

    /**
     * Handles the HTTP <code>POST</code> method.
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Returns a short description of the servlet.
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>
}
```

Uploadhpca.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */


import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.sql.*;

/**
 *
 * @author dell
 */
public class uploadhpca extends HttpServlet {

   /**
             * Processes requests for both HTTP <code>GET</code> and
<code>POST</code> methods.
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
      protected void processRequest(HttpServletRequest request, HttpServletResponse
response)
   throws ServletException, IOException {
       response.setContentType("text/html;charset=UTF-8");
       PrintWriter out = response.getWriter();
       try {

        Class.forName("com.mysql.jdbc.Driver");
                                                                          Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/project1","root","an
grybird");


          String uid=request.getParameter("qid");
          String urate=request.getParameter("rating");
          String uques=request.getParameter("question");
          String check="select questionid from hpca where questionid='"+uid+"'";
          Statement stat=con.createStatement();
          ResultSet rs=stat.executeQuery(check);
          if(rs.next())
          {
             response.sendRedirect("HPCA.html");
          }
```

```
         String upload="insert into hpca values(?,?,?);";

         PreparedStatement pst=con.prepareStatement(upload);

         pst.setString(1,uid);
         pst.setString(2,uques);
         pst.setString(3,urate);
         pst.executeUpdate();

         response.sendRedirect("HPCA.html");

      }
      catch(ClassNotFoundException e)
      {
         e.printStackTrace();
      }
      catch(SQLException e)
      {
         e.printStackTrace();
      }
         finally {
         out.close();
      }
   }

   // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the +
sign on the left to edit the code.">
   /**
    * Handles the HTTP <code>GET</code> method.
    * @param request servlet request
    * @param response servlet response
    * @throws ServletException if a servlet-specific error occurs
    * @throws IOException if an I/O error occurs
    */
   @Override
   protected void doGet(HttpServletRequest request, HttpServletResponse response)
   throws ServletException, IOException {
      processRequest(request, response);
   }

   /**
    * Handles the HTTP <code>POST</code> method.
    * @param request servlet request
    * @param response servlet response
    * @throws ServletException if a servlet-specific error occurs
    * @throws IOException if an I/O error occurs
    */
   @Override
```

```
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Returns a short description of the servlet.
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>

}
```

Uploadcs.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */


import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.sql.*;

/**
 *
 * @author dell
 */
public class uploadcs extends HttpServlet {

   /**
            * Processes requests for both HTTP <code>GET</code> and
<code>POST</code> methods.
    * @param request servlet request
    * @param response servlet response
    * @throws ServletException if a servlet-specific error occurs
    * @throws IOException if an I/O error occurs
    */
     protected void processRequest(HttpServletRequest request, HttpServletResponse
response)
   throws ServletException, IOException {
       response.setContentType("text/html;charset=UTF-8");
       PrintWriter out = response.getWriter();
       try {

        Class.forName("com.mysql.jdbc.Driver");
                                                                        Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/project1","root","an
grybird");


           String uid=request.getParameter("qid");
           String urate=request.getParameter("rating");
           String uques=request.getParameter("question");
           String check="select questionid from cs where questionid='"+uid+"'";
           Statement stat=con.createStatement();
           ResultSet rs=stat.executeQuery(check);
           if(rs.next())
           {
              response.sendRedirect("CS.html");
           }
```

```
            String upload="insert into cs values(?,?,?);";

            PreparedStatement pst=con.prepareStatement(upload);

            pst.setString(1,uid);
            pst.setString(2,uques);
            pst.setString(3,urate);
            pst.executeUpdate();

            response.sendRedirect("facultywelcomepage.html");

        }
        catch(ClassNotFoundException e)
        {
            e.printStackTrace();
        }
        catch(SQLException e)
        {
            e.printStackTrace();
        }
            finally {
            out.close();
        }
    }

    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the +
sign on the left to edit the code.">
    /**
     * Handles the HTTP <code>GET</code> method.
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Handles the HTTP <code>POST</code> method.
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Returns a short description of the servlet.
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>

}
```

Upload.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */


import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.sql.*;

/**
 *
 * @author dell
 */
public class upload extends HttpServlet {

   /**
            * Processes requests for both HTTP <code>GET</code> and
<code>POST</code> methods.
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
      protected void processRequest(HttpServletRequest request, HttpServletResponse
response)
   throws ServletException, IOException {
       response.setContentType("text/html;charset=UTF-8");
       PrintWriter out = response.getWriter();
       try {

        Class.forName("com.mysql.jdbc.Driver");
                                                                     Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/project1","root","an
grybird");


          String uid=request.getParameter("qid");
          String urate=request.getParameter("rating");
          String uques=request.getParameter("question");
          String check="select questionid from sub1ques where questionid='"+uid+"'";
          Statement stat=con.createStatement();
          ResultSet rs=stat.executeQuery(check);
          if(rs.next())
          {
             response.sendRedirect("upload.html");
          }
          String upload="insert into sub1ques values(?,?,?);";

          PreparedStatement pst=con.prepareStatement(upload);

          pst.setString(1,uid);
          pst.setString(2,uques);
          pst.setString(3,urate);
          pst.executeUpdate();

          response.sendRedirect("facultywelcomepage.html");

        }
        catch(ClassNotFoundException e)
        {
          e.printStackTrace();
        }
        catch(SQLException e)
        {
          e.printStackTrace();
        }
          finally {
          out.close();
        }
      }

      // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the +
sign on the left to edit the code.">
      /**
       * Handles the HTTP <code>GET</code> method.
       * @param request servlet request
       * @param response servlet response
       * @throws ServletException if a servlet-specific error occurs
       * @throws IOException if an I/O error occurs
       */
      @Override
      protected void doGet(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {
        processRequest(request, response);
      }

      /**
       * Handles the HTTP <code>POST</code> method.
       * @param request servlet request
       * @param response servlet response
       * @throws ServletException if a servlet-specific error occurs
       * @throws IOException if an I/O error occurs
       */
      @Override
```

```
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Returns a short description of the servlet.
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>

}
```

Studentreg.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */


import java.sql.*;
import java.io.*;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 *
 * @author dell
 */
public class studentreg extends HttpServlet {

   /**
              * Processes requests for both HTTP <code>GET</code> and
<code>POST</code> methods.
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
     protected void processRequest(HttpServletRequest request, HttpServletResponse
response)
   throws ServletException, IOException
   {
      PrintWriter out = response.getWriter();

      response.setContentType("text/html");

      try
      {
         Class.forName("com.mysql.jdbc.Driver");
                                                                         Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/project1","root","an
grybird");

         String uname=request.getParameter("username");
         String upass=request.getParameter("password");
         String utype="Student";

         String check="select studentid from student where studentid='"+uname+"'";
         Statement stat=con.createStatement();
         ResultSet rs=stat.executeQuery(check);
         if(rs.next())
         {
            response.sendRedirect("StudentReg.html");
         }


         String login="insert into login values(?,?,?);";
         String student="insert into student values(?,?,?,?,?,?,?);";
         PreparedStatement pst=con.prepareStatement(login);
         PreparedStatement pst1=con.prepareStatement(student);
         pst.setString(1,uname);
         pst.setString(2,upass);
         pst.setString(3,utype);
         pst.executeUpdate();

         String umail=request.getParameter("email");
         String ufull=request.getParameter("name");
         String uphone=request.getParameter("phone");
         String ucg=request.getParameter("cgpa");
         pst1.setString(1,uname);
         pst1.setString(2,umail);
         pst1.setString(3,ufull);
         pst1.setString(4,uphone);
         pst1.setString(5,ucg);
         pst1.setString(6,ucg);
         pst1.setString(7,ucg);
         pst1.executeUpdate();
         response.sendRedirect("index.html");

      }
      catch(ClassNotFoundException e)
      {
         e.printStackTrace();
      }
      catch(SQLException e)
      {
         e.printStackTrace();
      }
      finally
      {
         out.close();

      }
   }

   // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the +
sign on the left to edit the code.">
   /**
     * Handles the HTTP <code>GET</code> method.
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Handles the HTTP <code>POST</code> method.
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Returns a short description of the servlet.
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>

}
```

Login.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */


import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.sql.*;

/**
 *
 * @author dell
 */
public class login extends HttpServlet {

   /**
              * Processes requests for both HTTP <code>GET</code> and
<code>POST</code> methods.
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
      protected void processRequest(HttpServletRequest request, HttpServletResponse
response)
   throws ServletException, IOException {
       response.setContentType("text/html;charset=UTF-8");
       PrintWriter out = response.getWriter();
       try
       {
          Class.forName("com.mysql.jdbc.Driver");
                                                                        Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/project1","root","an
grybird");
          Statement st= con.createStatement();
          String username=request.getParameter("username");
          String password= request.getParameter("password");
              String login="select usertype from login where username='"+username+"'
"+"and password='"+password+"'";
          ResultSet rs=st.executeQuery(login);
          if(rs.next())
          {
             String usertype=rs.getString(1);
             if(usertype.equals("Student"))
             {
                response.sendRedirect("studentwelcomepage.html");
             }
```

```
            else
            {
               response.sendRedirect("facultywelcomepage.html");
            }

         }
         else
         {
            response.sendRedirect("login1.html");
         }

      }
      catch(ClassNotFoundException e)
      {

      }
      catch(SQLException e)
      {
      }
      finally {
         out.close();
      }
   }

   // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the +
sign on the left to edit the code.">
   /**
    * Handles the HTTP <code>GET</code> method.
    * @param request servlet request
    * @param response servlet response
    * @throws ServletException if a servlet-specific error occurs
    * @throws IOException if an I/O error occurs
    */
   @Override
   protected void doGet(HttpServletRequest request, HttpServletResponse response)
   throws ServletException, IOException {
      processRequest(request, response);
   }

   /**
    * Handles the HTTP <code>POST</code> method.
    * @param request servlet request
    * @param response servlet response
    * @throws ServletException if a servlet-specific error occurs
    * @throws IOException if an I/O error occurs
    */
   @Override
   protected void doPost(HttpServletRequest request, HttpServletResponse response)
   throws ServletException, IOException {
      processRequest(request, response);
   }

   /**
    * Returns a short description of the servlet.
    * @return a String containing servlet description
    */
   @Override
   public String getServletInfo() {
      return "Short description";
   }// </editor-fold>

}
```

Gensoa.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */


import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.sql.*;
/**
 *
 * @author dell
 */
public class gensoa extends HttpServlet {

   /**
            * Processes requests for both HTTP <code>GET</code> and
<code>POST</code> methods.
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
     protected void processRequest(HttpServletRequest request, HttpServletResponse
response)
   throws ServletException, IOException {
       response.setContentType("text/html;charset=UTF-8");
       PrintWriter out = response.getWriter();
       out.println("<html><body>");
       try {
          int i=1;
        Class.forName("com.mysql.jdbc.Driver");
                                                                                  Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/project1","root","an
grybird");
        String subject= "SOA";
              String uno=request.getParameter("no");
        Integer a= Integer.parseInt(uno);
        String check="select description from soa order by rand() limit "+a;
          Statement stat=con.createStatement();
          ResultSet rs=stat.executeQuery(check);
                   out.println("<p align=center><b>Jaypee University of Information
Technology</b></p>");
        out.println("<p align=center><b>SOA</b></p>");
        out.println();
        out.println("<table cellpadding=12>");
         while(rs.next())
          {
           String n = rs.getString(1);
           out.println("<tr><td>"+(i++)+"</td><td>"+n+"</td></tr>");
          }
        out.println("</table>");
        out.println("</body></html>");





       }
       catch(ClassNotFoundException e)
       {
       }
       catch(SQLException e)
       {
       }
       finally {
          out.close();
       }
   }

   // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the +
sign on the left to edit the code.">
   /**
     * Handles the HTTP <code>GET</code> method.
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
   @Override
   protected void doGet(HttpServletRequest request, HttpServletResponse response)
   throws ServletException, IOException {
       processRequest(request, response);
   }

   /**
     * Handles the HTTP <code>POST</code> method.
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
   @Override
   protected void doPost(HttpServletRequest request, HttpServletResponse response)
   throws ServletException, IOException {
       processRequest(request, response);
   }
   /**
     * Returns a short description of the servlet.
     * @return a String containing servlet description
     */
   @Override
   public String getServletInfo() {
       return "Short description";
   }// </editor-fold>}
```

Genos.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */


import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.sql.*;
/**
 *
 * @author dell
 */
public class genos extends HttpServlet {

   /**
              * Processes requests for both HTTP <code>GET</code> and
<code>POST</code> methods.
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
     protected void processRequest(HttpServletRequest request, HttpServletResponse
response)
   throws ServletException, IOException {
      response.setContentType("text/html;charset=UTF-8");
      PrintWriter out = response.getWriter();
      out.println("<html><body>");
      try {
         int i=1;
       Class.forName("com.mysql.jdbc.Driver");
                                                                          Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/project1","root","an
grybird");
       String subject= "os";
             String uno=request.getParameter("no");
       Integer a= Integer.parseInt(uno);
       String check="select description from os order by rand() limit "+a;
         Statement stat=con.createStatement();
         ResultSet rs=stat.executeQuery(check);
                 out.println("<p align=center><b>Jaypee University of Information
Technology</b></p>");
       out.println("<p align=center><b>OS</b></p>");
       out.println();
       out.println("<table cellpadding=12>");
        while(rs.next())
         {
          String n = rs.getString(1);
          out.println("<tr><td>"+(i++)+"</td><td>"+n+"</td></tr>");
         }
       out.println("</table>");
       out.println("</body></html>");




      }
      catch(ClassNotFoundException e)
      {
      }
      catch(SQLException e)
      {
      }
      finally {
         out.close();
      }
   }

   // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the +
sign on the left to edit the code.">
   /**
     * Handles the HTTP <code>GET</code> method.
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
   @Override
   protected void doGet(HttpServletRequest request, HttpServletResponse response)
   throws ServletException, IOException {
      processRequest(request, response);
   }

   /**
     * Handles the HTTP <code>POST</code> method.
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
   @Override
   protected void doPost(HttpServletRequest request, HttpServletResponse response)
   throws ServletException, IOException {
      processRequest(request, response);
   }
    /**
     * Returns a short description of the servlet.
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>}
```

Genhpca.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */


import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.sql.*;
/**
 *
 * @author dell
 */
public class genhpca extends HttpServlet {

   /**
             * Processes requests for both HTTP <code>GET</code> and
<code>POST</code> methods.
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
     protected void processRequest(HttpServletRequest request, HttpServletResponse
response)
   throws ServletException, IOException {
       response.setContentType("text/html;charset=UTF-8");
       PrintWriter out = response.getWriter();
       out.println("<html><body>");
       try {
          int i=1;
        Class.forName("com.mysql.jdbc.Driver");
                                                                              Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/project1","root","an
grybird");
        String subject= "hpca";
              String uno=request.getParameter("no");
        Integer a= Integer.parseInt(uno);
        String check="select description from hpca order by rand() limit "+a;
          Statement stat=con.createStatement();
          ResultSet rs=stat.executeQuery(check);
                   out.println("<p align=center><b>Jaypee University of Information
Technology</b></p>");
        out.println("<p align=center><b>HPCA</b></p>");
        out.println();
        out.println("<table cellpadding=12>");
         while(rs.next())
          {
           String n = rs.getString(1);
           out.println("<tr><td>"+(i++)+"</td><td>"+n+"</td></tr>");
          }
        out.println("</table>");
        out.println("</body></html>");




       }
       catch(ClassNotFoundException e)
       {
       }
       catch(SQLException e)
       {
       }
       finally {
          out.close();
       }
   }

   // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the +
sign on the left to edit the code.">
   /**
     * Handles the HTTP <code>GET</code> method.
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
   @Override
   protected void doGet(HttpServletRequest request, HttpServletResponse response)
   throws ServletException, IOException {
       processRequest(request, response);
   }

   /**
     * Handles the HTTP <code>POST</code> method.
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
   @Override
   protected void doPost(HttpServletRequest request, HttpServletResponse response)
   throws ServletException, IOException {
       processRequest(request, response);
   }
   /**
     * Returns a short description of the servlet.
     * @return a String containing servlet description
     */
   @Override
   public String getServletInfo() {
       return "Short description";
   }// </editor-fold>}
```

Generate1.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */


import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.sql.*;
/**
 *
 * @author dell
 */
public class generate1 extends HttpServlet {

   /**
            * Processes requests for both HTTP <code>GET</code> and
<code>POST</code> methods.
    * @param request servlet request
    * @param response servlet response
    * @throws ServletException if a servlet-specific error occurs
    * @throws IOException if an I/O error occurs
    */
     protected void processRequest(HttpServletRequest request, HttpServletResponse
response)
   throws ServletException, IOException {
       response.setContentType("text/html;charset=UTF-8");
       PrintWriter out = response.getWriter();
       try {
           int i=1;
        Class.forName("com.mysql.jdbc.Driver");
                                                                    Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/project1","root","an
grybird");
        String check="select description from sub1ques order by rand() limit "+10;
          Statement stat=con.createStatement();
          ResultSet rs=stat.executeQuery(check);
                  out.println("<p align=center><b>Jaypee University of Information
Technology</b></p>");
        out.println("<p align=center><b>B.Tech VII Semester</b></p>");
        out.println("<p align=center><b>Operating Systems Quiz</b></p>");
        out.println();
        out.println("<table cellpadding=12>");

         while(rs.next())
          {
             String n = rs.getString(1);
             out.println("<tr><td>"+(i++)+"</td><td>"+n+"</td></tr>");
          }
        out.println("</table>");
        out.println("</body></html>");





       }
       catch(ClassNotFoundException e)
       {
       }
       catch(SQLException e)
       {
       }
        finally {
          out.close();
       }
   }

   // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the +
sign on the left to edit the code.">
   /**
    * Handles the HTTP <code>GET</code> method.
    * @param request servlet request
    * @param response servlet response
    * @throws ServletException if a servlet-specific error occurs
    * @throws IOException if an I/O error occurs
    */
   @Override
   protected void doGet(HttpServletRequest request, HttpServletResponse response)
   throws ServletException, IOException {
       processRequest(request, response);
   }

   /**
    * Handles the HTTP <code>POST</code> method.
    * @param request servlet request
    * @param response servlet response
    * @throws ServletException if a servlet-specific error occurs
    * @throws IOException if an I/O error occurs
    */
   @Override
   protected void doPost(HttpServletRequest request, HttpServletResponse response)
   throws ServletException, IOException {
       processRequest(request, response);
   }

   /**
    * Returns a short description of the servlet.
    * @return a String containing servlet description
    */
   @Override
   public String getServletInfo() {
       return "Short description";
   }// </editor-fold>

}
```

Generate.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */


import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.sql.*;
/**
 *
 * @author dell
 */
public class generate extends HttpServlet {

   /**
         * Processes requests for both HTTP <code>GET</code> and
<code>POST</code> methods.
    * @param request servlet request
    * @param response servlet response
    * @throws ServletException if a servlet-specific error occurs
    * @throws IOException if an I/O error occurs
    */
     protected void processRequest(HttpServletRequest request, HttpServletResponse
response)
   throws ServletException, IOException {
      response.setContentType("text/html;charset=UTF-8");
      PrintWriter out = response.getWriter();
      out.println("<html><body>");
      try {
         int i=1;
       Class.forName("com.mysql.jdbc.Driver");
                                                                    Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/project1","root","an
grybird");
       String subject=request.getParameter("subject");
       String uno=request.getParameter("no");
       Integer a= Integer.parseInt(uno);
       String check="select description from mocktable order by rand() limit "+a;
         Statement stat=con.createStatement();
         ResultSet rs=stat.executeQuery(check);
                 out.println("<p align=center><b>Jaypee University of Information
Technology</b></p>");
       out.println("<p align=center><b>B.Tech VII Semester</b></p>");
       out.println("<p align=center><b>"+subject);
       out.println("</b></p>");
       out.println();
       out.println("<table cellpadding=12>");
      while(rs.next())
        {
           String n = rs.getString(1);
           out.println("<tr><td>"+(i++)+"</td><td>"+n+"</td></tr>");
        }
     out.println("</table>");
     out.println("</body></html>");




    }
    catch(ClassNotFoundException e)
    {
    }
    catch(SQLException e)
    {
    }
    finally {
       out.close();
    }
}

// <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the +
sign on the left to edit the code.">
/**
 * Handles the HTTP <code>GET</code> method.
 * @param request servlet request
 * @param response servlet response
 * @throws ServletException if a servlet-specific error occurs
 * @throws IOException if an I/O error occurs
 */
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    processRequest(request, response);
}

/**
 * Handles the HTTP <code>POST</code> method.
 * @param request servlet request
 * @param response servlet response
 * @throws ServletException if a servlet-specific error occurs
 * @throws IOException if an I/O error occurs
 */
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    processRequest(request, response);
}

/**
 * Returns a short description of the servlet.
 * @return a String containing servlet description
 */
@Override
public String getServletInfo() {
    return "Short description";
}// </editor-fold>

}
```

Firstuploadpage.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */


import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.sql.*;

/**
 *
 * @author dell
 */
public class firstuploadpage extends HttpServlet {

   /**
            * Processes requests for both HTTP <code>GET</code> and
<code>POST</code> methods.
    * @param request servlet request
    * @param response servlet response
    * @throws ServletException if a servlet-specific error occurs
    * @throws IOException if an I/O error occurs
    */
protected void processRequest(HttpServletRequest request, HttpServletResponse
response)
   throws ServletException, IOException {
      response.setContentType("text/html;charset=UTF-8");
      PrintWriter out = response.getWriter();
      out.println("<html><body>");
      try {
         int i=1;
       Class.forName("com.mysql.jdbc.Driver");
                                                                    Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/project1","root","an
grybird");
       String branch=request.getParameter("branch");
       String semester=request.getParameter("semester");
          PreparedStatement pst= con.prepareStatement("select link from subjectdata
where branch=? and semester=?");
       pst.setString(1, branch);
       pst.setString(2,semester);
       ResultSet rs=pst.executeQuery();
                 out.println("<p align=center><b>Jaypee University of Information
Technology</b></p>");
       out.println("<p align=center><b>Select Subject</b></p>");
       out.println();
       out.println("<table cellpadding=12>");
      while(rs.next())
        {
           String n = rs.getString(1);
           out.println("<tr><td>"+(i++)+"</td><td>"+n+"</td></tr>");
        }
     out.println("</table>");
     out.println("</body></html>");




    }
    catch(ClassNotFoundException e)
    {
    }
    catch(SQLException e)
    {
    }
    finally {
       out.close();
    }
  }

  // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the +
sign on the left to edit the code.">
  /**
   * Handles the HTTP <code>GET</code> method.
   * @param request servlet request
   * @param response servlet response
   * @throws ServletException if a servlet-specific error occurs
   * @throws IOException if an I/O error occurs
   */
  @Override
  protected void doGet(HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException {
    processRequest(request, response);
  }

  /**
   * Handles the HTTP <code>POST</code> method.
   * @param request servlet request
   * @param response servlet response
   * @throws ServletException if a servlet-specific error occurs
   * @throws IOException if an I/O error occurs
   */
  @Override
  protected void doPost(HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException {
    processRequest(request, response);
  }

  /**
   * Returns a short description of the servlet.
   * @return a String containing servlet description
   */
  @Override
  public String getServletInfo() {
    return "Short description";
  }// </editor-fold>

}
```

Firstgeneratepage.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */


import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.sql.*;

/**
 *
 * @author dell
 */
public class firstgeneratepage extends HttpServlet {

   /**
            * Processes requests for both HTTP <code>GET</code> and
<code>POST</code> methods.
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
protected void processRequest(HttpServletRequest request, HttpServletResponse
response)
   throws ServletException, IOException {
       response.setContentType("text/html;charset=UTF-8");
       PrintWriter out = response.getWriter();
       out.println("<html><body>");
       try {
          int i=1;
        Class.forName("com.mysql.jdbc.Driver");
                                                                    Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/project1","root","an
grybird");
        String branch=request.getParameter("branch");
        String semester=request.getParameter("semester");
            PreparedStatement pst= con.prepareStatement("select link from subdatagen
where branch=? and semester=?");
        pst.setString(1, branch);
        pst.setString(2,semester);
        ResultSet rs=pst.executeQuery();
                   out.println("<p align=center><b>Jaypee University of Information
Technology</b></p>");
        out.println("<p align=center><b>Select Subject</b></p>");
        out.println();
        out.println("<table cellpadding=12>");
```

```
      while(rs.next())
        {
           String n = rs.getString(1);
           out.println("<tr><td>"+(i++)+"</td><td>"+n+"</td></tr>");
        }
     out.println("</table>");
     out.println("</body></html>");





    }
    catch(ClassNotFoundException e)
    {
    }
    catch(SQLException e)
    {
    }
    finally {
      out.close();
    }
  }

  // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the +
sign on the left to edit the code.">
  /**
   * Handles the HTTP <code>GET</code> method.
   * @param request servlet request
   * @param response servlet response
   * @throws ServletException if a servlet-specific error occurs
   * @throws IOException if an I/O error occurs
   */
  @Override
  protected void doGet(HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException {
    processRequest(request, response);
  }

  /**
   * Handles the HTTP <code>POST</code> method.
   * @param request servlet request
   * @param response servlet response
   * @throws ServletException if a servlet-specific error occurs
   * @throws IOException if an I/O error occurs
   */
  @Override
  protected void doPost(HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException {
    processRequest(request, response);
    }

    /**
     * Returns a short description of the servlet.
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>

}
```

BIBLIOGRAPHY

1. All India Engineering Project Innovation Contest QPG, Available: http://www.siliconindia.com/aiepic/project/automatic_question_paper_generator-pid=2653.html , as on 5th August, 2012

2. Majumdar & Bhattacharya ,"Creating Schema", Database Management System , (2004),TMH publications, 113-159

3. Developing data flow diagram, Tutorial Available: as on 10th September,2012 http://www.visual-paradigm.com/product/lz/tutorials/dfd.jsp.

4. Gantt Chart Softwar, Available: http://www.smartdraw.com/specials/projectchart.asp, as on 25th July, 2012

5. Head First Servlets and Jsp, second edition by Kathy Sierra, O'Reilly publications.

6. Insight to HarshEduserve, Available: as on 5th August, 2012 http://www.harsheduserve.com/apg.

7. Java Servlet Programming By Jason Hunter, Publisher: O'Reilly Media, Released: November 1998.

8. JDBC Basics Tutorial, Available: as on 27th July, 2012 http://docs.oracle.com/javase/tutorial/jdbc/basics.

9. Prepared Statement in servlets, Available: as on 13 November, 2012. http://dev.mysql.com/doc/refman/5.0/en/sql-syntax-prepared-statements.html.

10. Question paper generator material, quickquest, Available: as on 7th August, 2012. http://srushti-soft.com/documents/quickquest.html.

11. Random generator paper, Available: as on 9th August, 2012. http://www.g-netsolutions.com/questionpapergeneration.htm.

12. Research Paper: Shuffling Algorithms for Automatic Generator Question Paper System, Nor Shahida bt Mohd Jamail, Abu Bakar Md Sultan, Computer and Information Science 01/2010.

13. UML Use Case, Available: as on 16th September, 2012. http://www.tutorialspoint.com/uml/uml_use_case_diagram.html.