PROJECT SYNOPSIS

**INDEX**

**Contents Page No.**

1. **Introduction 2**
   1. **Background 2**
   2. **Objective 3**
   3. **Purpose and scope 3**

**1.3.1 Purpose 3**

**1.3.2 Scope 3**

1. **Survey of Technologies 4-11**
2. **Requirements and Analysis 12**

**3.1 Problem Definition 12**

**3.2 Requirements Specification 13-15**

* 1. **Planning and Scheduling 15**
  2. **Software and Hardware Requirements 16**

**3.5 Conceptual Models 18-26**

1. **Preliminary Product Description 26**

**1 INTRODUCTION**

**1.1BACKGROUND**

With the passes of time, lots of changes have come in the examination systems. The manual examination system was meant for times when there were fewer students and courses. However, at present, with the gross enrolment ratio in higher education is going up, the examination system has to bear an increased load and leading towards inefficiencies [1]. Except a few institutions, most affiliated institutions depend heavily upon university for administrative, examination- related and curricular matters. This amounts to an unnecessary burden on the university as it is reduced to an administrative and exam conducting body [2]. The manual compilation of results takes very long time to declare results. This in turn affects students because some of them lose chances to get admissions in next higher classes. The students have to correspond by post or visit the university in person for examination-related queries. Sometimes, it becomes difficult to retrieve information manually for a specific candidate among huge volume of papers based databases.

The conventional paper-pen examination system is prone to errors, greater time consuming, inefficient and waste of valuable resources [1]. The students spent their valuable time and money for getting accurate information, but students are not getting this. They have to waste time and money for getting exam/result related information and sometimes to give bribes for the same due to several reasons. One of the main reasons is that in most of the Indian Universities, examination system is managed manually [3]. For universities, the ever-increasing paper-based record registers have made it difficult to store and manage information. There is repetition of work because the same data is represented in different forms by different branches. This leads to data duplication and huge money is being spent by the universities to buy paper and hire additional manpower. In addition to above, there are also chances of tampering with students‟ records, circulation of fake degrees, unfair practices, etc. [1].

The ICT has been facilitating some universities to manage examinations timely in a neat, clean and transparent manner. The main objectives of automation of examination systems are to minimize human intervention, curtail expenditures, bring efficiency, enhance productivity, optimal utilization of resources, better monitoring of examination activities to take quick decisions, timely availability of information/services for stakeholders, bring transparency, integration of isolated but related databases, minimize data redundancy, role-based access to users, reduce psychological pressure, improve public image, etc.

**1.2 OBJECTIVES**

Online test engine is an educational platform featuring the quickest, simplest way for organizations to train students, for FREE. The platform features an intuitive, easy-to-use interface to create, deliver and track training of students.

It offers a way to measure, analyses and view results of test takers. Its modern and clean theme is optimized for almost all devices including mobile and tabs. We believe preparation can be more social and fun and far more techno-friendly. With an internet-enabled device in the hands of most students, we felt a need for mobile test preparation platform giving students’ freedom to learn and prepare anywhere, anytime.

* Free service to all institutes.
* User Friendly interface.
* Online Tests.
* Discussion Forum.
* Leader board.
* Graphical Analysis of results.
* Accessibility anywhere any time.
* Zero installation.
* No development or maintenance cost.
* This can be used in educational institutions as well as in corporate world.
* Can be used anywhere any time as it is a web based application (user Location doesn’t matter).
* No restriction that examiner has to be present when the candidate takes the test.

**1.3** **PURPOSE, SCOPE**

**1.3.1** **PURPOSE**

The purpose of the project is to provide online facility to Institutes to conduct online exams and to Students to give online exams. Institutes can enter and edit the questions along with the students list. Also they can view the result. Students can login and give their respective exams and view their score then and there. Others can view sample papers to get look and feel of the online examination system.

**1.3.2 SCOPE**

## The website to conduct online examination is “Online Test Engine”. This website provides facility to institutes to conduct online exams. The institute provides questions along with positive and negative marks. Institute also enters the list of eligible students. All the information entered can be later edited by the institute.

## In turn student can login with their id, pass and institute name to give the exams and can view their result then and there. Institutes can also view the result of their students.

## SOFTWARE SCOPE

## Reusability: Reusability is possible as and when we require in this application. We can update it. Reusable software reduces design, coding, and testing cost by amortizing effort over several design. Reducing the amount of code also simplified understanding, which increases the likelihood that the code is correct.

## Extensibility: This software is extended in ways that its original developers may not expect. The following principles enhance extensibility like hide data structure, avoid traversing multiple links or methods, avoid case statement on object type and distinguish public and private operations.

* **Robustness:** Its method is robust if it does not fail even if it receives in proper parameters. There is some facilities like project against errors, optimize after the programs runs, validate argument and avoid predefined limits.

**2. Technology Survey**

1. **BOOTSTRAP -**

It is a CSS Framework for developing responsive and device friendly web applications. It uses jQuery as an internal implementation. Bootstrap makes front-end web development faster and easier. It's made for folks of all skill levels, devices of all shapes, and projects of all sizes.

|  |  |
| --- | --- |
| **Bootstrap** is a [free and open-source](http://en.wikipedia.org/wiki/Free_and_open-source_software) collection of tools for creating [websites](http://en.wikipedia.org/wiki/Website) and [web applications](http://en.wikipedia.org/wiki/Web_application). It contains [HTML](http://en.wikipedia.org/wiki/HTML)- and [CSS](http://en.wikipedia.org/wiki/CSS)-based design templates for [typography](http://en.wikipedia.org/wiki/Typography), forms, buttons, navigation and other interface components, as well as optional [JavaScript](http://en.wikipedia.org/wiki/JavaScript) extensions. The bootstrap framework aims to ease [web development](http://en.wikipedia.org/wiki/Web_development).  Bootstrap is a [front end](http://en.wikipedia.org/wiki/Front_and_back_ends) that is an [interface](http://en.wikipedia.org/wiki/Interface_(computer_science)) between the user and the server-side code which resides on the "back end" or [server](http://en.wikipedia.org/wiki/Server_(computing)). And it is a [web application framework](http://en.wikipedia.org/wiki/Web_application_framework), that is a [software framework](http://en.wikipedia.org/wiki/Software_framework) which is designed to support the development of [dynamic websites](http://en.wikipedia.org/wiki/Dynamic_web_page) and [web applications](http://en.wikipedia.org/wiki/Web_application).  Bootstrap is a sleek, intuitive, and powerful front-end framework for faster and easier web development.  Bootstrap tutorials will help you learn the essentials of Bootstrap, from the fundamentals to advanced topics, so that you can create web pages with much less efforts.  Tutorials are broken down into sections where each section containing a number of related topics that are packed with easy to understand explanations, practice examples, smart workarounds and useful tips.  You can save a lot of time and efforts with Bootstrap — So bookmark this website and continue on. |  |

If you're completely new to the Bootstrap we recommend you to start with the section that covers Bootstrap Basics and gradually move forward, by learning a little bit every day.

Twitter Bootstrap is the most popular front end framework in the recent time. It is sleek, intuitive, and powerful mobile first front-end framework for faster and easier web development. It uses HTML, CSS and Javascript.

1. **HTML -**

It is the basic language used to create web application structure. **Hyper Text Markup Language**, commonly referred to as HTML, is the standard markup language used to create web pages. It is written in the form of HTML elements consisting of tags enclosed in angle brackets (like <html>). HTML tags most commonly come in pairs like <h1> and </h1>, although some represent empty elements and so are unpaired, for example <img>. The first tag in such a pair is the start tag, and the second is the end tag (they are also called opening tags and closing tags).

Web browsers can read HTML files and render them into visible or audible web pages. Browsers do not display the HTML tags and scripts, but use them to interpret the content of the page. HTML describes the structure of a website semantically along with cues for presentation, making it a markup language, rather than a programming language.

HTML elements form the building blocks of all websites. HTML allows images and objects to be embedded and can be used to create interactive forms. It provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. It can embed scripts written in languages such as JavaScript which affect the behavior of HTML web pages.

1. **jQuery -**

It is a Javascript framework which provides some built-in functionalities for adding dynamic elements. **jQuery** is a [cross-platform](http://en.wikipedia.org/wiki/Cross-platform) [JavaScript library](http://en.wikipedia.org/wiki/JavaScript_library) designed to simplify the [client-side scripting](http://en.wikipedia.org/wiki/Client-side_scripting) of [HTML](http://en.wikipedia.org/wiki/HTML). jQuery is the most popular [JavaScript library](http://en.wikipedia.org/wiki/JavaScript_library) in use today.  jQuery is [free, open-source software](http://en.wikipedia.org/wiki/Free_and_open_source_software) licensed under the [MIT License](http://en.wikipedia.org/wiki/MIT_License).

jQuery's syntax is designed to make it easier to navigate a document, select [DOM](http://en.wikipedia.org/wiki/Document_Object_Model) elements, create [animations](http://en.wikipedia.org/wiki/Animation), handle [events](http://en.wikipedia.org/wiki/Event_(computing)), and develop [Ajax applications](http://en.wikipedia.org/wiki/Ajax_(programming)). jQuery also provides capabilities for developers to create [plug-ins](http://en.wikipedia.org/wiki/Plug-in_(computing)) on top of the JavaScript library. This enables developers to create [abstractions](http://en.wikipedia.org/wiki/Abstraction_(computer_science)) for low-level interaction and animation, advanced effects and high-level, theme-able widgets. The modular approach to the jQuery library allows the creation of powerful [dynamic web pages](http://en.wikipedia.org/wiki/Dynamic_web_page) and web applications.

JQuery UI is a curated set of user interface interactions, effects, widgets, and themes built on top of the jQuery JavaScript Library. Whether you're building highly interactive web applications or you just need to add a date picker to a form control, jQuery UI is the perfect choice.

jQuery UI is built for designers and developers alike. We've designed all of our plugins to get you up and running quickly while being flexible enough to evolve with your needs and solve a plethora of use cases. If you're new to jQuery UI, check out our [getting started guide](http://learn.jquery.com/jquery-ui/getting-started/) and [other tutorials](http://learn.jquery.com/jquery-ui/). Play around with the [demos](https://jqueryui.com/demos/) and read through the [API documentation](http://api.jqueryui.com/) to get an idea of what's possible.

1. **AJAX-**

**A**synchronous **J**avascript **A**nd **X**ml is used for server communications and fetching and loading data from the server. It makes fast web interactions by loading only the required data and not the underlying styling or HTML document structuring elements. **AJAX** (short for asynchronous [JavaScript](http://en.wikipedia.org/wiki/JavaScript) and [XML](http://en.wikipedia.org/wiki/XML))  is a group of interrelated [Web development](http://en.wikipedia.org/wiki/Web_development) techniques used on the [client-side](http://en.wikipedia.org/wiki/Client-side) to create [asynchronous](http://en.wikipedia.org/wiki/Asynchronous_I/O) [Web applications](http://en.wikipedia.org/wiki/Web_application). With Ajax, web applications can send data to and retrieve from a [server](http://en.wikipedia.org/wiki/Web_server) asynchronously (in the background) without interfering with the display and behavior of the existing page. Data can be retrieved using the [XMLHttpRequest](http://en.wikipedia.org/wiki/XMLHttpRequest) [object](http://en.wikipedia.org/wiki/Object_(computer_science)). Despite the name, the use of XML is not required ([JSON](http://en.wikipedia.org/wiki/JavaScript_Object_Notation) is often used in the [AJAJ](http://en.wikipedia.org/wiki/AJAJ) variant), and the requests do not need to be asynchronous.

Ajax is not a single technology, but a group of technologies. [HTML](http://en.wikipedia.org/wiki/Hypertext_Markup_Language) and [CSS](http://en.wikipedia.org/wiki/Cascading_Style_Sheets) can be used in combination to mark up and style information. The [DOM](http://en.wikipedia.org/wiki/Document_Object_Model) is accessed with JavaScript to dynamically display – and allow the user to interact with – the information presented. JavaScript and the XMLHttpRequest object provide a method for exchanging data asynchronously between browser and server to avoid full page reloads.

The [iframe](http://en.wikipedia.org/wiki/Iframe#Frames) tag was introduced by [Internet Explorer](http://en.wikipedia.org/wiki/Internet_Explorer) to load or to fetch content asynchronously.

1. **MySQL**

**MySQL** (but also called "My Sequel") is (as of July 2013) the world's second mostwidely used relational database management system (RDBMS) and most widely used open-source RDBMS. It is named after co-founder Michael Widenius's daughter, My. The SQL acronym stands for Structured Query Language.

The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL was owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now owned by Oracle Corporation.

MySQL is a popular choice of database for use in web applications, and is a central component of the widely used LAMP open source web application software stack (and other 'AMP' stacks). LAMP is an acronym for "Linux, Apache, MySQL, Perl/PHP/Python." Free-software-open source projects that require a full-featured database management system often use MySQL.

For proprietary use, several paid editions are available, and offer additional functionality. Applications which use MySQL databases include: [TYPO3](http://en.wikipedia.org/wiki/TYPO3), [MODx](http://en.wikipedia.org/wiki/MODx), [Joomla](http://en.wikipedia.org/wiki/Joomla), [WordPress](http://en.wikipedia.org/wiki/WordPress), [phpBB](http://en.wikipedia.org/wiki/PhpBB), [MyBB](http://en.wikipedia.org/wiki/MyBB), [Drupal](http://en.wikipedia.org/wiki/Drupal) and other software. MySQL is also used in many high-profile, large-scale [websites](http://en.wikipedia.org/wiki/Website), including [Google](http://en.wikipedia.org/wiki/Google) (though not for searches), [Facebook](http://en.wikipedia.org/wiki/Facebook), [Twitter](http://en.wikipedia.org/wiki/Twitter), [Flickr](http://en.wikipedia.org/wiki/Flickr), and [YouTube](http://en.wikipedia.org/wiki/YouTube).

1. **XML**

**Extensible Markup Language** (**XML**) is a markup language that defines a set of rules for encoding documents in a [format](http://en.wikipedia.org/wiki/File_format) which is both [human-readable](http://en.wikipedia.org/wiki/Human-readable_medium) and [machine-readable](http://en.wikipedia.org/wiki/Machine-readable_data). It is defined by the [W3C](http://en.wikipedia.org/wiki/World_Wide_Web_Consortium)'s XML 1.0 Specification and by several other related specifications, all of which are free [open standards](http://en.wikipedia.org/wiki/Open_standard).

The design goals of XML emphasize simplicity, generality and usability across the [Internet](http://en.wikipedia.org/wiki/Internet). It is a textual data format with strong support via [Unicode](http://en.wikipedia.org/wiki/Unicode) for different [human languages](http://en.wikipedia.org/wiki/Language). Although the design of XML focuses on documents, it is widely used for the representation of arbitrary [data structures](http://en.wikipedia.org/wiki/Data_structure) such as those used in [web services](http://en.wikipedia.org/wiki/Web_service).

Several [schema systems](http://en.wikipedia.org/wiki/XML_schema) exist to aid in the definition of XML-based languages, while many [application programming interfaces](http://en.wikipedia.org/wiki/Application_programming_interface) (APIs) have been developed to aid the processing of XML data.

Extensible Markup Language (XML) is a simple, very flexible text format derived from SGML (ISO 8879). Originally designed to meet the challenges of large-scale electronic publishing, XML is also playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere.

**7). AngularJS**, commonly referred to as Angular, is an open source web application framework maintained by Google and a community of individual developers and corporations to address many of the challenges encountered in developing single-page applications. Its goal is to simplify both development and testing of such applications by providing a framework for client-side model–view–controller (MVC) architecture, along with components commonly used in rich Internet applications.

The library works by first reading the HTML page, which has embedded into it additional custom tag attributes. Those attributes are interpreted as directives telling Angular to bind input or output parts of the page to a model that is represented by standard JavaScript variables. The values of those JavaScript variables can be manually set within the code, or retrieved from static or dynamic JSON resources.

It is a very powerful JavaScript library. It is used in Single Page Application (SPA) projects. It extends HTML DOM with additional attributes and makes it more responsive to user actions. AngularJS is open source, completely free, and used by thousands of developers around the world. It is licensed under the Apache license version 2.0.

It is based on the MVC pattern (Model View Control). Therefore AngularJS separates your RIA application into models, views and controllers. The views are specified using HTML + AngularJS's own template language. The models and controllers are specified via JavaScript objects and JavaScript functions. Thus, the views are specified declaratively, as HTML normally is, and the models and controllers are specified imperatively, as JavaScript normally is.

If you don't know the difference between declarative and imperative programming, don't worry. It is not important to know before learning AngularJS. Besides, it is pretty simple to find the definition on the web.

Despite AngularJS' success it is not without its challenges. AngularJS makes several bold claims about AngularJS on its project page. Personally, I believe some of those claims to be wrong or at least somewhat misguided. It's not like the model used by AngularJS has not been tried before in other incarnations. That is why I have added an [AngularJS critique](http://tutorials.jenkov.com/angularjs/critique.html) to this AngularJS tutorial series. If you decide to go ahead with AngularJS I would advise you to at least read the critique too. Too many tutorials just praise AngularJS without looking at its weak spots.

1. **JAVA**

**Java** is a general-purpose computer programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of computer architecture.

As of 2015, Java is one of the most popular programming languages in use, particularly for client-server web applications, with a reported 9 million developers.[citation needed] Java was originally developed by James Gosling at Sun Microsystems (which has since been acquired by Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++, but it has fewer low-level facilities than either of them.

The original and reference implementation Java compilers, virtual machines, and class libraries were originally released by Sun under proprietary licenses. As of May 2007, in compliance with the specifications of the Java Community Process, Sun relicensed most of its Java technologies under the GNU General Public License. Others have also developed alternative implementations of these Sun technologies, such as the GNU Compiler for Java (bytecode compiler), GNU Classpath (standard libraries), and IcedTea-Web (browser plugin for applets).

**USING JSP (Java Server Pages)**

Java Server Pages (JSP) is released in 1999 by SUN Microsystems. JSP is a technology for developing web pages. JSP is similar to PHP, but it uses the Java programming language. It follows the characteristics of Java ‘write once and run anywhere. JSP pages are platform independent means it can run in any platform. Java Server Pages (JSP) is a web technology that helps software developers to create dynamic content based web pages. Unlike a plain HTML page, which contains static content that always remains the same but in JSP; you can change content dynamically with the help of Java Bean and JSP elements.

**INTRODUCTION TO ECLIPSE IDE**

Eclipse is an integrated development environment (IDE) for developing primarily with Java, but also with other languages, in particular PHP, C/C++, and HTML5. It is also an application platform framework for Java desktop applications and others. The Eclipse IDE is written in Java and can run on Windows, OS X, Linux, Solaris and other platforms supporting a compatible JVM. The Eclipse Platform allows applications to be developed from a set of modular software components called modules. Applications based on the Eclipse Platform (including the Eclipse IDE itself) can be extended by third party developers. Eclipse IDE Mars 2.0 was released in February 2013 which added support for HTML5 and web technologies.Eclipse IDE Mars 2.0 is currently in final testing and expected later in 2013.Eclipse has a roadmap document for release plans.

## Eclipse Platform:

## Framework for simplifying the development of java swing desktop applications. The Eclipse IDE bundle for Java SE contains what is needed to start developing Eclipse plug-in and Eclipse Platform based applications; no additional SDK is required. Applications can install modules dynamically. Any application can include the Update Centre module to allow users of the application to download digitally signed upgrades and new features directly into the running application. Reinstalling an upgrade or a new release does not force users to download the entire application again.

The platform offers reusable services common to desktop applications, allowing developers to focus on the logic specific to their application. Among the features of the platform are:

* User interface management (e.g. menus and toolbars)
* User settings management
* Storage management (saving and loading any kind of data)
* Window management
* Wizard framework (supports step-by-step dialogs)
* Eclipse Visual Library
* Integrated development tools

Eclipse IDE is a free, open-source, cross-platform IDE with built-in-support for Java Programming Language.

**3. REQUIREMENTS AND ANALYSIS**

**3.1 PROBLEM DEFINITION:**

**NEEDS FOR THE SYSTEM**

It was decided to Computerize the System in order to overcome the following problems:-

1. To increase the Examination percentage.

2. To make student easy to test from anywhere.

3. To avoid thing like examination center.

4. Not a time saving.

**PROPOSED SYSTEM**

There will be three major components:

1. Reducing workload.

2. Data security.

3. Student account maintenance.

4. Faculty account maintenance.

5. Result generation.

6. Reducing paper work.

**Proposed System Provides Following Solution:**

1. It provides “better and efficient” service to students.

2. Reduce the workload of Administrator.

3. Faster retrieval of information about the different party’s candidates.

4. Provide facility for proper monitoring reduces paper work and provide data security.

5. All details will be available on a click.

**3.2 Requirement Specification**

**3.2.1 Software Requirement Specification**

**Introduction**

**Purpose:**

“Online Test Engine” is intended to help the student to test and learn their courses from anywhere easily. Good choice of student is always very helpful to grasp the collection easily.

**Scope**

“Online Test Engine” is a website in which student can test from anywhere for his/her place. With the help of this software result will make easy and fast and no any exam can missed.

**Overview**

The rest of this SRS is organized as follow: Section 2 gives us overall description of the software. It gives what level of proficiency is expected of the user, some general constraints while making the software and some assumption and dependencies that are assumed. Section 3 gives specific requirements, Functional requirements which the software is expected to deliver. Some performance requirements and design constraints are also given.

**3.2.2 Overall Description:**

**Product Perspective:**

“Online Test Engine” is aimed towards a person who wants to give the test from anywhere. It help to the student in the choice, searching or savings time. Online Test Engine should be user-friendly, ‘quick to learn’ reliable and time-saver software for the above purpose. This system is web-based application.

**Product Functions:**

This system performs the function of showing all the information to the student. It stores the information about the online exam and provides the online examination.

**User Characteristics**

* Student can search the exam list.
* User can register himself.
* User can login.
* User can test online.
* The process of conducting exams was secure, reliable and accountable.

**3.2.3 Principal Actors:**

The three principal actors in Online Test Engine are “student”, “Faculty” and “Admin”.

**3.2.4 Assumptions and Constraints:**

1. Full working of Online Test Engine is dependent on the availability of Internet connection.
2. Student is considered as novice user.
3. GUI based environment is needed.

**3.2.5 Specific Requirements:**

**Functional Requirements:**

1. Registration.

2. Login.

3. Insert, Update, Delete information.

4. Manage student, faculty and admin’s information.

5. Result generation.

**Performance Requirements:**

1. Should run on AMD IIX2 260 3.20 GHz or Higher, 512 MB machine.
2. 90% of the response should be within 5 second.

**3.2.6 Design Constraints:**

1. **Security:** The files in which the information regarding securities should be secured against malicious deformation.
2. **Fault** **Tolerance**: Data should not become corrupted in case of system crash or power failure.
3. **Interactive:** System should be user friendly and interactive
4. **Maintainable:** System should be easy to maintain and easy to handle.

**3 Planning and Scheduling**

**Work Tasks Week 1 Week 2 Week3 Week4**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Identifying Necessity |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| identifying the furniture types and new updation |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| identifyingDetails& needs |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Defining desired output/input/control |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Defining Validations |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Define the functionality Behavior |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Describe modes of interaction |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Research on file management |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| *Milestone : Software elements defined* |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Define Technical feasibility |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| *Milestone: Technical feasibility achieved* |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Creating DFD |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Coding |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Documentation |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Testing |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

**3.4 Tools and Environment User (S/W & H/W Requirements)**

**Hardware Requirements:**

1 System with following configuration (minimum)

* Dual 2GHz+ CPU
* 4GB+ RAM
* 80MB database space
* 100GB disk space

**Software Requirements**

* The Eclipse IDE ,with JAVA and Android plug in installed
* MySQL Server

**Front End:**

* Bootstrap
* HTML5 , CSS3
* JSP
* jQuery
* AJAX
* JavaScript
* AngularJS

**Back End:**

* **MySQL**
* **Java**

**Server** → Apache Tomcat

**3.5. Conceptual Models**

**3.5.1 DFD (Data flow Diagram)**

**0 level DFD**

Get Course

View Course

Give Test

Test Result

Student

Student

Response quarry

Post quarry

View Result

Check result

View Test

Admin

Admin

Get Test

Faculty updated

Add faculty

## 1st Level DFD

query\_db

course\_db

Get Course

Get quarry

response quarry

Get test

View Course

tests\_db

Give test

Get Result

Student

Stud details

Get result

userdetails\_db

Upload test

Upload success

Get details

Get student

View student

Manages deatils

Admin

admin details

userdetial\_ db

View

get details

ACK

Check test

View course

Course details

View test

manages faculty

userdetails \_db

tests\_db

course\_db

## 2nd Level DFD

## 2 LEVEL’S DFD (STUDENT LOGIN PROCESS)

Take test

Quarries response

Ask quarries

Login success & rejected

Enter User id & pass

Students

query\_db

test\_db

Get result

get course

course\_db

read course

**2 LEVEL’S DFD (STUDENT REGISTRATION PROCESS)**

Students

userdetails\_db

Check user id & Pass

Exists user id & Pass or not

Enter user id & Pass

user id & Pass accepted

student info updated

student details

student info updated

student details

Enter user Id

Voters

**3 LEVEL’S DFD (FACULTY LOGIN PROCESS)**

Faculty

course\_db

add course

Successful updated

Enter user id & Pass

user id & Pass accepted

faculty info updated

update details

test uploaded

View student

upload test

get details

userdetails\_db

userdetails\_db

tests\_db

**2nd Level ADMIN LOGIN PROCESS**

get details

ACK

course\_db

tests\_db

student deatils

Retrieve Info

Add batch

Add faculty

View faculty

Upload test

View test

Request verification

Success or Fail

Admin Login

userlogin\_db

Admin

ad

View batch

Get info

view student

Update faculty

view course

Get details

Test view

Get details

userdetails\_db

userdetails\_db

**2nd Level Data Flow Diagram**

userdetails\_db

userdetails\_db

Test completed

Uploads test

tests\_db

Admin details

admin Login

Login success

Student details

Admin Login

Check details

Student Login

Login success

Success register

admin register

admin updated

student details

admin details

student updated

Success register

Student register

Check result

Test result

Student Login

check details

Faculty login

Upload test

Success login

faculty

View details

Uploaded test

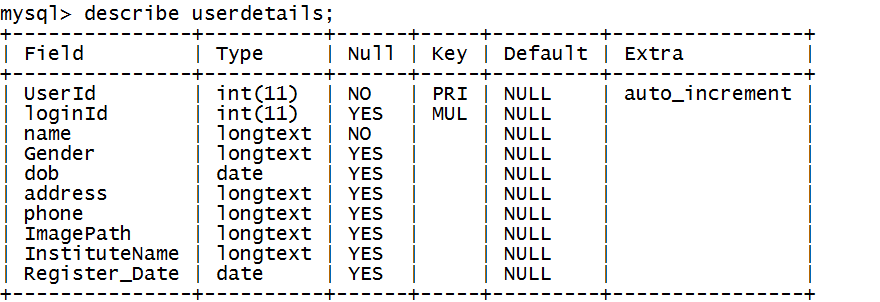
details update

**usertestresult** \_db

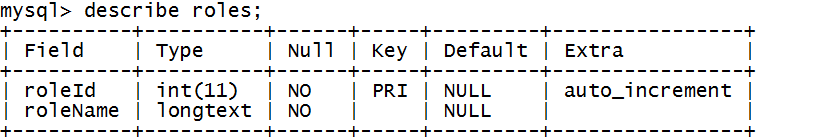
userdetail\_db

* + 1. **Data Base Design**

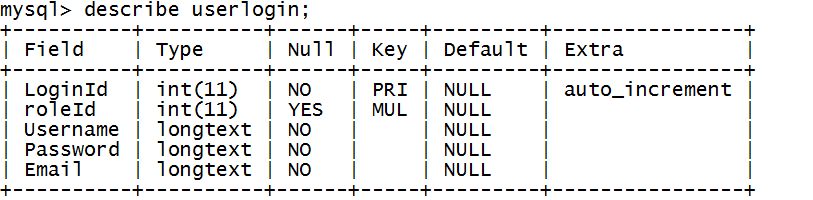
1. **userdetails Table**

****

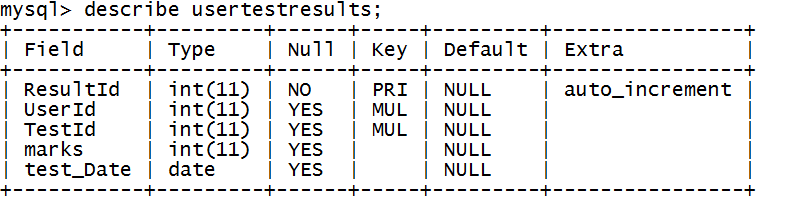
1. **roles Table**

****

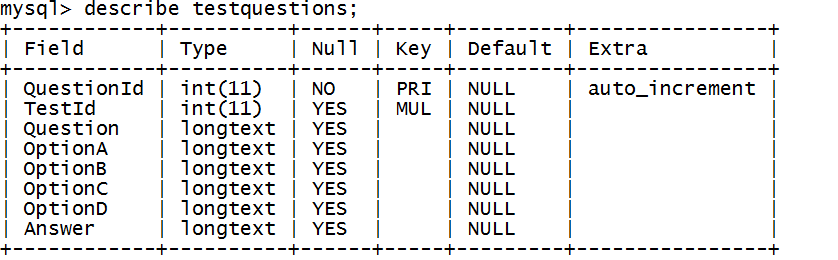
1. **userlogin Table**

****

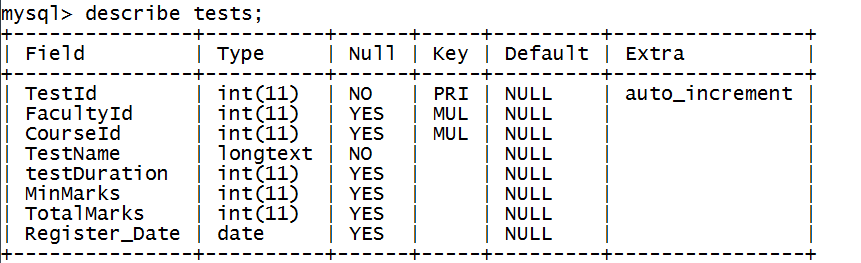
1. **usertestresult Table**

****

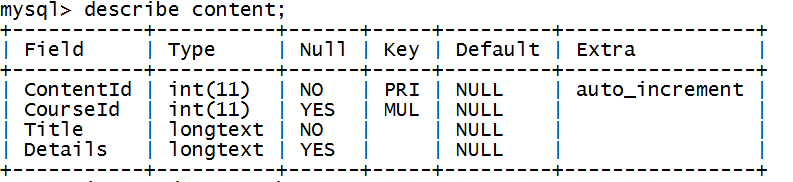
1. **testquestions Table**

****

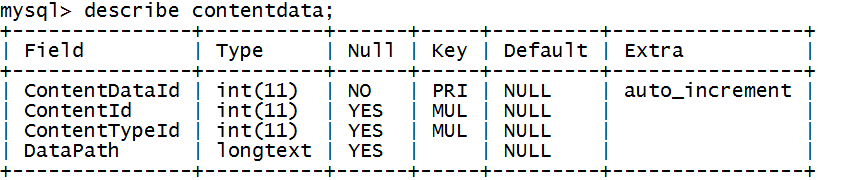
1. **tests Table**

****

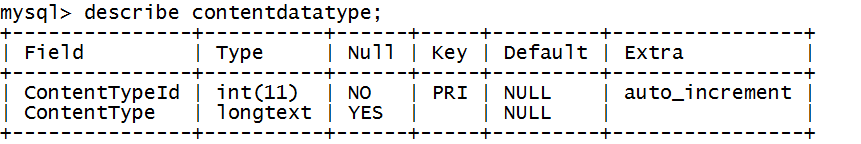
1. **content Table**

****

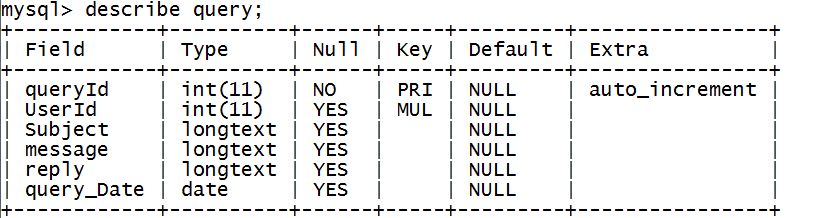
1. **ContentData Table**

****

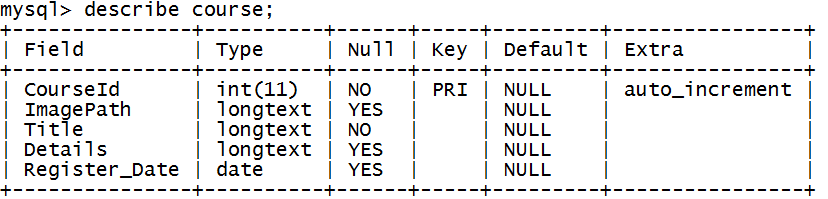
1. **ContentDataType Table**

****

1. **query Table**

****

1. **Course Table**

****

**4 PRELIMINARY PRODUCT DESCRIPTION**

**4.1 Modules Description**

Here in this module we identify the requirements and objective of all the new system. Define the function and operation of the application/system. There are six modules in this project and their description is as follows:-

* **Home**: It is the link itself, the shown page is the home page. Which cantinas the notice board and Discussion Forum and tests etc. related information.
* **Explore**: It tells about the features of the site. This provides the details
* **Registration:** This module describes entire information related to the student. Registration information will be saved student and admin table. With the help of this module unregistered student, s/he will be linked to the registration page.
* **Login:** The application wed based and supported to support many students simultaneously. In this module student, admin and faculty can login, but student will find student dash board page, faculty will find faculty dash board and administrator will find admin dash board page.
* **Tests:** This module is used to check results, student information, result percentage, student details etc. Administrator can be asked to print results.
* **Course:** This module is used to provide the course content to student, which is uploaded by the faculty and admin also can view the courses and course content of the student.
* **Contact us**: This link to the contact us page. Where institute address and contact details will be mentioned.

PROJECT REPORT

**TABLE OF CONTENT**

**TOPIC Page No.**

**TITLE OF THE PROJECT**

**SYNOPSIS OF PROJECT**

**PROJECT REPORT**

**CHAPTER 1: INTRODUCTION OF PROJECT**

**Introduction 32**

**Objective 32**

**Purpose, Scope & Availability 33**

**CHAPTER 2: SYSTEM ANALYSIS**

**Problem Definition 36**

**Feasibility Study 37**

**Project Planning 38**

**Project Scheduling (PERT & GANTT charts) 38-40**

**Software Requirement Specification (SRS) 41**

**Software Engineering Paradigm Applied 45-46**

**Conceptual Models (DFD’s, ERD, Use Case) 47-55**

**CHAPTER 3: SYSTEM DESIGN**

**Basic Modules 56**

**Data Design 57-60**

**User Interface Design 61-71**

**CHAPTER 4: CODING 72-155**

**Database coding 72-96**

**Project coding 96-241**

**CHAPTER 5: IMPLEMENTATION AND TESTING**

**Testing 242-247**

**CHAPTER 6: SYSTEM SECURITY MEASURES 248-251**

**CHAPTER 7: COST ESTIMATION OF PROJECT 252-255**

**CHAPTER 8: FUTURE SCOPE AND FURTHER 256-257**

**ENHANCEMENT**

**CHAPTER: 9 CONCLUSION AND BIBLIOGRAPHY 258-261**

**APPENDIX A**

**Acronyms**

**CHAPTER -1**

**INTRODUCTION OF PROJECT**

**INTRODUCTION**

**1.1BACKGROUND**

With the passes of time, lots of changes have come in the examination systems. The manual examination system was meant for times when there were fewer students and courses. However, at present, with the gross enrolment ratio in higher education is going up, the examination system has to bear an increased load and leading towards inefficiencies [1]. Except a few institutions, most affiliated institutions depend heavily upon university for administrative, examination- related and curricular matters. This amounts to an unnecessary burden on the university as it is reduced to an administrative and exam conducting body [2]. The manual compilation of results takes very long time to declare results. This in turn affects students because some of them lose chances to get admissions in next higher classes. The students have to correspond by post or visit the university in person for examination-related queries. Sometimes, it becomes difficult to retrieve information manually for a specific candidate among huge volume of papers based databases.

The conventional paper-pen examination system is prone to errors, greater time consuming, inefficient and waste of valuable resources [1]. The students spent their valuable time and money for getting accurate information, but students are not getting this. They have to waste time and money for getting exam/result related information and sometimes to give bribes for the same due to several reasons. One of the main reasons is that in most of the Indian Universities, examination system is managed manually [3]. For universities, the ever-increasing paper-based record registers have made it difficult to store and manage information. There is repetition of work because the same data is represented in different forms by different branches. This leads to data duplication and huge money is being spent by the universities to buy paper and hire additional manpower. In addition to above, there are also chances of tampering with students‟ records, circulation of fake degrees, unfair practices, etc. [1].

The ICT has been facilitating some universities to manage examinations timely in a neat, clean and transparent manner. The main objectives of automation of examination systems are to minimise human intervention, curtail expenditures, bring efficiency, enhance productivity, optimal utilization of resources, better monitoring of examination activities to take quick decisions, timely availability of information/services for stakeholders, bring transparency, integration of isolated but related databases, minimise data redundancy, role-based access to users, reduce psychological pressure, improve public image, etc.

**1.2 OBJECTIVES**

Online test engine is an educational platform featuring the quickest, simplest way for organizations to train students, for FREE. The platform features an intuitive, easy-to-use interface to create, deliver and track training of students.

It offers a way to measure, analyses and view results of test takers. Its modern and clean theme is optimized for almost all devices including mobile and tabs. We believe preparation can be more social and fun and far more techno-friendly. With an internet-enabled device in the hands of most students, we felt a need for mobile test preparation platform giving students’ freedom to learn and prepare anywhere, anytime.

* Free service to all institutes.
* User Friendly interface.
* Online Tests.
* Leader board.
* Graphical Analysis of results.
* Accessibility anywhere any time.
* No development or maintenance cost.
* This can be used in educational institutions as well as in corporate world.
* Can be used anywhere any time as it is a web based application (user Location doesn’t matter).
* No restriction that examiner has to be present when the candidate takes the test.

**1.3** **PURPOSE, SCOPE**

**1.3.1** **PURPOSE**

The purpose of the project is to provide online facility to Institutes to conduct online exams and to Students to give online exams. Institutes can enter and edit the questions along with the students list. Also they can view the result. Students can login and give their respective exams and view their score then and there. Others can view sample papers to get look and feel of the online examination system.

**1.3.2 SCOPE**

## The website to conduct online examination is “Online Test Engine”. This website provides facility to institutes to conduct online exams. The institute provides questions along with positive and negative marks. Institute also enters the list of eligible students. All the information entered can be later edited by the institute.

## In turn student can login with their id, pass and institute name to give the exams and can view their result then and there. Institutes can also view the result of their students.

## SOFTWARE SCOPE

## Reusability: Reusability is possible as and when we require in this application. We can update it. Reusable software reduces design, coding, and testing cost by amortizing effort over several design. Reducing the amount of code also simplified understanding, which increases the likelihood that the code is correct.

## Extensibility: This software is extended in ways that its original developers may not expect. The following principles enhance extensibility like hide data structure, avoid traversing multiple links or methods, avoid case statement on object type and distinguish public and private operations.

* **Robustness:** Its method is robust if it does not fail even if it receives in proper parameters. There is some facilities like project against errors, optimize after the programs runs, validate argument and avoid predefined limits.

**CHAPTER-2**

**SYSTEM ANALYSIS**

**PROBLEM DEFINITION:**

**NEEDS FOR THE SYSTEM**

It was decided to Computerize the System in order to overcome the following problems:-

1. To increase the Examination percentage.

2. To make student easy to test from anywhere.

3. To avoid thing like examination center.

4. Not a time saving.

**PROPOSED SYSTEM**

There will be three major components:

1. Reducing workload.

2. Data security.

3. Student account maintenance.

4. Faculty account maintenance.

5. Result generation.

6. Reducing paper work.

**Proposed System Provides Following Solution:**

1. It provides “better and efficient” service to students.

2. Reduce the workload of Administrator.

3. Faster retrieval of information about the different party’s candidates.

4. Provide facility for proper monitoring reduces paper work and provide data security.

5. All details will be available on a click.

**FEASIBILITY STUDY**

**Each and every project needs to have a feasibility study for the complete understandability of the project. We will consider following types of feasibility study they are technical feasibility, operational feasibility and economic feasibility etc. as follows:**

***Technical feasibility:*-This new system requires 6 fully trained people to run the system perfectly. 1 Admin person to maintain database and other 5 to handle the system interface and order making things. As our existing system is purely manual, so we need a onetime investment of Rs. 4 Lacks for the purchase of 6 computers, 5 invoice printers, a laser printer, AC and networking etc. It requires approx. 10 Lacks PA as an operating cost.**

**With the above details our system is technically feasible as after investing 14 lacks in a year, the company is still saving Rs. 15 Lacks PA.**

***Operational Feasibility:*-The new solution is feasible in all sense but operationally it is not. The new system demands the expulsion of at least 15 people from the company. It creates an environment of joblessness and fear among the employees. It can lead to an indefinite strike in the company also. So the management must take corrective actions prior in advance in order to start the further proceedings.**

***Economic Feasibility:*-With the manual system the operating cost of the system is about 60 Lacks P.A. This cost comprises salary of 25 people, stationary, building rent, electricity, water, telephone etc. But with the new system this reoccurring cost comes out to be about 20 lacks P.A. Hence the new system is economically feasible.**

***Schedule Feasibility:*** - Does the company currently have the time resources to undertake the project? Can the project be completed in the available time?

***Cultural Feasibility:*** - What will be the impact on both local and general cultures? What sort of environmental implications does the feasibility study have?

***Legal/Ethical Feasibility:***-

What are the legal implications of the project? What sort of ethical considerations are there? You need to make sure that any project undertaken will meet all legal and ethical requirements before the project is on the table.

***Resource Feasibility:*** - Do you have enough resources, what resources will be required, what facilities will be required for the project, etc.

***Marketing Feasibility:*** - Will anyone want the product once it’s done? What is the target demographic? Should there be a test run? Is there enough buzz that can be created for the product?

***Real Estate Feasibility*:**-What kind of land or property will be required to undertake the project? What is the market like? What are the zoning laws? How will the business impact the area?

**Planning and Scheduling**

**Gantt chart**

**Work Tasks Week 1 Week 2 Week3 Week4**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Identifying Necessity |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| identifying the Students and requirements for their career guidance |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| identifying Details & needs |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Defining desired output/input/control |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Defining Validations |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Define the functionality Behavior |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Describe modes of interaction |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Research on file management |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| *Milestone : Software elements defined* |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Define Technical feasibility |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| *Milestone: Technical feasibility achieved* |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Creating DFD |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Coding |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Documentation |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Testing |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

**PERT CHART FOR THE PROJECT**

It is very important for any multi-people project to time estimation. It insures best recourse and manpower utilization. In case of this project, there is only one man is involve and takes very few resources. But according to program structure and available time, there are a schedule must be follow. And it was achieved by PERT and Timeline Chart.

PERT (program evaluation and review technique) – PERT provides quantitative tools that allow the software planner to

1. Determine the critical path
2. established the “most likely” time estimation
3. Calculate boundary time that define a time windows for a particular task

**Major Tasks of the Project are:**

Task  time (in week)

1. Requirement analysis 3

2. System Design 4

3. Interface design 3

4. Animation file design 2

5. JAVA source code development and testing 1

6. Authoring 2

7. Integration and testing 2

8. Implementation 1

3

Start

Requirement and

Analysis

Design

Server module

Server Setting

coding

Integration

& testing

Implementation

4

3

3

1

2

1

Critical Path = 16

3

***Fig. – Pert chart***

In above graph bold line show the critical and their value= 16 week

**SOFTWARE REQUIREMENT & SPECIFICATION (SRS)**

We know that software requirement specification (**SRS)** is the very important part of the software development process. This describes the actual or real user level requirement from technical point of view. I.e. what is the users exactly wanting. It means that what is the main target or expectation of users from this project. The objective of preparing the SRS is to represent and describes all the requirements of the user. It is the main task of system analysts. The main goal of a system analyst is to prepare the SRS in such a manner that ultimately leads to successful software specification actually; Software Requirement Specification (SRS) is the outcome of System Analysis. This is the result of this phase which describes all the requirements of user in complete and most understandable format.

The SRS outlines the following information about the project.

1. **Introduction:** this is the first section which introduces the project with their objective and scope. This section includes following sub-section to introduce the project in more enhanced manner.
   * **Purpose:** describes the main goal/objectives of the project.
   * **Scope:** what is the scope and chances of future enhancement of the project are described in this section
   * **Definition, Acronyms:** this sub-section makes the project more comprehensive.
   * **Overview:** this sub-section summarizes the introduction.
2. **Overall Description: -** this section completely describes the project. This section also contains sub-sections as follow:

* **Product Perspective: -** this sub-section describes the concerned product.
* **Product Function: -** The functionality of the project is described in this part of overall description.
* **User Characteristics: -** this sub-section recognizes the type of user to use the system. What is the main characteristic of user, how they should be interacted through system.
* **Constraints: -** Every project have some constraints which should be highlighted otherwise this may create a long-lasted problem in future. This sub-section throws the light on the main constraints of the project.
* **Assumptions and Dependencies: -** This sub-section describes the dependency of the system.

1. **Specific Requirement: -** this section specifies the requirements of the system/user. The system utilizes the overall description to specify the constraints. This section has the following the sub-section.

Let I prepare a Software Requirement Specification (SRS) for my project.

**Software Requirement Specification (SRS) Document**

**Introduction: -** With the passes of time, lots of changes have come in the examination systems. The manual examination system was meant for times when there were fewer students and courses. However, at present, with the gross enrolment ratio in higher education is going up, the examination system has to bear an increased load and leading towards inefficiencies [1]. Except a few institutions, most affiliated institutions depend heavily upon university for administrative, examination- related and curricular matters.

**Purpose: -** The purpose of the project is to provide online facility to Institutes to conduct online exams and to Students to give online exams. Institutes can enter and edit the questions along with the students list. Also they can view the result. Students can login and give their respective exams and view their score then and there.

**Scope:** - By this website we can providing the information to the Users and Visitors. And find out the better the performance and smooth working by this Software.

**Overall Description: -** This section of the SRS provides an overall description of all the related information of the project.

**Project Perspective: -** This project is a web-based application which can connect the people all over the world. The main advantage of the project is that this project supports the advanced network technology like web-services and AJAX.

**Functionality:-**  this system performs the function of showing all the information to user and visitors. It stores the information about the library.

**User Characteristics**: - The user of the system may be novice for the computers so we have provided him an easy and interactive user-interface. It is quite interesting to tell that the user can know by looking up the icons and menus written in English that what has to do.

**Database Requirements: -** The database requirement of the system is very important thing. Every system needs data for their process to produce information. For this, they need an on-line storage to store data and can retrieve the data as per need.

To make a physical storage it is very necessary to design the databases logically. We can implement the database by looking-up the logical Design of data base.

**Design Constraints:-**  The system is designed in such a way that it would be:

* User-friendly
* Interactive
* Easy Maintainable
* Easy to Handle etc.

My project is following everything listed above in the designed constraint.

**Comments: -** This system provides services to the Users in electronic form that can be accessed anywhere, anytime, and in any condition through Internet.

**SYSTEM SPECIFICATIONS**

**Hardware Requirements:**

1 System with following configuration (minimum)

* Dual 2GHz+ CPU
* 4GB+ RAM
* 80MB database space
* 100GB disk space

**Software Requirements**

* The Eclipse IDE ,with JAVA and Android plug in installed
* MySQL Workbench

**Front End:**

* Bootstrap
* HTML5 , CSS3
* JSP
* Jquery
* AJAX
* JavaScript
* AngularJS

**Back End:**

* **MySQL**
* **Java**

**Server** → Apache Tomcat

**SOFTWARE ENGINEERING PARADIGMS APPLIED**

The model that is basically being followed is the AGILE MODEL, which is a combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working software product.

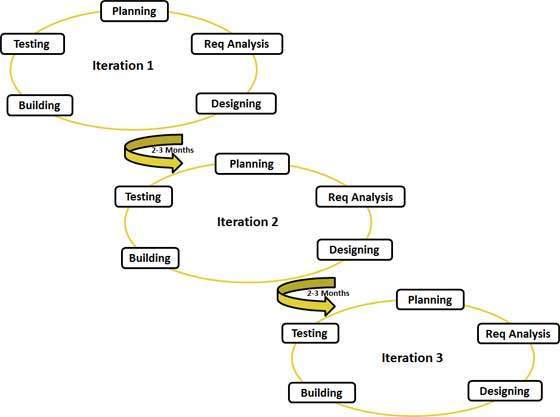
Agile Methods break the product into small incremental builds. These builds are provided in iterations. Each iteration typically lasts from about one to three weeks. Every iteration involves cross functional teams working simultaneously on various areas like planning, requirements analysis, design, coding, unit testing, and acceptance testing.

At the end of the iteration a working product is displayed to the customer and important stakeholders.

In this model, every project needs to be handled differently and the existing methods need to be tailored to best suit the project requirements. In agile the tasks are divided to time boxes (small time frames) to deliver specific features for a release.

Iterative approach is taken and working software build is delivered after each iteration. Each build is incremental in terms of features; the final build holds all the features required by the customer.

Here is a graphical illustration of the Agile Model:

****

**Fig: 2.1 Graphical illustration of the Agile Model**

Following are the Agile Manifesto principles:

* **Individuals and interactions** - in agile development, self-organization and motivation are important, as are interactions like co-location and pair programming.
* **Working software** - Demo working software is considered the best means of communication with the customer to understand their requirement, instead of just depending on documentation.
* **Customer collaboration** - As the requirements cannot be gathered completely in the beginning of the project due to various factors, continuous customer interaction is very important to get proper product requirements.
* **Responding to change** - agile development is focused on quick responses to change and continuous development.

**DFD (Data flow Diagram)**

**0 level DFD**

Get Course

View Course

Give Test

Test Result

Student

Student

Response quarry

Post quarry

View Result

Check result

View Test

Admin

Admin

Get Test

Faculty updated

Add faculty

## 1st Level DFD

query\_db

course\_db

Get Course

Get quarry

response quarry

Get test

View Course

tests\_db

Give test

Get Result

Student

Stud details

Get result

userdetails\_db

Upload test

Upload success

Get details

Get student

View student

Manages deatils

Admin

admin details

userdetial\_ db

View

get details

ACK

Check test

View course

Course details

View test

manages faculty

userdetails \_db

tests\_db

course\_db

## 2nd Level DFD

## 2 LEVEL’S DFD (STUDENT LOGIN PROCESS)

Take test

Quarries response

Ask quarries

Login success & rejected

Enter User id & pass

Students

query\_db

test\_db

Get result

get course

course\_db

read course

**2 LEVEL’S DFD (STUDENT REGISTRATION PROCESS)**

Students

userdetails\_db

Check user id & Pass

Exists user id & Pass or not

Enter user id & Pass

user id & Pass accepted

student info updated

student details

student info updated

student details

Enter user Id

Voters

**3 LEVEL’S DFD (FACULTY LOGIN PROCESS)**

Faculty

course\_db

Add course

Successful updated

Enter user id & Pass

user id & Pass accepted

faculty info updated

update details

test uploaded

View student

upload test

get details

userdetails\_db

userdetails\_db

tests\_db

**2nd Level ADMIN LOGIN PROCESS**

get details

ACK

course\_db

tests\_db

student deatils

Add batch

Add faculty

View faculty

Upload test

View test

Request verification

Success or Fail

Admin Login

userlogin\_db

Admin

ad

Retrieve Info

View batch

Get info

view student

Update faculty

view course

Get details

Test view

Get details

userdetails\_db

userdetails\_db

**2nd Level Data Flow Diagram**

userdetails\_db

userdetails\_db

Test completed

Uploads test

tests\_db

Admin details

admin Login

Login success

Student details

Admin Login

Check details

Student Login

Login success

Success register

admin register

admin updated

student details

admin details

student updated

Success register

Student register

Test result

Student Login

check details

Faculty login

Upload test

Success login

faculty

View details

Uploaded test

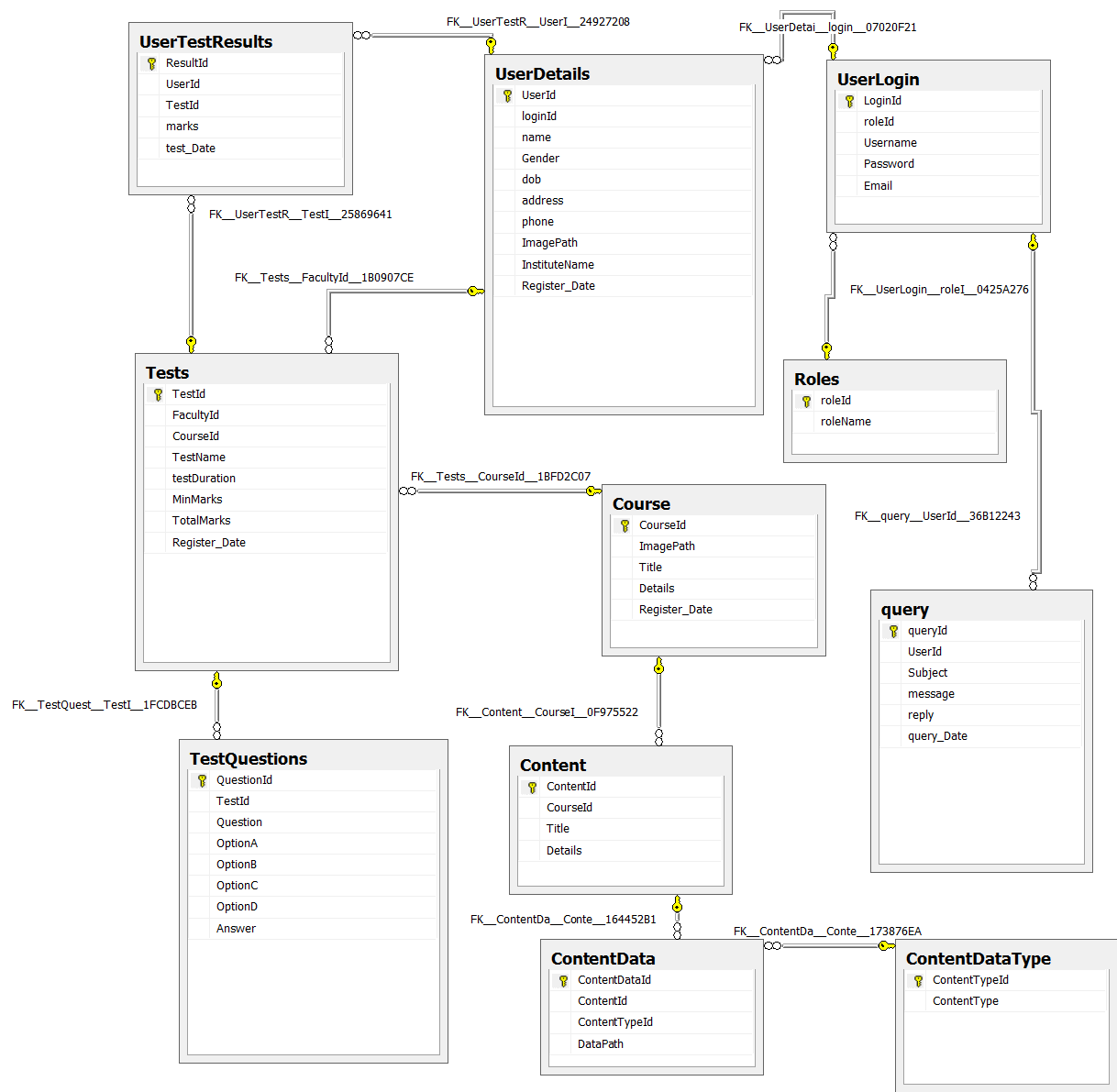
details update

**usertestresult** \_db

userdetail\_db

Check result

**E-R Diagram**

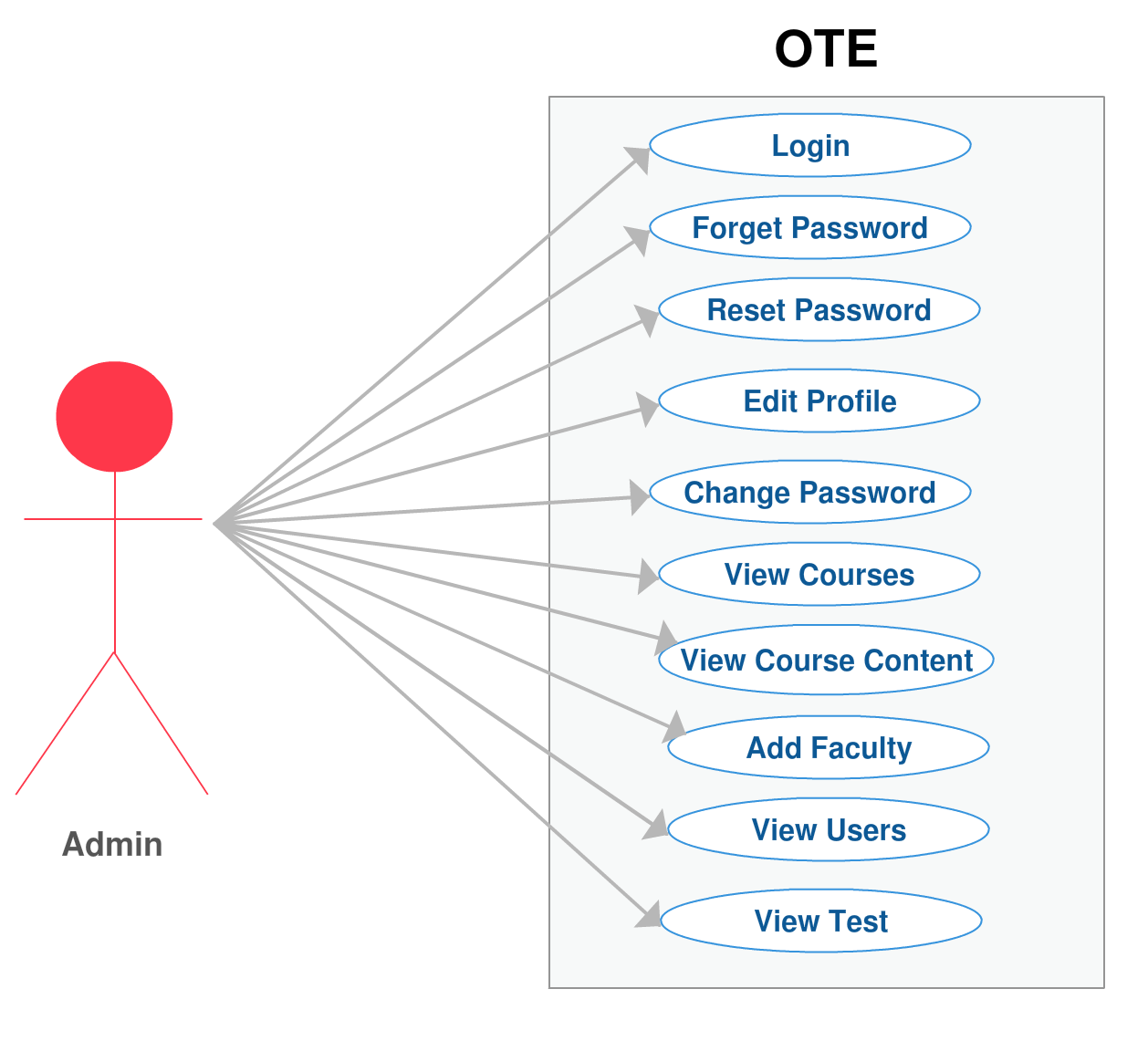
****

**Use Case Diagrams**

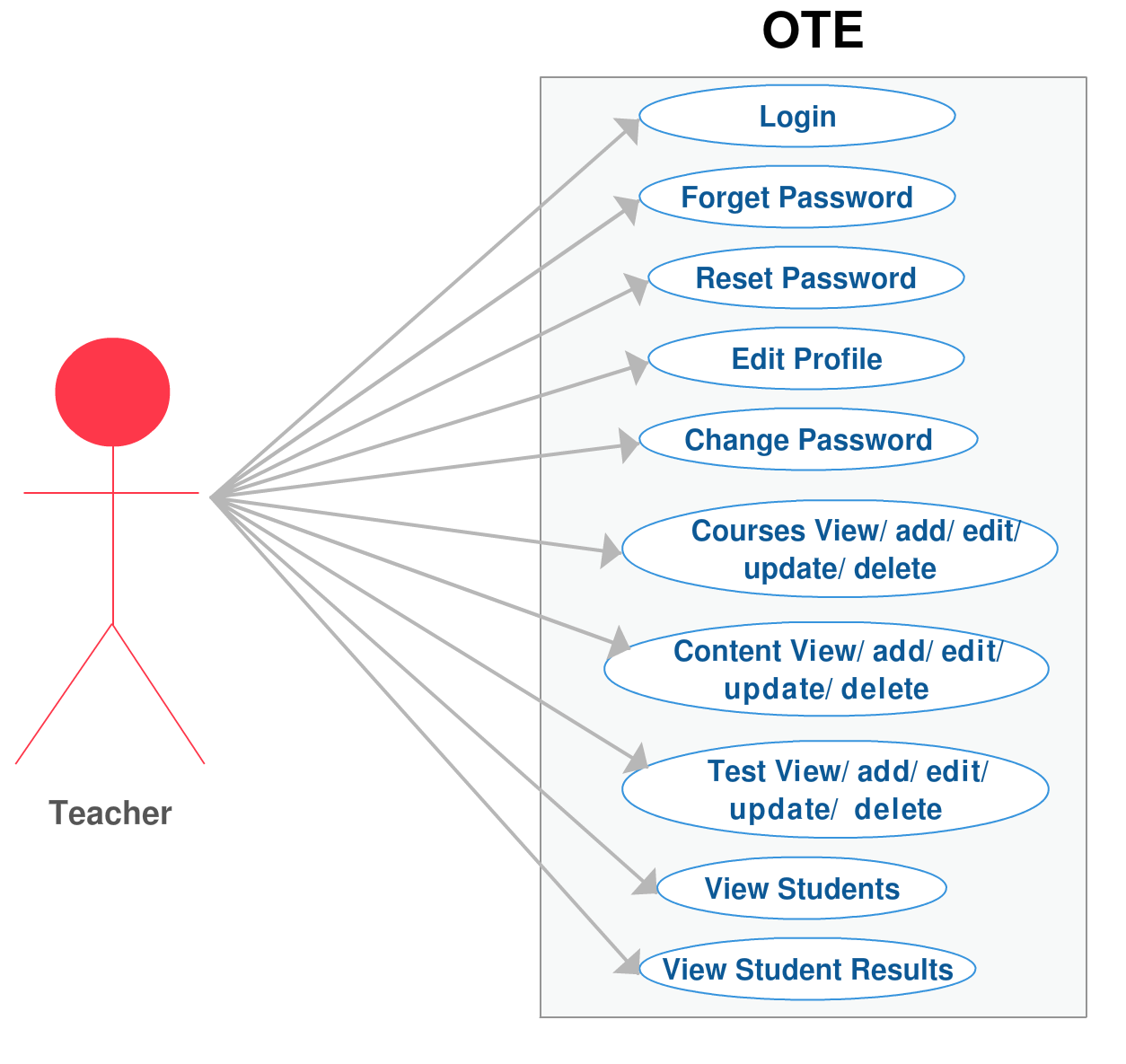
**Student Use Case Diagram**

****

**Admin Use Case Diagrams**

****

**Faculty Use Case Diagram**

****

**CHAPTER 3**

**SYSTEM DESIGN**

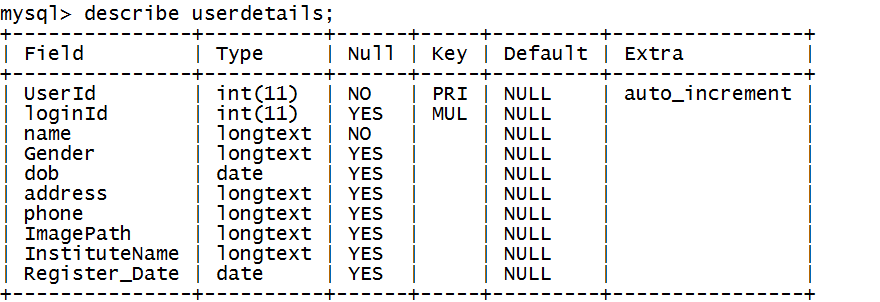
**Modules Description**

Here in this module we identify the requirements and objective of all the new system. Define the function and operation of the application/system. There are six modules in this project and their description is as follows:-

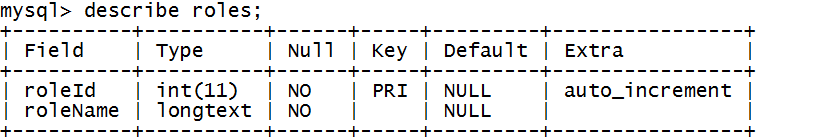
* **Home**: It is the link itself, the shown page is the home page. Which cantinas the notice board and Discussion Forum and tests etc. related information.
* **Registration:** This module describes entire information related to the student. Registration information will be saved student and admin table. With the help of this module unregistered student, s/he will be linked to the registration page.
* **Login:** The application wed based and supported to support many students simultaneously. In this module student, admin and faculty can login, but student will find student dash board page, faculty will find faculty dash board and administrator will find admin dash board page.
* **Tests:** This module is used to check results, student information, result percentage, student details etc. Administrator can be asked to print results.
* **Course:** This module is used to provide the course content to student, which is uploaded by the faculty and admin also can view the courses and course content of the student.
* **Contact us**: This link to the contact us page. Where institute address and contact details will be mentioned.

**Data Base Design**

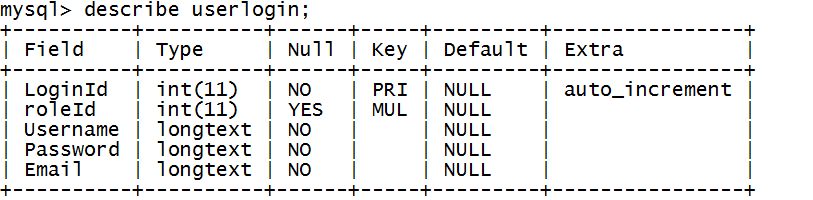
**1. userdetails Table**

****

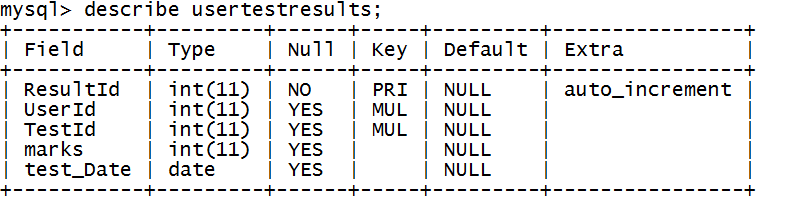
1. **roles Table**

****

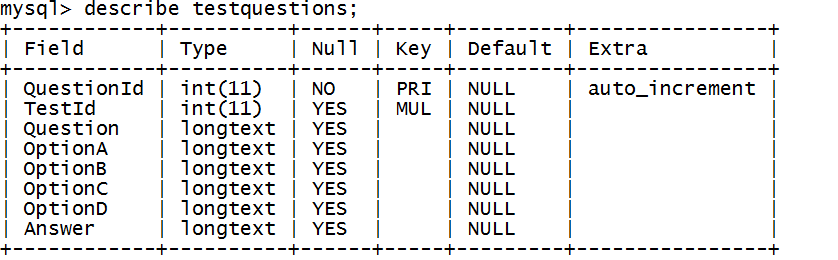
1. **userlogin Table**

****

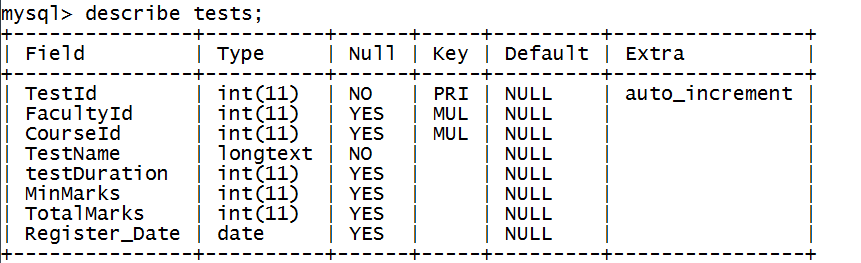
1. **usertestresult Table**

****

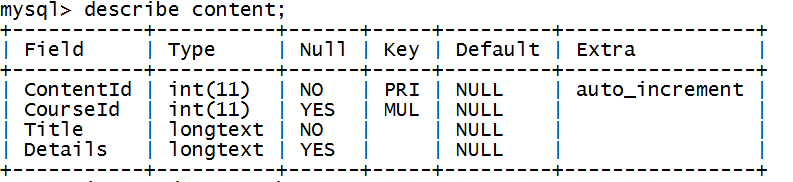
1. **testquestions Table**

****

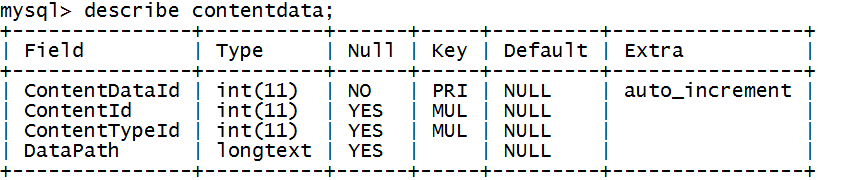
1. **tests Table**

****

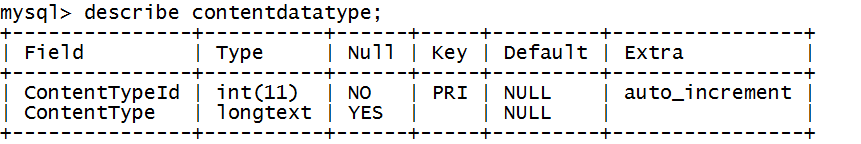
1. **content Table**

****

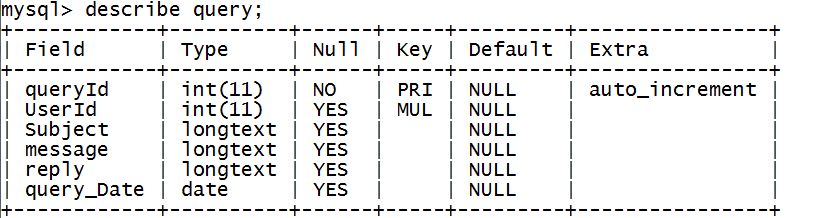
1. **ContentData Table**

****

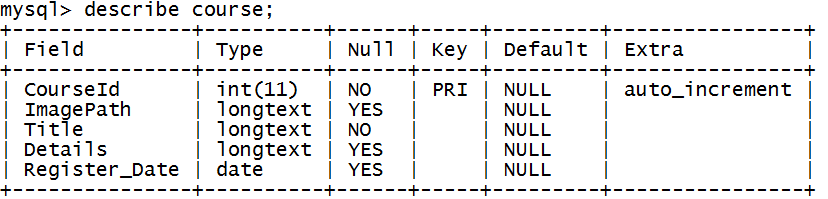
1. **ContentDataType Table**

****

1. **query Table**

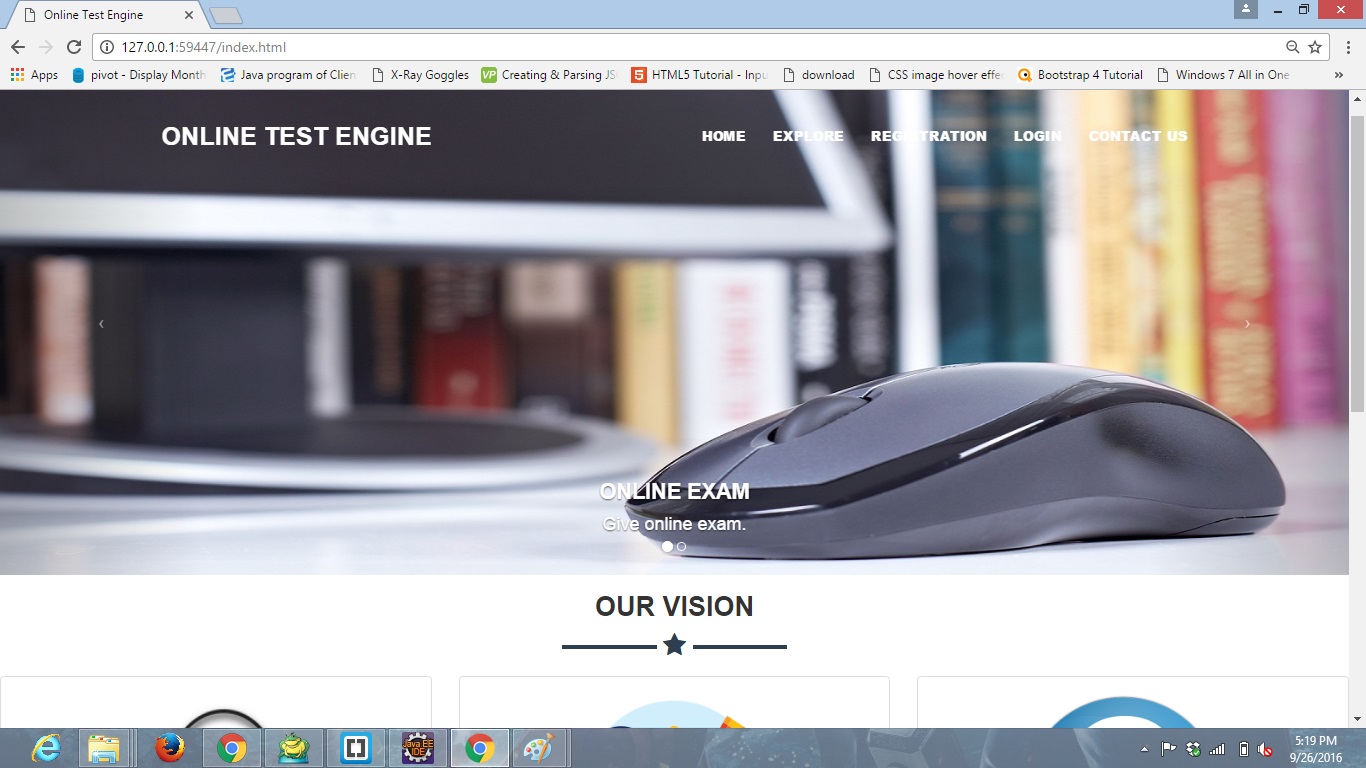
****

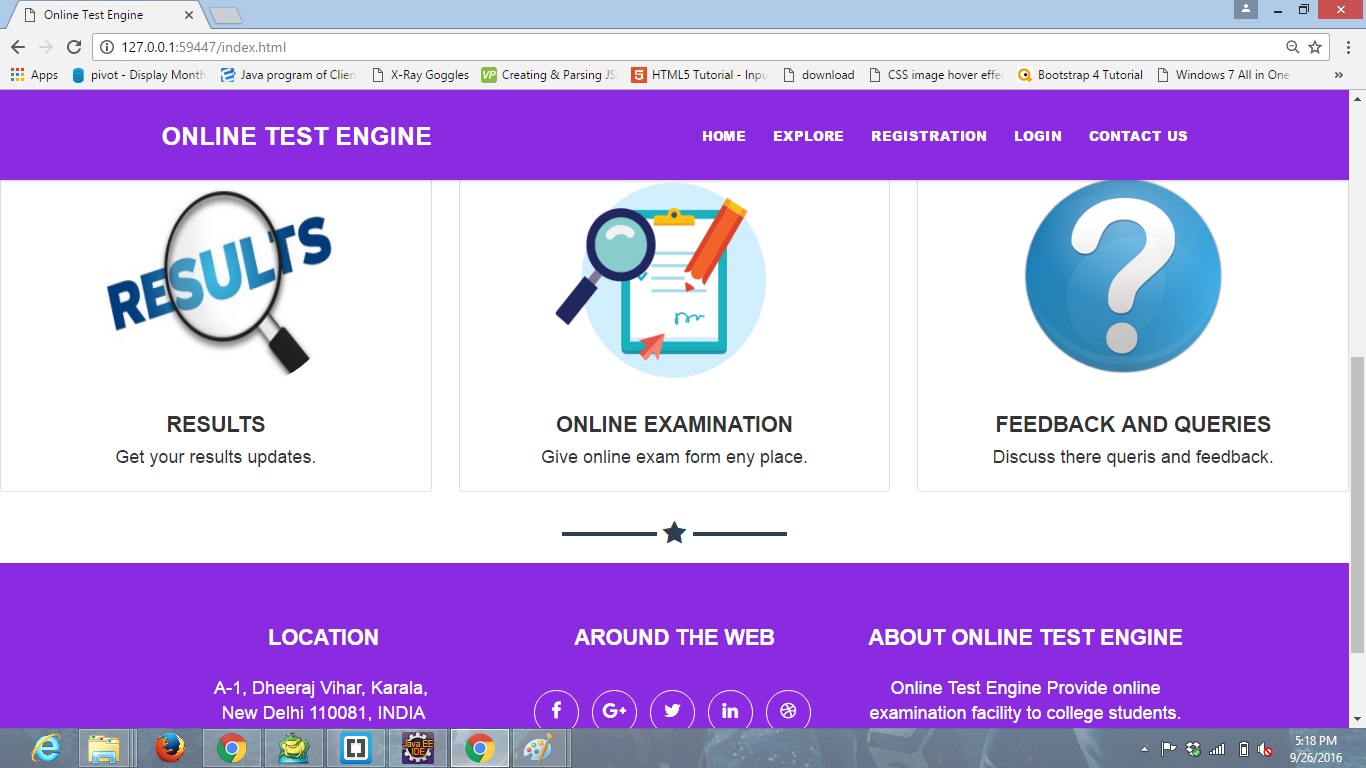
1. **Course Table**

****

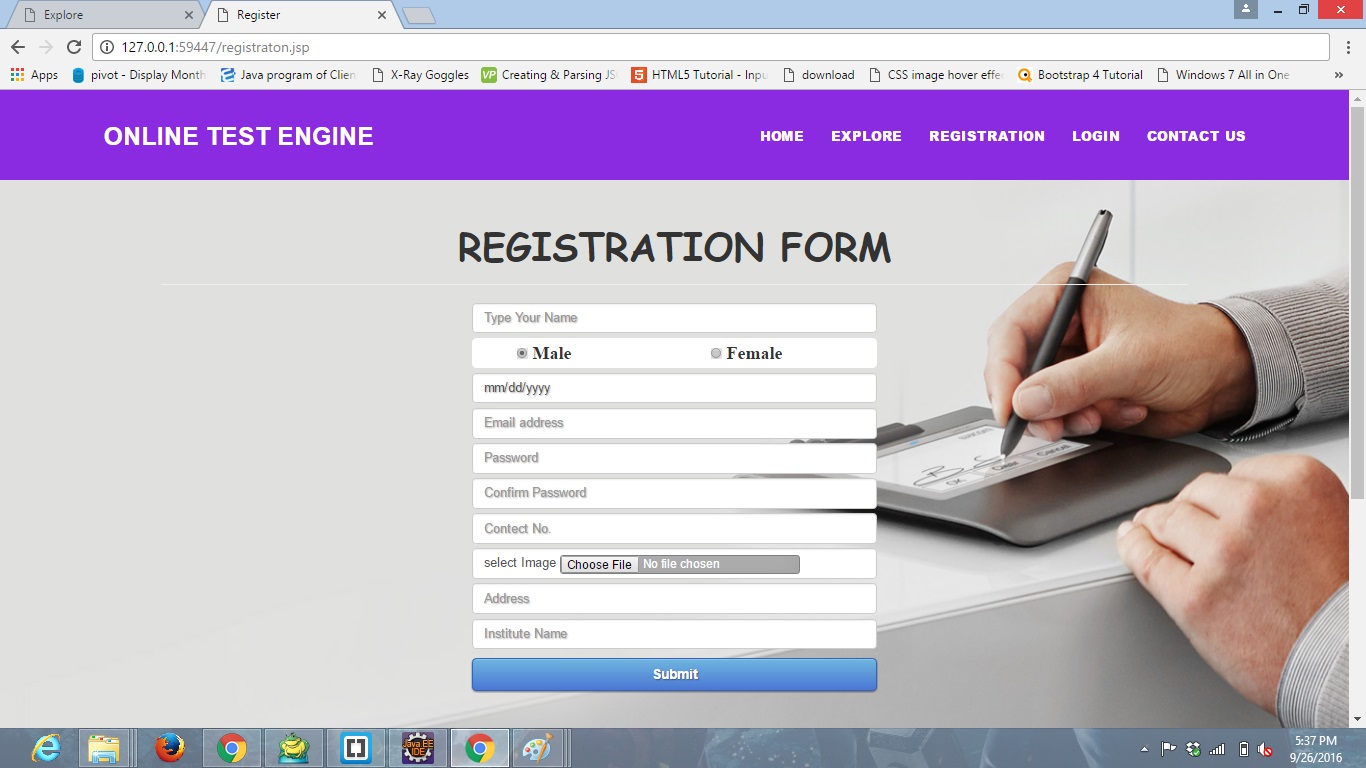
**User Interface Design**

**Index.jsp**

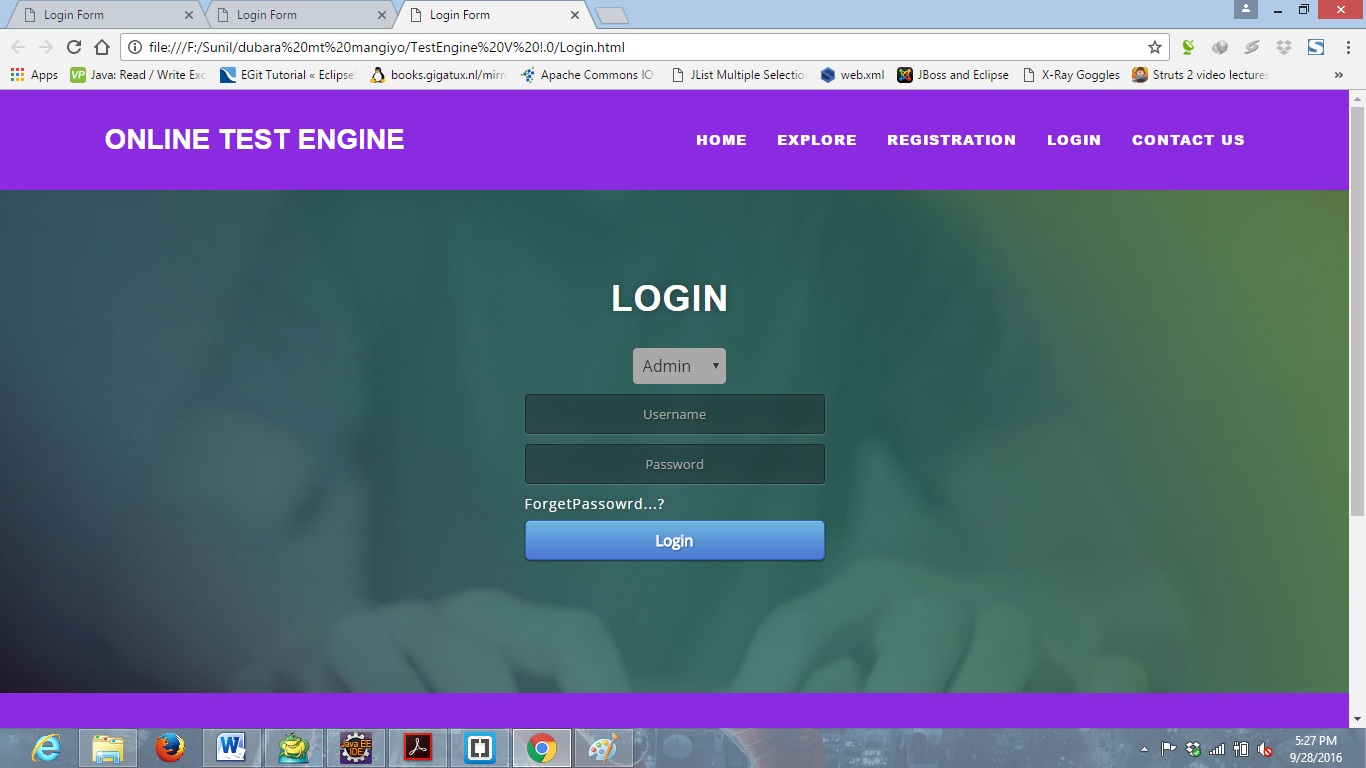
****

****

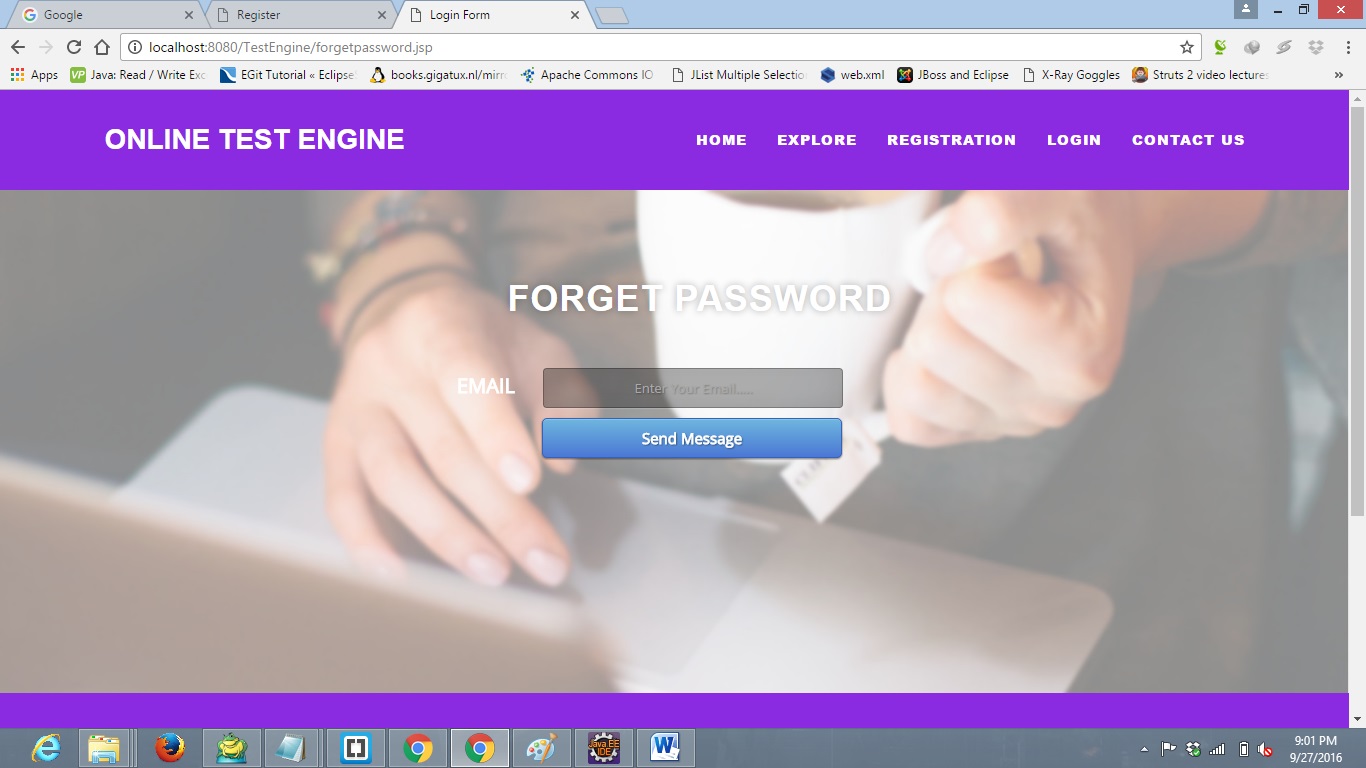
**Register.jsp**

****

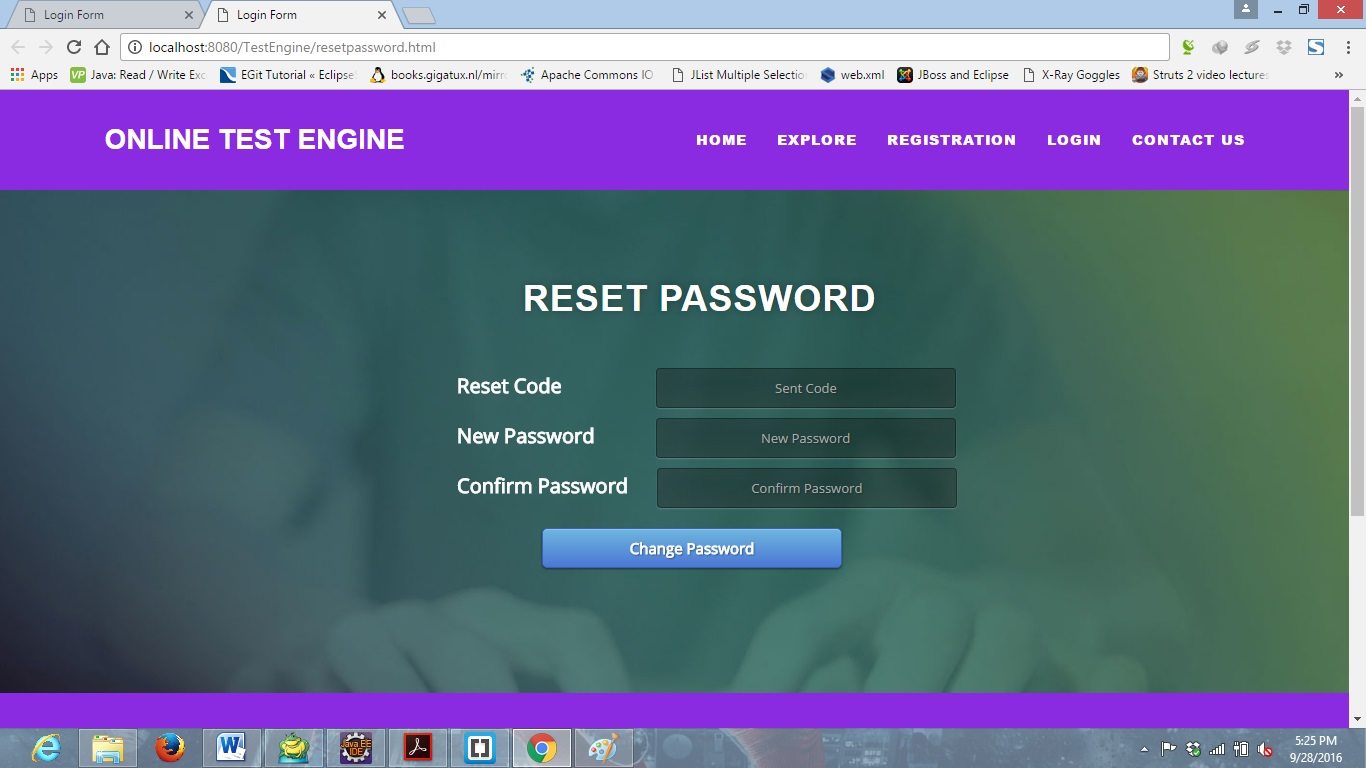
**Login.jsp**

****

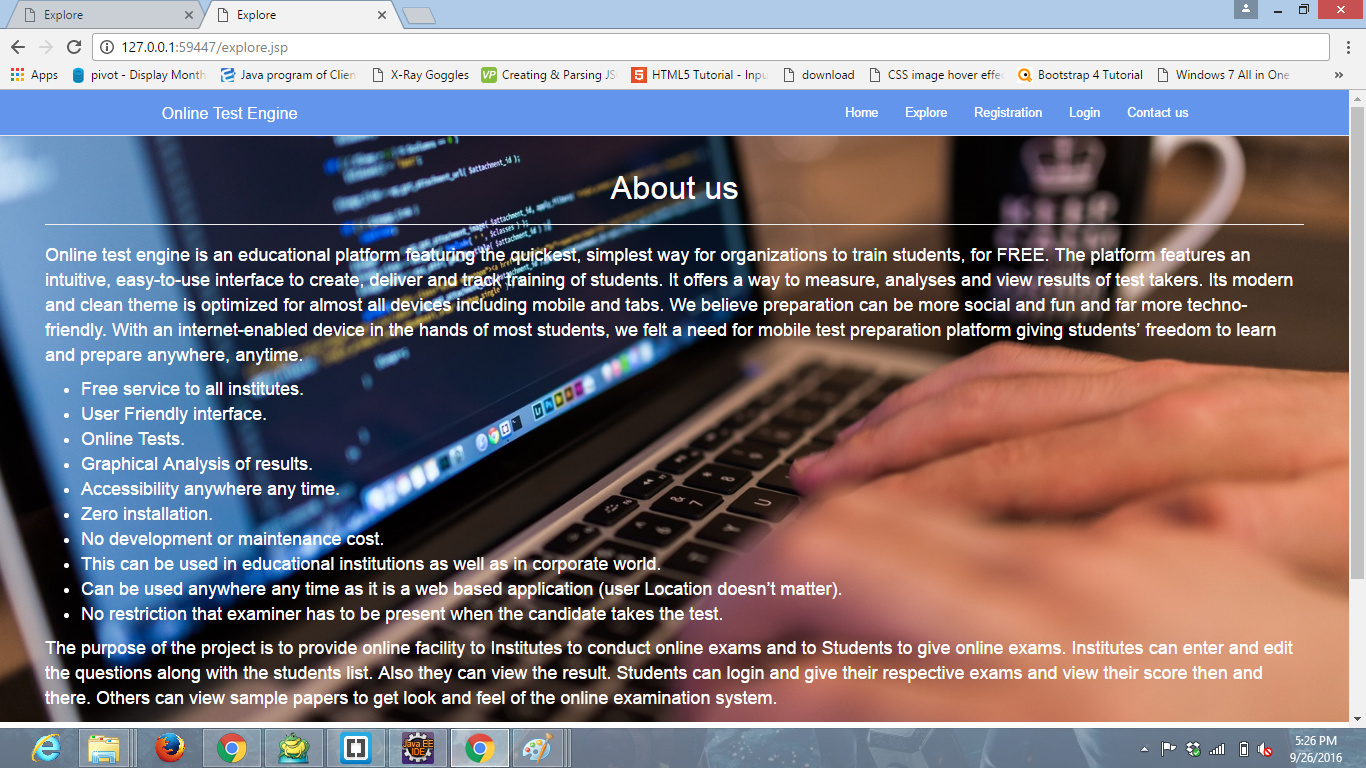
**Forgetpassword.jsp**

****

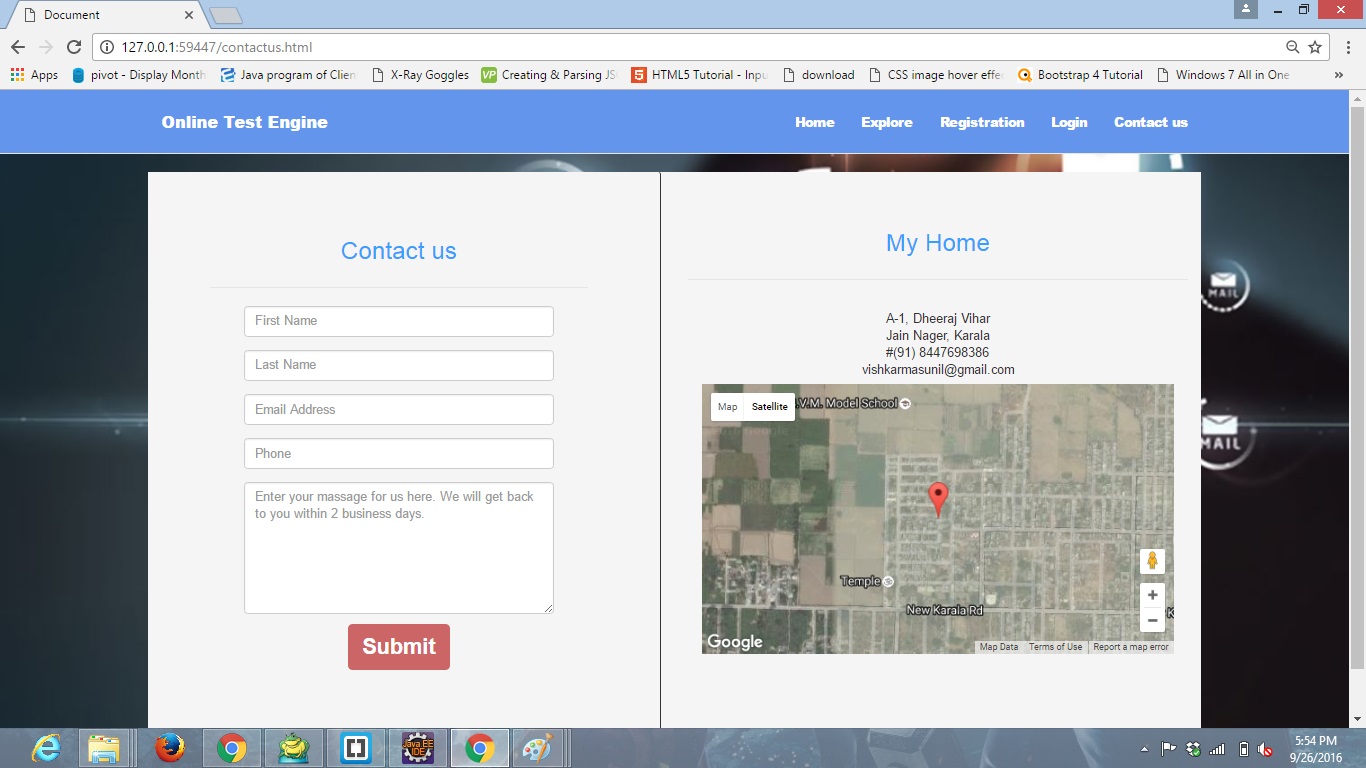
**Resetpassword.jsp**

****

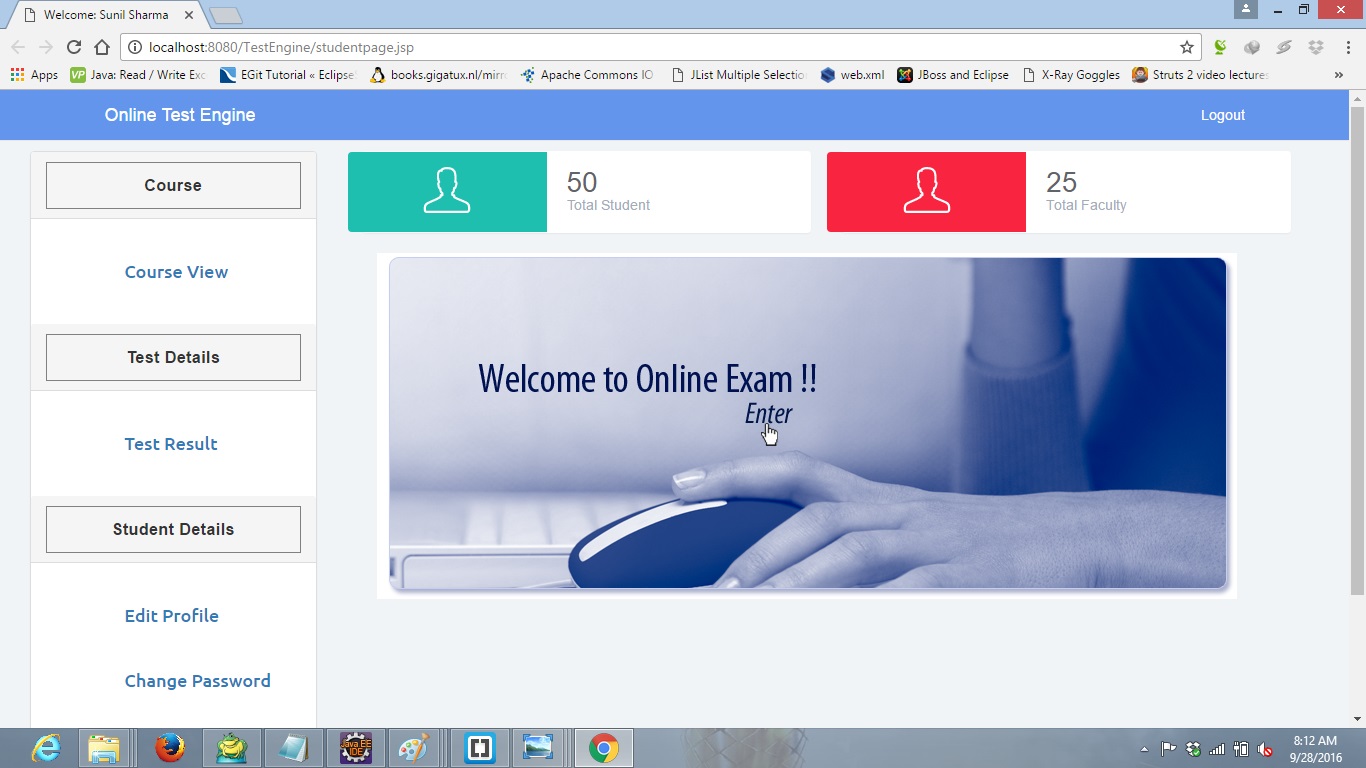
**Explore.jsp**

****

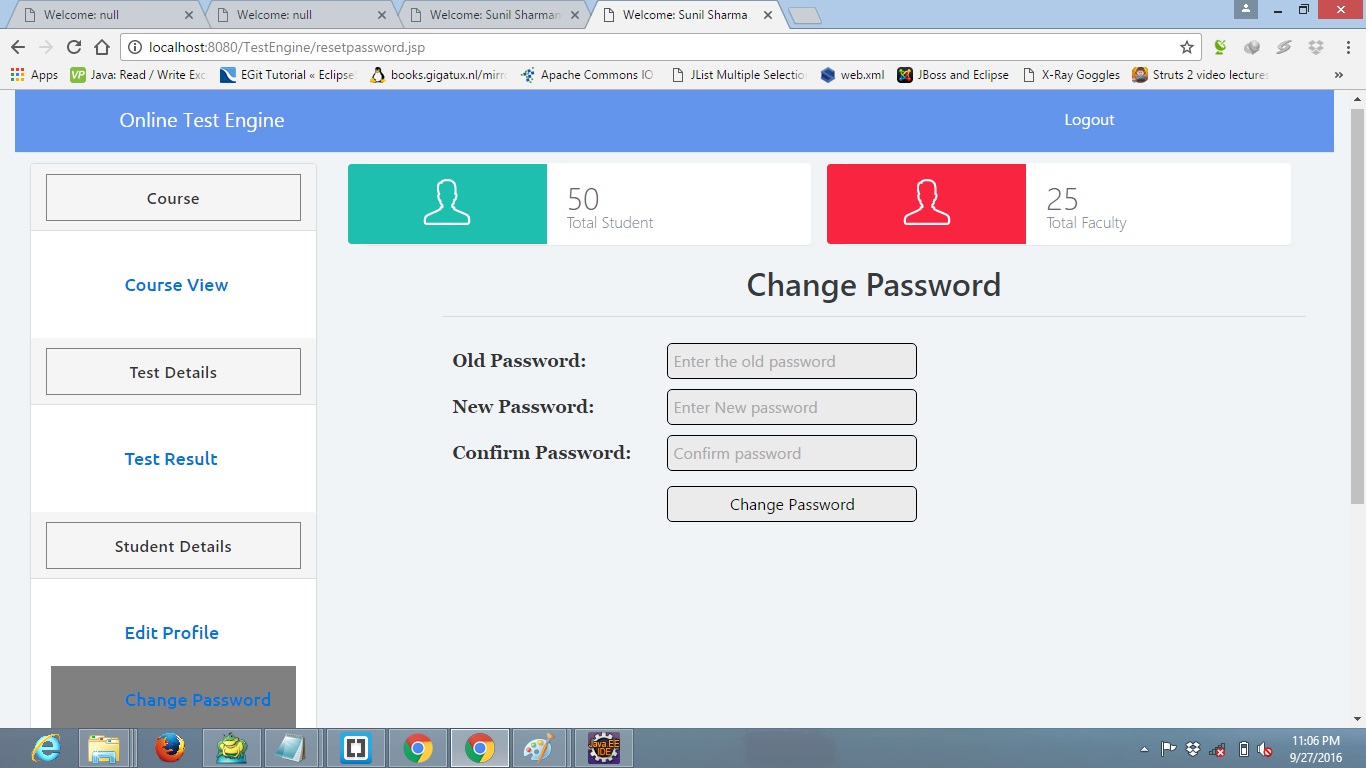
**Contactus.jsp**

****

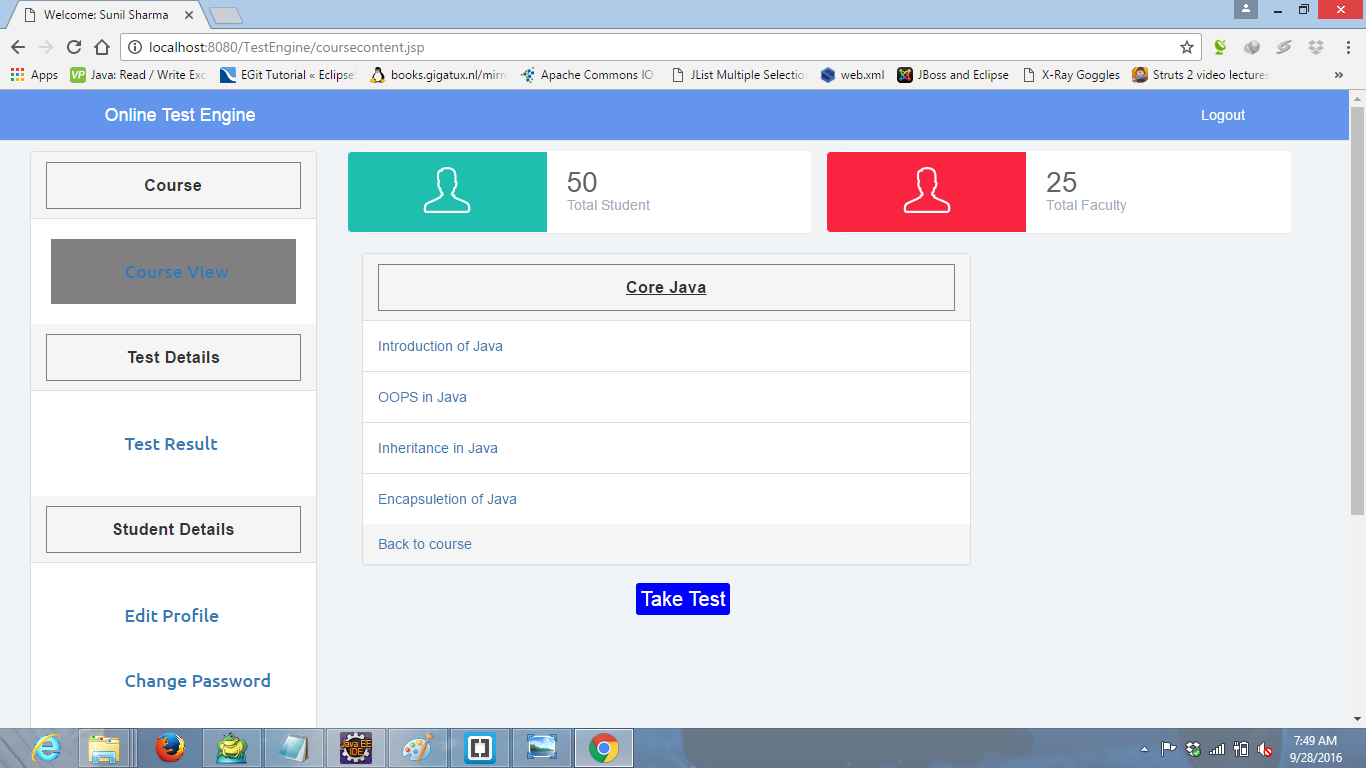
**Studentpage.jsp**

****

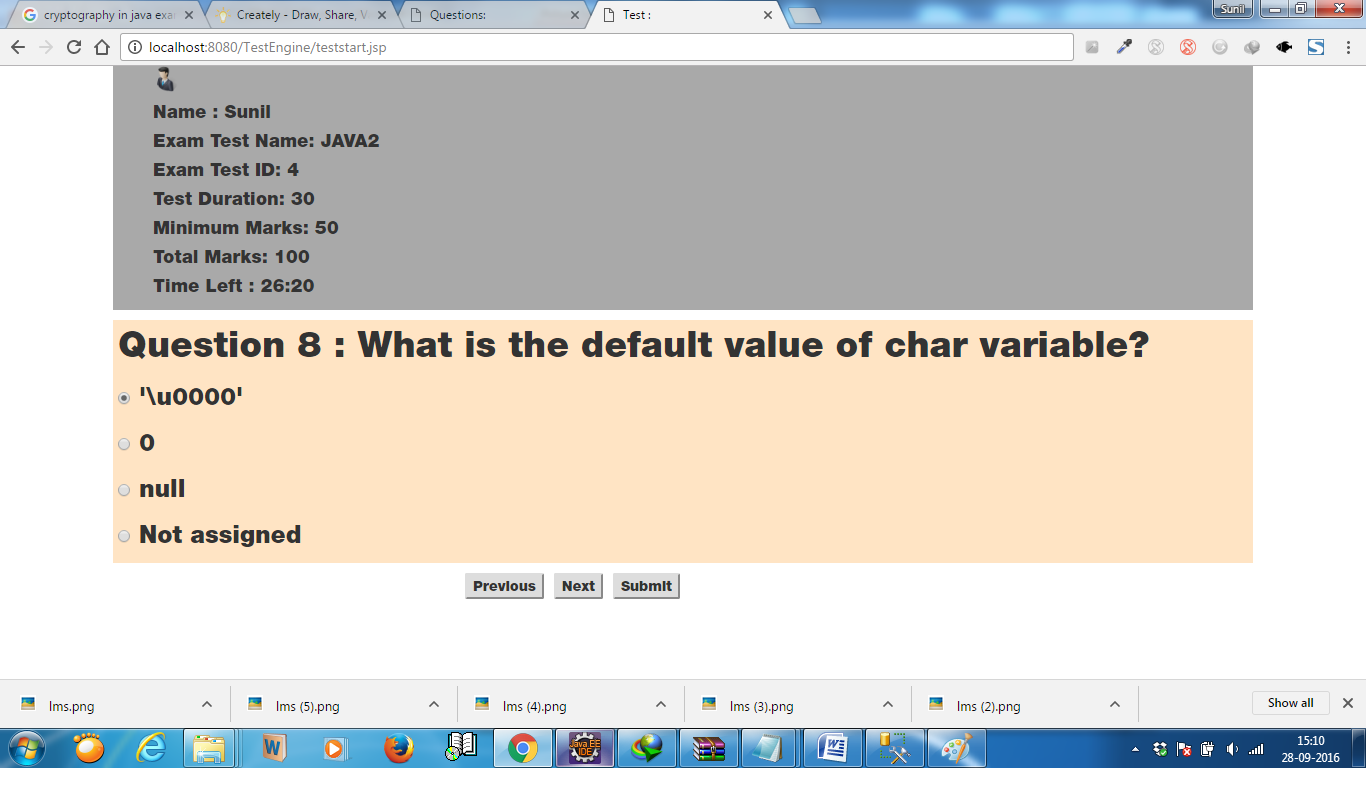
**resetpassword.jsp**

****

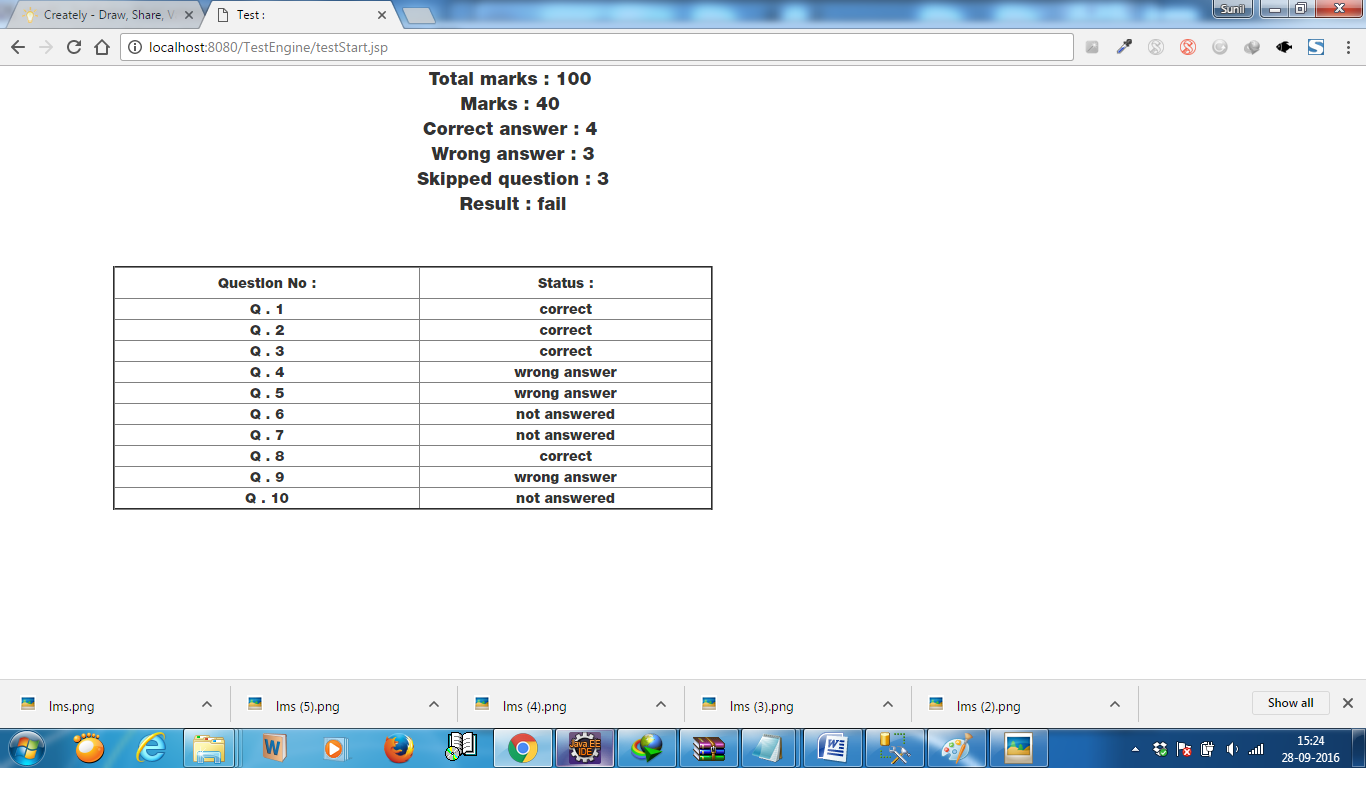
**Studentcousecontent.jsp**

****

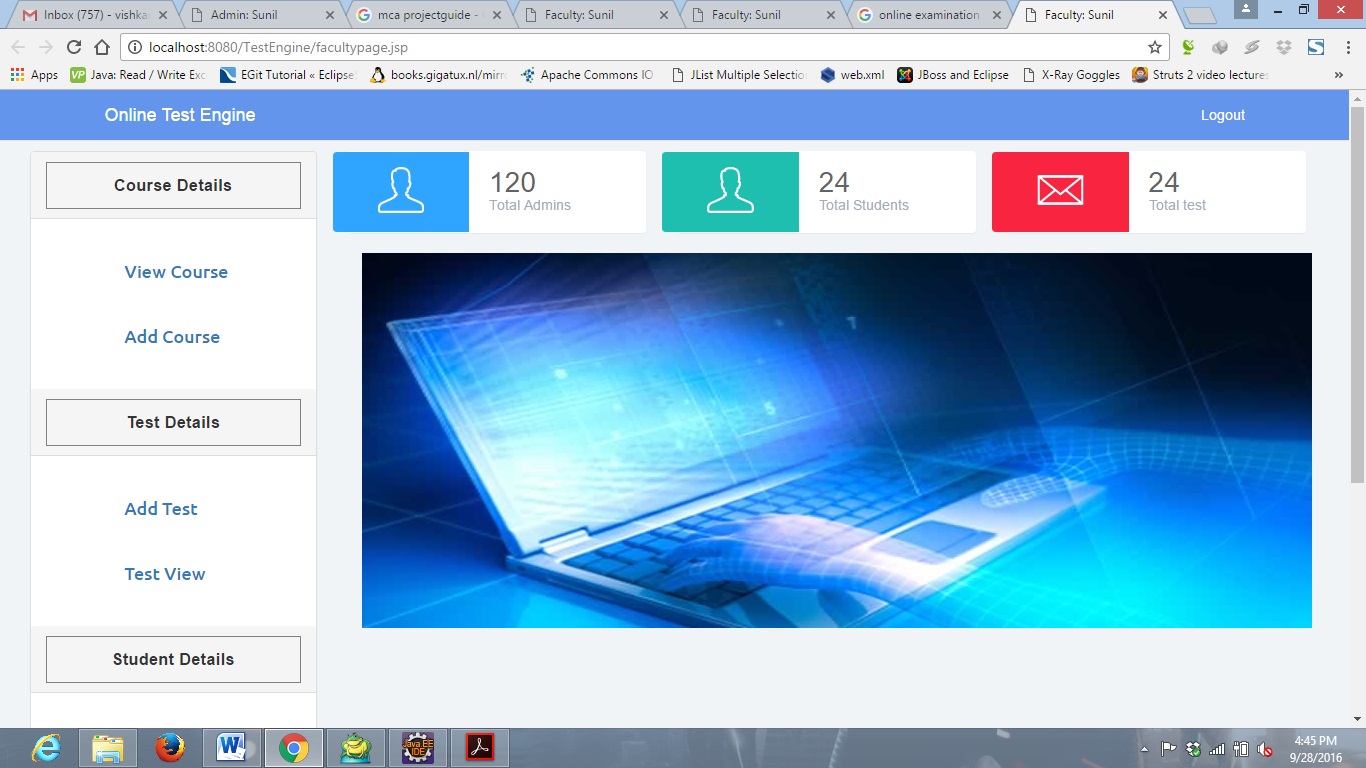
**Teststart.jsp**

****

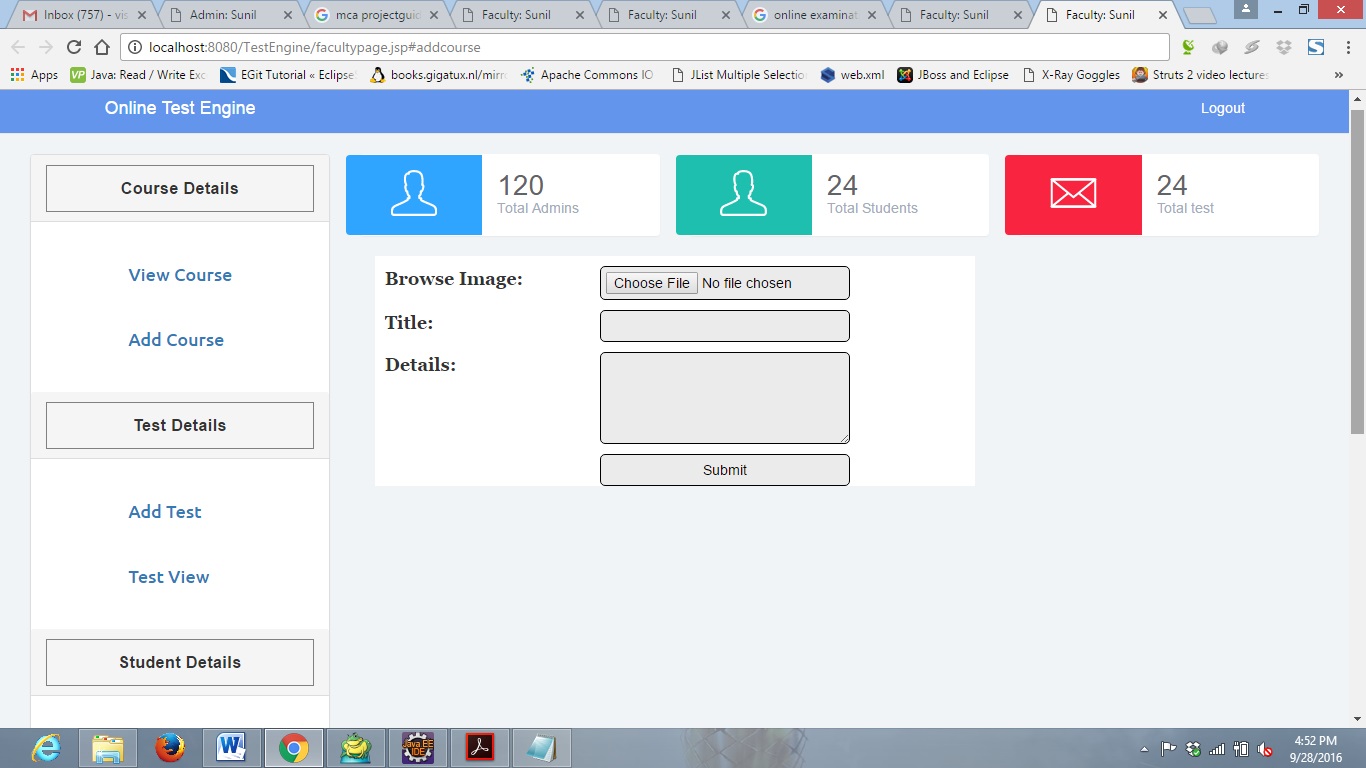
**Result report**

****

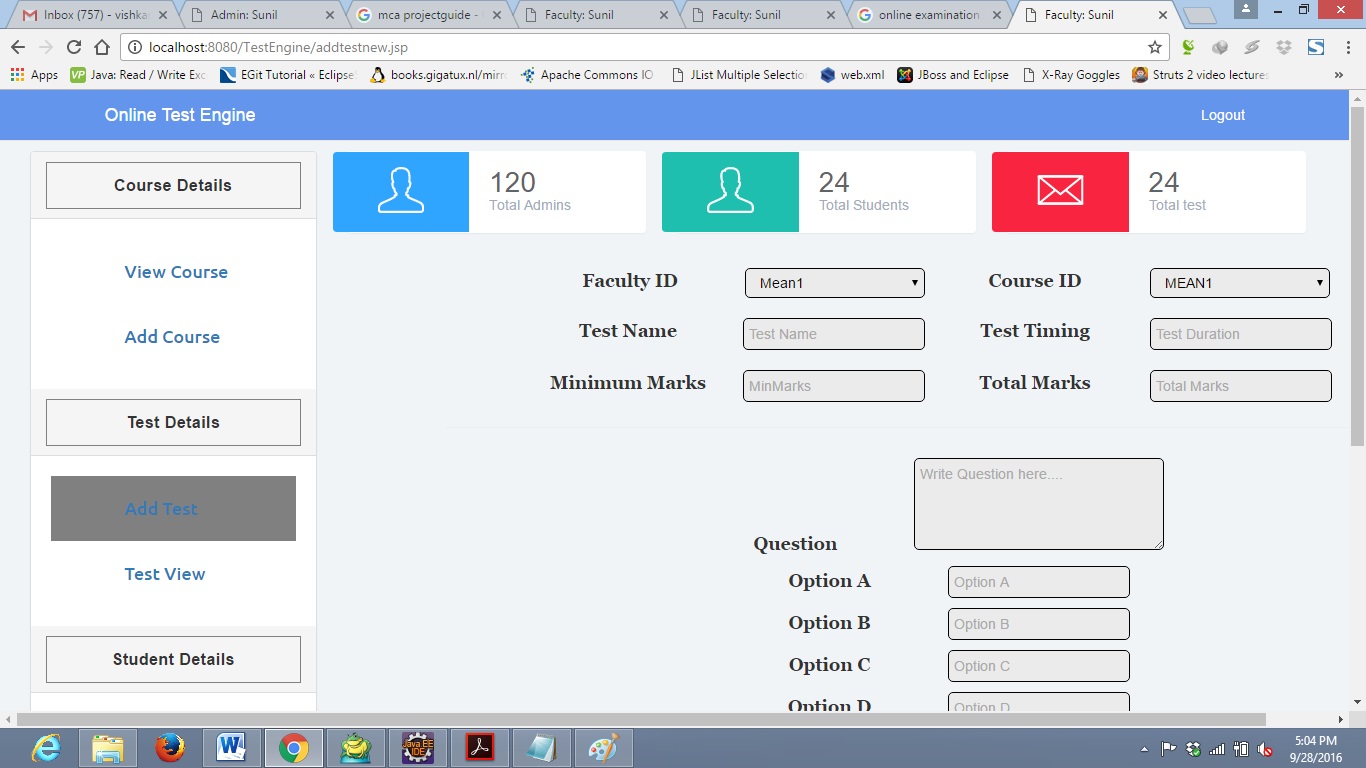
**Facultypage.jsp**

****

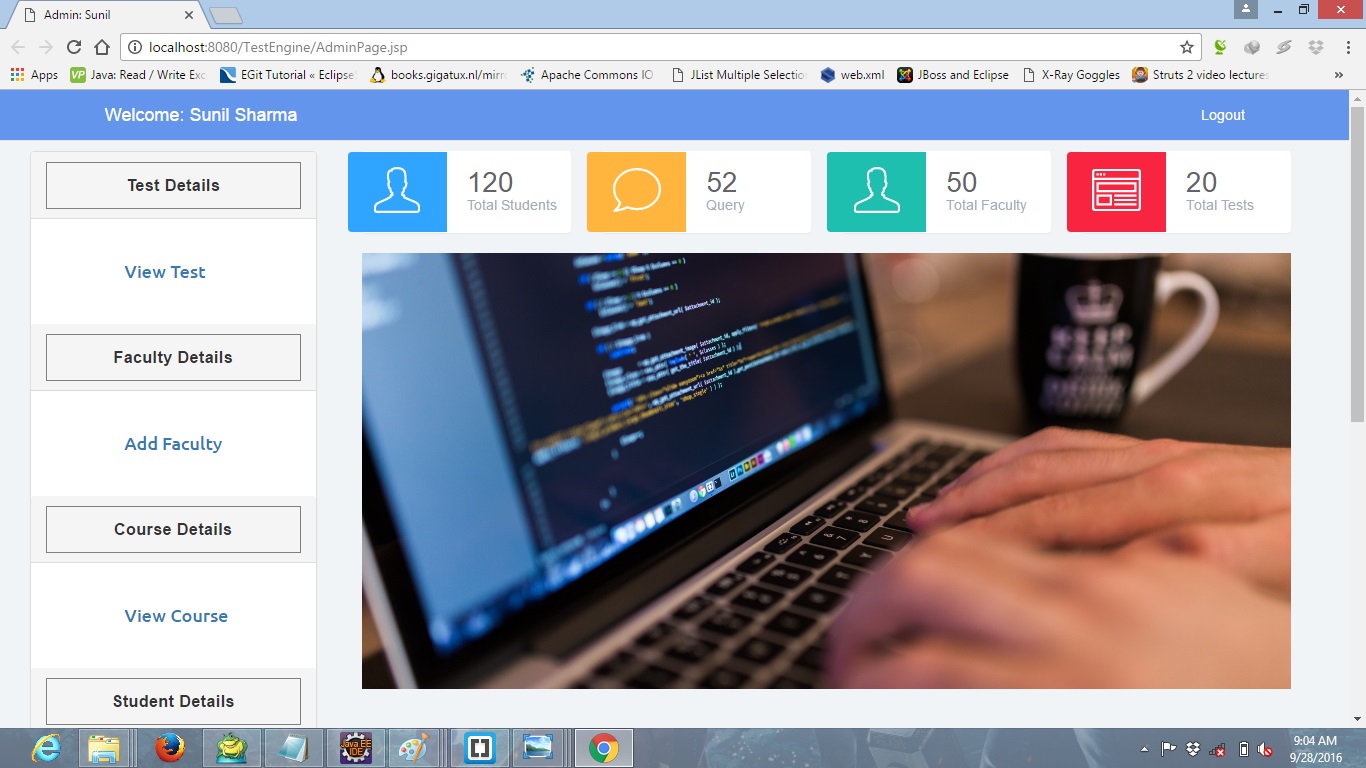
**Addcourse.jsp**

****

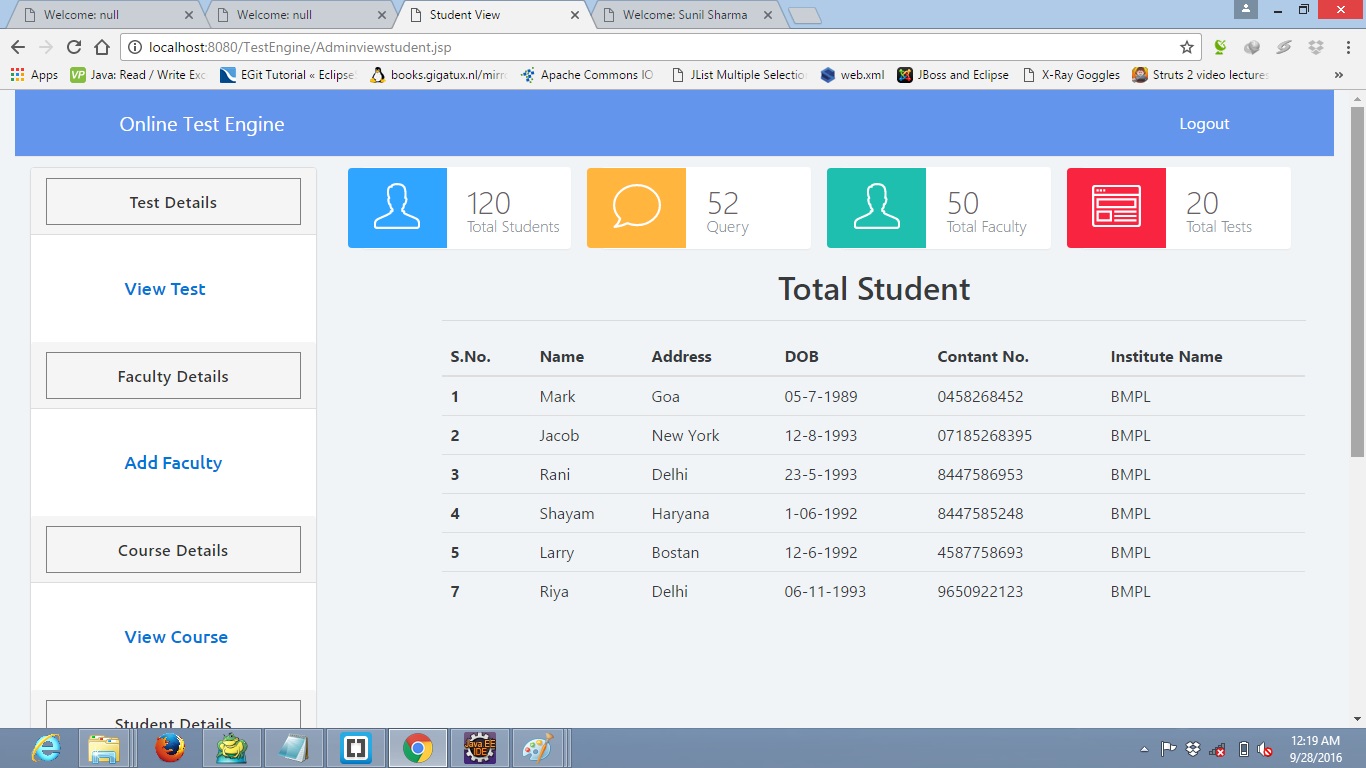
**Addtest.jsp**

****

**AdminPage.jsp**

****

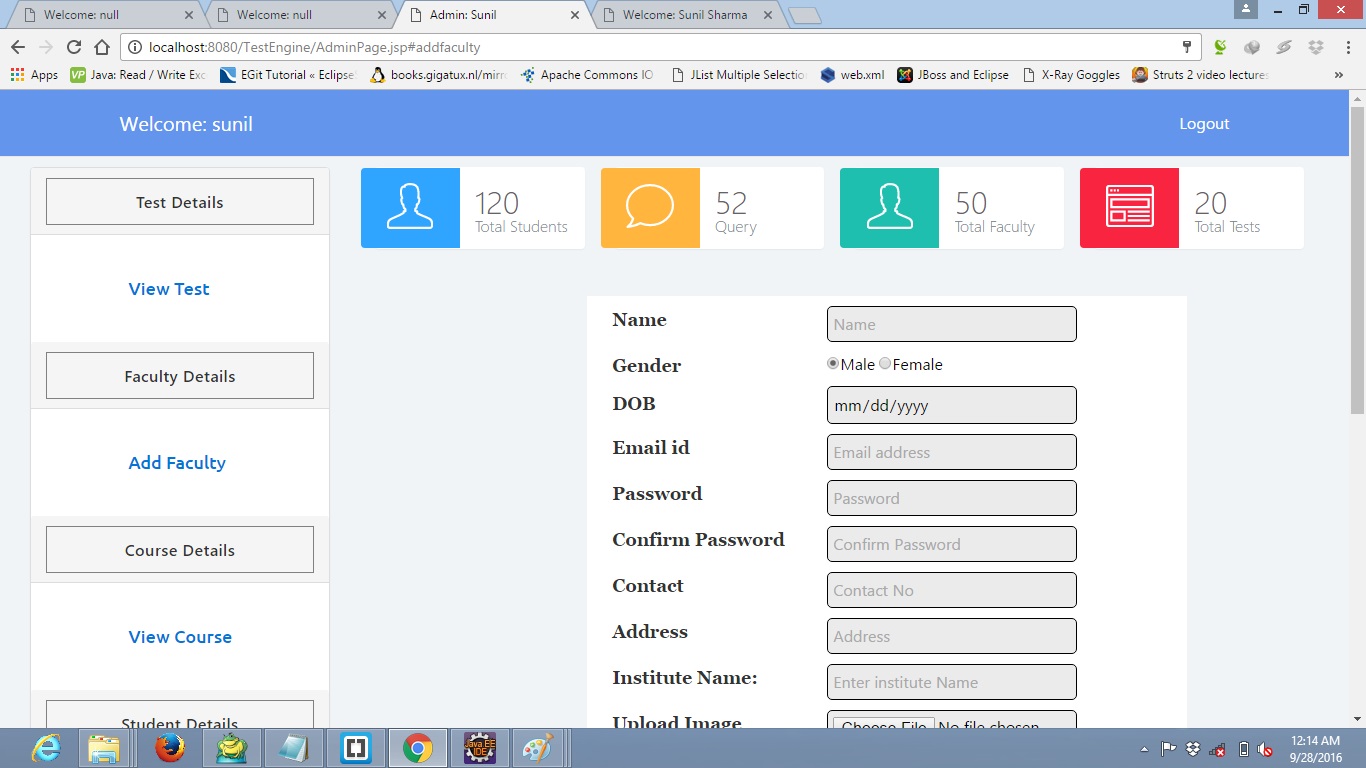
**Studentview.jsp**

****

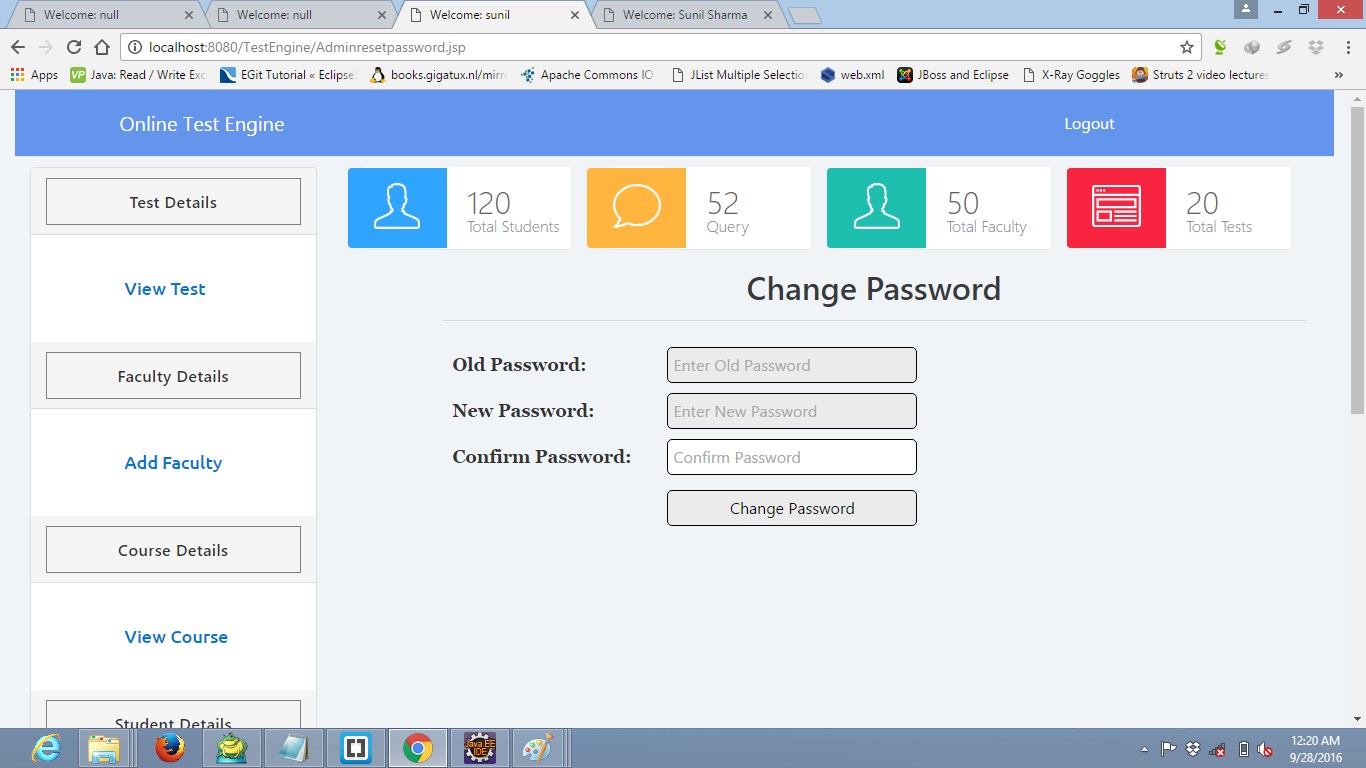
**Adminviewtest.jsp**

****

**AdminAddfaculty.jsp**

****

**Adminchangepassword.jsp**

****

**CHAPTER-4**

**CODING**

**Database Coding**

**create DATABASE lmsdb;**

**use lmsdb;**

**create table Roles(**

**roleId int AUTO\_INCREMENT primary key,**

**roleName Longtext NOT NULL**

**);**

**insert into Roles(roleName)values('Admin');**

**insert into Roles(roleName)values('Faculty');**

**insert into Roles(roleName)values('Student');**

**CREATE TABLE UserLogin(**

**`LoginId` int AUTO\_INCREMENT primary key,**

**roleId int not null,**

**`Username` longtext not null,**

**`Password` longtext NOT NULL,**

**`Email` longtext NOT NULL,**

**CONSTRAINT ULLIPK FOREIGN KEY (roleId)**

**REFERENCES Roles(roleId)**

**);**

**insert into UserLogin(roleId,Username,Password,Email)values(1,'sunil','sunil','sunil@gmail.com');**

**create table UserDetails(**

**`UserId` int AUTO\_INCREMENT primary key,**

**loginId int not null,**

**name longtext not null,**

**Gender longtext not null,**

**dob date not null,**

**address longtext not null,**

**phone longtext not null,**

**ImagePath longtext not null,**

**InstituteName longtext not null,**

**Register\_Date date not null,**

**CONSTRAINT UDUIPK FOREIGN KEY (loginId)**

**REFERENCES UserLogin(loginId)**

**);**

**DELIMITER //**

**CREATE PROCEDURE UserLogin\_Insert (**

**p\_Username LONGTEXT,**

**p\_roleId int,**

**p\_Password LONGTEXT,**

**p\_Email LONGTEXT,**

**p\_name longtext,**

**p\_Gender longtext,**

**p\_dob date,**

**p\_address longtext,**

**p\_phone longtext,**

**p\_ImagePath longtext,**

**p\_InstituteName longtext)**

**BEGIN**

**declare v\_userid int;**

**IF EXISTS(SELECT LoginId FROM UserLogin WHERE Username = p\_Username)**

**THEN**

**SELECT -1; -- Username exists.**

**ELSE IF EXISTS(SELECT LoginId FROM UserLogin WHERE Email = p\_Email)**

**THEN**

**SELECT -2; -- Email exists.**

**ELSE**

**INSERT INTO UserLogin**

**(roleid**

**,Username**

**,Password**

**,Email**

**)**

**VALUES**

**(p\_roleId**

**,p\_Username**

**,p\_Password**

**,p\_Email**

**);**

**-- @userid=select SCOPE\_IDENTITY() -- UserId**

**set v\_userid = LAST\_INSERT\_ID();**

**Insert into userdetails(**

**userId ,**

**name ,**

**Gender ,**

**dob ,**

**address ,**

**phone ,**

**ImagePath ,**

**InstituteName ,**

**Register\_Date**

**)**

**values(**

**v\_userid ,**

**p\_name ,**

**p\_Gender ,**

**p\_dob ,**

**p\_address ,**

**p\_phone ,**

**p\_ImagePath ,**

**p\_InstituteName ,**

**NOW()**

**);**

**SELECT LAST\_INSERT\_ID(); -- UserId**

**END IF;**

**END IF;**

**END;**

**//**

**DELIMITER ;**

**DELIMITER //**

**CREATE PROCEDURE UserLogin\_Validate (**

**p\_roleId int,**

**p\_Username LONGTEXT,**

**p\_Password LONGTEXT)**

**BEGIN**

**DECLARE v\_UserId INT;**

**SELECT loginid INTO v\_UserId**

**FROM UserLogin WHERE Username = p\_Username AND Password = p\_Password and roleId=p\_roleId ;**

**IF v\_UserId IS NOT NULL**

**THEN**

**SELECT '1'; -- User Valid**

**ELSE**

**SELECT '-1'; -- invalid user.**

**END IF;**

**END;**

**//**

**DELIMITER ;**

**create table Course(**

**CourseId int AUTO\_INCREMENT primary key,**

**ImagePath longtext,**

**Title longtext not null,**

**Details longtext,**

**Register\_Date date**

**);**

**DELIMITER //**

**CREATE PROCEDURE Course\_CRUD (**

**p\_Action VARCHAR(10),**

**p\_CourseId int,**

**p\_ImagePath longtext,**

**p\_Title longtext,**

**p\_Details longtext)**

**BEGIN**

**-- SELECT**

**IF p\_Action = 'SELECT'**

**THEN**

**select \* from Course;**

**END IF;**

**-- INSERT**

**IF p\_Action = 'INSERT'**

**THEN**

**INSERT INTO Course**

**(CourseId,**

**ImagePath ,**

**Title ,**

**Details ,**

**Register\_Date)**

**VALUES(p\_CourseId,**

**p\_ImagePath ,**

**p\_Title ,**

**p\_Details ,NOW());**

**SELECT LAST\_INSERT\_ID(); -- UserId**

**END IF;**

**-- UPDATE**

**IF p\_Action = 'UPDATE'**

**THEN**

**UPDATE Course**

**SET**

**ImagePath=p\_ImagePath,**

**Title=p\_Title ,**

**Details=p\_Details**

**WHERE CourseId = p\_CourseId;**

**END IF;**

**-- DELETE**

**IF p\_Action = 'DELETE'**

**THEN**

**DELETE FROM Course**

**WHERE CourseId = p\_CourseId;**

**END IF;**

**END;**

**//**

**DELIMITER ;**

**create table Content(**

**ContentId int AUTO\_INCREMENT primary key,**

**CourseId int not null,**

**Title longtext not null,**

**Details longtext not null,**

**CONSTRAINT COCIPK FOREIGN KEY (CourseId)**

**REFERENCES Course(CourseId)**

**);**

**DELIMITER //**

**CREATE PROCEDURE Content\_CRUD (**

**p\_Action VARCHAR(10),**

**p\_ContentId int,**

**p\_CourseId int,**

**p\_Title longtext,**

**p\_Details longtext)**

**BEGIN**

**-- SELECT**

**IF p\_Action = 'SELECT'**

**THEN**

**select \* from Content;**

**END IF;**

**-- INSERT**

**IF p\_Action = 'INSERT'**

**THEN**

**INSERT INTO Content**

**(CourseId,**

**Title,**

**Details**

**)**

**VALUES(**

**p\_CourseId,**

**p\_Title,**

**p\_Details);**

**SELECT LAST\_INSERT\_ID(); -- UserId**

**END IF;**

**-- UPDATE**

**IF p\_Action = 'UPDATE'**

**THEN**

**UPDATE Content**

**SET**

**Title=p\_Title,**

**Details=p\_Details**

**WHERE ContentId = p\_ContentId;**

**END IF;**

**-- DELETE**

**IF p\_Action = 'DELETE'**

**THEN**

**DELETE FROM Content**

**WHERE ContentId = p\_ContentId;**

**END IF;**

**END;**

**//**

**DELIMITER ;**

**create table ContentDataType(**

**ContentTypeId int AUTO\_INCREMENT primary key,**

**ContentType longtext**

**);**

**DELIMITER //**

**CREATE PROCEDURE ContentDataType\_CRUD (**

**p\_Action VARCHAR(10),**

**p\_ContentTypeId int,**

**p\_ContentType longtext)**

**BEGIN**

**-- SELECT**

**IF p\_Action = 'SELECT'**

**THEN**

**select \* from ContentDataType;**

**END IF;**

**-- INSERT**

**IF p\_Action = 'INSERT'**

**THEN**

**INSERT INTO ContentDataType**

**(ContentType)**

**VALUES(p\_ContentType);**

**SELECT LAST\_INSERT\_ID(); -- UserId**

**END IF;**

**-- UPDATE**

**IF p\_Action = 'UPDATE'**

**THEN**

**UPDATE ContentDataType**

**SET ContentType=p\_ContentType WHERE ContentTypeId = p\_ContentTypeId;**

**END IF;**

**-- DELETE**

**IF p\_Action = 'DELETE'**

**THEN**

**DELETE FROM ContentDataType**

**WHERE ContentTypeId = p\_ContentTypeId;**

**END IF;**

**END;**

**//**

**DELIMITER ;**

**create table ContentData(**

**ContentDataId int AUTO\_INCREMENT primary key,**

**ContentId int,**

**ContentTypeId int,**

**DataPath longtext,**

**foreign key (ContentId) references Content(ContentId),**

**foreign key (ContentTypeId) references ContentDataType(ContentTypeId)**

**)**

**DELIMITER //**

**CREATE PROCEDURE ContentData\_CRUD (**

**p\_Action VARCHAR(10),**

**p\_ContentDataId int,**

**p\_ContentId int,**

**p\_ContentTypeId int,**

**p\_DataPath longtext)**

**BEGIN**

**-- SELECT**

**IF p\_Action = 'SELECT'**

**THEN**

**select \* from ContentData;**

**END IF;**

**-- INSERT**

**IF p\_Action = 'INSERT'**

**THEN**

**INSERT INTO ContentData**

**(ContentId,**

**ContentTypeId,**

**DataPath**

**)**

**VALUES(p\_ContentId,**

**p\_ContentTypeId,**

**p\_DataPath);**

**SELECT LAST\_INSERT\_ID(); -- UserId**

**END IF;**

**-- UPDATE**

**IF p\_Action = 'UPDATE'**

**THEN**

**UPDATE ContentData**

**SET**

**DataPath=DataPath**

**WHERE ContentDataId = p\_ContentDataId;**

**END IF;**

**-- DELETE**

**IF p\_Action = 'DELETE'**

**THEN**

**DELETE FROM ContentData**

**WHERE ContentDataId = p\_ContentDataId;**

**END IF;**

**END;**

**//**

**DELIMITER ;**

**create table Tests(**

**TestId int AUTO\_INCREMENT primary key,**

**FacultyId int ,**

**CourseId int,**

**TestName longtext not null,**

**testDuration int,**

**MinMarks int,**

**TotalMarks int,**

**Register\_Date date,**

**foreign key (FacultyId) references UserDetails(UserId),**

**foreign key (CourseId) references Course(CourseId)**

**)**

**DELIMITER //**

**CREATE PROCEDURE Tests\_CRUD (**

**p\_Action VARCHAR(10),**

**p\_TestId int,**

**p\_FacultyId int,**

**p\_CourseId int,**

**p\_TestName longtext,**

**p\_testDuration int,**

**p\_MinMarks int,**

**p\_TotalMarks int)**

**BEGIN**

**-- SELECT**

**IF p\_Action = 'SELECT'**

**THEN**

**select \* from Tests;**

**END IF;**

**-- INSERT**

**IF p\_Action = 'INSERT'**

**THEN**

**INSERT INTO Tests**

**(FacultyId,**

**CourseId,**

**TestName,**

**testDuration,**

**MinMarks,**

**TotalMarks,**

**Register\_Date**

**)**

**VALUES(**

**p\_FacultyId,**

**p\_CourseId,**

**p\_TestName,**

**p\_testDuration,**

**p\_MinMarks,**

**p\_TotalMarks,**

**now()**

**);**

**SELECT LAST\_INSERT\_ID(); -- UserId**

**END IF;**

**-- UPDATE**

**IF p\_Action = 'UPDATE'**

**THEN**

**UPDATE Tests**

**SET**

**TestName=p\_TestName,**

**testDuration=p\_testDuration,**

**MinMarks=p\_MinMarks,**

**TotalMarks=p\_TotalMarks**

**WHERE TestId = p\_TestId ;**

**END IF;**

**-- DELETE**

**IF p\_Action = 'DELETE'**

**THEN**

**DELETE FROM Tests**

**WHERE TestId = p\_TestId;**

**END IF;**

**END;**

**//**

**DELIMITER ;**

**create table TestQuestions(**

**QuestionId int AUTO\_INCREMENT primary key,**

**TestId int ,**

**Question longtext,**

**OptionA longtext,**

**OptionB longtext,**

**OptionC longtext,**

**OptionD longtext,**

**Answer longtext,**

**foreign key (TestId) references tests(testid)**

**);**

**DELIMITER //**

**CREATE PROCEDURE TestQuestions\_CRUD (**

**p\_Action VARCHAR(10),**

**p\_QuestionId int,**

**p\_TestId int,**

**p\_Question longtext ,**

**p\_OptionA longtext ,**

**p\_OptionB longtext ,**

**p\_OptionC longtext ,**

**p\_OptionD longtext ,**

**p\_Answer longtext)**

**BEGIN**

**-- SELECT**

**IF p\_Action = 'SELECT'**

**THEN**

**select \* from TestQuestions;**

**END IF;**

**-- INSERT**

**IF p\_Action = 'INSERT'**

**THEN**

**INSERT INTO TestQuestions**

**(TestId ,**

**Question ,**

**OptionA ,**

**OptionB ,**

**OptionC ,**

**OptionD ,**

**Answer**

**)**

**VALUES(**

**p\_TestId ,**

**p\_Question ,**

**p\_OptionA ,**

**p\_OptionB ,**

**p\_OptionC ,**

**p\_OptionD ,**

**p\_Answer**

**);**

**SELECT LAST\_INSERT\_ID(); -- UserId**

**END IF;**

**-- UPDATE**

**IF p\_Action = 'UPDATE'**

**THEN**

**UPDATE TestQuestions**

**SET**

**Question=p\_Question ,**

**OptionA =p\_OptionA ,**

**OptionB =p\_OptionB ,**

**OptionC=p\_OptionC ,**

**OptionD=p\_OptionD ,**

**Answer =p\_Answer**

**where**

**QuestionId = p\_QuestionId;**

**END IF;**

**-- DELETE**

**IF p\_Action = 'DELETE'**

**THEN**

**DELETE FROM TestQuestions**

**WHERE QuestionId = p\_QuestionId;**

**END IF;**

**END;**

**//**

**DELIMITER ;**

**create table UserTestResults(**

**ResultId int AUTO\_INCREMENT primary key,**

**UserId int ,**

**TestId int,**

**marks int,**

**test\_Date date,**

**foreign key (UserId) references UserDetails(UserId),**

**foreign key (TestId) references tests(testid)**

**);**

**DELIMITER //**

**CREATE PROCEDURE UserTestResults\_CRUD (**

**p\_Action VARCHAR(10),**

**p\_ResultId int,**

**p\_UserId int,**

**p\_TestId int,**

**p\_marks int)**

**BEGIN**

**-- SELECT**

**IF p\_Action = 'SELECT'**

**THEN**

**select \* from UserTestResults;**

**END IF;**

**-- INSERT**

**IF p\_Action = 'INSERT'**

**THEN**

**INSERT INTO UserTestResults**

**(**

**UserId ,**

**TestId ,**

**marks ,**

**test\_Date**

**)**

**VALUES(**

**p\_UserId ,**

**p\_TestId ,**

**p\_marks ,**

**now()**

**);**

**SELECT LAST\_INSERT\_ID(); -- UserId**

**END IF;**

**-- UPDATE**

**IF p\_Action = 'UPDATE'**

**THEN**

**UPDATE UserTestResults**

**SET**

**UserId =p\_UserId,**

**TestId =p\_TestId,**

**marks=p\_marks**

**where**

**ResultId =p\_ResultId;**

**END IF;**

**-- DELETE**

**IF p\_Action = 'DELETE'**

**THEN**

**DELETE FROM UserTestResults**

**WHERE ResultId =p\_ResultId;**

**END IF;**

**END;**

**//**

**DELIMITER ;**

**Project Coding**

**Index.jsp**

<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*

pageEncoding=*"ISO-8859-1"*%>

<!DOCTYPE>

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>

<meta charset=*"utf-8"*>

<meta http-equiv=*"X-UA-Compatible"* content=*"IE=edge"*>

<meta name=*"viewport"* content=*"width=device-width, initial-scale=1"*>

<meta name=*"description"* content=*""*>

<meta name=*"author"* content=*""*>

<title>Online Test Engine</title>

<link href=*"css/bootstrap.min.css"* rel=*"stylesheet"* type=*"text/css"*>

<link href=*"css/freelancer.min.css"* rel=*"stylesheet"* type=*"text/css"*>

<link href=*"css/font-awesome.min.css"* rel=*"stylesheet"* type=*"text/css"*>

<link rel=*"stylesheet"* href=*"css/animate.css"*>

<link href=*"css/custom.css"* rel=*"stylesheet"* type=*"text/css"*>

<!-- <link href="https://fonts.googleapis.com/css?family=Montserrat:400,700" rel="stylesheet" type="text/css">

<link href="https://fonts.googleapis.com/css?family=Lato:400,700,400italic,700italic" rel="stylesheet" type="text/css"> -->

</head>

<body id=*"page-top"* class=*"index"*>

<nav id=*"mainNav"* class=*"navbar navbar-default navbar-fixed-top navbar-custom colornav color"*>

<div class=*"container"*>

<!-- Brand and toggle get grouped for better mobile display -->

<div class=*"navbar-header page-scroll"*>

<button type=*"button"* class=*"navbar-toggle"* data-toggle=*"collapse"* data-target=*"#bs-example-navbar-collapse-1"*>

<span class=*"sr-only"*>Toggle navigation</span> Menu <i class=*"fa fa-bars"*></i>

</button>

<a class=*"navbar-brand"* href=*"#page-top"*>Online Test Engine</a>

</div>

<!-- Collect the nav links, forms, and other content for toggling -->

<div class=*"collapse navbar-collapse"* id=*"bs-example-navbar-collapse-1"*>

<ul class=*"nav navbar-nav navbar-right"*>

<li class=*"page-scroll"*>

<a href=*"index.jsp"*>Home</a>

</li>

<li class=*"page-scroll"*>

<a href=*"explore.jsp"*>Explore</a>

</li>

<li class=*"page-scroll"*>

<a href=*"register.jsp"*>Registration</a>

</li>

<li class=*"page-scroll"*>

<a href=*"Login.jsp"*>Login</a>

</li>

<li class=*"page-scroll dropdown"*>

<a href=*"contecus.jsp"*>Contact us</a>

</li>

</ul>

</div>

<!-- /.navbar-collapse -->

</div>

<!-- /.container-fluid -->

</nav>

<header id=*"myCarousel"* class=*"carousel slide"*>

<!-- Indicators -->

<ol class=*"carousel-indicators"*>

<li data-target=*"#myCarousel"* data-slide-to=*"0"* class=*"active"*></li>

<li data-target=*"#myCarousel"* data-slide-to=*"1"*></li>

</ol>

<!-- Wrapper for slides -->

<div class=*"carousel-inner"*>

<div class=*"item active"*>

<img src=*"images/banner4.jpeg"* width=*"auto"* height=*"1600"*>

<div class=*"carousel-caption"* class=*"animated rubberBand"*>

<h3>Online Exam</h3>

<p>Give online exam.</p>

</div>

</div>

<div class=*"item"*>

<img src=*"images/banner\_3.jpg"* width=*"auto"* height=*"1600"*>

<div class=*"carousel-caption"* class=*"animated bounceInLeft"*>

<h3>Result</h3>

<p>Get your result easily</p>

</div>

</div>

</div>

<!-- Controls -->

<a class=*"left carousel-control"* href=*"#myCarousel"* data-slide=*"prev"*>

<span class=*"icon-prev"*></span>

</a>

<a class=*"right carousel-control"* href=*"#myCarousel"* data-slide=*"next"*>

<span class=*"icon-next"*></span>

</a>

</header>

<div class=*"row"*>

<div class=*"col-lg-12 text-center"*>

<hr class=*"star-primary"*>

</div>

</div>

<div class=*"thumbnails"*>

<div class=*"row"*>

<div class=*"col-sm-6 col-md-4"*>

<div class=*"thumbnail"*>

<img src=*"images/Results-images.jpg"* alt=*"Result"* class=*"img-circle"*>

<div class=*"caption text-center"*>

<h3>Results</h3>

<p>Get your results updates.</p>

</div>

</div>

</div>

<div class=*"col-sm-6 col-md-4"*>

<div class=*"thumbnail"*>

<img src=*"images/online-test.png"* alt=*"OnlineTest"* class=*"img-circle"*>

<div class=*"caption text-center"*>

<h3>Online Examination</h3>

<p>Give online exam form eny place.</p>

</div>

</div>

</div>

<div class=*"col-sm-6 col-md-4"*>

<div class=*"thumbnail"*>

<img src=*"images/1\_009.png"* alt=*"Feedback"* class=*"img-circle"*>

<div class=*"caption text-center"*>

<h3>Feedback and Queries</h3>

<p>Discuss there queries and feedback.</p>

</div>

</div>

</div>

</div>

</div>

<div class=*"row"*>

<div class=*"col-lg-12 text-center"*>

<hr class=*"star-primary"*>

</div>

</div>

<!-- Footer -->

<footer class=*"text-center id="*contactus">

<div class=*"footer-above"*>

<div class=*"container"*>

<div class=*"row"*>

<div class=*"footer-col col-md-4"*>

<h3>Location</h3>

<p>A-1, Dheeraj Vihar, Karala,

<br>New Delhi 110081, INDIA</p>

</div>

<div class=*"footer-col col-md-4"*>

<h3>Around the Web</h3>

<ul class=*"list-inline"*>

<li>

<a href=*"#"* class=*"btn-social btn-outline"*><i class=*"fa fa-fw fa-facebook"*></i></a>

</li>

<li>

<a href=*"#"* class=*"btn-social btn-outline"*><i class=*"fa fa-fw fa-google-plus"*></i></a>

</li>

<li>

<a href=*"#"* class=*"btn-social btn-outline"*><i class=*"fa fa-fw fa-twitter"*></i></a>

</li>

<li>

<a href=*"#"* class=*"btn-social btn-outline"*><i class=*"fa fa-fw fa-linkedin"*></i></a>

</li>

<li>

<a href=*"#"* class=*"btn-social btn-outline"*><i class=*"fa fa-fw fa-dribbble"*></i></a>

</li>

</ul>

</div>

<div class=*"footer-col col-md-4"*>

<h3>About Online Test Engine</h3>

<p>Online Test Engine Provide online examination facility to College students.</p>

</div>

</div>

</div>

</div>

<div class=*"footer-below"*>

<div class=*"container"*>

<div class=*"row"*>

<div class=*"col-lg-12"*>

Copyright &copy; Your WebSite 2016

</div>

</div>

</div>

</div>

</footer>

</body>

<script type=*"application/javascript"* src=*"js/jquery-1.12.3.min.js"*></script>

<script type=*"application/javascript"* src=*"js/bootstrap.min.js"*></script>

<script type=*"application/javascript"* src=*"js/homepage.js"*></script>

</html>

**login.jsp**

<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*

pageEncoding=*"ISO-8859-1"*%>

<!DOCTYPE html>

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>

<meta charset=*"utf-8"*>

<meta http-equiv=*"X-UA-Compatible"* content=*"IE=edge"*>

<meta name=*"viewport"* content=*"width=device-width, initial-scale=1"*>

<title>Login Form</title>

<link href=*"css/bootstrap.min.css"* rel=*"stylesheet"* type=*"text/css"*>

<link href=*"css/freelancer.min.css"* rel=*"stylesheet"* type=*"text/css"*>

<link href=*"css/font-awesome.min.css"* rel=*"stylesheet"* type=*"text/css"*>

<link rel=*"stylesheet"* href=*"css/custom.css"*>

<script type=*"application/javascript"* src=*"js/jquery-1.12.3.min.js"*></script>

<script type=*"application/javascript"* src=*"js/bootstrap.min.js"*></script>

<script src=*"controller/angular.min.js"*></script>

<script>

**var** app=angular.module("myApp",[]);

app.controller('roleCtrl', **function**($scope,$http){

$scope.rolesList=[];

$scope.fetchRole=**function**(){

$http({

method : 'POST',

//url : "courseServlet?action=SELECT"

url : "loginCommonValues",

data : {'action':'COMMONVALUES'}

,

headers: {

'Content-Type': 'application/json'

}

}).success(**function**(data, status, headers, config) {

$scope.rolesList = data;

console.log(JSON.stringify(rolesList));

}).error(**function**(data, status, headers, config) {

// called asynchronously if an error occurs

// or server returns response with an error status.

});

};

});

</script>

</head>

<body class=*"loginpage"* ng-app=*"myApp"* ng-controller=*"roleCtrl"* ng-init=*"fetchRole()"*>

<nav id=*"mainNav"* class=*"navbar navbar-default navbar-fixed-top navbar-custom"*>

<div class=*"container"*>

<!-- Brand and toggle get grouped for better mobile display -->

<div class=*"navbar-header page-scroll"*>

<button type=*"button"* class=*"navbar-toggle"* data-toggle=*"collapse"* data-target=*"#bs-example-navbar-collapse-1"*>

<span class=*"sr-only"*>Toggle navigation</span> Menu <i class=*"fa fa-bars"*></i>

</button>

<a class=*"navbar-brand"* href=*"#page-top"*>Online Test Engine</a>

</div>

<!-- Collect the nav links, forms, and other content for toggling -->

<div class=*"collapse navbar-collapse"* id=*"bs-example-navbar-collapse-1"*>

<ul class=*"nav navbar-nav navbar-right"*>

<li class=*"page-scroll"*>

<a href=*"HomePage.html"*>Home</a>

</li>

<li class=*"page-scroll"*>

<a href=*"#page-top"*>Explore</a>

</li>

<li class=*"page-scroll"*>

<a href=*"register.jsp"*>Registration</a>

</li>

<li class=*"page-scroll"*>

<a href=*"#page-top"*>Login</a>

</li>

<li class=*"page-scroll"*>

<a href=*"#contact"*>Contact us</a>

</li>

</ul>

</div>

<!-- /.navbar-collapse -->

</div>

<!-- /.container-fluid -->

</nav>

<div class=*"login"*>

<h1>Login</h1>

<form method=*"post"* action=*"loginServlet"*>

<select name=*"roles"* class=*"Options"* ng-model=*"courseId"* >

<option ng-repeat=*"role in rolesList"* value=*"{{role.identity}}"*>{{role.roleName}}</option>

</select>

<input type=*"text"* name=*"username"* placeholder=*"Username"* required=*"required"*/>

<input type=*"password"* name=*"pwd"* placeholder=*"Password"* required=*"required"*/>

<% String error="";

**if**(request.getParameter("error")!=**null**){

error=request.getParameter("error");

}

%>

<p style="color: *white*"><%=error %></p>

<input type=*"submit"* class=*"btn btn-primary btn-block btn-large"* value=*"Submit"*/>

</form>

</div>

<footer class=*"text-center"* id=*"logincontactus"*>

<div class=*"footer-above"*>

<div class=*"container"*>

<div class=*"row"*>

<div class=*"footer-col col-md-4"*>

<h3>Location</h3>

<p>A-1, Dheeraj Vihar, Karala,

<br>New Delhi 110081, INDIA</p>

</div>

<div class=*"footer-col col-md-4"*>

<h3>Around the Web</h3>

<ul class=*"list-inline"*>

<li>

<a href=*"#"* class=*"btn-social btn-outline"*><i class=*"fa fa-fw fa-facebook"*></i></a>

</li>

<li>

<a href=*"#"* class=*"btn-social btn-outline"*><i class=*"fa fa-fw fa-google-plus"*></i></a>

</li>

<li>

<a href=*"#"* class=*"btn-social btn-outline"*><i class=*"fa fa-fw fa-twitter"*></i></a>

</li>

<li>

<a href=*"#"* class=*"btn-social btn-outline"*><i class=*"fa fa-fw fa-linkedin"*></i></a>

</li>

<li>

<a href=*"#"* class=*"btn-social btn-outline"*><i class=*"fa fa-fw fa-dribbble"*></i></a>

</li>

</ul>

</div>

<div class=*"footer-col col-md-4"*>

<h3>About Online Test Engine</h3>

<p>Online Test Engine Provide online examination facility to collage students.</p>

</div>

</div>

</div>

</div>

<div class=*"footer-below"*>

<div class=*"container"*>

<div class=*"row"*>

<div class=*"col-lg-12"*>

Copyright &copy; Your Website 2016

</div>

</div>

</div>

</div>

</footer>

</body>

</html>

**loginServlet.java**

package com.edu.testengine.servlets;

import java.io.IOException;

/\*import java.io.PrintWriter;\*/

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.Cookie;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import com.edu.testengine.dao.loginDAO;

import com.edu.testengine.dto.loginDTO;

import com.edu.testengine.dto.userDTO;

import com.edu.testengine.utils.Encryption2;

/\*\*

\* Servlet implementation class loginServlet

\*/

@WebServlet("/loginServlet")

public class loginServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

/\*PrintWriter out = response.getWriter();\*/

String username = request.getParameter("username");

String pwd = request.getParameter("pwd");

int role=Integer.parseInt(request.getParameter("roles"));

loginDTO logindto= new loginDTO();

loginDAO loginDAO = new loginDAO();

System.out.println("hello");

logindto.setUsername(username);

logindto.setRoleid(role);

System.out.println(logindto.getUsername());

try {

logindto.setPassword(Encryption2.encrypt(pwd));

} catch (Exception e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

//System.out.println(logindto.getPassword());

//System.out.println(logindto.getRoleid());

try{

int result=loginDAO.authenticate(logindto);

switch (result)

{

case 1:

System.out.println("login success");

if(role==1)

{

System.out.println("Admin");

HttpSession session=request.getSession(true);

session.setAttribute("username", logindto.getUsername());

session.setAttribute("role", logindto.getRoleid());

session.setMaxInactiveInterval(10\*60);

Cookie userName = new Cookie("username", logindto.getUsername());

userName.setMaxAge(5\*60);

response.addCookie(userName);

response.sendRedirect("AdminPage.jsp");

}

else if(role==2)

{

System.out.println("Faculty");

HttpSession session=request.getSession(true);

session.setAttribute("username", logindto.getUsername());

session.setAttribute("role", logindto.getRoleid());

Cookie userName = new Cookie("username", logindto.getUsername());

userName.setMaxAge(5\*60);

response.addCookie(userName);

response.sendRedirect("facultypage.jsp");

}

else

{

if(role==3)

{

System.out.println("Studnet");

HttpSession session=request.getSession(true);

session.setAttribute("username", logindto.getUsername());

session.setAttribute("role", logindto.getRoleid());

Cookie userName = new Cookie("username", logindto.getUsername());

userName.setMaxAge(5\*60);

response.addCookie(userName);

response.sendRedirect("studentpage.jsp");

}

}

//Session["data"] = null;

break;

case -1:

System.out.println("login failed");

response.sendRedirect("Login.jsp?error=Invalid User Id Or Password !");

//Session["data"] = null;

break;

default:

System.out.println("login at error");

response.sendRedirect("Login.jsp?error=login at error !");

break;

}

/\*

if(loginFields==1)

{

System.out.println("Admin");

}

else if(loginFields.getRoleid()==2)

{

System.out.println("Faculty");

}

else

{

if(loginFields.getRoleid()==3)

{

System.out.println("Studnet");

}

else

{

System.out.println("Please Selecet Role of the user....Thanks!");

}

}

\*/

}catch(Exception ex){

ex.printStackTrace();

}

}

}

**UserDTO**

**package** com.edu.testengine.dto;

**import** java.util.Date;

//import java.util.Date;

**public** **class** userDTO {

**private** String userName;

**public** String getUserName() {

**return** userName;

}

**public** **void** setUserName(String userName) {

**this**.userName = userName;

}

**private** **int** roleID;

**private** String password;

**private** String email;

**private** String name;

**private** String gender;

//private Date DOB;

**private** String DOB;

**public** String getDOB() {

**return** DOB;

}

**public** **void** setDOB(String dOB) {

DOB = dOB;

}

**private** String address;

**private** String phone;

**private** String ImagePath;

**private** String Institute\_name;

**public** String getEmail() {

**return** email;

}

**public** **void** setEmail(String email) {

**this**.email = email;

}

**public** **int** getRoleID() {

**return** roleID;

}

**public** **void** setRoleID(**int** roleID) {

**this**.roleID = roleID;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getGender() {

**return** gender;

}

**public** **void** setGender(String gender) {

**this**.gender = gender;

}

**public** String getAddress() {

**return** address;

}

**public** **void** setAddress(String address) {

**this**.address = address;

}

**public** String getPhone() {

**return** phone;

}

**public** **void** setPhone(String phone) {

**this**.phone = phone;

}

**public** String getImagePath() {

**return** ImagePath;

}

**public** **void** setImagePath(String imagePath) {

ImagePath = imagePath;

}

**public** String getInstitute\_name() {

**return** Institute\_name;

}

**public** **void** setInstitute\_name(String institute\_name) {

Institute\_name = institute\_name;

}

**public** String getPassword() {

**return** password;

}

**public** **void** setPassword(String password) {

**this**.password = password;

}

}

**loginDAO.JAVA**

package com.edu.testengine.dao;

import java.sql.CallableStatement;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import com.edu.testengine.dto.loginDTO;

import com.edu.testengine.utils.CommonDAO;

import com.edu.testengine.utils.CommonSQLConstants;

public class loginDAO {

public int authenticate(loginDTO logindto) throws ClassNotFoundException, SQLException{

//String sql = CommonSQLConstants.LOGIN\_SQLADMIN;

Connection con = null;

PreparedStatement pstmt = null;

ResultSet rs = null;

//loginDTO loginDTOUser=null;

int result=0;

try{

con= CommonDAO.getConnection();

String query = "{CALL UserLogin\_Validate(?,?,?)}";

CallableStatement stmt = con.prepareCall(query);

stmt.setInt(1, logindto.getRoleid());

stmt.setString(2, logindto.getUsername());

stmt.setString(3, logindto.getPassword());

rs=stmt.executeQuery();

System.out.println(rs.toString());

if(rs.next()){

result=rs.getInt(1);

}

}

finally{

if(rs!=null){

rs.close();

}

if(pstmt!=null){

pstmt.close();

}

if(con!=null){

con.close();

}

}

return result;

}

}

**CommonDAO.JAVA**

package com.edu.testengine.utils;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

import java.util.ResourceBundle;

public class CommonDAO {

private CommonDAO() {

}

public static Connection getConnection() throws ClassNotFoundException, SQLException {

ResourceBundle rb = ResourceBundle.getBundle("db");

Class.forName(rb.getString("drivername"));

Connection con = DriverManager.getConnection(rb.getString("dburl"), rb.getString("userid"),

rb.getString("password"));

return con;

}

}

**Encryption.JAVA**

**package** com.edu.testengine.utils;

**import** java.security.\*;

**import** java.security.spec.InvalidKeySpecException;

**import** javax.crypto.\*;

**import** javax.crypto.spec.SecretKeySpec;

**import** sun.misc.\*;

**public** **class** Encryption2 {

**private** **static** **final** String ***ALGO*** = "AES";

**private** **static** **final** **byte**[] ***keyValue*** =

**new** **byte**[] { 'T', 'h', 'e', 'B', 'e', 's', 't','S', 'e', 'c', 'r','e', 't', 'K', 'e', 'y' };

**public** **static** String encrypt(String Data) **throws** Exception {

Key key = *generateKey*();

Cipher c = Cipher.*getInstance*(***ALGO***);

c.init(Cipher.***ENCRYPT\_MODE***, key);

**byte**[] encVal = c.doFinal(Data.getBytes());

String encryptedValue = **new** BASE64Encoder().encode(encVal);

**return** encryptedValue;

}

**public** **static** String decrypt(String encryptedData) **throws** Exception {

Key key = *generateKey*();

Cipher c = Cipher.*getInstance*(***ALGO***);

c.init(Cipher.***DECRYPT\_MODE***, key);

**byte**[] decordedValue = **new** BASE64Decoder().decodeBuffer(encryptedData);

**byte**[] decValue = c.doFinal(decordedValue);

String decryptedValue = **new** String(decValue);

**return** decryptedValue;

}

**private** **static** Key generateKey() **throws** Exception {

Key key = **new** SecretKeySpec(***keyValue***, ***ALGO***);

**return** key;

}

/\*

public static void main(String[] args) throws Exception {

String password = "mypassword";

String passwordEnc = encrypt(password);

String passwordDec = decrypt(passwordEnc);

System.out.println("Plain Text : " + password);

System.out.println("Encrypted Text : " + passwordEnc);

System.out.println("Decrypted Text : " + passwordDec);

}

\*/

}

**db.properties**

drivername=com.mysql.jdbc.Driver

dburl=jdbc:mysql://localhost:3306/lms

userid=root

password=root

**studentpage.jsp**

<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*

pageEncoding=*"ISO-8859-1"*%>

<%

String user = **null**;

**if**(session.getAttribute("username") == **null** || session.getAttribute("roles")==**null**){

response.sendRedirect("index.jsp");

}**else**{

**if**(session.getAttribute("roles").toString().equals("3")){

user = (String) session.getAttribute("username");

}

**else**{

response.sendRedirect("index.jsp");

}

}

String userName = **null**;

String sessionID = **null**;

response.setHeader("Cache-Control", "private, no-store, no-cache, must-revalidate");

//Set standard HTTP/1.0 no-cache header.

response.setDateHeader("Expires", 0);

response.setHeader("Pragma", "no-cache");

Cookie[] cookies = request.getCookies();

**if**(cookies !=**null**){

**for**(Cookie cookie : cookies){

**if**(cookie.getName().equals("username")) userName = cookie.getValue();

**if**(cookie.getName().equals("JSESSIONID")) sessionID = cookie.getValue();

}

%>

<!DOCTYPE html>

<html lang=*"en"*>

<head>

<meta charset=*"utf-8"*>

<meta http-equiv=*"X-UA-Compatible"* content=*"IE=edge"*>

<meta name=*"viewport"* content=*"width=device-width, initial-scale=1"*>

<meta name=*"description"* content=*""*>

<meta name=*"author"* content=*""*>

<title>Welcome: <%=session.getAttribute("username")%></title>

<link href=*"css/bootstrap.min.css"* rel=*"stylesheet"* type=*"text/css"*>

<link rel=*"stylesheet"* href=*"css/datepicker3.css"*>

<!--<link href="css/freelancer.min.css" rel="stylesheet" type="text/css">-->

<link href=*"css/font-awesome.min.css"* rel=*"stylesheet"* type=*"text/css"*>

<link href=*"https://fonts.googleapis.com/css?family=Ubuntu:500|Vollkorn"* rel=*"stylesheet"*>

<link rel=*"stylesheet"* href=*"css/animate.css"*>

<link rel=*"stylesheet"* href=*"css/studentpage.css"*>

</head>

<body id=*"page-top"* class=*"index"*>

<nav id=*"mainNav"* class=*"navbar navbar-default navbar-fixed-top navbar-custom colornav"*>

<div class=*"container"*>

<!-- Brand and toggle get grouped for better mobile display -->

<div class=*"navbar-header page-scroll"*>

<button type=*"button"* class=*"navbar-toggle"* data-toggle=*"collapse"* data-target=*"#bs-example-navbar-collapse-1"*>

<span class=*"sr-only"*>Toggle navigation</span> Menu <i class=*"fa fa-bars"*></i>

</button>

<a class=*"navbar-brand"* href=*"#page-top"*>Online Test Engine</a>

</div>

<!-- Collect the nav links, forms, and other content for toggling -->

<div class=*"collapse navbar-collapse"* id=*"bs-example-navbar-collapse-1"*>

<ul class=*"nav navbar-nav navbar-right"*>

<li class=*"page-scroll"*>

<a onclick="logout()" href=*"#"*>Logout</a>

</li>

</ul>

</div>

<!-- /.navbar-collapse -->

</div>

<!-- /.container-fluid -->

</nav>

<div class=*"group col-xs-12"*>

<aside class=*"left col-xs-3"*>

<div class=*"panel-group"* id=*"accordion"* role=*"tablist"* aria-multiselectable=*"true"*>

<div class=*"panel panel-default"*>

<div class=*"panel-heading"* role=*"tab"* id=*"headingtwo"*>

<h4 class=*"panel-title"*>

<a role=*"button"* data-toggle=*"collapse"* data-parent=*"#accordion"* href=*"#"* aria-expanded=*"true"* aria-controls=*"collapsetwo"*>

Course

</a>

</h4>

</div>

<div id=*"collapsetwo"* class=*"panel-collapse collapse in"* role=*"tabpanel"* aria-labelledby=*"headingtwo"*>

<div class=*"panel-body"*>

<ul class=*"nav-sidebar nav"*>

<li><a href=*"viewcourse.jsp"*>Course View</a></li>

</ul>

</div>

</div>

<div class=*"panel-heading"* role=*"tab"* id=*"headingOne"*>

<h4 class=*"panel-title"*>

<a role=*"button"* data-toggle=*"collapse"* data-parent=*"#accordion"* href=*"#"* aria-expanded=*"false"* aria-controls=*"collapseOne"*>

Test Details

</a>

</h4>

</div>

<div id=*"collapseOne"* class=*"panel-collapse collapse in"* role=*"tabpanel"* aria-labelledby=*"headingOne"*>

<div class=*"panel-body"*>

<ul class=*"nav-sidebar nav"*>

<li><a href=*"testresult.jsp"*> Test Result</a></li>

</ul>

</div>

</div>

<div class=*"panel-heading"* role=*"tab"* id=*"headingfour"*>

<h4 class=*"panel-title"*>

<a role=*"button"* data-toggle=*"collapse"* data-parent=*"#accordion"* href=*"#"* aria-expanded=*"true"* aria-controls=*"collapsefour"*>

Studnet Details

</a>

</h4>

</div>

<div id=*"collapsefour"* class=*"panel-collapse collapse in"* role=*"tabpanel"* aria-labelledby=*"headingfour"*>

<div class=*"panel-body"*>

<ul class=*"nav-sidebar nav"*>

<li><a href=*"editprofile.jsp"*> Edit Profile </a></li>

<li><a href=*"resetpassword.jsp"*> Change Password</a></li>

<li><a href=*"sendquery.jsp"*> Send Query</a></li>

</ul>

</div>

</div>

</div>

</div>

</aside>

<!--<section class="right">-->

<div class=*"col-xs-9"*>

<div class=*"col-xs-12 col-md-6 col-lg-6"*>

<div class=*"panel panel-teal panel-widget"*>

<div class=*"row no-padding"*>

<div class=*"col-sm-3 col-lg-5 widget-left"*>

<svg class=*"glyph stroked male-user"*><use xlink:href=*"#stroked-male-user"*></use></svg>

</div>

<div class=*"col-sm-9 col-lg-7 widget-right"*>

<div class=*"large"*>50</div>

<div class=*"text-muted"*>Total Student</div>

</div>

</div>

</div>

</div>

<div class=*"col-xs-12 col-md-6 col-lg-6"*>

<div class=*"panel panel-widget"*>

<div class=*"row no-padding"*>

<div class=*"col-sm-3 col-lg-5 widget-left panel-red"*>

<svg class=*"glyph stroked email"*><use xlink:href=*"#stroked-email"*/></svg></svg>

</div>

<div class=*"col-sm-9 col-lg-7 widget-right"*>

<div class=*"large"*>25</div>

<div class=*"text-muted"*>Total Faculty</div>

</div>

</div>

</div>

</div>

<div class=*"col-xs-8"*>

<div id=*"mainContent"*>

<div class=*"col-xs-12"*>

<img src=*"images/Online-exam.png"* alt=*"online Exam"*>

</div>

</div>

</div>

</div>

</div>

<footer class=*"footer"*>

<div class=*"container"*>

<div class=*"row"*>

<div class=*"col-lg-12"*>

Copyright &copy; Your Website 2016

</div>

</div>

</div>

</footer>

<script type=*"application/javascript"* src=*"js/jquery-1.12.3.min.js"*></script>

<script type=*"application/javascript"* src=*"js/bootstrap.min.js"*></script>

<script src=*"js/lumino.glyphs.js"*></script>

<script src=*"js/ajax.js"*></script>

</body>

</html>

<%}%>

**viewcourse.jsp**

<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*

pageEncoding=*"ISO-8859-1"*%>

<%

String user = **null**;

**if**(session.getAttribute("username") == **null** || session.getAttribute("roles")==**null**){

response.sendRedirect("index.jsp");

}**else**{

**if**(session.getAttribute("roles").toString().equals("3")){

user = (String) session.getAttribute("username");

}

**else**{

response.sendRedirect("index.jsp");

}

}

String userName = **null**;

String sessionID = **null**;

Cookie[] cookies = request.getCookies();

**if**(cookies !=**null**){

**for**(Cookie cookie : cookies){

**if**(cookie.getName().equals("username")) userName = cookie.getValue();

**if**(cookie.getName().equals("JSESSIONID")) sessionID = cookie.getValue();

}

%>

<!DOCTYPE html>

<html lang=*"en"*>

<head>

<meta charset=*"utf-8"*>

<meta http-equiv=*"X-UA-Compatible"* content=*"IE=edge"*>

<meta name=*"viewport"* content=*"width=device-width, initial-scale=1"*>

<meta name=*"description"* content=*""*>

<meta name=*"author"* content=*""*>

<title>Welcome: <%=session.getAttribute("username")%></title>

<link href=*"css/bootstrap.min.css"* rel=*"stylesheet"* type=*"text/css"*>

<link href=*"css/bootstrap.min2.css"* rel=*"stylesheet"*>

<!--<link href="css/freelancer.min.css" rel="stylesheet" type="text/css">-->

<link href=*"css/font-awesome.min.css"* rel=*"stylesheet"* type=*"text/css"*>

<link href=*"https://fonts.googleapis.com/css?family=Ubuntu:500|Vollkorn"* rel=*"stylesheet"*>

<link rel=*"stylesheet"* href=*"css/animate.css"*>

<link rel=*"stylesheet"* href=*"css/adminpage.css"*>

<link rel=*"stylesheet"* href=*"css/modify.css"*>

</head>

<body id=*"page-top"* class=*"index"*>

<div class=*"col-xl-12"*>

<nav id=*"mainNav"* class=*"navbar navbar-default navbar-fixed-top navbar-custom colornav"*>

<div class=*"container"*>

<!-- Brand and toggle get grouped for better mobile display -->

<div class=*"navbar-header page-scroll"*>

<button type=*"button"* class=*"navbar-toggle"* data-toggle=*"collapse"* data-target=*"#bs-example-navbar-collapse-1"*>

<span class=*"sr-only"*>Toggle navigation</span> Menu <i class=*"fa fa-bars"*></i>

</button>

<a class=*"navbar-brand"* href=*"#page-top"*>Online Test Engine</a>

</div>

<!-- Collect the nav links, forms, and other content for toggling -->

<div class=*"collapse navbar-collapse"* id=*"bs-example-navbar-collapse-1"*>

<ul class=*"nav navbar-nav navbar-right"*>

<li class=*"page-scroll"*>

<a onclick="logout()" href=*"#"*>Logout</a>

</li>

</ul>

</div>

<!-- /.navbar-collapse -->

</div>

<!-- /.container-fluid -->

</nav>

</div>

<div class=*"group col-xs-12"*>

<aside class=*"left col-xs-3"*>

<div class=*"panel-group"* id=*"accordion"* role=*"tablist"* aria-multiselectable=*"true"*>

<div class=*"panel panel-default"*>

<div class=*"panel-heading"* role=*"tab"* id=*"headingtwo"*>

<h4 class=*"panel-title"*>

<a role=*"button"* data-toggle=*"collapse"* data-parent=*"#accordion"* href=*"#"* aria-expanded=*"true"* aria-controls=*"collapsetwo"*>

Course

</a>

</h4>

</div>

<div id=*"collapsetwo"* class=*"panel-collapse collapse in"* role=*"tabpanel"* aria-labelledby=*"headingtwo"*>

<div class=*"panel-body"*>

<ul class=*"nav-sidebar nav"*>

<li style="background-color: *gray*;"><a href=*"viewcourse.jsp"*>Course View</a></li>

</ul>

</div>

</div>

<div class=*"panel-heading"* role=*"tab"* id=*"headingOne"*>

<h4 class=*"panel-title"*>

<a role=*"button"* data-toggle=*"collapse"* data-parent=*"#accordion"* href=*"#"* aria-expanded=*"false"* aria-controls=*"collapseOne"*>

Test Details

</a>

</h4>

</div>

<div id=*"collapseOne"* class=*"panel-collapse collapse in"* role=*"tabpanel"* aria-labelledby=*"headingOne"*>

<div class=*"panel-body"*>

<ul class=*"nav-sidebar nav"*>

<li><a href=*"testresult.jsp"*> Test Result</a></li>

</ul>

</div>

</div>

<div class=*"panel-heading"* role=*"tab"* id=*"headingfour"*>

<h4 class=*"panel-title"*>

<a role=*"button"* data-toggle=*"collapse"* data-parent=*"#accordion"* href=*"#"* aria-expanded=*"true"* aria-controls=*"collapsefour"*>

Studnet Details

</a>

</h4>

</div>

<div id=*"collapsefour"* class=*"panel-collapse collapse in"* role=*"tabpanel"* aria-labelledby=*"headingfour"*>

<div class=*"panel-body"*>

<ul class=*"nav-sidebar nav"*>

<li><a href=*"editprofile.jsp"*> Edit Profile </a></li>

<li><a href=*"resetpassword.jsp"*> Change Password</a></li>

<li><a href=*"sendquery.jsp"*> Send Query</a></li>

</ul>

</div>

</div>

</div>

</div>

</aside>

<!--<section class="right">-->

<div class=*"col-xs-9"*>

<div class=*"col-xs-12 col-md-6 col-lg-6"*>

<div class=*"panel panel-teal panel-widget"*>

<div class=*"row no-padding"*>

<div class=*"col-sm-3 col-lg-5 widget-left"*>

<svg class=*"glyph stroked male-user"*><use xlink:href=*"#stroked-male-user"*></use></svg>

</div>

<div class=*"col-sm-9 col-lg-7 widget-right"*>

<div class=*"large"*>50</div>

<div class=*"text-muted"*>Total Student</div>

</div>

</div>

</div>

</div>

<div class=*"col-xs-12 col-md-6 col-lg-6"*>

<div class=*"panel panel-widget"*>

<div class=*"row no-padding"*>

<div class=*"col-sm-3 col-lg-5 widget-left panel-red"*>

<svg class=*"glyph stroked email"*><use xlink:href=*"#stroked-email"*/></svg></svg>

</div>

<div class=*"col-sm-9 col-lg-7 widget-right"*>

<div class=*"large"*>25</div>

<div class=*"text-muted"*>Total Faculty</div>

</div>

</div>

</div>

</div>

<div id=*"mainContent"*>

<h2 style="text-align: *center*;">Total Courses</h2>

<hr>

<table class=*"table sidestyle2"*>

<thead>

<tr>

<th>S.No.</th>

<th>Course Name</th>

<th>Details</th>

<th>Ragister Date</th>

<th>View Course</th>

</tr>

</thead>

<tbody>

<tr>

<th scope=*"row"*>1</th>

<td>Java Core</td>

<td>Java Basic course</td>

<td>05-7-2016</td>

<td><a href=*"coursecontent.jsp"*>Start Course</a></td>

</tr>

<tr>

<th scope=*"row"*>2</th>

<td>Mean</td>

<td>Mean for user interface development</td>

<td>12-8-2016</td>

<td><a href=*"coursecontent.jsp"*>Start Course</a></td>

</tr>

<tr>

<th scope=*"row"*>3</th>

<td>Html&JS</td>

<td>Basic of Website development</td>

<td>12-5-2016</td>

<td><a href=*"coursecontent.jsp"*>Start Course</a></td>

</tr>

<tr>

<th scope=*"row"*>4</th>

<td>Advance Java</td>

<td>Java advance course </td>

<td>1-06-2015</td>

<td><a href=*"coursecontent.jsp"*>Start Course</a></td>

</tr>

<tr>

<th scope=*"row"*>5</th>

<td>Ruby</td>

<td>Ruby Course Basic</td>

<td>12-6-2016</td>

<td><a href=*"coursecontent.jsp"*>Start Course</a></td>

</tr>

<tr>

<th scope=*"row"*>7</th>

<td>Python</td>

<td>Python Course Basic</td>

<td>06-11-2015</td>

<td><a href=*"coursecontent.jsp"*>Start Course</a></td>

</tr>

</tbody>

</table>

</div>

</div>

</div>

<footer class=*"footer"*>

<div class=*"container"*>

<div class=*"row"*>

<div class=*"col-lg-12"*>

Copyright &copy; Your Website 2016

</div>

</div>

</div>

</footer>

<script src=*"js/jquery-2.0.3.js"*></script>

<script type=*"application/javascript"* src=*"js/bootstrap.min.js"*></script>

<script src=*"js/lumino.glyphs.js"*></script>

<script src=*"js/ajax.js"*></script>

</body>

</html>

<%}%>

**addcourse.jsp**

<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*

pageEncoding=*"ISO-8859-1"*%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>

<meta charset=*"utf-8"*>

<meta http-equiv=*"X-UA-Compatible"* content=*"IE=edge"*>

<meta name=*"viewport"* content=*"width=device-width, initial-scale=1"*>

<title>Courses</title>

<link href=*"css/bootstrap.min.css"* rel=*"stylesheet"*>

<link rel=*"stylesheet"* href=*"css/addcourse.css"*>

<script type=*"application/javascript"* src=*"js/jquery-1.12.3.min.js"*></script>

<script type=*"application/javascript"* src=*"js/bootstrap.min.js"*></script>

</head>

<body>

<section class=*"main"*>

<div id=*"form"*>

<form action=*"courseServlet"* method=*"post"* enctype=*"multipart/form-data"* class=*"register"*>

<fieldset>

<label for=*"ImageUpload"*>Browse Image:</label>

<input type=*"file"* value=*"Browse...."* name=*"uploadimage"*/>

<label for=*"datalist"*>Title:</label>

<input id=*"text"* id=*"datalist"* list=*"coursetitle"* name=*"coursetitle"*/>

<label for=*"title"*>Details:</label>

<textarea class=*"textareadetails"* name=*"coursedetails"* id=*"title"* cols=*"50px"* rows=*"4"*></textarea>

<input type=*"submit"* name=*"submit"* class=*"button"* value=*"Submit"* /></p>

</fieldset>

</form>

</div>

</section>

</body>

</html>

**courseServlet.JAVA**

package com.edu.testengine.servlets;

import java.io.IOException;

import java.io.PrintWriter;

import java.sql.SQLException;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import com.edu.testengine.dao.courseDAO;

import com.edu.testengine.dto.courseDTO;

import java.io.File;

import java.io.IOException;

import java.io.PrintWriter;

import java.util.HashMap;

import java.util.Iterator;

import java.util.List;

import org.apache.commons.fileupload.FileItem;

import org.apache.commons.fileupload.disk.DiskFileItemFactory;

import org.apache.commons.fileupload.servlet.ServletFileUpload;

/\*\*

\* Servlet implementation class courseServlet

\*/

@WebServlet("/courseServlet")

public class courseServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

private static final String UPLOAD\_DIRECTORY = "courseImage";

private static final int MEMORY\_THRESHOLD = 1024 \* 1024 \* 3; // 3MB

private static final int MAX\_FILE\_SIZE = 1024 \* 1024 \* 40; // 40MB

private static final int MAX\_REQUEST\_SIZE = 1024 \* 1024 \* 50; // 50MB

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

courseDAO coursedao=new courseDAO();

courseDTO coursedto = new courseDTO();

if (!ServletFileUpload.isMultipartContent(request)) {

PrintWriter writer = response.getWriter();

writer.println("Error: Form must has enctype=multipart/form-data.");

writer.flush();

return;

}

HashMap<String,String> value=new HashMap<>();

DiskFileItemFactory factory = new DiskFileItemFactory();

factory.setSizeThreshold(MEMORY\_THRESHOLD);

factory.setRepository(new File(System.getProperty("java.io.tmpdir")));

ServletFileUpload upload = new ServletFileUpload(factory);

upload.setFileSizeMax(MAX\_FILE\_SIZE);

upload.setSizeMax(MAX\_REQUEST\_SIZE);

String uploadPath = getServletContext().getRealPath("")

+ File.separator + UPLOAD\_DIRECTORY;

File uploadDir = new File(uploadPath);

if (!uploadDir.exists()) {

uploadDir.mkdir();

}

try {

@SuppressWarnings("unchecked")

List<FileItem> formItems = upload.parseRequest(request);

if (formItems != null && formItems.size() > 0) {

for (FileItem item : formItems) {

if (!item.isFormField()) {

String fileName = new File(item.getName()).getName();

String filePath = uploadPath + File.separator + fileName;

File storeFile = new File(filePath);

System.out.println("file path :"+storeFile);

//my code

String folderPath = uploadPath + File.separator;

int index=fileName.lastIndexOf(".");

String fileN=fileName.substring(0, index);

String extension=fileName.substring(index,fileName.length());

System.out.println("file name :"+ fileN);

System.out.println("file extension :"+ extension);

File storeFileNew=new File(filePath);

for(int i=1;i<=100000;i++){

if(storeFileNew.exists()){

storeFileNew = new File(folderPath+fileN+i+extension);

continue;

}

storeFile.renameTo(storeFileNew);

//coursedto.setImagePath(UPLOAD\_DIRECTORY+"/"+fileName);

coursedto.setImagePath(UPLOAD\_DIRECTORY+"/"+storeFileNew.getName());

break;

}

//

item.write(storeFile);

request.setAttribute("message",

"Upload has been done successfully!");

}

if (item.isFormField())

{

String name = item.getFieldName();

String value2 = item.getString();

value.put(name,value2);

System.out.println(name+":"+value2);

}

}

}

//System.out.println("user Name : "+value.get("userName"));

//System.out.println("phone : "+value.get("phone"));

String title = value.get("coursetitle");

String details = value.get("coursedetails");

coursedto.setDetails(details);

coursedto.setTitle(title);

} catch (Exception ex) {

request.setAttribute("message",

"There was an error: " + ex.getMessage());

}

try {

if(coursedao.insert(coursedto)){

System.out.println("course added successfully");

}

else{

System.out.println("course adding failed");

}

} catch (Exception e1) {

// TODO Auto-generated catch block

e1.printStackTrace();

}

}

}

**courseDTO.java**

**package** com.edu.testengine.dto;

**import** java.util.Date;

**public** **class** courseDTO {

**private** **int** courseID;

**private** String title;

**private** String imagePath;

**private** String Details;

**private** Date ragister\_date;

**public** **int** getCourseID() {

**return** courseID;

}

**public** **void** setCourseID(**int** courseID) {

**this**.courseID = courseID;

}

**public** String getTitle() {

**return** title;

}

**public** **void** setTitle(String title) {

**this**.title = title;

}

**public** String getImagePath() {

**return** imagePath;

}

**public** **void** setImagePath(String imagePath) {

**this**.imagePath = imagePath;

}

**public** String getDetails() {

**return** Details;

}

**public** **void** setDetails(String details) {

Details = details;

}

**public** Date getRagister\_date() {

**return** ragister\_date;

}

**public** **void** setRagister\_date(Date ragister\_date) {

**this**.ragister\_date = ragister\_date;

}

}

**courseDAO.JAVA**

package com.edu.testengine.dao;

import java.sql.CallableStatement;

import java.sql.Connection;

import java.sql.ResultSet;

import java.sql.SQLException;

import com.edu.testengine.dto.courseDTO;

import com.edu.testengine.utils.CommonDAO;

public class courseDAO {

public ResultSet select() throws ClassNotFoundException, SQLException{

Connection con= CommonDAO.getConnection();

String query = "{CALL Course\_CRUD(?,?,?,?,?)}";

CallableStatement stmt = con.prepareCall(query);

stmt.setString(1, "SELECT");

stmt.setInt(2, 0);

stmt.setString(3, "");

stmt.setString(4, "");

stmt.setString(5, "");

ResultSet rs = stmt.executeQuery();

return rs;

}

public boolean insert(courseDTO course) throws ClassNotFoundException, SQLException{

Connection con= CommonDAO.getConnection();

String query = "{CALL Course\_CRUD(?,?,?,?,?)}";

CallableStatement stmt = con.prepareCall(query);

stmt.setString(1, "INSERT");

stmt.setInt(2, 0);

stmt.setString(3, course.getTitle());

stmt.setString(4, course.getDetails());

stmt.setString(5, course.getImagePath());

boolean rs = stmt.execute();

return rs;

}

public boolean update(courseDTO course) throws ClassNotFoundException, SQLException{

Connection con= CommonDAO.getConnection();

String query = "{CALL Course\_CRUD(?,?,?,?,?)}";

CallableStatement stmt = con.prepareCall(query);

stmt.setString(1, "UPDATE");

stmt.setInt(2, course.getCourseID());

stmt.setString(3, course.getTitle());

stmt.setString(4, course.getDetails());

stmt.setString(5, course.getImagePath());

boolean rs = stmt.execute();

return rs;

}

public boolean delete(courseDTO course) throws ClassNotFoundException, SQLException{

Connection con= CommonDAO.getConnection();

String query = "{CALL Course\_CRUD(?,?,?,?,?)}";

CallableStatement stmt = con.prepareCall(query);

stmt.setString(1, "DELETE");

stmt.setInt(2, course.getCourseID());

stmt.setString(3, course.getTitle());

stmt.setString(4, course.getDetails());

stmt.setString(5, course.getImagePath());

boolean rs = stmt.execute();

return rs;

}

/\*

public boolean insert(courseDTO coursedto){

boolean isFound=false;

try{

Connection con=CommonDAO.getConnection();

CallableStatement csetmt=con.prepareCall("{call Course\_CRUD(?,?,?,?,?)}");

csetmt.setString(1,"INSERT");

csetmt.setInt(2, 0);

csetmt.setString(3, coursedto.getImagePath());

csetmt.setString(4, coursedto.getTitle());

csetmt.setString(5, coursedto.getDetails());

isFound=csetmt.execute();

con.close();

}

catch(Exception e)

{

e.printStackTrace();

}

return isFound;

}

\*/

}

**register.jsp**

<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*

pageEncoding=*"ISO-8859-1"*%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>

<meta charset=*"utf-8"*>

<meta http-equiv=*"X-UA-Compatible"* content=*"IE=edge"*>

<meta name=*"viewport"* content=*"width=device-width, initial-scale=1"*>

<meta name=*"description"* content=*""*>

<meta name=*"author"* content=*""*>

<title>Register</title>

<!-- Bootstrap Core CSS -->

<link href=*"css/bootstrap.min.css"* rel=*"stylesheet"*>

<!-- Theme CSS -->

<link href=*"css/freelancer.min.css"* rel=*"stylesheet"*>

<link href=*"css/Registration.css"* rel=*"stylesheet"*>

<link rel=*"stylesheet"* href=*"css/custom.css"*>

<!-- Custom Fonts -->

<link href=*"css/font-awesome.min.css"* rel=*"stylesheet"* type=*"text/css"*>

<!--<link href="https://fonts.googleapis.com/css?family=Montserrat:400,700" rel="stylesheet" type="text/css">

<link href="https://fonts.googleapis.com/css?family=Lato:400,700,400italic,700italic" rel="stylesheet" type="text/css">-->

<script src=*"js/jquery-1.12.3.min.js"*></script>

<script src=*"js/bootstrap.min.js"*></script>

</head>

<body id=*"page-top"* class=*"index"* >

<div class=*"container"*>

<!-- Navigation -->

<nav id=*"mainNav"* class=*"navbar navbar-default navbar-fixed-top navbar-custom"*>

<!-- Brand and toggle get grouped for better mobile display -->

<div class=*"navbar-header page-scroll"*>

<button type=*"button"* class=*"navbar-toggle"* data-toggle=*"collapse"* data-target=*"#bs-example-navbar-collapse-1"*>

<span class=*"sr-only"*>Toggle navigation</span> Menu <i class=*"fa fa-bars"*></i>

</button>

<a class=*"navbar-brand"* href=*"HomePage.html"*>Online Test Engine</a>

</div>

<!-- Collect the nav links, forms, and other content for toggling -->

<div class=*"collapse navbar-collapse"* id=*"bs-example-navbar-collapse-1"*>

<ul class=*"nav navbar-nav navbar-right"*>

<li class=*"page-scroll"*>

<a href=*"HomePage.html"*>Home</a>

</li>

<li class=*"page-scroll"*>

<a href=*"#page-top"*>Explore</a>

</li>

<li class=*"page-scroll"*>

<a href=*"registraton.html"*>Registration</a>

</li>

<li class=*"page-scroll"*>

<a href=*"index.html"*>Login</a>

</li>

<li class=*"page-scroll"*>

<a href=*"#contact"*>Contact us</a>

</li>

</ul>

</div>

</nav> <!-- /.navbar-collapse -->

</div>

<!-- /.container-fluid -->

<!-- Portfolio Grid Section -->

<section id=*"portfolio"*>

<div class=*"container"*>

<div class=*"row"*>

<div class=*"col-lg-12 text-center"*>

<h2>REGISTRATION FORM</h2>

<hr>

</div>

<!-- / form starts

<p>Use tab keys to move from one input field to the next.</p>-->

<form class=*"form-signin"* action=*"registerServlet"* method=*"post"* enctype=*"multipart/form-data"*>

<!-- <h2 class="form-signin-heading">Please sign in</h2>

<br>-->

<!--<label for="inputname" class="sr-only">Name</label>-->

<input type=*"name"* name=*"username"* class=*"form-control"* placeholder=*"Type Your User Name"* required=*"required"*/>

<input type=*"name"* name=*"inputname"* class=*"form-control"* placeholder=*"Type Your Name"* required=*"required"*/>

<div class=*"radiobutton"*>

<input type=*"radio"* name=*"gender"* value=*"male"* checked> Male

<input type=*"radio"* name=*"gender"* value=*"female"*> Female

</div>

<label for=*"inputage"* class=*"sr-only"*>Age</label>

<input type=*"date"* id=*"inputage"* class=*"form-control"* name=*"Age"* required=*"required"* />

<label for=*"inputEmail"* class=*"sr-only"*>Email address</label>

<input type=*"email"* id=*"inputEmail"* class=*"form-control"* name=*"email"* placeholder=*"Email address"* required=*"required"* />

<label for=*"passid"* class=*"sr-only"*>Password</label>

<input type=*"password"* id=*"passid"* class=*"form-control"* name=*"password"* placeholder=*"Password"* required=*"required"* />

<label for=*"confirmPassword"* class=*"sr-only"*> ConfirmPassword</label>

<input type=*"password"* id=*"confirmPassword"* class=*"form-control"* name=*"confirmpassword"* placeholder=*"Confirm Password"* required=*"required"* />

<label for=*"Phone"* class=*"sr-only"*> Contact No.</label>

<input type=*"text"* id=*"number"* class=*"form-control"* name=*"mobileno"* placeholder=*"Contect No."* required=*"required"* />

<div class=*"form-control image-file"*>

select Image <input type=*"file"* name=*"filePath"*>

<!--<input type="file" id="image" class="form-image form-control" name="selectedimage" placeholder="Browse..." required="required"/>-->

</div>

<label for=*"address"* class=*"sr-only"*> Address</label>

<input type=*"textarea"* id=*"address"* class=*"form-control"* name=*"address"* placeholder=*"Address"* required=*"required"* />

<label for=*"institute"* class=*"sr-only"*> Institute Name</label>

<input type=*"text"* id=*"institute"* class=*"form-control"* name=*"institute"* placeholder=*"Institute Name"* required=*"required"*/>

<div class=*"checkbox"*>

<button type=*"submit"* class=*"btn btn-primary btn-block btn-large"*>Register</button>

</div>

</form>

</div>

</div>

</section>

<footer class=*"text-center"* id=*"contactus"*>

<div class=*"footer-above"*>

<div class=*"container"*>

<div class=*"row"*>

<div class=*"footer-col col-md-4"*>

<h3>Location</h3>

<p>A-1, Dheeraj Vihar, Karala,

<br>New Delhi 110081, INDIA</p>

</div>

<div class=*"footer-col col-md-4"*>

<h3>Around the Web</h3>

<ul class=*"list-inline"*>

<li>

<a href=*"#"* class=*"btn-social btn-outline"*><i class=*"fa fa-fw fa-facebook"*></i></a>

</li>

<li>

<a href=*"#"* class=*"btn-social btn-outline"*><i class=*"fa fa-fw fa-google-plus"*></i></a>

</li>

<li>

<a href=*"#"* class=*"btn-social btn-outline"*><i class=*"fa fa-fw fa-twitter"*></i></a>

</li>

<li>

<a href=*"#"* class=*"btn-social btn-outline"*><i class=*"fa fa-fw fa-linkedin"*></i></a>

</li>

<li>

<a href=*"#"* class=*"btn-social btn-outline"*><i class=*"fa fa-fw fa-dribbble"*></i></a>

</li>

</ul>

</div>

<div class=*"footer-col col-md-4"*>

<h3>About Online Test Engine</h3>

<p>Online Test Engine Provide online examination facility to collage students.</p>

</div>

</div>

</div>

</div>

<div class=*"footer-below"*>

<div class=*"container"*>

<div class=*"row"*>

<div class=*"col-lg-12"*>

Copyright &copy; Your Website 2016

</div>

</div>

</div>

</div>

</footer>

</body>

</html>

**registerServlet.JAVA**

package com.edu.testengine.servlets;

import java.sql.SQLException;

import java.text.DateFormat;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Date;

//import java.util.Date;

import java.util.HashMap;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import com.edu.testengine.dao.userDAO;

import com.edu.testengine.dto.userDTO;

import java.io.File;

import java.io.IOException;

import java.io.PrintWriter;

import java.util.Iterator;

import java.util.List;

import org.apache.commons.fileupload.FileItem;

import org.apache.commons.fileupload.disk.DiskFileItemFactory;

import org.apache.commons.fileupload.servlet.ServletFileUpload;

/\*\*

\* Servlet implementation class registerServlet

\*/

@WebServlet("/registerServlet")

public class registerServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

private static final String UPLOAD\_DIRECTORY = "upload";

private static final int MEMORY\_THRESHOLD = 1024 \* 1024 \* 3; // 3MB

private static final int MAX\_FILE\_SIZE = 1024 \* 1024 \* 40; // 40MB

private static final int MAX\_REQUEST\_SIZE = 1024 \* 1024 \* 50; // 50MB

/\*\*

\* @see HttpServlet#HttpServlet()

\*/

public registerServlet() {

super();

// TODO Auto-generated constructor stub

}

/\*\*

\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

PrintWriter out = response.getWriter();

userDAO userdao=new userDAO();

userDTO userdto = new userDTO();

if (!ServletFileUpload.isMultipartContent(request)) {

PrintWriter writer = response.getWriter();

writer.println("Error: Form must has enctype=multipart/form-data.");

writer.flush();

return;

}

HashMap<String,String> value=new HashMap<>();

DiskFileItemFactory factory = new DiskFileItemFactory();

factory.setSizeThreshold(MEMORY\_THRESHOLD);

factory.setRepository(new File(System.getProperty("java.io.tmpdir")));

ServletFileUpload upload = new ServletFileUpload(factory);

upload.setFileSizeMax(MAX\_FILE\_SIZE);

upload.setSizeMax(MAX\_REQUEST\_SIZE);

String uploadPath = getServletContext().getRealPath("")

+ File.separator + UPLOAD\_DIRECTORY;

File uploadDir = new File(uploadPath);

if (!uploadDir.exists()) {

uploadDir.mkdir();

}

try {

@SuppressWarnings("unchecked")

List<FileItem> formItems = upload.parseRequest(request);

if (formItems != null && formItems.size() > 0) {

for (FileItem item : formItems) {

if (!item.isFormField()) {

String fileName = new File(item.getName()).getName();

String filePath = uploadPath + File.separator + fileName;

File storeFile = new File(filePath);

System.out.println("file path :"+storeFile);

//my code

String folderPath = uploadPath + File.separator;

int index=fileName.lastIndexOf(".");

String fileN=fileName.substring(0, index);

String extension=fileName.substring(index,fileName.length());

System.out.println("file name :"+ fileN);

System.out.println("file extension :"+ extension);

File storeFileNew=new File(filePath);

for(int i=1;i<=100000;i++){

if(storeFileNew.exists()){

storeFileNew = new File(folderPath+fileN+i+extension);

continue;

}

storeFile.renameTo(storeFileNew);

userdto.setImagePath(UPLOAD\_DIRECTORY+"/"+storeFileNew.getName());

break;

}

//

item.write(storeFile);

request.setAttribute("message",

"Upload has been done successfully!");

}

if (item.isFormField())

{

String name = item.getFieldName();

String value2 = item.getString();

value.put(name,value2);

System.out.println(name+":"+value2);

}

}

}

//System.out.println("user Name : "+value.get("userName"));

//System.out.println("phone : "+value.get("phone"));

String username = value.get("username");

String name = value.get("inputname");

String age=value.get("Age");

//DateFormat df=new SimpleDateFormat("mm-dd-yyyy");

//Date age = df.parse(value.get("Age"));

String password = value.get("password");

String mobileno = value.get("mobileno");

String institute = value.get("institute");

//String filepath = request.getParameter("filePath");

String address = value.get("address");

String Gender= value.get("gender");

String email= value.get("email");

System.out.println("value in name variable : "+name);

userdto.setName(name);

userdto.setUserName(username);

userdto.setGender(Gender);

//userdto.setImagePath(filepath);

userdto.setDOB(age);

userdto.setPhone(mobileno);

userdto.setInstitute\_name(institute);

userdto.setPassword(password);

userdto.setAddress(address);

userdto.setEmail(email);

} catch (Exception ex) {

request.setAttribute("message",

"There was an error: " + ex.getMessage());

}

try {

if(userdao.register(userdto)){

System.out.println("register successfully");

}

else{

System.out.println("registration failed");

}

} catch (ClassNotFoundException | SQLException e1) {

// TODO Auto-generated catch block

e1.printStackTrace();

}

try {

boolean isFound=userDAO.register(userdto);

} catch (ClassNotFoundException e) {

e.printStackTrace();

} catch (SQLException e) {

e.printStackTrace();

}

}

}

**userDAO.java**

package com.edu.testengine.dao;

import java.sql.CallableStatement;

import java.sql.Connection;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.text.DateFormat;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Date;

import com.edu.testengine.dto.userDTO;

import com.edu.testengine.utils.CommonDAO;

public class userDAO {

public ResultSet selected(String id) throws ClassNotFoundException, SQLException, ParseException{

Connection con= CommonDAO.getConnection();

String query = "{CALL UserDetails\_CRUD(?,?,?,?,?,?,?,?,?,?)}";

CallableStatement stmt = con.prepareCall(query);

stmt.setString(1, "SELECTED");

stmt.setInt(2, 0);

stmt.setString(3, id);

stmt.setString(4, "");

stmt.setString(5, "");

DateFormat df=new SimpleDateFormat("yyyy-MM-dd");

java.util.Date dd=new java.util.Date();

dd.setDate(2);

dd.setMonth(2);

dd.setYear(2016);

Date dob = df.parse(dd.toString());

java.sql.Date sqlDate = new java.sql.Date(dob.getTime());

stmt.setDate(6,sqlDate);

stmt.setString(7, "");

stmt.setString(8, "");

stmt.setString(9, "");

stmt.setString(10, "");

ResultSet rs = stmt.executeQuery();

return rs;

}

public static boolean register(userDTO uDTO) throws ClassNotFoundException,SQLException{

boolean isFound=false;

try{

Connection con=CommonDAO.getConnection();

CallableStatement csetmt=con.prepareCall("{call UserLogin\_Insert(?,?,?,?,?,?,?,?,?,?,?)}");

/\* PreparedStatement pstmt = con.prepareStatement("insert into registration\_master(first\_name,last\_name,date\_of\_birth,user\_name,password,re\_enter\_password) values(?,?,?,?,?,?)");\*/

csetmt.setString(1, uDTO.getUserName());

System.out.println("in dao user name :"+uDTO.getName());

csetmt.setInt(2, 3);

csetmt.setString(3, uDTO.getPassword());

csetmt.setString(4, uDTO.getEmail());

csetmt.setString(5, uDTO.getName());

csetmt.setString(6, uDTO.getGender());

DateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");

//

DateFormat df=new SimpleDateFormat("yyyy-MM-dd");

Date age = df.parse(uDTO.getDOB());

java.sql.Date sqlDate = new java.sql.Date(age.getTime());

csetmt.setDate(7,sqlDate);

//

//csetmt.setDate(7,(Date)sdf.parse(uDTO.getDOB()));

csetmt.setString(8, uDTO.getAddress());

csetmt.setString(9, uDTO.getPhone());

csetmt.setString(10, uDTO.getImagePath());

csetmt.setString(11, uDTO.getInstitute\_name());

isFound=csetmt.execute();

con.close();

}

catch(Exception e)

{

e.printStackTrace();

}

return isFound;

}

}

**addtest.jsp**

<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*

pageEncoding=*"ISO-8859-1"*%>

<!DOCTYPE html>

<html>

<head lang=*"en"*>

<meta charset=*"UTF-8"*>

<title>Add Test</title>

<link href=*"css/bootstrap.min.css"* rel=*"stylesheet"*>

<link rel=*"stylesheet"* href=*"css/addtest.css"*>

<script type=*"application/javascript"* src=*"js/jquery-1.12.3.min.js"*></script>

<script type=*"application/javascript"* src=*"js/bootstrap.min.js"*></script>

<script type=*"application/javascript"* src=*"js/addtest.js"*></script>

<script src=*"controller/angular.min.js"*></script>

<script src=*"controller/mainController.js"*></script>

<script src=*"controller/addTestController.js"*></script>

</head>

<body ng-app=*"myApp"* ng-controller=*"uploadQuestionCtrl"* ng-init=*"getCommon()"*>

<form method=*"post"*>

<div>

<table class=*"sidestyle"*>

<!-- <form action="" method="post" name="addtest" onsubmit="return validate();">-->

<tr>

<td class=*"tablestyle"*>

<label for=*"facultyid"*>Faculty ID</label>

<!-- <label ng-model=<%=session.getAttribute("username")%>><%=session.getAttribute("username")%></label> -->

<label for=*"course\_id"*>Course ID</label>

<select name=*"course\_id"* ng-model=*"courseId"* ng-change=*"changeevtcourse()"* >

<option ng-repeat=*"course in courseIdList"* value=*"{{course.courseID}}"*>{{course.title}}</option>

</select>

</td>

</tr>

<tr>

<td class=*"tablestyle"*><label for=*"testname"*>Test Name</label>

<input type=*"text"* name=*"testname"* ng-model=*"testName"* placeholder=*"Test Name"*>

<label for=*"testtime"*>Test Timing</label>

<input type=*"number"* name=*"testtime"* ng-model=*"testTime"* placeholder=*"Test Duration"*></td>

</tr>

<tr>

<td class=*"tablestyle"*>

<label for=*"minmarks"*>Minimum Marks</label>

<input type=*"number"* name=*"minmarks"* ng-model=*"minMarks"* placeholder=*"MinMarks"*>

<label for=*"testtime"*>Total Marks</label>

<input type=*"number"* name=*"totalmarks"* ng-model=*"totalMarks"* placeholder=*"Total Marks"*>

</td>

</tr>

<tr>

<td>

<label>Total Question :</label>

<p>{{totalQuestions}}</p>

</td>

</tr>

<tr>

<td>

<hr/>

<label for=*"textareaname"* >Question </label>

<textarea class=*"textareadetails"* name=*"testquestion"* ng-model=*"question"* id=*"textareaname"* placeholder=*"Write Question here...."* cols=*"50px"* rows=*"4"*/></textarea>

</td>

<tr><td>

<label for=*"option\_a"*>Option A</label>

<input type=*"text"* placeholder=*"Option A"* name=*"a"* ng-model=*"optionA"*/>

</td></tr>

<tr><td>

<label for=*"option\_a"*>Option B</label>

<input type=*"text"* placeholder=*"Option B"* name=*"b"* ng-model=*"optionB"*/>

</td></tr>

<tr><td>

<label for=*"option\_a"*>Option C</label>

<input type=*"text"* placeholder=*"Option C"* name=*"c"* ng-model=*"optionC"*/>

</td></tr>

<tr><td>

<label for=*"option\_a"*>Option D</label>

<input type=*"text"* placeholder=*"Option D"* name=*"d"* ng-model=*"optionD"*/>

</td></tr>

<tr><td>

<label for=*"option\_a"*>Answer</label>

<input type=*"TEXT"* placeholder=*"Answer"* name=*"correct\_answer"* ng-model=*"answer"*/>

</td></tr>

<tr>

<td>

<button ng-click=*"submit()"* class=*"button padstyle"*>Add Question</button><button ng-click=*"upload()"* name=*"publish test"*>Publish Test</button>

<p>{{resultMsg.msg}}</p>

</td>

</tr>

</table>

<hr/>

<table class=*"table sidestyle2"*>

<thead>

<tr>

<th>Q no. </th>

<th>Question:</th>

<th>Option A</th>

<th>Option B</th>

<th>Option C</th>

<th>Option D</th>

<th>Answer</th>

</tr>

</thead>

<tbody>

<tr ng-repeat=*"q in questionListUpload"*>

<td>{{q.questionNumber}}</td>

<td>{{q.question}}</td>

<td>{{q.optionA}}</td>

<td>{{q.optionB}}</td>

<td>{{q.optionC}}</td>

<td>{{q.optionD}}</td>

<td>{{q.answer}}</td>

</tr>

</tbody>

</table>

</div>

</form>

</body>

</html>

**addTestController.js**

app.controller('uploadQuestionCtrl', **function**($scope,$http){

$scope.facultyId="sunil";

$scope.totalQuestions=0;

/\*

$scope.courseIdList = [{

value: '17',

label: 'JAVA2'

}, {

value: '18',

label: 'JAVA3'

}];

\*/

$scope.getCommon=**function**(){

$http({

method : 'POST',

//url : "courseServlet?action=SELECT"

url : "addTestServlet",

data : {'action':'COMMONVALUES','id':$scope.facultyId}

,

headers: {

'Content-Type': 'application/json'

}

}).success(**function**(data, status, headers, config) {

$scope.courseIdList = data;

}).error(**function**(data, status, headers, config) {

// called asynchronously if an error occurs

// or server returns response with an error status.

});

};

**function** submit(courseId,facultyId,testName,testTime,minMarks,totalMarks,questions){

**this**.testName=testName;

**this**.testTime=testTime;

**this**.minMarks=minMarks;

**this**.totalMarks=totalMarks;

**this**.questions=questions;

}

$scope.questionListUpload=[];

**if**(localStorage.questionList){

$scope.questionListUpload=JSON.parse(localStorage.questionList);

$scope.totalQuestions=localStorage.total;

}

$scope. submit= **function**(){

console.log('UploadQuestionCtrl Submit Call...');

$scope.totalQuestions++;

**var** questionObj=**new** Question($scope.totalQuestions,$scope.question,$scope.optionA,$scope.optionB,$scope.optionC,$scope.optionD,$scope.answer);

//$scope.questionListUpload.push({questionNumber:$scope.questionNumber,question:$scope.question,optionA:$scope.optionA,optionB:$scope.optionB,optionC:$scope.optionC,optionD:$scope.optionD,answer:$scope.answer});

$scope.questionListUpload.push(questionObj);

localStorage.questionList=JSON.stringify($scope.questionListUpload);

localStorage.total=$scope.totalQuestions;

}

$scope.upload= **function**(){

console.log('UploadQuestionCtrl Upload Call...');

**var** dataSubmit=**new** submit($scope.courseId.value,$scope.facultyId.value,$scope.testName,$scope.testTime,$scope.minMarks,$scope.totalMarks,$scope.questionListUpload);

//var dataSubmit=new submit($scope.testName,$scope.questionListUpload)

console.log("data in dataSubmit :"+JSON.stringify(dataSubmit));

$http({

method : 'POST',

//url : "courseServlet?action=SELECT"

url : "addTestServlet",

//data : {'action':'ADD',course:obj}

data : {'action':'INSERT','test':dataSubmit}

,

headers: {

'Content-Type': 'application/json'

}

}).success(**function**(data, status, headers, config) {

//$scope.courseData = data;

$scope.result=data.data;

localStorage.clear();

localStorage.total=0;

$scope.questionListUpload=[];

}).error(**function**(data, status, headers, config) {

// called asynchronously if an error occurs

// or server returns response with an error status.

});

}

**function** Question(questionNumber,question,optionA,optionB,optionC,optionD,answer){

**this**.questionNumber=questionNumber;

**this**.question=question;

**this**.optionA=optionA;

**this**.optionB=optionB;

**this**.optionC=optionC;

**this**.optionD=optionD;

**this**.answer=answer;

}

});

**addTestServlet.JAVA**

**package** com.edu.testengine.servlets;

**import** java.io.BufferedReader;

**import** java.io.IOException;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**import** java.util.ArrayList;

**import** javax.servlet.ServletException;

**import** javax.servlet.annotation.WebServlet;

**import** javax.servlet.http.HttpServlet;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

**import** javax.servlet.http.HttpSession;

**import** org.apache.catalina.Session;

**import** org.json.JSONArray;

**import** org.json.JSONException;

**import** org.json.JSONObject;

**import** com.edu.testengine.dao.courseDAO;

**import** com.edu.testengine.dao.testDAO;

**import** com.edu.testengine.dto.courseDTO;

**import** com.edu.testengine.dto.testDTO;

**import** com.edu.testengine.dto.testQuestionDTO;

**import** com.google.gson.Gson;

**import** com.google.gson.JsonArray;

**import** com.google.gson.JsonElement;

**import** com.google.gson.JsonObject;

**import** com.google.gson.JsonParser;

/\*\*

\* Servlet implementation class addTestServlet

\*/

@WebServlet("/addTestServlet")

**public** **class** addTestServlet **extends** HttpServlet {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**protected** **void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

StringBuilder sb = **new** StringBuilder();

BufferedReader br = request.getReader();

String str = **null**;

**while** ((str = br.readLine()) != **null**) {

sb.append(str);

}

System.***out***.println(sb.toString());

JsonParser parser = **new** JsonParser();

JsonObject jObj=(JsonObject) parser.parse(sb.toString());

testDTO test=**new** testDTO();

String action=jObj.get("action").getAsString();

**switch**(action){

**case** "COMMONVALUES":

response.setContentType("application/json");

HttpSession session=request.getSession(**false**);

//String user=session.getAttribute("username").toString();

String user="sunil";

response.getWriter().write(selectCommon(user));

**break**;

**case** "SELECT":

response.setContentType("application/json");

response.getWriter().write(select());

**break**;

**case** "INSERT":

JsonObject testData=jObj.get("test").getAsJsonObject();

//(courseId,facultyId,testName,questions,testTime,minMarks,totalMarks)

//test.setCourseid(testData.get("courseId").getAsInt());

//test.setFacultyID(testData.get("facultyId").getAsInt());

test.setTestName(testData.get("testName").getAsString());

test.setMinMarks(testData.get("minMarks").getAsInt());

test.setTotalMarks(testData.get("totalMarks").getAsInt());

test.setTestDuration(testData.get("testTime").getAsInt());

JsonArray questions=testData.get("questions").getAsJsonArray();

ArrayList<testQuestionDTO> questionList = **new** ArrayList<>();

**if** (questions.size() > 0) {

testQuestionDTO q;

**for** (**int** i = 0;i < questions.size();i++) {

**try** {

//JsonElement q = questions.get(i);

JsonObject object= (JsonObject)questions.get(i);

q=**new** testQuestionDTO();

//q.setQuestionNo(object.get("questionNumber").getAsInt());

q.setQuestion( object.get("question").getAsString());

q.setOptionA( object.get("optionA").getAsString());

q.setOptionB(object.get("optionB").getAsString());

q.setOptionC(object.get("optionC").getAsString());

q.setOptionD(object.get("optionD").getAsString());

q.setAnswer( object.get("answer").getAsString());

questionList.add(q);

} **catch** (Exception e) {

}

}

}

test.setQuestionList(questionList);

add(test);

**break**;

}

}

**private** String select(){

String json="";

courseDAO dao=**new** courseDAO();

ResultSet rs=**null**;

**try** {

rs=dao.select();

ArrayList<courseDTO> courseData=**new** ArrayList<>();

courseDTO obj;

**while** (rs.next()) {

obj=**new** courseDTO();

obj.setCourseID(rs.getInt("CourseId"));

obj.setTitle(rs.getString("Title"));

obj.setDetails(rs.getString("Details"));

obj.setImagePath(rs.getString("ImagePath"));

courseData.add(obj);

}

json = **new** Gson().toJson(courseData);

System.***out***.println(json);

} **catch** (ClassNotFoundException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

} **catch** (SQLException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

**return** json;

}

**private** String selectCommon(String id){

String json="";

courseDAO dao=**new** courseDAO();

ResultSet rs=**null**;

**try** {

rs=dao.select();

ArrayList<courseDTO> courseData=**new** ArrayList<>();

courseDTO obj;

**while** (rs.next()) {

obj=**new** courseDTO();

obj.setCourseID(rs.getInt("CourseId"));

obj.setTitle(rs.getString("Title"));

courseData.add(obj);

}

json = **new** Gson().toJson(courseData);

System.***out***.println(json);

} **catch** (ClassNotFoundException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

} **catch** (SQLException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

**return** json;

}

**private** String add(testDTO test){

String json="";

testDAO dao=**new** testDAO();

**try** {

**if**(dao.insert(test)){

json="{\"data\":\"Added successfully\"}";

}

**else**{

json="{\"data\":\"Not Added\"}";

}

} **catch** (ClassNotFoundException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

} **catch** (SQLException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

**return** json;

}

}

**testQuestionDTO.java**

**package** com.edu.testengine.dto;

**public** **class** testQuestionDTO {

**private** **int** questionNo;

**private** **int** questionID;

**public** **int** getQuestionID() {

**return** questionID;

}

**public** **void** setQuestionID(**int** questionID) {

**this**.questionID = questionID;

}

**public** **int** getQuestionNo() {

**return** questionNo;

}

**public** **void** setQuestionNo(**int** questionNo) {

**this**.questionNo = questionNo;

}

**private** **int** testid;

**private** String question;

**private** String optionA;

**private** String optionB;

**private** String optionC;

**public** **int** getTestid() {

**return** testid;

}

**public** **void** setTestid(**int** testid) {

**this**.testid = testid;

}

**public** String getQuestion() {

**return** question;

}

**public** **void** setQuestion(String question) {

**this**.question = question;

}

**public** String getOptionA() {

**return** optionA;

}

**public** **void** setOptionA(String optionA) {

**this**.optionA = optionA;

}

**public** String getOptionB() {

**return** optionB;

}

**public** **void** setOptionB(String optionB) {

**this**.optionB = optionB;

}

**public** String getOptionC() {

**return** optionC;

}

**public** **void** setOptionC(String optionC) {

**this**.optionC = optionC;

}

**public** String getOptionD() {

**return** optionD;

}

**public** **void** setOptionD(String optionD) {

**this**.optionD = optionD;

}

**public** String getAnswer() {

**return** answer;

}

**public** **void** setAnswer(String answer) {

**this**.answer = answer;

}

**private** String optionD;

**private** String answer;

}

**testDTO.java**

**package** com.edu.testengine.dto;

**import** java.sql.Date;

**import** java.util.ArrayList;

**public** **class** testDTO {

**private** **int** testid;

**private** **int** facultyID;

**private** **int** courseid;

**private** String testName;

**private** **int** testDuration;

**private** **int** MinMarks;

**private** **int** totalMarks;

**public** **int** getTotalMarks() {

**return** totalMarks;

}

**public** **void** setTotalMarks(**int** totalMarks) {

**this**.totalMarks = totalMarks;

}

**private** Date register\_date;

**private** ArrayList<testQuestionDTO> questionList;

**public** ArrayList<testQuestionDTO> getQuestionList() {

**return** questionList;

}

**public** **void** setQuestionList(ArrayList<testQuestionDTO> questionList) {

**this**.questionList = questionList;

}

**public** **int** getTestid() {

**return** testid;

}

**public** **void** setTestid(**int** testid) {

**this**.testid = testid;

}

**public** **int** getFacultyID() {

**return** facultyID;

}

**public** **void** setFacultyID(**int** facultyID) {

**this**.facultyID = facultyID;

}

**public** **int** getCourseid() {

**return** courseid;

}

**public** **void** setCourseid(**int** courseid) {

**this**.courseid = courseid;

}

**public** **int** getTestDuration() {

**return** testDuration;

}

**public** **void** setTestDuration(**int** testDuration) {

**this**.testDuration = testDuration;

}

**public** **int** getMinMarks() {

**return** MinMarks;

}

**public** **void** setMinMarks(**int** minMarks) {

MinMarks = minMarks;

}

**public** Date getRegister\_date() {

**return** register\_date;

}

**public** **void** setRegister\_date(Date ragister\_date) {

**this**.register\_date = ragister\_date;

}

**public** String getTestName() {

**return** testName;

}

**public** **void** setTestName(String testName) {

**this**.testName = testName;

}

}

**testDAO.java**

**package** com.edu.testengine.dao;

**import** java.sql.CallableStatement;

**import** java.sql.Connection;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**import** com.edu.testengine.dto.courseDTO;

**import** com.edu.testengine.dto.testDTO;

**import** com.edu.testengine.dto.testQuestionDTO;

**import** com.edu.testengine.utils.CommonDAO;

**public** **class** testDAO {

**public** ResultSet selectedTest(**int** id) **throws** ClassNotFoundException, SQLException{

Connection con= CommonDAO.*getConnection*();

String query = "{CALL Tests\_CRUD(?,?,?,?,?,?,?,?)}";

CallableStatement stmt = con.prepareCall(query);

stmt.setString(1, "SELECTED");

stmt.setInt(2, id);

stmt.setInt(3, 0);

stmt.setInt(4, 0);

stmt.setString(5, "");

stmt.setInt(6, 0);

stmt.setInt(7, 0);

stmt.setInt(8, 0);

ResultSet rs = stmt.executeQuery();

**return** rs;

}

**public** ResultSet select() **throws** ClassNotFoundException, SQLException{

Connection con= CommonDAO.*getConnection*();

String query = "{CALL Tests\_CRUD(?,?,?,?,?,?,?,?)}";

CallableStatement stmt = con.prepareCall(query);

stmt.setString(1, "SELECT");

stmt.setInt(2, 0);

stmt.setString(3, "");

stmt.setString(4, "");

stmt.setString(5, "");

ResultSet rs = stmt.executeQuery();

**return** rs;

}

**public** **boolean** insert(testDTO test) **throws** ClassNotFoundException, SQLException{

Connection con= CommonDAO.*getConnection*();

String query = "{CALL Tests\_CRUD(?,?,?,?,?,?,?,?)}";

CallableStatement stmt = con.prepareCall(query);

stmt.setString(1, "INSERT");

//stmt.setInt(2, test.getTestid());

//stmt.setInt(3, test.getFacultyID());

stmt.setInt(2, 0); //testID

stmt.setInt(3, 2); //facultyid

stmt.setInt(4, 17); //courseid

stmt.setString(5, test.getTestName());

stmt.setInt(6, test.getTestDuration());

stmt.setInt(7, test.getMinMarks());

stmt.setInt(8, test.getTotalMarks());

/\*

stmt.setInt(2, 0);

stmt.setInt(3, 0);

stmt.setInt(4, 0);

stmt.setString(5, "");

stmt.setInt(6, 0);

stmt.setInt(7, 0);

stmt.setInt(8, 0);

\*/

ResultSet rs=stmt.executeQuery();

**int** id=0;

**if**(rs.next()){

id=rs.getInt(1);

}

**boolean** result =**false**;

**if**(id>0){

result=**true**;

testQuestionDAO qdao=**new** testQuestionDAO();

**for** (testQuestionDTO question : test.getQuestionList()) {

qdao.insert(question,id);

}

}

**return** result;

}

**public** **boolean** update(courseDTO course) **throws** ClassNotFoundException, SQLException{

Connection con= CommonDAO.*getConnection*();

String query = "{CALL Course\_CRUD(?,?,?,?,?)}";

CallableStatement stmt = con.prepareCall(query);

stmt.setString(1, "UPDATE");

stmt.setInt(2, course.getCourseID());

stmt.setString(3, course.getTitle());

stmt.setString(4, course.getDetails());

stmt.setString(5, course.getImagePath());

**boolean** rs = stmt.execute();

**return** rs;

}

**public** **boolean** delete(courseDTO course) **throws** ClassNotFoundException, SQLException{

Connection con= CommonDAO.*getConnection*();

String query = "{CALL Course\_CRUD(?,?,?,?,?)}";

CallableStatement stmt = con.prepareCall(query);

stmt.setString(1, "DELETE");

stmt.setInt(2, course.getCourseID());

stmt.setString(3, course.getTitle());

stmt.setString(4, course.getDetails());

stmt.setString(5, course.getImagePath());

**boolean** rs = stmt.execute();

**return** rs;

}

}

**testQuestionDAO.java**

package com.edu.testengine.dao;

import java.sql.CallableStatement;

import java.sql.Connection;

import java.sql.ResultSet;

import java.sql.SQLException;

import com.edu.testengine.dto.courseDTO;

import com.edu.testengine.dto.testQuestionDTO;

import com.edu.testengine.utils.CommonDAO;

public class testQuestionDAO {

public ResultSet selectedTest(int id) throws ClassNotFoundException, SQLException{

Connection con= CommonDAO.getConnection();

String query = "{CALL TestQuestions\_CRUD(?,?,?,?,?,?,?,?,?)}";

CallableStatement stmt = con.prepareCall(query);

stmt.setString(1, "SELECTED");

stmt.setInt(2, 0);

stmt.setInt(3, id);

stmt.setString(4, "");

stmt.setString(5, "");

stmt.setString(6, "");

stmt.setString(7, "");

stmt.setString(8, "");

stmt.setString(9, "");

ResultSet rs = stmt.executeQuery();

return rs;

}

public ResultSet select() throws ClassNotFoundException, SQLException{

Connection con= CommonDAO.getConnection();

String query = "{CALL TestQuestions\_CRUD(?,?,?,?,?,?,?,?,?)}";

CallableStatement stmt = con.prepareCall(query);

stmt.setString(1, "SELECT");

stmt.setInt(2, 0);

stmt.setString(3, "");

stmt.setString(4, "");

stmt.setString(5, "");

ResultSet rs = stmt.executeQuery();

return rs;

}

public boolean insert(testQuestionDTO object,int id) throws ClassNotFoundException, SQLException{

Connection con= CommonDAO.getConnection();

String query = "{CALL TestQuestions\_CRUD(?,?,?,?,?,?,?,?,?)}";

CallableStatement stmt = con.prepareCall(query);

stmt.setString(1, "INSERT");

stmt.setInt(2, 0);

stmt.setInt(3, id);

stmt.setString(4,object.getQuestion());

stmt.setString(5,object.getOptionA());

stmt.setString(6,object.getOptionB());

stmt.setString(7,object.getOptionC());

stmt.setString(8,object.getOptionD());

stmt.setString(9,object.getAnswer());

boolean rs = stmt.execute();

return rs;

}

public boolean update(courseDTO course) throws ClassNotFoundException, SQLException{

Connection con= CommonDAO.getConnection();

String query = "{CALL TestQuestions\_CRUD(?,?,?,?,?,?,?,?,?)}";

CallableStatement stmt = con.prepareCall(query);

stmt.setString(1, "UPDATE");

stmt.setInt(2, course.getCourseID());

stmt.setString(3, course.getTitle());

stmt.setString(4, course.getDetails());

stmt.setString(5, course.getImagePath());

boolean rs = stmt.execute();

return rs;

}

public boolean delete(courseDTO course) throws ClassNotFoundException, SQLException{

Connection con= CommonDAO.getConnection();

String query = "{CALL TestQuestions\_CRUD(?,?,?,?,?,?,?,?,?)}";

CallableStatement stmt = con.prepareCall(query);

stmt.setString(1, "DELETE");

stmt.setInt(2, course.getCourseID());

stmt.setString(3, course.getTitle());

stmt.setString(4, course.getDetails());

stmt.setString(5, course.getImagePath());

boolean rs = stmt.execute();

return rs;

}

}

**teststart.jsp**

<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*

pageEncoding=*"ISO-8859-1"*%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>

<meta charset=*"utf-8"*>

<meta http-equiv=*"X-UA-Compatible"* content=*"IE=edge"*>

<meta name=*"viewport"* content=*"width=device-width, initial-scale=1"*>

<title>Test :</title>

<link href=*"css/bootstrap.min.css"* rel=*"stylesheet"*>

<link rel=*"stylesheet"* href=*"css/question.css"*>

<script type=*"application/javascript"* src=*"js/jquery-1.12.3.min.js"*></script>

<script type=*"application/javascript"* src=*"js/bootstrap.min.js"*></script>

<script src=*"controller/angular.min.js"*></script>

<script src=*"controller/mainController.js"*></script>

<script src=*"controller/testStartController.js"*></script>

</head>

<body class=*"container"*>

<form method=*"post"* ng-app=*"myApp"* ng-controller=*"testController"* ng-init=*"loadTest()"*>

<div ng-show=*"testVisible"* >

<div class=*"head"*>

<header class=*"group"* >

<div class=*"studentdetails"*>

<ul>

<li><img src=*"images/sunil.jpg"* alt=*"sunil.jpg"*></li></h4>

<li><h4>Name : Sunil</h4></li>

<li><h4>Exam Test Name: {{testDetails.testName}}</h4></li>

<li><h4>Exam Test ID: {{testDetails.testid}}</h4></li>

<li><h4>Test Duration: {{testDetails.testTime}}</h4></li>

<li><h4>Minimum Marks: {{testDetails.minMarks}}</h4></li>

<li><h4>Total Marks: {{testDetails.totalMarks}}</h4></li>

<li><h4>Time Left : <label ng-bind=*"Message"*></label></h4></li>

</ul>

</div>

<div class=*"navigation-menu"*>

</div>

</header>

</div>

<div id=*"content"*>

<main class=*"main"*>

<div class=*"question-contant"*>

<h1><strong>Question {{questionNo}} : </strong>{{question}}</h1>

<h3><input type=*"radio"* name=*"question"* ng-click=*"answered('A')"*> {{optionA}}<br></h3>

<h3><input type=*"radio"* name=*"question"* ng-click=*"answered('B')"*> {{optionB}}<br></h3>

<h3><input type=*"radio"* name=*"question"* ng-click=*"answered('C')"*> {{optionC}}<br></h3>

<h3><input type=*"radio"* name=*"question"* ng-click=*"answered('D')"*> {{optionD}}</h3>

</p>

</div>

</main>

<aside class=*"sidebar"*>

<Ul>

<li>

<h3>Missed</h3>

</li>

<li>

<h3>Attemped</h3>

</li>

<li>

<h3>Wrong</h3>

</li>

<li>

<h3>Left</h3>

</li>

</Ul>

</aside>

</div>

<footer class=*"footeroption"*>

<button value=*"Previous"* ng-click=*"previousQ()"*>Previous</button><button value=*"Next"* ng-click=*"nextQ()"*>Next</button><button value=*"Submit"* ng-click=*"submit()"*>Submit</button>

</footer>

</div>

<div ng-hide=*"testVisible"* style="font-size: *large*;text-align: *center*;width: *800px*">

<div style="text-align: *center*;">

Total marks : <b>{{result.totalMarks}} </b><br>

Marks : <b>{{result.marks}} </b><br>

Correct answer : <b>{{result.correctAnswer}} </b><br>

Wrong answer :<b> {{result.wrongAnswer}}</b><br>

Skipped question : <b>{{result.skipQuestion}}</b><br>

Result : <b>{{result.status}}</b><br>

<br> <br>

</div>

<div style="text-align: *center*;width: *800px*" >

<table border=*"2"* style="text-align: *center*;width: *600px*" >

<tr>

<td style="padding: *5px*;">

Question No :

</td>

<td style="padding: *5px*;">

Status :

</td>

</tr>

<tr ng-repeat=*"question in result.result"* align=*"center"*>

<td>

Q . {{question.questionNo}}

</td>

<td>

{{question.result}}

</td>

</tr>

</table>

</div>

</div>

</form>

</body>

</html>

**testStartController.js**

app.controller('testController', **function** ($scope,$http, $interval, $filter) {

console.log("entered in controller");

$scope.testVisible=**true**;

$scope.count=-1;

$scope.questionList=[];

$scope.loadTest=**function**(){

console.log("controller called");

//timerCode

//$scope.callAtInterval();

**if**(localStorage.questionList){

console.log("inside if value of count"+$scope.count);

$scope.questionList=JSON.parse(localStorage.getItem("questionList"));

$scope.count=parseInt(localStorage.getItem("currentQuestion"));

console.log("inside if value of count after local"+$scope.count);

$scope.testDetails=JSON.parse(localStorage.getItem("testDetails"));

$scope.StartTimer();

$scope.nextQ();

}

**else**{

$http({

method : 'POST',

//url : "courseServlet?action=SELECT"

url : "getTestServlet",

data : {'action':'SELECT'}

,

headers: {

'Content-Type': 'application/json'

}

}).success(**function**(data, status, headers, config) {

**var** dataSubmit=**new** submit(data.courseid,data.facultyID,data.testName,data.testDuration,data.MinMarks,data.totalMarks,data.testid);

localStorage.testDetails=JSON.stringify(dataSubmit);

localStorage.questionList=JSON.stringify(data.questionList);

//localStorage.currentQuestion=-1;

//$scope.count=localStorage.getItem("currentQuestion");

$scope.testDetails=JSON.parse(localStorage.getItem("testDetails"));

$scope.questionList=JSON.parse(localStorage.getItem("questionList"));

$scope.StartTimer();

$scope.nextQ();

//$scope.testOperation.questionList = data;

}).error(**function**(data, status, headers, config) {

// called asynchronously if an error occurs

// or server returns response with an error status.

});

}

};

$scope.Timer = **null**;

//$scope.finishTime=Date().setMinutes(new Date().getMinutes()+$scope.testDetails.testTime);

$scope.finishTime=**new** Date().setMinutes(**new** Date().getMinutes());

//Timer start function.

$scope.StartTimer = **function** () {

//Set the Timer start message.

//$scope.Message = "Timer started. ";

//Initialize the Timer to run every 1000 milliseconds i.e. one second.

$scope.Timer = $interval(**function** () {

$scope.startTime=**new** Date();

//Display the current time.

finishMinutes=$scope.finishTime-$scope.startTime;

//var time = $filter('date')(new Date(), 'mm:ss');

**var** time = $filter('date')(finishMinutes, 'mm:ss');

$scope.Message = time;

}, 1000);

};

$scope.callAtInterval = **function**() {

//console.log("$scope.callAtInterval - Interval occurred");

$scope.submit();

}

//$interval( function(){ $scope.callAtInterval(); }, 1000\*$scope.testDetails.testTime);

$interval( **function**(){ $scope.callAtInterval(); }, 1000\*60\*30);

/\*

//Timer stop function.

$scope.StopTimer = function () {

//Set the Timer stop message.

$scope.Message = "Timer stopped.";

//Cancel the Timer.

if (angular.isDefined($scope.Timer)) {

$interval.cancel($scope.Timer);

}

};

\*/

$scope.nextQ=**function**(){

console.log("next called");

$scope.count++;

$scope.questionNo=$scope.questionList[$scope.count].questionNo;

$scope.question=$scope.questionList[$scope.count].question;

$scope.optionA=$scope.questionList[$scope.count].optionA;

$scope.optionB=$scope.questionList[$scope.count].optionB;

$scope.optionC=$scope.questionList[$scope.count].optionC;

$scope.optionD=$scope.questionList[$scope.count].optionD;

localStorage.currentQuestion=$scope.count-1;

};

$scope.previousQ=**function**(){

console.log("previous called");

$scope.count--;

$scope.questionNo=$scope.questionList[$scope.count].questionNo;

$scope.question=$scope.questionList[$scope.count].question;

$scope.optionA=$scope.questionList[$scope.count].optionA;

$scope.optionB=$scope.questionList[$scope.count].optionB;

$scope.optionC=$scope.questionList[$scope.count].optionC;

$scope.optionD=$scope.questionList[$scope.count].optionD;

localStorage.currentQuestion=$scope.count-1;

};

$scope.answered=**function**(val){

$scope.questionList[$scope.count].answer=val;

console.log($scope.questionList[$scope.count].answer);

localStorage.currentQuestion=$scope.count-1;

localStorage.questionList=JSON.stringify($scope.questionList);

};

$scope.submit=**function**(){

console.log("submit called");

$http({

method : 'POST',

//url : "courseServlet?action=SELECT"

url : "getTestServlet",

//data : {'action':'ADD',course:obj}

data : {'action':'SUBMIT','userid':'sunil', 'testDetails':$scope.testDetails,'questionList':JSON.parse(localStorage.getItem("questionList"))}

,

headers: {

'Content-Type': 'application/json'

}

}).success(**function**(data, status, headers, config) {

//$scope.courseData = data;

$scope.result=data;

$scope.testVisible=**false**;

localStorage.result=JSON.stringify($scope.result);

}).error(**function**(data, status, headers, config) {

// called asynchronously if an error occurs

// or server returns response with an error status.

});

};

});

**function** submit(courseId,facultyId,testName,testTime,minMarks,totalMarks,testid){

**this**.courseId=courseId;

**this**.facultyId=facultyId;

**this**.testName=testName;

**this**.testTime=testTime;

**this**.minMarks=minMarks;

**this**.totalMarks=totalMarks;

**this**.testid=testid;

}

**getTestServlet.java**

**package** com.edu.testengine.servlets;

**import** java.io.BufferedReader;

**import** java.io.IOException;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**import** java.util.ArrayList;

**import** javax.servlet.ServletException;

**import** javax.servlet.annotation.WebServlet;

**import** javax.servlet.http.HttpServlet;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

**import** com.edu.testengine.dao.courseDAO;

**import** com.edu.testengine.dao.testDAO;

**import** com.edu.testengine.dao.testQuestionDAO;

**import** com.edu.testengine.dao.usertestResultDAO;

**import** com.edu.testengine.dto.courseDTO;

**import** com.edu.testengine.dto.finalResultDTO;

**import** com.edu.testengine.dto.resultDTO;

**import** com.edu.testengine.dto.testDTO;

**import** com.edu.testengine.dto.testQuestionDTO;

**import** com.edu.testengine.dto.usertestResultDTO;

**import** com.google.gson.Gson;

**import** com.google.gson.JsonArray;

**import** com.google.gson.JsonObject;

**import** com.google.gson.JsonParser;

/\*\*

\* Servlet implementation class getTestServlet

\*/

@WebServlet("/getTestServlet")

**public** **class** getTestServlet **extends** HttpServlet {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

/\*\*

\* **@see** HttpServlet#HttpServlet()

\*/

**public** getTestServlet() {

**super**();

// **TODO** Auto-generated constructor stub

}

**protected** **void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

StringBuilder sb = **new** StringBuilder();

BufferedReader br = request.getReader();

String str = **null**;

**while** ((str = br.readLine()) != **null**) {

sb.append(str);

}

System.***out***.println(sb.toString());

JsonParser parser = **new** JsonParser();

JsonObject jObj=(JsonObject) parser.parse(sb.toString());

String action=jObj.get("action").getAsString();

**switch**(action){

**case** "SELECT":

response.setContentType("application/json");

**int** testID=4;

response.getWriter().write(select(testID));

**break**;

**case** "SUBMIT":

JsonObject testData=jObj.get("testDetails").getAsJsonObject();

testDTO test=**new** testDTO();

//(courseId,facultyId,testName,questions,testTime,minMarks,totalMarks)

String userid=jObj.get("userid").getAsString();

test.setCourseid(testData.get("courseId").getAsInt());

test.setFacultyID(testData.get("facultyId").getAsInt());

test.setTestid(testData.get("testid").getAsInt());

test.setTestName(testData.get("testName").getAsString());

test.setMinMarks(testData.get("minMarks").getAsInt());

test.setTotalMarks(testData.get("totalMarks").getAsInt());

test.setTestDuration(testData.get("testTime").getAsInt());

JsonArray questions=jObj.get("questionList").getAsJsonArray();

ArrayList<testQuestionDTO> questionList = **new** ArrayList<>();

**if** (questions.size() > 0) {

testQuestionDTO q;

**for** (**int** i = 0;i < questions.size();i++) {

**try** {

//JsonElement q = questions.get(i);

JsonObject object= (JsonObject)questions.get(i);

q=**new** testQuestionDTO();

//q.setQuestionNo(object.get("questionNumber").getAsInt());

q.setQuestion( object.get("question").getAsString());

q.setQuestionNo(object.get("questionNo").getAsInt());

q.setQuestionID(object.get("questionID").getAsInt());

q.setOptionA( object.get("optionA").getAsString());

q.setOptionB(object.get("optionB").getAsString());

q.setOptionC(object.get("optionC").getAsString());

q.setOptionD(object.get("optionD").getAsString());

q.setAnswer( object.get("answer").getAsString());

questionList.add(q);

} **catch** (Exception e) {

}

}

}

test.setQuestionList(questionList);

response.getWriter().write(submit(test,userid));

//submit(test,userid);

**break**;

}

}

**private** String submit(testDTO test,String userid){

String json="";

testQuestionDAO dao=**new** testQuestionDAO();

ResultSet rs=**null**;

**try** {

rs=dao.selectedTest(test.getTestid());

ArrayList<testQuestionDTO> questionList=**new** ArrayList<>();

testQuestionDTO obj;

**int** qno=1;

**while** (rs.next()){

obj=**new** testQuestionDTO();

obj.setQuestionNo(qno);

obj.setQuestionID(rs.getInt("QuestionId"));

obj.setQuestion(rs.getString("Question"));

obj.setOptionA(rs.getString("OptionA"));

obj.setOptionB(rs.getString("OptionB"));

obj.setOptionC(rs.getString("OptionC"));

obj.setOptionD(rs.getString("OptionD"));

obj.setAnswer(rs.getString("Answer"));

questionList.add(obj);

qno+=1;

}

ArrayList<testQuestionDTO> userQ=test.getQuestionList();

ArrayList<resultDTO> result=**new** ArrayList<>();

resultDTO res;

**int** totalMarks=100;

**int** perQuestionMark=10;

**int** marks=0;

**int** minMarks=50;

**int** correctAnswer=0;

**int** wrongAnswer=0;

**int** skipQuestion=0;

String status;

**for** (testQuestionDTO q : userQ) {

**for** (testQuestionDTO r : questionList) {

**if**(q.getQuestionID()==r.getQuestionID()){

res=**new** resultDTO();

res.setQuestionNo(q.getQuestionNo());

**if**(q.getAnswer().equalsIgnoreCase(r.getAnswer())){

res.setResult("correct");

correctAnswer++;

}

**else** **if**(q.getAnswer().trim().equals("") ||q.getAnswer()==**null**){

res.setResult("not answered");

skipQuestion++;

}

**else**{

res.setResult("wrong answer");

wrongAnswer++;

}

result.add(res);

**break**;

}

}

}

marks=correctAnswer\*perQuestionMark;

**if**(marks>=minMarks){

status="pass";

}

**else**{

status="fail";

}

finalResultDTO finalRes=**new** finalResultDTO();

finalRes.setCorrectAnswer(correctAnswer);

finalRes.setMarks(marks);

finalRes.setSkipQuestion(skipQuestion);

finalRes.setStatus(status);

finalRes.setTotalMarks(totalMarks);

finalRes.setWrongAnswer(wrongAnswer);

finalRes.setResult(result);

usertestResultDTO testResult=**new** usertestResultDTO();

testResult.setMarks(marks);

testResult.setResultid(0);

testResult.setTestid(test.getTestid());

testResult.setUserName("sunil");

usertestResultDAO resultdao=**new** usertestResultDAO();

resultdao.insert(testResult);

json = **new** Gson().toJson(finalRes);

System.***out***.println(json);

} **catch** (ClassNotFoundException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

} **catch** (SQLException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

**return** json;

}

**private** String select(**int** testID){

String json="";

testDTO test=**new** testDTO();

testDAO testdao=**new** testDAO();

testQuestionDAO dao=**new** testQuestionDAO();

ResultSet rs=**null**;

**try** {

rs=testdao.selectedTest(testID);

**while** (rs.next()){

test.setTestid(rs.getInt("TestId"));

test.setFacultyID(rs.getInt("FacultyId"));

test.setCourseid(rs.getInt("CourseId"));

test.setTestName(rs.getString("TestName"));

test.setTestDuration(rs.getInt("testDuration"));

test.setMinMarks(rs.getInt("MinMarks"));

test.setTotalMarks(rs.getInt("TotalMarks"));

}

rs=dao.selectedTest(testID);

ArrayList<testQuestionDTO> questionList=**new** ArrayList<>();

testQuestionDTO obj;

**int** qno=1;

**while** (rs.next()){

obj=**new** testQuestionDTO();

obj.setQuestionNo(qno);

obj.setQuestionID(rs.getInt("QuestionId"));

obj.setQuestion(rs.getString("Question"));

obj.setOptionA(rs.getString("OptionA"));

obj.setOptionB(rs.getString("OptionB"));

obj.setOptionC(rs.getString("OptionC"));

obj.setOptionD(rs.getString("OptionD"));

obj.setAnswer("");

questionList.add(obj);

qno+=1;

}

test.setQuestionList(questionList);

json = **new** Gson().toJson(test);

System.***out***.println(json);

} **catch** (ClassNotFoundException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

} **catch** (SQLException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

**return** json;

}

}

**userTestResultDAO.java**

package com.edu.testengine.dao;

import java.sql.CallableStatement;

import java.sql.Connection;

import java.sql.ResultSet;

import java.sql.SQLException;

import com.edu.testengine.dto.testDTO;

import com.edu.testengine.dto.testQuestionDTO;

import com.edu.testengine.dto.usertestResultDTO;

import com.edu.testengine.utils.CommonDAO;

public class usertestResultDAO {

public boolean insert(usertestResultDTO test) throws ClassNotFoundException, SQLException{

Connection con= CommonDAO.getConnection();

String query = "{CALL UserTestResults\_CRUD(?,?,?,?,?)}";

CallableStatement stmt = con.prepareCall(query);

stmt.setString(1, "INSERT");

stmt.setInt(2, test.getResultid()); //testID

stmt.setString(3, test.getUserName()); //facultyid

stmt.setInt(4, test.getTestid()); //courseid

stmt.setInt(5,test.getMarks());

ResultSet rs=stmt.executeQuery();

int id=0;

if(rs.next()){

id=rs.getInt(1);

}

boolean result =false;

if(id>0){

result=true;

}

return result;

}

}

**CHAPTER 5**

**IMPLEMENTATION AND TESTING**

**TESTING**

**Testing Methodology**

System testing is a critical aspect of Software Quality Assurance and represents the ultimate review of specification, design and coding. Testing is a process of executing a program with the intent of finding an error. A good test is one that has a probability of finding an as yet undiscovered error. The purpose of testing is to identify and correct bugs in the developed system. Nothing is complete without testing.

Testing is the vital to the success of the system.  
In the code testing the logic of the developed system is tested. For this every module of the program is executed to find an error. To perform specification test, the examination of the specifications stating what the program should do and how it should perform under various conditions.

Unit testing focuses first on the modules in the proposed system to locate errors. This enables to detect errors in the coding and logic that are contained within that module alone. Those resulting from the interaction between modules are initially avoided. In unit testing step each module has to be checked separately.

System testing does not test the software as a whole, but rather than integration of each module in the system. The primary concern is the compatibility of individual modules. One has to find areas where modules have been designed with different specifications of data lengths, type and data element name.

Testing and validation are the most important steps after the implementation of the developed system. The system testing is performed to ensure that there are no errors in the implemented system. The software must be executed several times in order to find out the errors in the different modules of the system.  
Validation refers to the process of using the new software for the developed system in a live environment i.e., new software inside the organization, in order to find out the errors.  
The validation phase reveals the failures and the bugs in the developed system. It will be come to know about the practical difficulties the system faces when operated in the true environment. By testing the code of the implemented software, the logic of the program can be examined. A specification test is conducted to check whether the specifications stating the program are performing under various conditions. Apart from these tests, there are some special tests conducted which are given below:

1. **Peak Load Tests:**  This determines whether the new system will handle the volume of activities when the system is at the peak of its processing demand. The test has revealed that the new software for the agency is capable of handling the demands at the peak time.
2. **Storage Testing:** This determines the capacity of the new system to store transaction data on a disk or on other files. The proposed software has the required storage space available, because of the use of a number of hard disks.
3. **Performance Time Testing:** This test determines the length of the time used by the system to process transaction data.

In this phase the software developed Testing is exercising the software to uncover errors and ensure the system meets defined requirements.

Testing may be done at different levels:-

* + - 1. Unit level testing
      2. Module level testing
      3. Integration and System testing
      4. Black Box Testing
      5. White Box Testing
      6. Acceptance Testing

**Unit level testing**

A Unit corresponds to a screen /form in the package. Unit testing focuses on verification of the corresponding class or Screen. This testing includes testing of control paths, interfaces, local data structures, logical decisions, boundary conditions, and error handling. Unit testing may use Test Drivers, which are control programs to co-ordinate test case inputs and outputs, and Test stubs, which replace low-level modules. A stub is a dummy subprogram.

**Module level testing**

Module Testing is done using the test cases prepared earlier. Module is defined during the time of design.

**Integration and System testing**

Integration testing is used to verify the combining of the software modules. Integration testing addresses the issues associated with the dual problems of verification and program construction. System testing is used to verify, whether the developed system meets the requirements.

**Black-Box testing**

Black-box testing is a method of software testing that examines the functionality of an application without peering into its internal structures or workings. This method of test can be applied to virtually every level of software testing: unit, integration, system and acceptance.

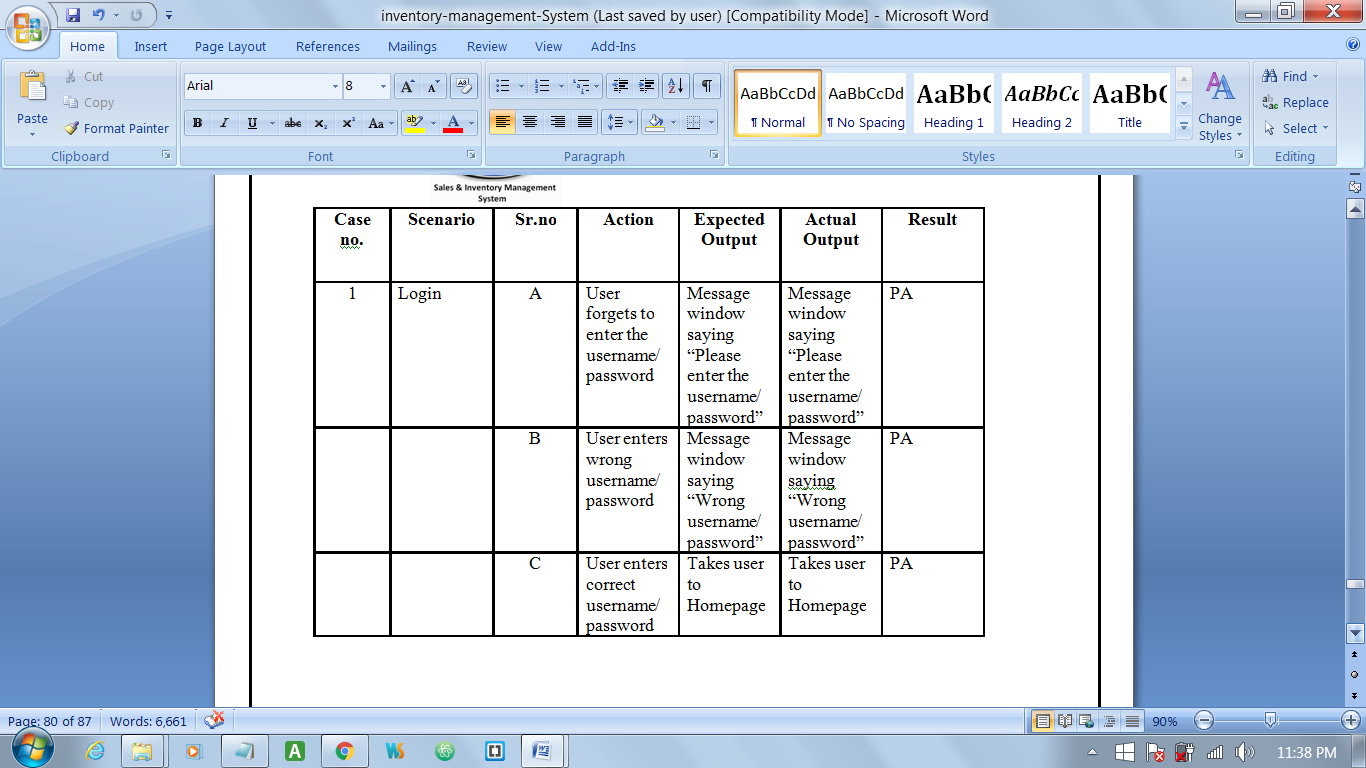
**White-Box testing**

White-box testing (also known as clear box testing, glass box testing, transparent box testing, and structural testing) is a method of testing software that tests internal structures or workings of an application, as opposed to its functionality (i.e. black-box testing).

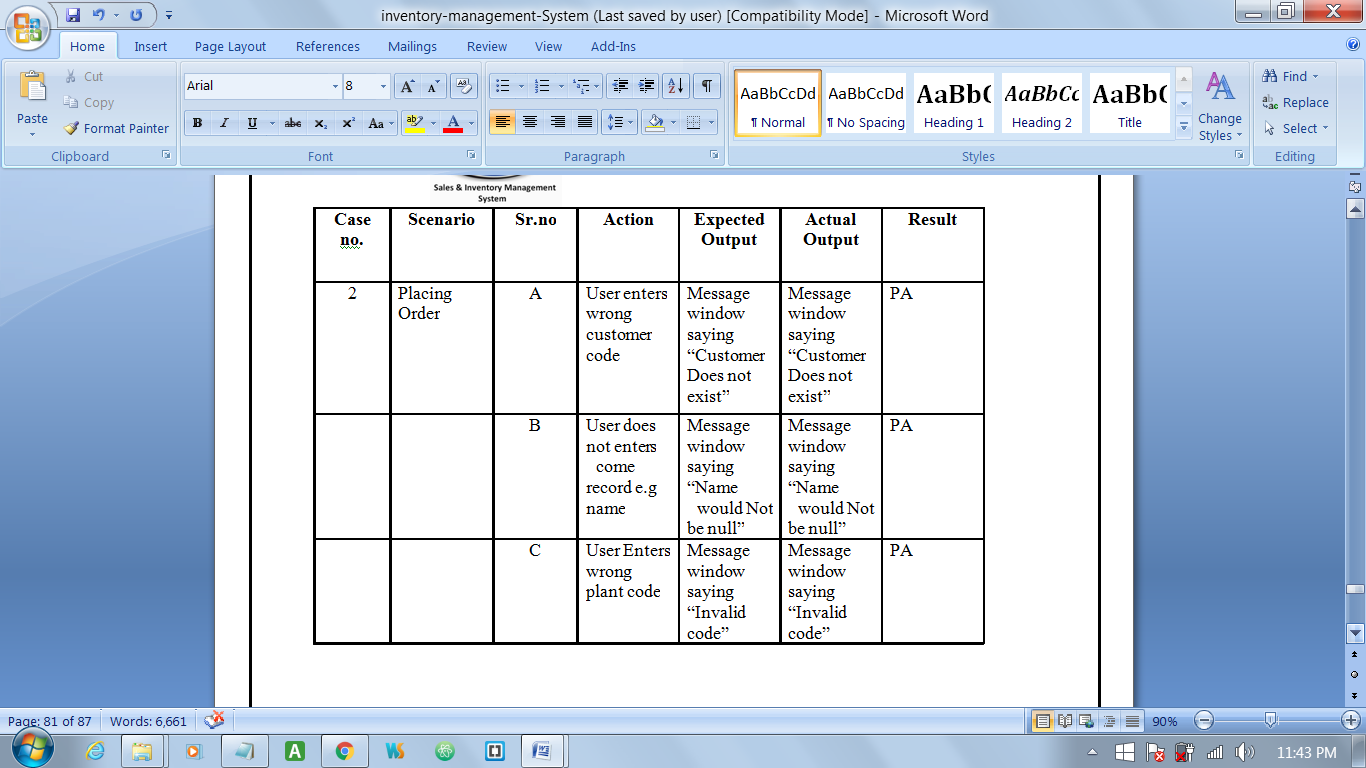
**Acceptance testing**

The acceptance testing part of a testing methodology is the final phase of functional software testing and involves making sure that all the product/project requirements have been met and that the end-users and customers have tested the system to make sure it operates as expected and meets all their defined requirements.

**TEST DATA AND TEST CASES**

****

**Fig: 4.1**

****

**Fig: 4.2**

**CHAPTER 6**

**SYSTEM SECURITY MEASURES**

**System Security Measures**

Security is the most important part of any system. It can be either the security of system program functionalities or underlying database. We have very cautious process of authentication of user that no one could change its contents in unauthorized manner.

Security and integrity of database are very important for any software system because databases are the backbone of the system. Security need to be implemented at every level of the system so that only authorized user can access the system for updating and other significance process.

Entering correct password while opening the system or we can say that entering the system is the process of authentication. If anyone is entering the password is wrong then he/she cannot access the system for any change purpose. The main purpose of the security is to save system from accidentally changes or loss of information or also getting wrong information. The system administrator is the person that can change the information or update the information. He can also grant the permission that who has to enter the system and what can he do. So security is the most important topic to be concerned.

**Data Security:**

Data security is protection of data in the database or any storage. In the case of database the term data integrity and data security confuse the users some time. They understand both terms the same but these both term have a main difference.

Security means protecting data against authorized.

**Rights and Permissions**:

The more you trusted, the more access you will get for example of you are fully trusted them you will find the permission to access the network as well as the database (an case of web-based Application).

The word permission is not same as right in the case of database (MYSQL Server). Permission is what you have been given by other to do. But right is what you have your own. So we can say that one has a list of permission to access and manipulate object.

The permission on the projects may of following type.

* Read Permission.
* Write Permission
* Delete Permission

I have used two mode of security in my project.

1. Authentication
2. Authorization

**Authentication:**

Authentication is the process of checking valid user. One cannot claim to be an owner of database that is under authority of other. You have to first prove your ownership that you are owner you ownership can only be contested, however after identity is verified. The process of verifying your identity is called authentication.

My project “Online Test Engine” I have username and password to verify the identity i.e. for authentication process. The user’s once entering valid username and password. User can access select (Admin) process.

**Authorization:**

Authorization is the process of specifying the permission for authorized user whether they can modify data or not.

I have implemented it in my project by specifying membership.

I have used MVC to design the project. It helps to find the error of project easily. Which reduce the bug finding time.

Anything changed in this file affect all the use

**Security Criteria in My Project:-**  I have used both authentication, encryption and autherisation process in my project. In JAVA there are three types of authentication process these are

* Form Authentication
* Window Authentication

In the form authentication we protect the particular form accessed from the unautherised user.

In Window authentication we protect the particular application from the unautherised user.

**Encryption**

Encryption is used in the project which make the student, faculty, and admin password hacking form the herker.

**CHAPTER 7:**

**COST ESTIMATION OF PROJECT**

**Cost Estimation**

Cost estimation is one very important activity. Today, Software is the most expensive element in many computer-based systems. Large cost estimation errors can make the difference between profit and loss. Cost overrun can be disastrous for the developers.

Software cost and effort estimation will never be an exact science. To many variables such as human, technical, environmental can affect the ultimate cost of software and effort applied to develop it.

To achieve reliable cost estimation a number of options are there:

Delay estimation until late in project.

Use relatively simple decomposition techniques to generate project cost and effort estimations.

Develop an empirical model for software cost and effort.

LOC and FP can be the basic data from which productivity metrics can be computed. LOC, Line of code and FP, Function Point are distinct estimation techniques. The LOC and FP techniques differ in the level of details required for decomposition. When LOC is used as the estimation variable, function decomposition is absolutely essential. The expected value for LOC and FP is computed. The expected value for the estimation variable E can be computed as weighted average of the optimistic (a), most likely (m) and pessimistic (b) LOC or FP estimation.

E=(A+4M+B)/6

Gives highest credence to the most likely estimation

**Cost Estimation**

Software cost comprises a small percentage of overall computer-based system cost. There are a number of factors, which are considered, that can affect the ultimate cost of the software such as - human, technical, Hardware and Software availability etc.

The main point that was considered during the cost estimation of project was its sizing. In spite of complete software sizing, function point and approximate lines of code were also used to "size" each element of the Software and their costing.

The cost estimation done by me for Project also depend upon the baseline metrics collected from past projects and these were used in conjunction with estimation variables to develop cost and effort projections.

We have basically estimated this project mainly on two bases -

**1) Effort Estimation -** This refers to the total man-hours required for the development of the project. It even includes the time required for doing documentation and user manual.

**2) Hardware Required Estimation -** This includes the cost of the PCs and the hardware cost required for development of this project.

**1. EFFORT ESTIMATION**

|  |  |
| --- | --- |
| Project Activities | Person Days |
| **System Requirement Analysis and its Feasibility study** | 20 |
| System Study  Preparation of the High Level Design of the system  Preparation of Low Level design –  Procedural Design  Database Design | 15 |
| 8 |
| 5 |
| **Designing Of Input/output screens** | 16 |
| Coding –  Front End Coding  Back End Coding | 35 |
| 15 |
| **Preparation of test cases and Test data** | 8 |
| Unit Testing  Integration Testing  Acceptance Testing | 15 |
| 18 |
| **Documentation** | **9** |
| Total Person days | 156 |
| Total Person hours | 1064 |
| Person Cost **(@45/- Per Hour Person Cost)** | Rs 14,560.00 |
| Guide Cost | Rs 3,500.00 |
| Total Cost | Rs 18,060.00 |

**CHAPTER 8:**

**FUTURE SCOPE AND FURTHER ENHANCEMENT**

**FUTURE SCOPE**

Application scope dependent the data and control to be processed function performance, constraints, interfaces and reliability. Function describes in the Statement of scope are evaluated and in some case refined to provide more detail prior to the beginning of the estimation. Because both cost and schedule estimates are functionally oriented, some degree of decomposition is often useful.

We can implement easily this application. Reusability is possible as and when we require in this application. We can update it In near Future. We can add new features as and when we require. There is flexibility in all the modules. Scope of this document is to put down the requirements, clearly identifying the information needed by the user, the source of the information and outputs expected from the system.

It is directly dependent on the lay stone of the project that is we will have to design an application which when the time passes having a better Application initially should not become a joke later.

It is highly likely that the scope will change as the web application project moves forward; the web process model should be incremental. This allows the development team to “freeze” the scope for one increment so that an operational web application release can be created. The next increment may scope changes suggested by a review of the preceding increment, but once the second increment commences, scope is again frozen temporarily. This approach enables the Web App team to work without having to accommodate a continual stream of changes but still recognizes the continuous evolution characteristics of most web application. Besides that, the following basic quality in the Website always safeguards the future scope of the Website

**CHAPTER: 9**

**CONCLUSION AND BIBLIOGRAPHY**

**CONCLUSION**

While developing the system a conscious effort has been made to create and develop a software package, making use of available tools, techniques and resources – that would generate a proper System. While making the system, an eye has been kept on making it as user-friendly, as cost-effective and as flexible as possible. As such one may hope that the system will be acceptable to any user and will adequately meet his/her needs.

As in case of any system development processes where there are a number of short comings, there have been some shortcomings in the development of this system also. The project is still under modification.

This system is to provide better optimization, cost reduction for the users of this system and to process inventory information in an online environment, so specific online procedures are performed in seconds rather than the hours required by traditional batch methods of computer processing. This IM system performs the basic inventory management objectives by providing a complete and accurate record of all stock transactions affecting each item or product in inventory.

* This system is to keep records of the complete inventory.
* It support for inventory management helps us record and track materials on the basis of both quantity and value.
* For warehouse management, inventory manager can track quantity and value of all our materials, perform physical inventory, and optimize our warehouse resources.
* Automatically update the inventory from completed sales orders in the System so manual entry of withdrawals (sales) is minimized.
* Maintain Product Master Records on each item in the inventory that is handled by the company, including description, unit price, list price, Item-Group code, cost, quantity on-hand and other data fields.
* Handle multiple warehouses by defining a separate Warehouse Code.
* Print management reports, such as Stock Status Report, Product Activity Report and others, to provide an audit trail of stock transactions entered into the IM System etc.

**BIBLIOGRAPHY**

**WEB LINKS:**

* <https://www.lynda.com/Java-tutorials/Java-Advanced-Training/107061-2.html>
* <http://www.wideskills.com/jdbc-tutorial/introduction-to-jdbc>
* <http://www.java-made-easy.com/java-programming-tutorials.html>
* <http://www.dickbaldwin.com/tocadv.htm>
* <http://getbootstrap.com/>
* <http://fontawesome.io/?utm_source=hackernewsletter>
* <https://www.codecademy.com/learn/javascript>
* <http://www.tutorialspoint.com/javascript/>

**BOOKS REFERRED:**

# Java SE 8 for Programmers, **by** Paul Deitel and Harvey Deitel, **Publisher:** Prentice Hall, **Published Date:** April 9, 2014

# [Java 8](http://www.goodreads.com/book/show/21944284-java-8): The Fundamentals by [Dane Cameron](http://www.goodreads.com/author/show/7419604.Dane_Cameron), Kindle Edition, 204 pages, Published March 15th 2014 by Cisdal Publishing, ASIN: B00J1BQ4G8 Edition Language: English

* **Murach's Java Programming by Joel Murach**, Published November 2011,

ISBN: 978-1-890774-65-3

* **Java The Complete Reference, 8th Edition by Herbert Schildt**
* **Software Engineering: A Practitioner's Approach, 7/e, Roger S Pressman, R. S. Pressman & Associates, Inc., ISBN:** 0073375977, **Copyright year:** 2010

### APPENDIX A

### Acronyms

* DFD – Data Flow Diagram
* DOB – Date of Birth
* DOM – Date of Marriage
* FD – Functional Dependencies
* IP – Internet Protocol
* LAN -- Local Area Network
* NF-- Normal Form
* PV – Present Value
* RDBMS—Relational Database Management System
* SCM – Software Configuration Management
* SDLC—Software Development Life Cycle
* SRS-- Software Requirement Specification