**TECHNICAL SPECIFICATION Documentation**

**Advanced Java – Quiz Application**

**Balepu Ushasri**

**Table of Contents**

* **Introduction**
* **Objectives & Concentrations**
* **Scope & Limitations**
* **Software Requirements**
* **Global Application Flow**
* **Conception**
  + Authentication(Signup)
  + Authentication Success(Sign In)
  + Start the Quiz Application
  + Verifying MCQ Questions
  + Results
* **Bibliography**

**Introduction**

This document will propose all features and technical specifications of the quiz web application project. It contains details about project, scope limitation, primary requirements, possible project risks, database design, and application flow.

The goal of this project is to develop an application (API oriented, Web based) that deals with quiz assessments.

The usual problem while preparing an evaluation, are to:

* Constitute an appropriate evaluation regarding of the required level
* Reuse former questions
* Organize sample evaluations
* Correct automatically the MCQ questions.

The implementation of this project has been done by using Frontend (HTML, Angular JS), Backend (REST, Java 8), Database (Mysql), Business Logic Framework (Hibernate, Spring) and Server (Tomcat).sample application flow is as shown in below diagram.

*Angular* is a JavaScript MVC or Model-View-Controller framework developed by Google that lets developers build well structured, easily testable, and maintainable front-end applications.

This section discusses the main principles or approach towards the integration of Spring MVC with AngularJS. Before integrating, it needs to be understood that the roles of both Spring MVC and AngularJS are different though the objective is the same. While the Spring MVC will provide a framework that works for the server-end of the web-based application, the AngularJS will be working on the client or front-end of the application. The main principles are shown in below quiz application.

A screenshot of a cell phone

Description automatically generated

# Objectives and concentrations:

This Quiz Management Application handles any user, that should have signed

up with proper email id.

Students or any kind of Users are having privilege to create the account, choose the relevant quiz type and quiz name, attend the quiz, view the result at the same time.

User on the other hand is able to create his/her account, edit/delete/view the registered user’s details with their quiz score are send to their registered email id. Apart, from managing the user’s details they are able to create the quiz specifying name, and type.

# Scope and limitations:

* + The system handles all the operations, and generates reports as soon as the test is finish, that includes name, mark, time spent to solve the exam.
  + Allow students to see or display his answers after the exam is finish.
  + The type of questions is multiple choice questions and
  + User can Take/Create or can edit or update the quiz and can review the test.

**Software Requirements:**

Technology Used:

### Building and starting the server

To build the backend and start the server, run the following command on the root folder of the repository:

mvn clean install tomcat7:run-war -Dspring.profiles.active=test

The spring test profile will activate an in-memory database. After the server starts, the application is accessible at the following URL:

<http://localhost:8080/>

Backend/Webserver:

JAVA-Version 8,

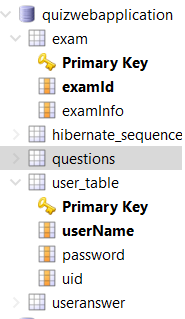
Spring Version 5.1

Tomcat Version8.5

FrontEnd:HTML,JSP,Angular.

Database: MySQL(5.1).

**Data Model:**



# Testing:

To satisfy the requirements and check whether every build services in the application are executed as expected, one of the earliest, testing efforts performed on the code is Unit Testing. To quickly test any new code or changes to existing code without the overhead and additional time involved in tasks such as server configuration, services setup and application deployment, we integrated the popular testing framework JUnit Framework to execute the unit test on every small code module.

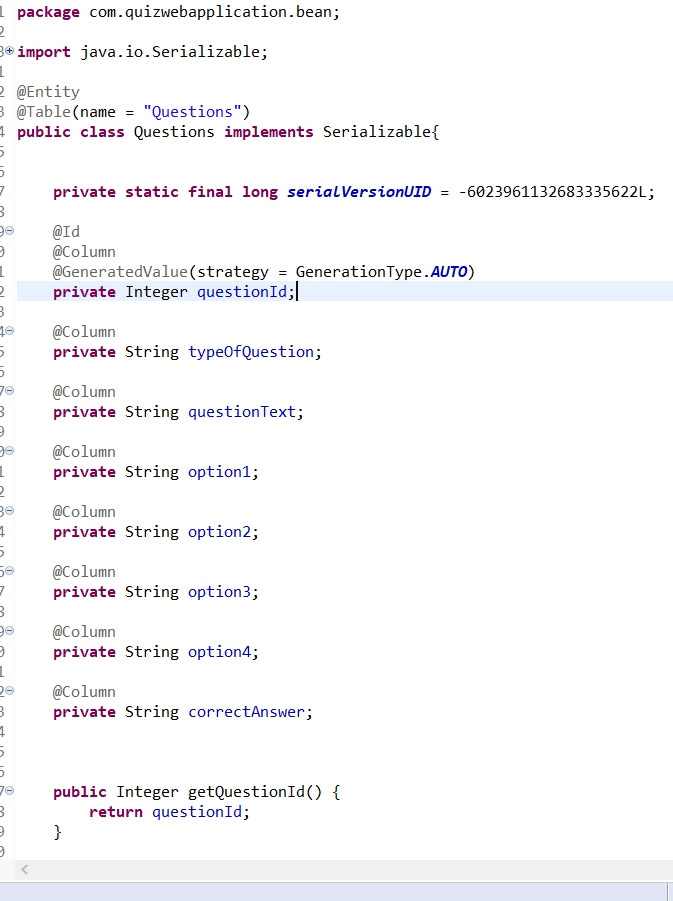
Our objectives are in writing and executing the tests

* We want to code and run the tests without leaving the IDE (Eclipse).
* There should be no special deployment of the code required
* We should be able to exploit other code analysis tools such as Metrics and find bugs right from within the IDE so we can find any bugs right away and fix those issues.

ExamController.java

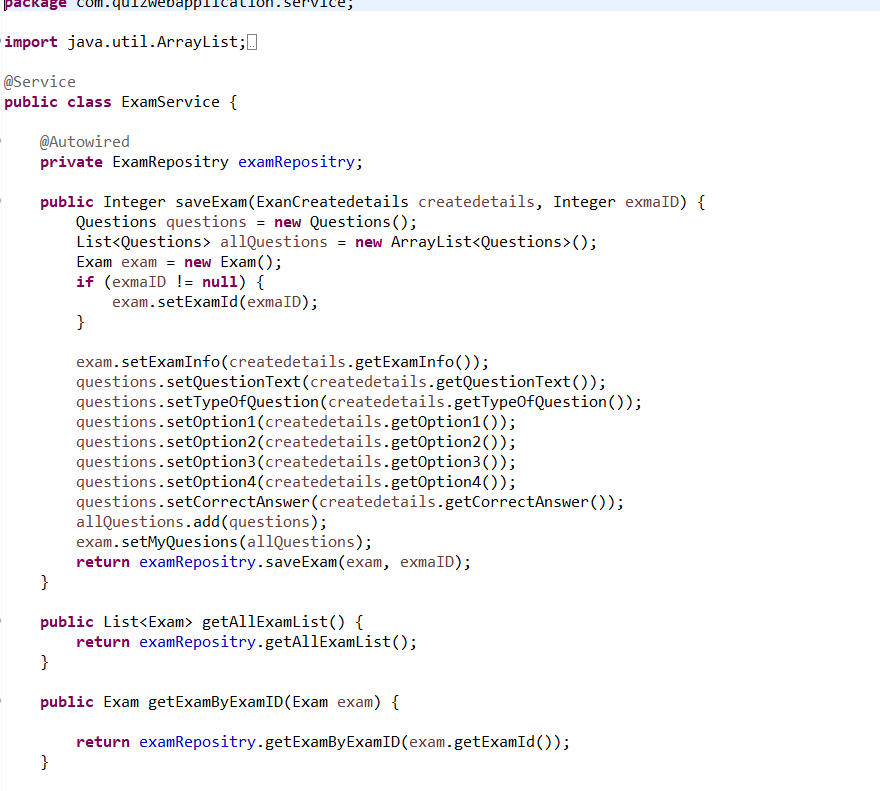


Questions.java



Connection with database:DBConnection.java





SpringContext.xml



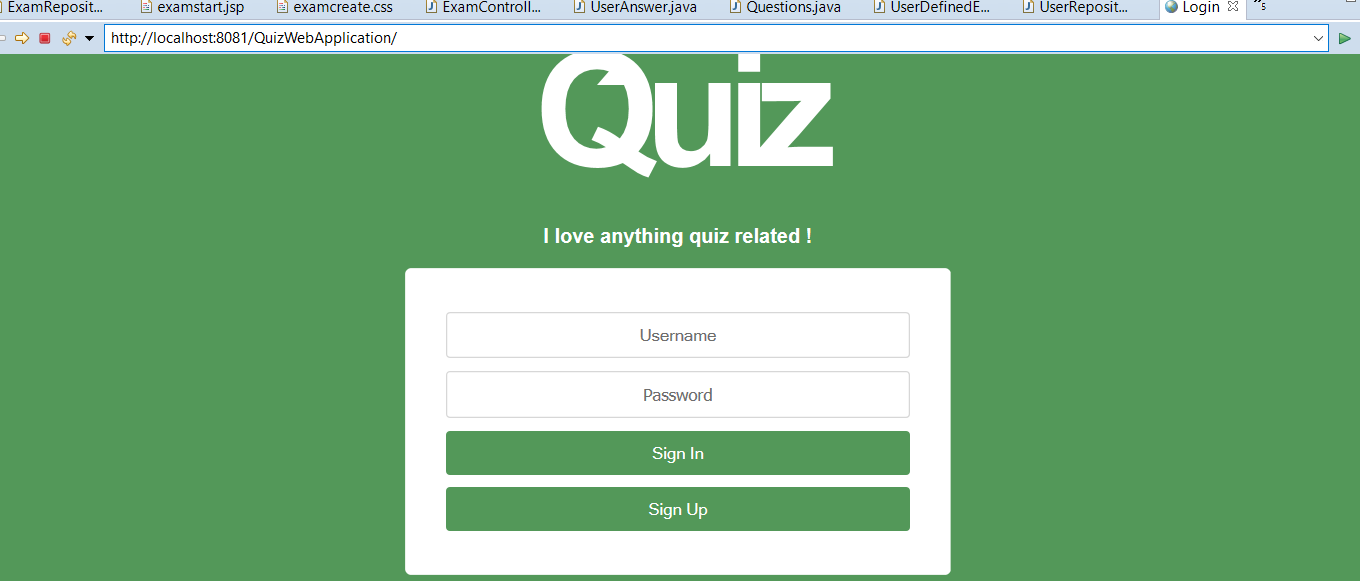
Web.xml



Welcome.js

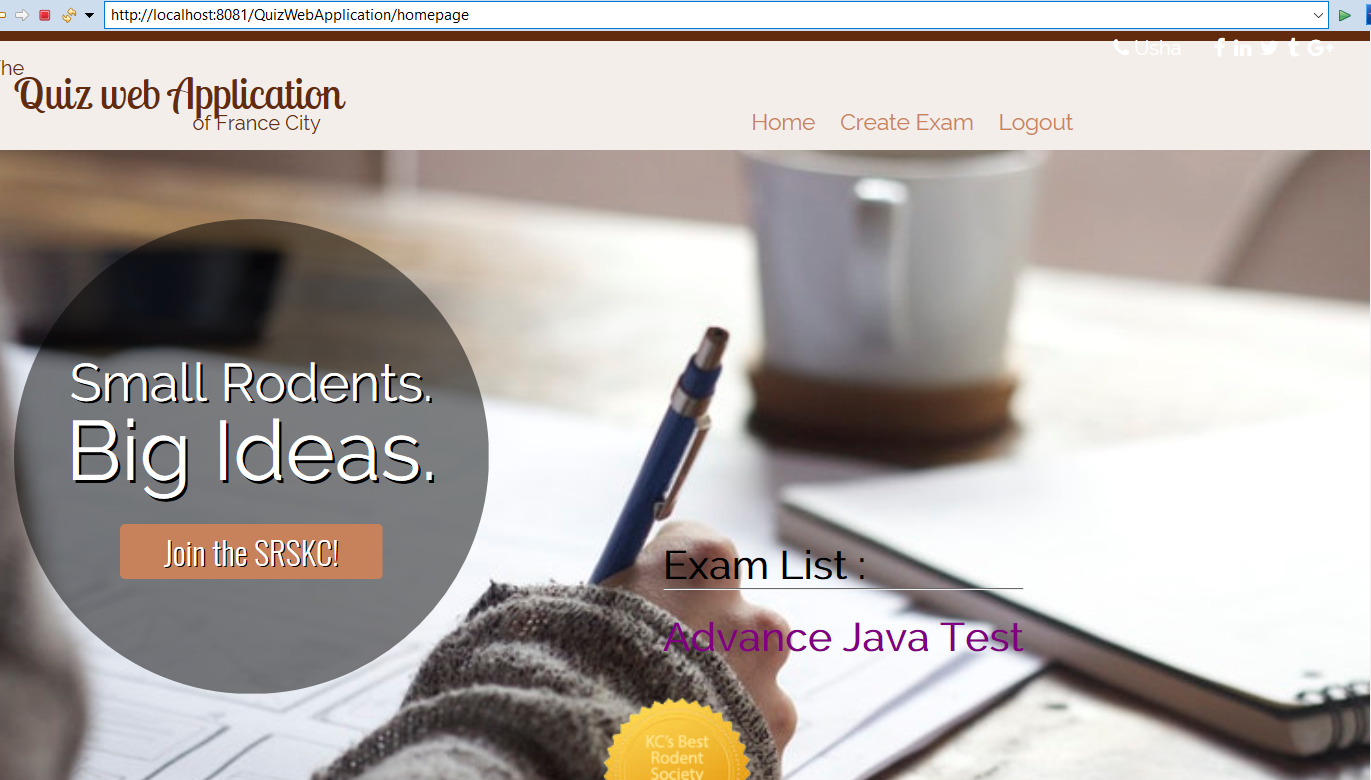


Login page:



**as Userid is Usha and Password 123456**

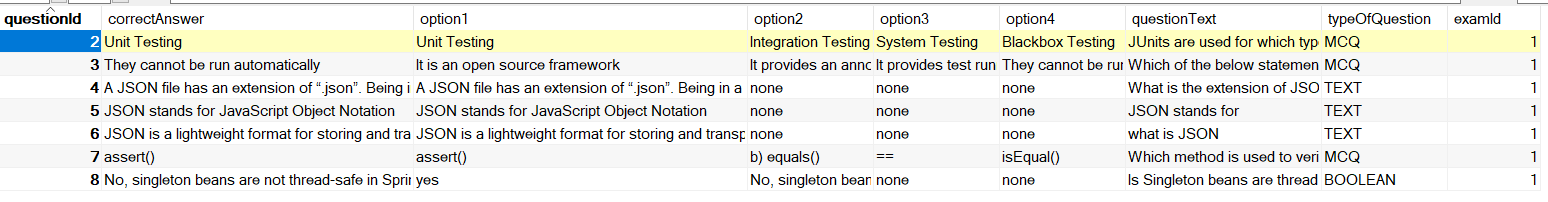
Homepage:



**Created Exam Test as**

**Advance Java Test:**

Questions are saved into DB in questions table:



**Application Module:**

This section gives a functional requirement that applicable to the Quiz Application.

* + **User module**: The candidate will logon to the application and take his quiz. He can check the list of question type, name and then attend the quiz as per his preference. The candidate will get result immediately after the completion of the examination.

**Major Features**

* + **Login/Sign Up Authentication**
  + **Manage Identities**
  + **Manage Questions**
  + **Take Quiz**

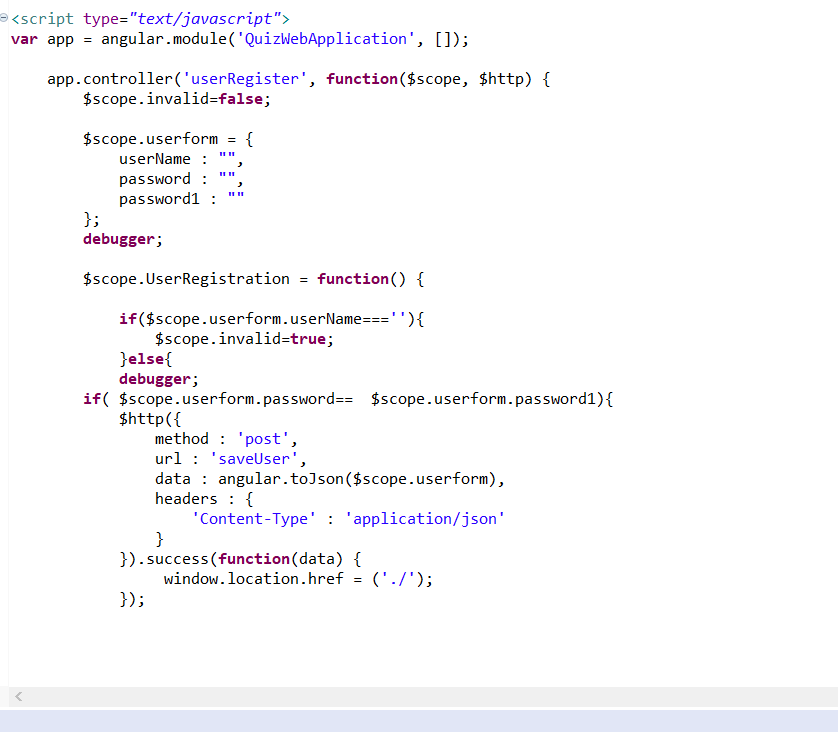
# Global Application Flow:

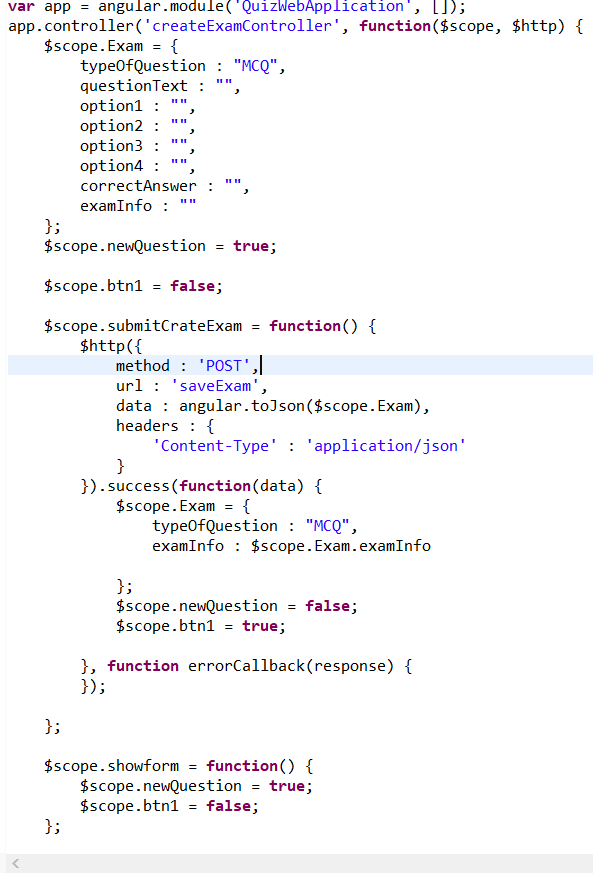
**Conception**

3.1 Authentication:

The web application cannot move forward without the login. There are two logins for student and admin. And a register page for the student, in-order to login.

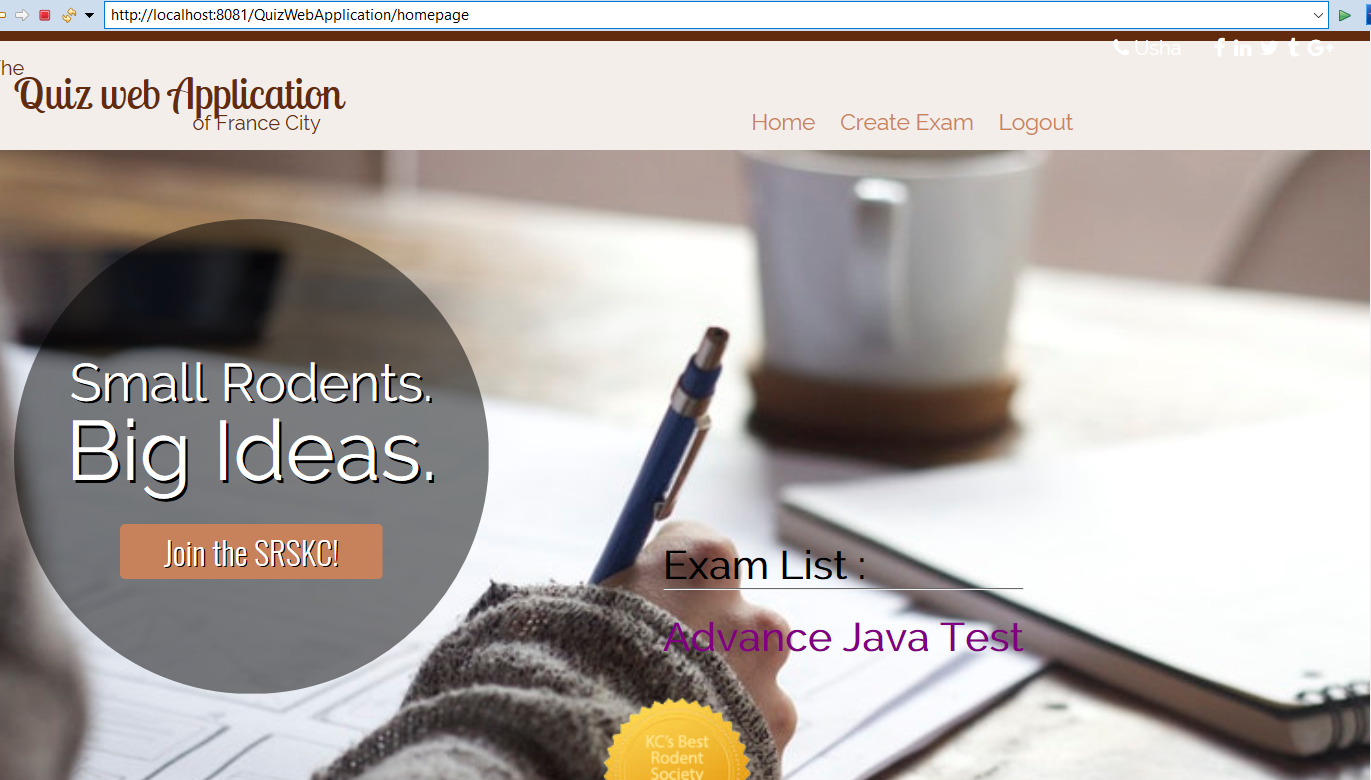
User Registering





* 1. Authentication successful:

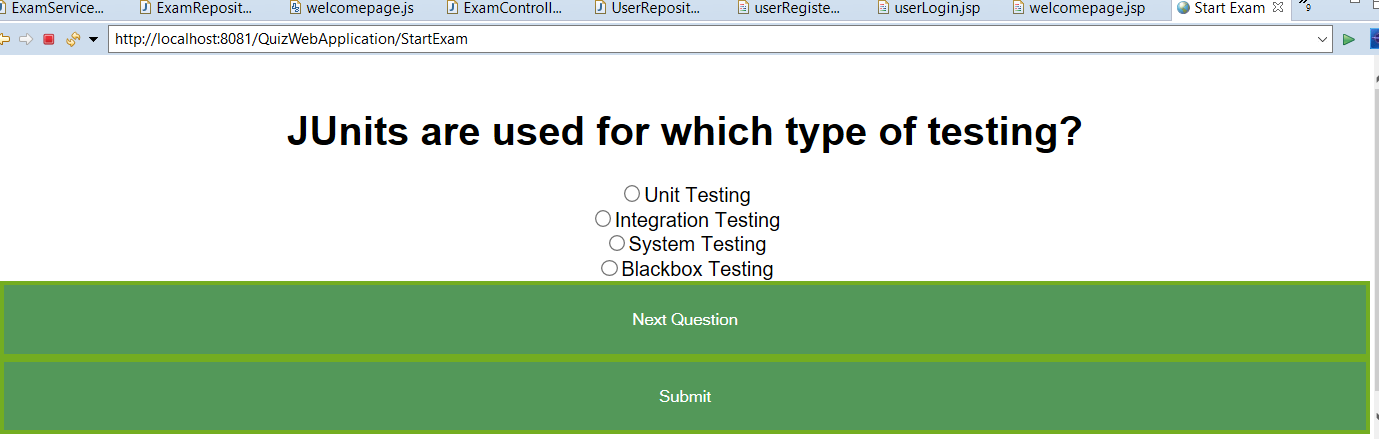
After successfully registering and logging into the application. The user can take the test.



## Start the quiz application:

Here we have implemented many service calls. The MVC architecture makes the code very feasible to understand.

<http://localhost:8081/QuizWebApplication/StartExam> starts exam



* 1. Start the quiz application:

Here we have implemented many service calls. The MVC architecture makes the code very feasible to understand.



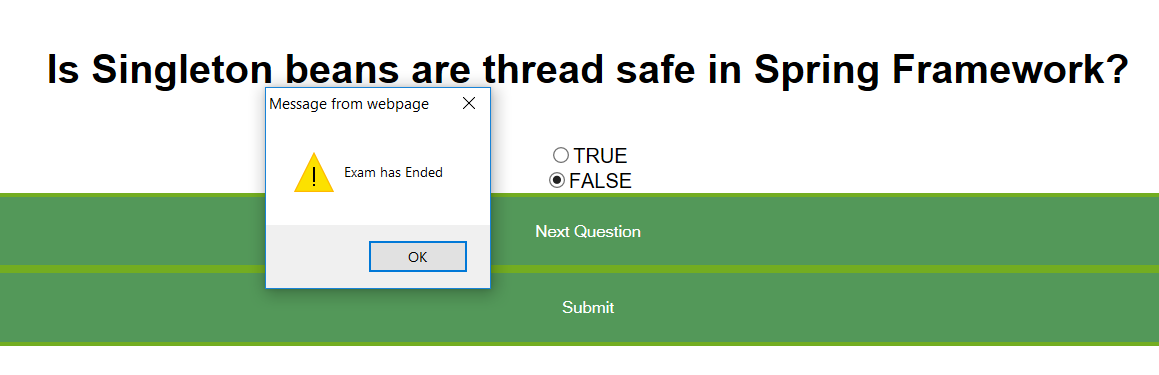
* 1. Verifying MCQ Questions:

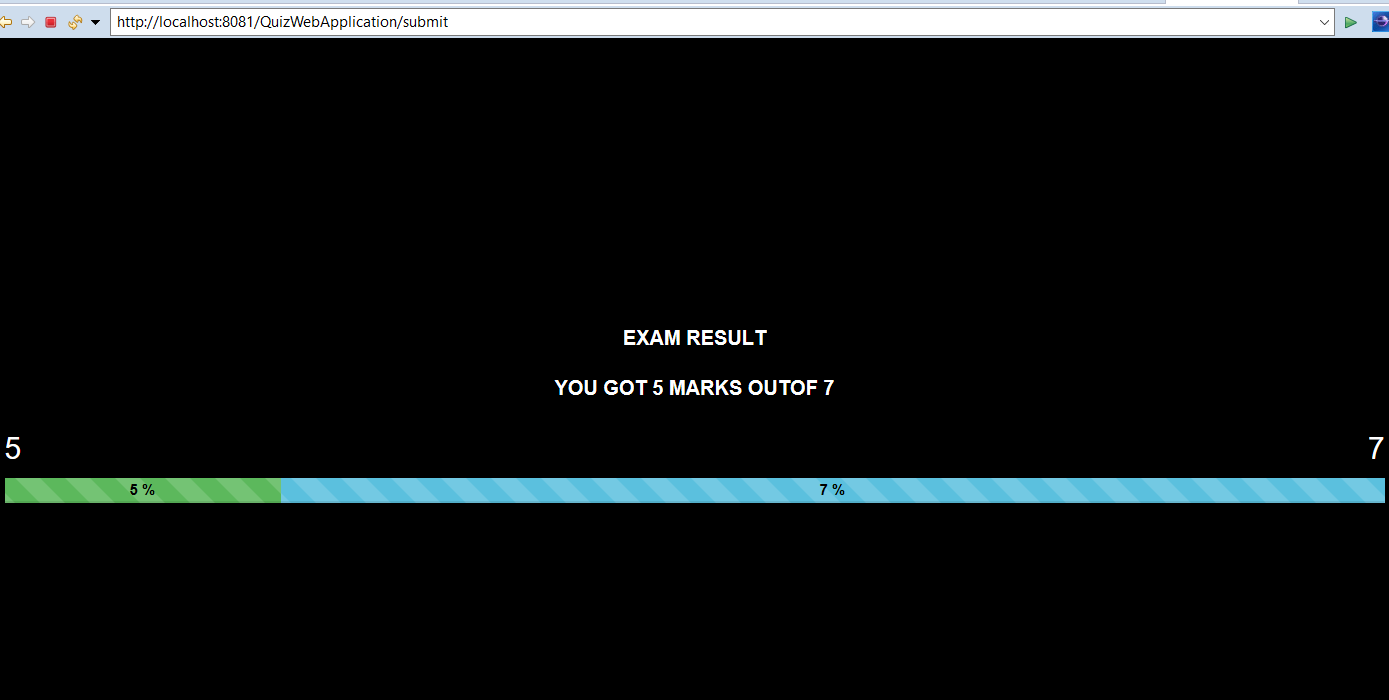
User will get MCQ questions and admin can reset the questions.



## Results: User will get result after submitting Results.

if questions over means as shown below dialogue box

After After Submit:

1. 

**Bibliography:** <https://github.com/thomasbroussard>

<https://thomas-broussard.fr/work/java/courses/project/advanced.xhtml>

<https://www.w3schools.com/angular/>

<https://www.javatpoint.com/spring-mvc-tutorial>

<https://www.java4s.com/hibernate/>

<https://maven.apache.org/>