

CALIFORNIA STATE UNIVERSITY, NORTHRIDGE

Enterprise Quiz Application with a new approach for better performance

A graduate project submitted in the partial fulfillment of the requirements
for the degree of Master of Science in Computer Science

By

Harish Pindi

December 2017

The graduate project of Harish Pindi is approved:

Prof. Sevada Isayan

Date

Dr. Richard Covington

Date

Dr. Robert McIlhenny, Chair

Date

California State University, Northridge

Acknowledgements

I would like to acknowledge and thank my thesis committee chair for Dr. Robert Mcilhenny for his continued guidance throughout the semester, his experience, and knowledge that he shared with me over the period along with his patient temperament that helped me in reaching the level of sophistication my thesis has.

I would also like to thank Dr. Richard Covington and Prof. Sevada Isayan for their support and agreeing to be a part of my thesis committee. Special thanks go to my graduate coordinator Dr. Ani Nahapetian who was there to resolve any questions concerning the thesis process and kept up to date with the deadlines and gave information related to any helpful workshops.

Table of Contents

Signatures Page	ii
Acknowledgements	iii
List of Figures	vii
List of Tables	ix
Abstract	x
Chapter 1: Introduction	1
1.1 Industry Problem.....	1
1.2 Proposed Solution	1
1.3 Resultant Solution.....	2
Chapter 2: Background/Trade Study	3
2.1 Enterprise Quiz App	3
2.2 Popular Applications in the Market	4
2.3 Impact of Current Applications and Research Work.....	5
Chapter 3: System Design.....	8
3.1 Overview of Features	8
3.2 Selection of Architecture, Framework, and Technologies.....	10
3.2.1 Architecture Pattern	10
3.2.2 System Front-end	11
3.2.3 System Back-end	12
3.2.4 System Components.....	13
Chapter 4: Implementation	15
4.1 Technologies and Framework.....	15

4.1.1 MVC Framework	15
4.1.2 Other Technologies to obtain Maven Structure	16
4.2 Languages and Tools:	26
4.2.1 J2EE7 (J2EE - Java 2 Enterprise Edition)	26
4.2.2 Maven illustration	26
4.2.3 Eclipse IDE (Neon):.....	26
4.2.4 Jersey Restful web services	27
4.2.5 Angular JS:.....	27
4.2.6 MySQL (5.1).....	29
4.3 Schedule and Approach	30
4.4 Deployment.....	31
4.5 User Interface.....	34
4.5.1 Sign-In Screens	34
4.5.2 User Home Page	34
4.5.3 User Dashboard Screen.....	35
4.5.4 Please select the exam Screen.....	35
4.5.5 Exam Drop down list Screen:	36
4.5.6 Start Exam Screen:.....	36
4.5.7 Exam Screen	37
4.5.8 Score Screen.....	37
4.5.9 Leader Board Screen.....	38
4.5.10 Admin Home/Dashboard page.....	38
4.5.11 Create Exam screen.....	39
4.5.12 Add Questions screen	39
4.5.13 Create Exam Modal class.....	40

4.6 Source Code	40
4.6.1 Enterprise QuizAppservice (Web services)	40
4.6.2 QuizAppRest.java	42
4.6.3 Quiz App Service internal functionality	44
4.6.4 QuizApp (AngularJS)Source code.....	47
4.6.5 REST API calls	49
Chapter 5: Conclusion.....	53
5.1 Summary	53
5.2 Development Experience	53
5.3 Challenges and Future Scope	54
References	56

List of Figures

Figure 1: Quiz Application Flow Chart	9
Figure 2: Quiz Application Architecture	11
Figure 3: Importing (QuizApp service) folder in Eclipse by existing Maven project	18
Figure 4: System screenshot on updating QuizAppService as a Maven project.	18
Figure 5: System screenshot on updating Maven project.	19
Figure 6: System screenshot of Maven Structure	20
Figure 7: System screenshot of QuizAppservice Conventions	21
Figure 8: System screenshot of Maven before the Execution of "mvn clean" command	22
Figure 9: System screenshot of Maven After Execution of “mvn clean” command	22
Figure 10: System screenshot of Maven After Execution of "mvn install" command	23
Figure 11: System screenshot of adding Jars explicitly in EclipseIDE.	24
Figure 12: System Screenshot of Maven dependencies instead adding JARs explicitly.	25
Figure 13: System screenshot of MVC pattern in Angular JS	27
Figure 14: System screenshot of controller’s role in Angular JS	28
Figure 15: System screenshot of Quizapp Database	29
Figure 16: System screenshot of QuizApp EER Diagram from existing DB MySQL	29
Figure 17: System screenshot of Quiz (App +Service) Tomcat Deployment	32
Figure 18: System screenshot of Tomcat server startup.bat for Deployment	33
Figure 19: System screenshot of entire Tomcat server startup for Deployment	33
Figure 20: Login Screens	34
Figure 21: System screenshot of User homepage	34
Figure 22: System screenshot of the User Dashboard	35

Figure 23: System screenshot of user “Please select Exam” option	35
Figure 24: System screenshot of the user’s view of Selecting the exam option	36
Figure 25: System screenshot of the user’s view of “Start Exam” button	36
Figure 26: System screenshot of the user Exam screen	37
Figure 27: System screenshot of the user’s resultant Score display	37
Figure 28: System screenshot of user’s resultant Score and Leaderboard	38
Figure 29: System screenshot of the Admin’s Home/dashboard page	38
Figure 30: System screenshot of Admin’s Create Exam options page	39
Figure 31: System screenshot of Admin’s Add questions page	39
Figure 32: System screenshot of Admin’s Create Exam Modal	40
Figure 33: System screenshot of Maven project directory structure	41
Figure 34: System screenshot of Deployment descriptor	42
Figure 35: GET and POST methods in QuizAppRest.java	43
Figure 36: System screenshot of QuizAppService Interface	44
Figure 37: System screenshot of Implementation of QuizAppService	45
Figure 38: System screenshot of homeCntrl.js	48
Figure 39: System screenshot of Gett call(getquestions) through Postman App	49
Figure 40: Screenshot of POST call (validate, create) user & GET call(activeexams)	50
Figure 41: Screen of GET (leaderboard,notifications) & POST call(createquestions)	51
Figure 42: Screen of GET (leaderboard, notification) & POST call (createleaderboard). 52	

List of Tables

Table1: System Sprint Table.....	31
----------------------------------	----

Abstract

Enterprise Quiz Application

By

Harish Pindi

Master of Science in Computer Science

In this project, I came up with a web application, where the user can access the application through the user login method and can attempt the exam. Since the exams are assigned only for a short period, the user can skip questions and proceed further to avoid negative marking. After the user attempts, the user can get access to check the result on a leaderboard. Where the user, used to know about his position out of several people. QuizApp has been designed to have a tiered architecture. The real strength of QuizApp is in its robust and flexible data model. However, not all users want to have to know this data model inside and out. The API layer allows a developer to know Rest endpoints and read/save them. This layer also can be used on different platforms like Windows, IOS, Android, etc.

Chapter 1: Introduction

The primary objective of this project is to develop a Quiz application, which provides a new approach for displaying ranking position based on results after each exam through a Leaderboard. A template of this project will be used to showcase my abilities to solve industry problems by applying knowledge acquired during my pursuit of the Master's degree.

1.1 Industry Problem

In general, the administrator of an organization conducts several quizzes to their prospective candidates. The candidates might be able to attempt these exams and get to know only about their results. Also, for the future quizzes, they might follow the same pattern.

The main problem arises in the way candidates perceive the results. Most of the methods just display the achieved result. The candidate might take the result as granted and show no further improvements. The result of one exam should motivate the candidate to excel on upcoming exams.

1.2 Proposed Solution

The way the candidate knows about his/her ranking position would encourage the chances of improvement in their perspective field. For instance, if an organization consists of thousands of people who are familiar with certain technologies, and if there is a sudden occurrence of new technology which might be trending, the administrator would like to

know how often their candidates are using the new technology. The administrator might be using this application to conduct quizzes. At the end of the exam, the user might know the result and their rank out of thousands of users. This ranking out of thousands of users would make them think of the situation and would seek for a better performance in the upcoming quiz exam.

1.3 Resultant Solution

I came up with a web application, where the user can access the application through the user login method and can attempt the exam. Since the exams are assigned only for a short period, the user can skip questions and proceed further to avoid negative marking. After each attempt, the user can get access to check the results through a leaderboard, where the user can know about his/her position out of several people. QuizApp has a very granulated permit system. Every action is performed with a privilege. The Administrator can add the Exam, Questions and can see the Leaderboard, whereas a user can only attempt an exam and view the Leaderboard.

Chapter 2: Background/Trade Study

2.1 Enterprise Quiz App

Apart from widely used applications for gauging pupil's capabilities on a specific topic, I am more interested in developing a forum where employees of an enterprise will be engaged to audit their capabilities on upcoming project technologies. It will help the employer to choose the best team to fulfill the project requirements easily. For gauging employee's capabilities, there are very few applications such as "Basecamp" which primarily focuses on employee engagement. Most of the Software Companies are still relying on face to face interviews to select the team to take up the new project. It will take valuable time and will be very tedious for a project manager. Also, there will be a very confined set of employees to gauge.

The enterprise Quiz app will decrease manual efforts in selecting the right set of employees for the upcoming project. The Employer can conduct quizzes on the topics based on the requirements of the new project, and all the employees in the organization can attempt the test.

The Employer can go through the Leaderboard console and select the best team that can fulfill the project's requirement. This application can have a huge impact on larger scale organizations.

Let's take an example of an Organization which has 500 employees. It will be very tedious to choose the best team for an upcoming project by interviewing each employee in person. Hence the project manager can conduct a few tests on the technologies used in the

upcoming project and can filter top 10 of the Leaderboard. On the other hand, even the employee will know his rank is seeing the Leaderboard and try focusing on the future endures. The gaming like the design of the system, especially with the Leaderboard will be fun for an employee to attempt the tests.

2.2 Popular Applications in the Market

There are many applications to gauge employees [1],

1) Basecamp

Representatives include their assignments for the day/week/month and verify them as they are finished. It enables the administration to perceive what's on a worker's plate for the day and what they could get past without grilling them.

2) Desk Time

DeskTime is a straightforward yet effective device that gives continuous programmed following abilities. It enables the administration to order applications as gainful, ineffective and impartial to genuinely gage exactly how profitable every single worker in the association is. The product likewise enables offices to track billable hours among their representatives. It has been with everything considered an awesome application for any business.

3) Asana + Harvest Integration

Asana coordinated with Harvest, empowers users to track their opportunity to deal with a financial plan productively. Undertakings can be relegated independently and followed continuously.

4) iDoneThis

iDoneThis is a straightforward route for workers to screen their execution and gives the group a shot praises their achievements.

5) 15Five

15Five is an electronic apparatus that makes clear correspondence and association with their workers. It addresses them all much of the time. There are times when users are recently too overwhelmed to assess how they are getting along and what they may require help. 15Five enables every user to cooperate to take care of issues, keep ventures moving and keep up a transparent working environment.

2.3 Impact of Current Applications and Research Work

All the mentioned systems primarily focus on day-to-day employee project and delivery monitoring. In contrast, Enterprise QuizApp primarily focuses on evaluating the employee's caliber for upcoming projects and emerging technologies. Most of the systems available in the market will gauge the employee's performance in the current project. Hence there is very much need for an application which primarily focuses on developing the employee's capabilities on emerging technologies and upcoming project requirements, Enterprise QuizApp is developed to address such issues and will be very helpful in the constant development of the employee.

One of the strategic areas identified in the Global Human Capital trend study [2] of 2016 by Deloitte is “attract and engage.” The point merits consideration because 78% of the directors who partook in the investigation appraised maintenance and engagement as pressing or vital. Representative engagement is the degree to which workers feel enthusiastic about their employment, are focused on their associations, and put optional exertion into their work. Connecting with individuals have turned into a wellspring of upper hand for the associations. Despite the fact that many merchants are offering approved studies and benchmarking devices, administrators feel these as inadequate, mostly because the present procedure is neither nitty sufficiently gritty nor constant. Moreover, with the current generation at work, who looks at the experience rather than engagement, employee engagement garners attention furthermore. In this unique situation, the idea of engagement needs reclassifying. The reason for this applied examination is to display a general perspective of the new engagement models went for making "overwhelming associations." Seminal deals with the subject are recognized and assessed for a superior comprehension of the improvements in the field. Rising and also steady indicators of representative engagement in the Industrial setting. Additionally, the article investigates the up and coming apparatuses and methodologies which better measure joy, arrangement, and occupation fulfillment continuously. While organizations have begun to perceive the effect of representative engagement, a substantial extent is yet to comprehend the degree of the genuine test. The roadblocks and implications for organizations bring this article to a close.

Enterprise QuizApp addresses the employee engagement by having an interactive console between employees and management. Management can continuously conduct Quiz on upcoming, and emerging topics and employee is engaged by the new and attractive

Leaderboard console which will increase the employee's competitive spirit. It is a new and innovative way to keep employee's competency levels high; ultimately employer will have the benefit of choosing the best team for the upcoming projects.

In general, most of the applications in the market are web-based and 2-tier architecture. Enterprise QuizApp is a 3-tier Architecture where we have decoupled the logic tier and presentation tier, which will help in easily enhancing the presentation tier based on the device (Desktop, Android or IOS mobile).

Chapter 3: System Design

The main objective of this research is to develop a Quiz Application, which provides a new approach toward displaying the results. Whenever the user logs in to the application, he/she can have access to try the perspective exams. After the attempt of the quiz/exam, he/she included with a Leaderboard through which he/she can not only see his/her results but also his/her position among the already attempted applicants.

It is a new approach toward displaying the results through a Leader Board, where his/her position can motivate the user and try to perform much better for the upcoming exams. This application is built utilizing the emerging technologies from the market. It provides a structural framework with high-end usability to the user.

3.1 Overview of Features

There are two versions of this application. One version for the administrator and the other version is for the user. The administrator is the one who creates the questions and posts them to an exam. He/she is the one who can update the exams by reviewing them. The 'user version' is the person who makes use of these exams within a time limit. Each of this version has their dashboard.

The administrator is the critical person in this application. He is the one who assigns exams to the users of this app. Once, he/she signs in the form, he/she includes in a dashboard of vertical navigation bars. This navigation bar often provides Home, Quiz, leader Board, Create Exam navigation bars. The administrator has access to update the questions to the prospective exam. Moreover, he/she can add the quizzes. Finally, he/she can also view the Leaderboard through which he/she can get the statistical analysis of the

people who attempted the examination and their ranking. Once he/she has finished with their role, they can log-out from this page.

The User, on the other hand, focuses entirely on attempting the exams that were assigned by the administrator. Once, the user signs into the application, the user is provided

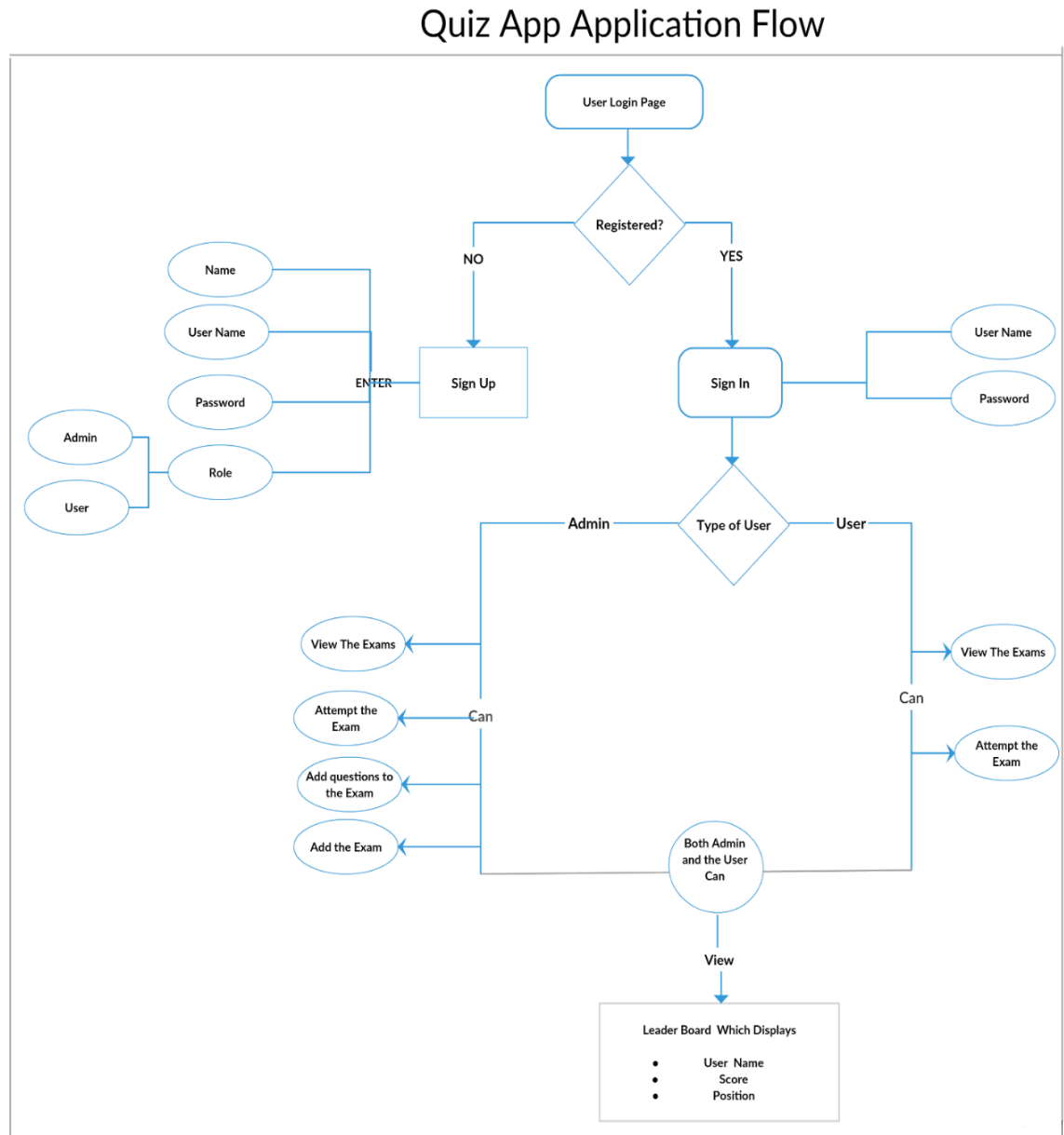


Figure 1: Quiz Application Flow Chart

with a welcome note and has vertical navigation bars to his/her left-hand side. This vertical navigation bar has Home, Quiz, Leader Board bars. Once, the user clicks the Quiz bar, he/she then get access to a drop-down list with a note “Please Select Exam.”

Once the user selects this required exam, he/she will be provided with a ‘START EXAM’ button below with a description of the time limit to finish the exam. Since most of the exams were conducted based on negative marking, the user can skip questions or else can proceed to further questions. Once the user finishes the exam, the score displayed on the dashboard with ‘GO TO LEADERBOARD’ button below it. Once he/she clicks this button, he/she will be redirected to select the Exam to view his/her leader Board view. The Leader Board displays the current user marks and his/her rank among the applicants who attempted the exam so far. Thus, the user can only attempt the exam and view the Leaderboard.

3.2 Selection of Architecture, Framework, and Technologies

In general, most web applications are built using a structured software framework intended to support web resources, services and API’s [3]. It is the most common way to build and deploy web applications. This section elaborates on the selection of the essential framework and choosing the front-end and back-end technologies respectively. All these patterns work with each other’s to obtain a perfect communication with each other frame. Every connection here in this framework is through REST calls. In general, every web application should follow a specific architecture so that web application will remain dynamic to the structure defined.

3.2.1 Architecture Pattern

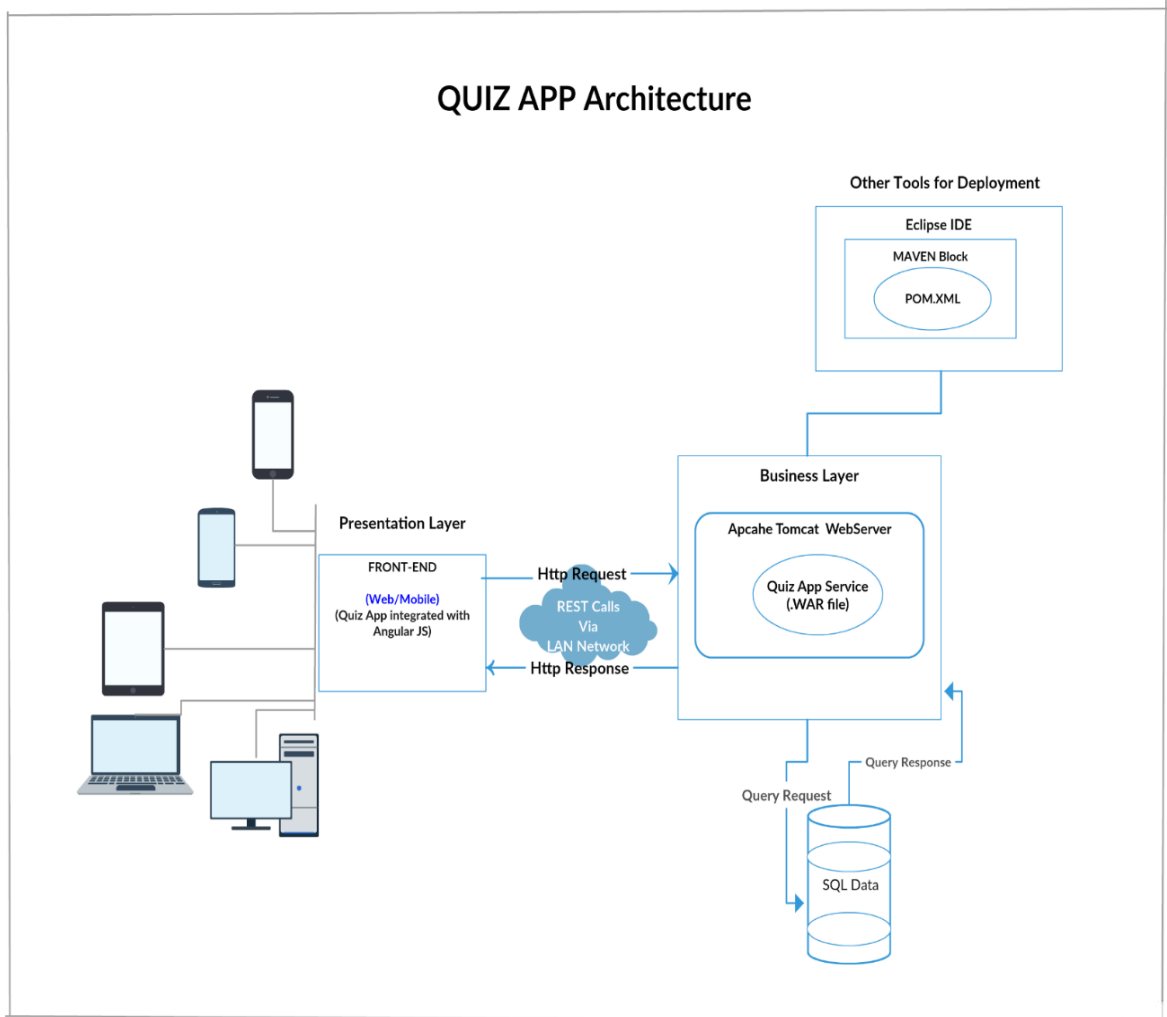


Figure 2: Quiz Application Architecture

Quiz app is built using a three-tier architecture pattern. This architecture provides a secure methodology for each of these designs, which also includes abstractions for better performance. It presents a structured model to complete the whole application. Quiz App has a three-tier architecture [4] which provides for (i) presentation tier, (ii) logic tier, and (iii) data tier. Each of these layers has its services, for which they are in the system. The Presentation Layer handles the user requests. The Business Layer includes the Core logic

of the System. The final tier used for the data storage also provides a Secured level of operations among the layers.

While handling the sensitive data, each of these layers uses its security services. For instance, the Presentation tier uses Presentation security services. Whenever the user of the app requests the app, it goes into the presentation layer for processing. The Presentation layer here calls the Quiz app service through the REST calls via the LAN network of the business layer. In the business layer, the controller takes the request from the presentation layer and calls the Quiz App Service. Since the service needs the Data Base (DB) layer for data retrieval, the business layer calls the DB layer through the DB Queries. Once the required data is obtained, it is processed using the controller. All the data requested from the DB layer would transform back to the Presentation Layer. All the communication between the layers is done through the JSON methodology (JavaScript Object Communication) and follows the granulated permit system using the authorization methods to secure the data transfer.

3.2.2 System Front-end

Whenever the client interacts with the system, The UI layer acts as communication with the system back-end. It is better to choose the framework trending in the market. Consider the statistics of mobile users (Apple, Android, Windows, etc.) and Web users. The figures of the web-traffic exhibit 45% of usage on laptops& desktops, 50% on mobile phones, 5% on tablet devices, 0.12% on other devices, as reported at the end of Jan 2017 [5]. Also, it is good to work with the framework which works with different environments. The end-user who is using this progressive web-app should feel comfortable in using this

user-interface. Thus, it is good to choose a framework which is platform-independent. Therefore, the code of the structure should be universal. Thus, the front-end technology contributes to the best interaction with a progressive web nature to the end user of the application.

3.2.3 System Back-end

The core logic of the system lies here. All the business Logic is done in the backend. After choosing the back-end language for the system, the application has built in an IDE following the basic rules of the selected framework pattern. It makes the form confined to the reusability. This layer lies in between the Presentation and DB Layer. Whatever the client requests, the middle layer conceptually handles the requests from the client using a controller and communicates with the DB Layer. After obtaining the requested data, the controller itself processes the data to will be displayed through the Presentation Layer of the system using REST calls.

3.2.4 System Components

There are mostly two main components considered during the process of applying the Quiz App System.

1. Admin:

The administrator is the crucial person in the System. He/she has overall control over the system. Whatever actions he performs, it will be reflected in the display to the end-user. The administrator can view all the exams, attempt the exams, add questions to the exam, add exams, and also can consider the Leader

Board, which is used to display the Name of the user, Score, and Position of the person comparatively out of already attempted users.

2. User:

The user is the one who tries the quizzes created by the administrator. Whenever the user is finished attempting the exam, the score will be displayed. He/she is provided with a view of the exams, an attempt for the exams and can view his Leader Board, after the end of the exam.

Chapter 4: Implementation

4.1 Technologies and Framework

4.1.1 MVC Framework

The Model-View-Controller design utilized. The MVC web framework holds extensive pieces concerning non-MVC web toolboxes. The Model-View-Controller design is the most utilized example of present world web applications. It has been utilized without precedent for Smalltalk and has been advanced by Java. The MVC design isolates an application into three modules: Model, View, and Controller. The Model is capable of dealing with the information. It stores and recovers substances utilized by an application, often from a database, and contains the rationale actualized by the application. The View is capable of showing the information given by the model in an organization. The Controller handles the Model and View layers for cooperating-on. The Controller receives a request from the client, generates the Model and sends the information to the View. The View organizes the information to be introduced to the client, in a web application as an HTML yield [6].

Quiz App is built using the basic principles of the MVC Pattern. The basic idea of the application is reusability, and hence the service and UI layers are decoupled. All the DB interactions were exposed as REST-endpoints, and these REST-endpoints also can be utilized in all the channels irrespective of the domain and technology they incorporate.

QuizApp utilizes the MVC pattern regarding Model (DAO), which consists of the application data, all the business rules and the Controller (Servlet) which acts as an

interface between the View and the Model to the View Layer. The servlet here further meditates as an input, converting further into commands for the Model or View. This View can be any output representation of data. The DAO (Data Access Object) concentrates on Business Logic, DB Server Connections, and other operations to access the process and extract the data from the database. This entire Model was constructed using an Open Source Tool (IDE) called Eclipse.

- Model: It contains all the Bean, DAO, DTO classes.
 - Bean: It contains all the getter and setter methods which are generally for storing or retrieving data in Object-oriented fashion.
 - DAO: Data Access Object (DAO) queries the database and sets back the data into Bean classes.
 - DTO: All the data defined as data objects and means of JSON establishes communication
- View: The View section handles the user interface of the application. It contains all the JSP, CSS, JavaScript files, etc.
- Controller: The controller (Servlet) acts as an event handler, It calls the service whenever it receives the data from the view section and calls the necessary function to handle it.

4.1.2 Other Technologies to obtain Maven Structure

After building the Source code using utilizing the MVC pattern in the open Source Tool (Eclipse IDE), it is the time to Run the Project folder using Maven.

a) Maven

Maven [7] is an open-source build tool customarily utilized as a part of Java and Java EE projects to arrange source documents, execute unit tests and gather appropriation artifacts. Apache Maven is a software tool which works on the concept of a Project Object Model (POM). Maven can deal with an undertaking's manufacturer, detailing and documentation from a vocal snippet of data. We can construct and deal with any Java-based task using Maven. In the event of different advancement group conditions, Maven can set up the best approach to fill in according to measures in a brief timeframe. As most of the project setups are simple and reusable, Maven makes the life of the designer simpler while making reports, checks, constructs and testing mechanization setups. This Maven tool provides developers with a complete build lifecycle framework. The build process of the project can be standardized and simplified using Maven. In this way, it utilizes the property of reusability and handles all kinds of Build related tasks.

POM stands for Project Object Model [8]. It is the principal unit of work in Maven. It is an XML record that lives in the base registry of the task as pom.xml. The POM contains data about the venture and different design details utilized by Maven to fabricate the project(s). POM additionally contains the objectives and modules. While executing an assignment or objective, Maven searches for the POM in the momentum catalog. It peruses the POM, gets the required setup data, and executes the objective. A portion of the arrangement determined in the POM is the following – Project conditions, modules, objectives, build profiles, Project form, developers, mailing list. Before making a POM, one should first choose the project

gathering (groupId), its name (artifactId) and its form an in a way to recognize them in a storehouse. QuizApp built with Maven and all the artifacts required are assembled by Maven, It also takes care of creating final a Web App (.war) file deployed on the server.

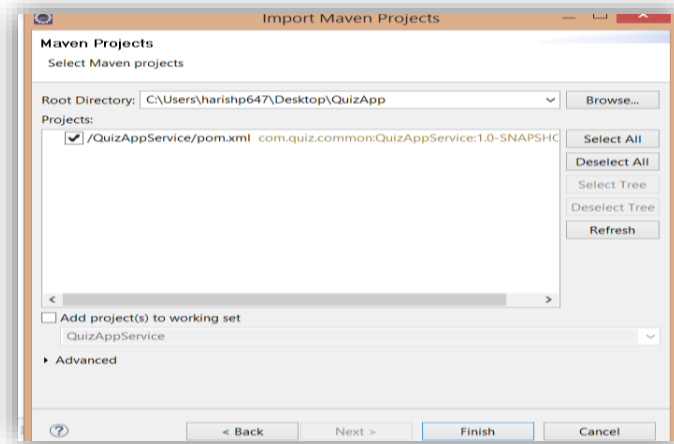


Figure 3: Importing (QuizApp service) folder in Eclipse by existing Maven project

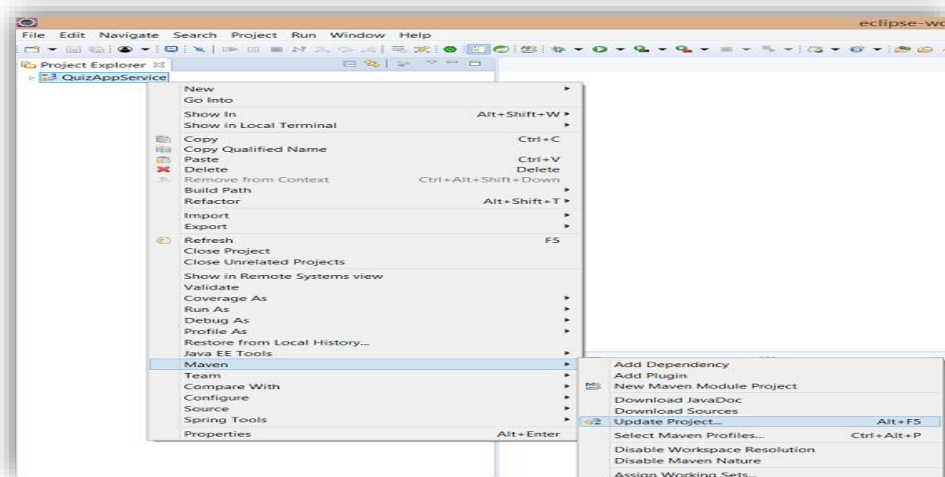


Figure 4: System screenshot on updating QuizAppService as a Maven project.

First, the folder (QuizAppService) is imported in Eclipse using the existing Maven project option. Second, after importing the project folder of (QuizAppService) into Eclipse, now we need to update it as a Maven project. Third, Maven updates the project. During the final phases, building workspace follows by updating the Maven project.

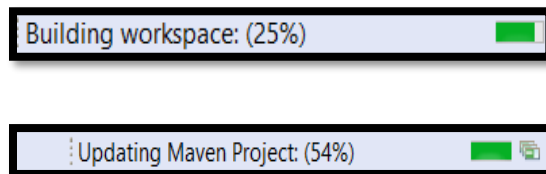
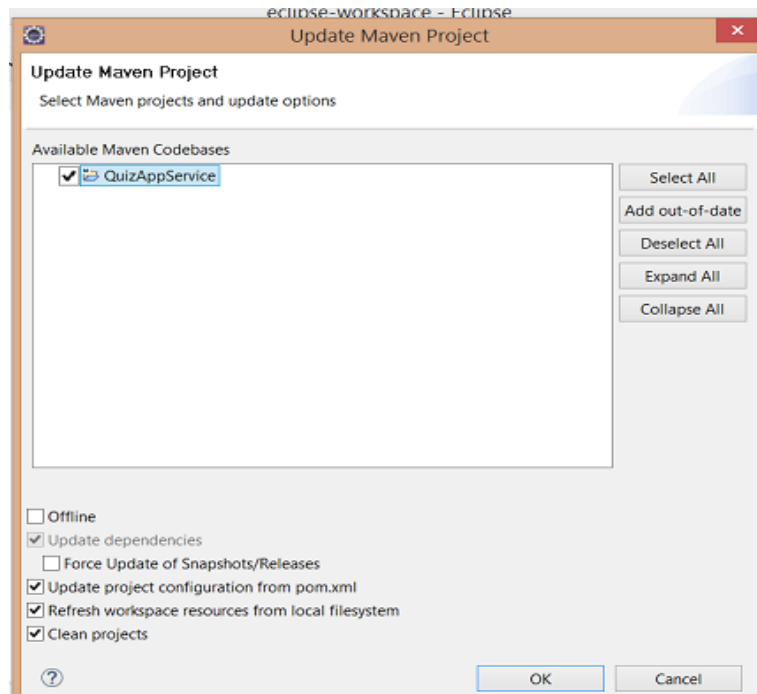


Figure 5: System screenshot on updating Maven project.

The resultant Maven structure is as follows:

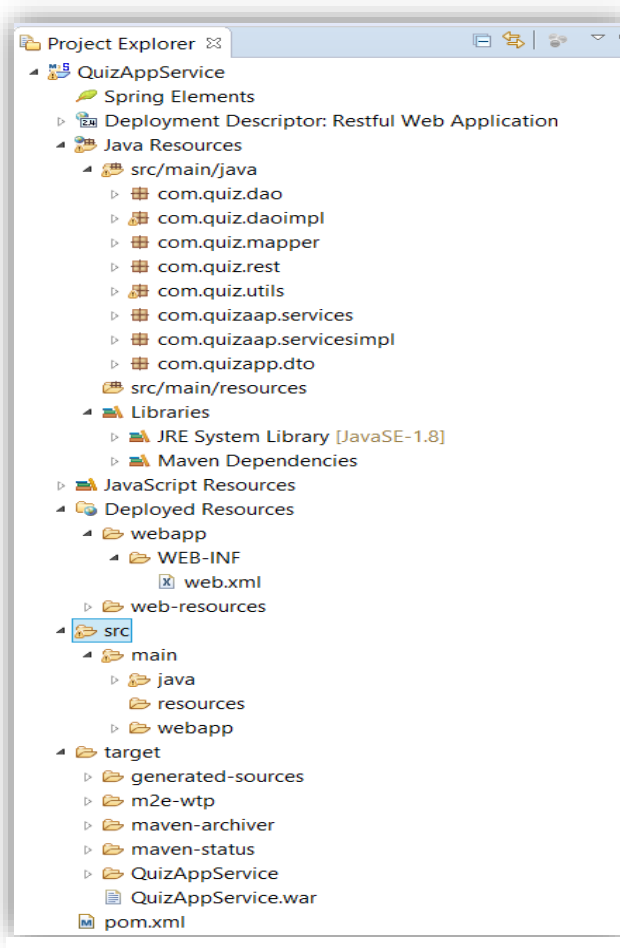
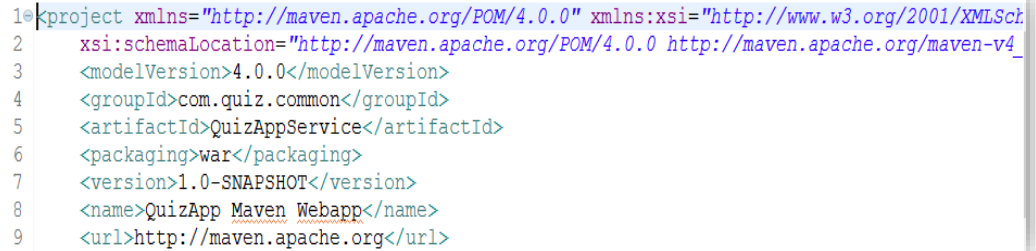


Figure 6: System screenshot of Maven Structure

In this (POM.xml) file, all the dependencies placed. Each dependency here defined as a Java Archive. A Java Archive [9] is a kind of bundle record design which has numerous Java class documents and corresponds to all the related metadata and their assets (pictures, content and so forth) into one record for a better conveyance. All these JARs required for the project placed in POM.xml. Whenever all the JARs installed, can be exposed in the repository. The (.war) file is

responsible for running on the server and deploying it in the form of a web application.

Here QuizApp conventions in (POM.xml) are termed in the form of



```
1<?xml version="1.0" encoding="UTF-8" ?>
2<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
4  <groupId>com.quiz.common</groupId>
5  <artifactId>QuizAppService</artifactId>
6  <packaging>war</packaging>
7  <version>1.0-SNAPSHOT</version>
8  <name>QuizApp Maven Webapp</name>
9  <url>http://maven.apache.org</url>
```

Figure 7: System screenshot of QuizAppservice Conventions

b) Build Life Cycle of Maven in Quiz App:

1. Prepare Resources: All the dependencies will be prepared here in the (pom.xml) file.
2. Validate: This method checks for the entire project validation & required info availability.
3. Compile: The entire source code is compiled here.
4. Testing: The compiled source code is available for the testing framework.
5. Packaging: Checks whether the package mentioned in POM.xml is a (.war) or a (.jar) file.
6. Install: In this phase, from remote to Local all the JARs are installed in the repository.

c) Executing the project in Maven

In general, the QuizAppService is the project folder. Here “target” is the Maven folder. In the target class, the (QuizAppService.war) file will be generated by Maven using Maven commands. In the command prompt, the path of the Quiz App service used for execution at the command prompt. When the command “mvn clean” executed, the target-folder will get diminished from the Quiz App Service.

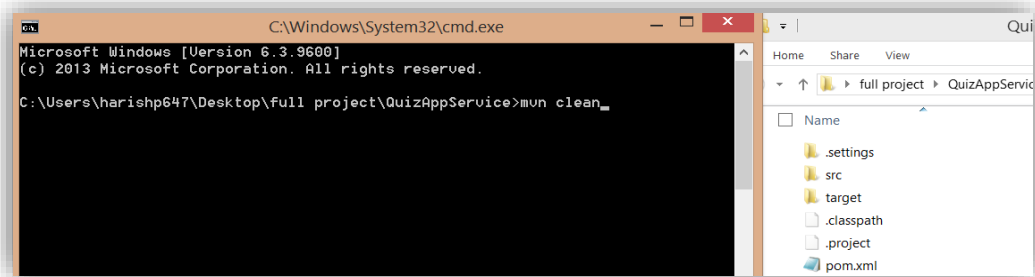


Figure 8: System screenshot of Maven before the Execution of "mvn clean" command

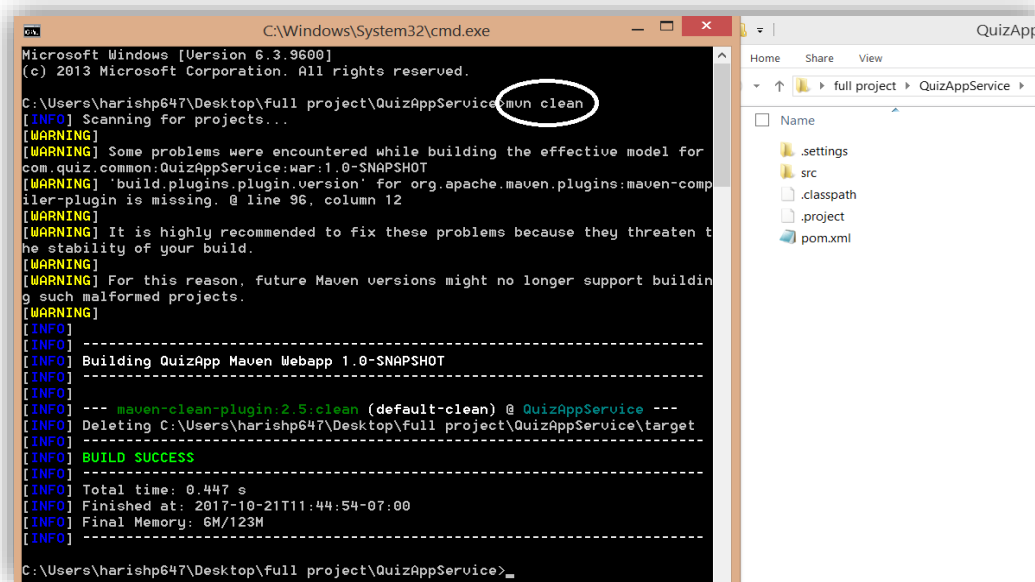


Figure 9: System screenshot of Maven After Execution of “mvn clean” command

Here the target folder diminishes from the Quiz App Service folder. After this, we run the command "mvn install" at the command prompt. Whenever this command gets executed, Maven will check all the resources. First, it will form the target folder. Afterward, it will download all the JARs. It will build the required (.war) file in the Project folder. This (.war) file deployed by Apache Tomcat Server.

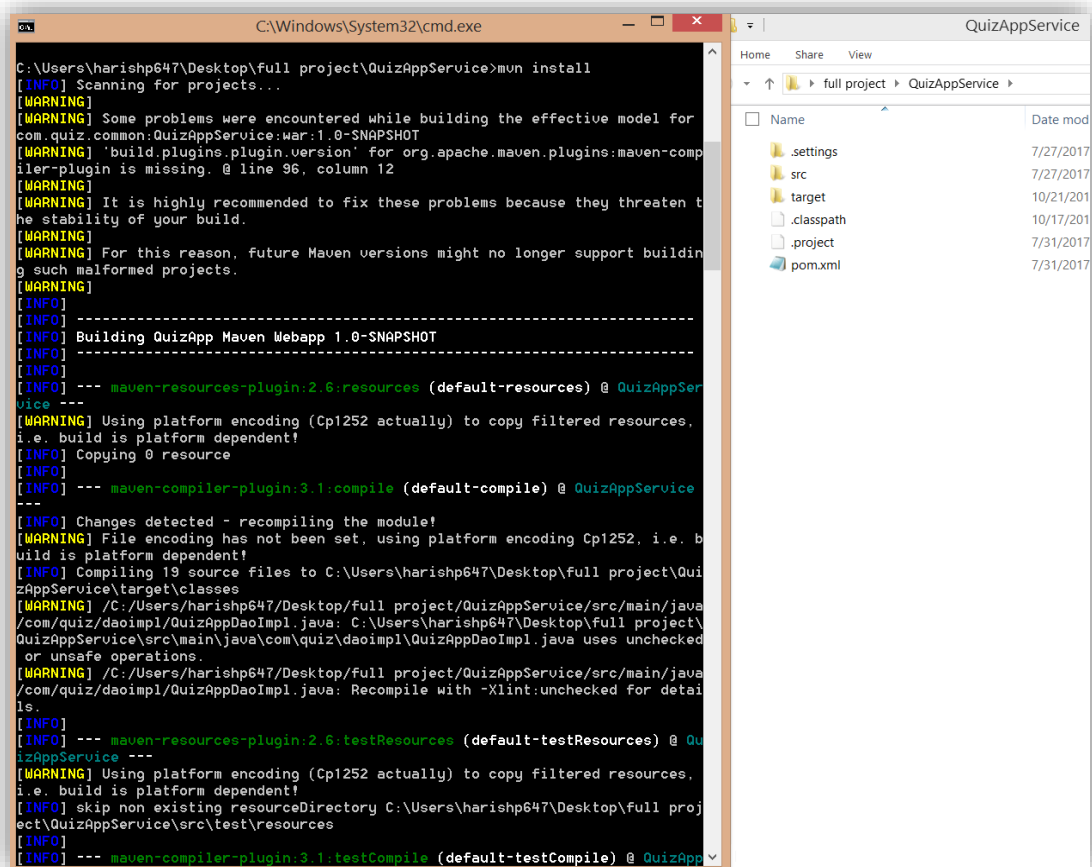


Figure 10: System screenshot of Maven After Execution of "mvn install" command

d) Further Illustration of Maven in QuizApp

Whenever we run the command "mvn clean install," based on the package, it will generate the (.jar) or (.war) file. To execute Maven, one needs to define the

JARs explicitly by adding them to the library. Maven itself generates them. The command “mvn clean install” generates the Maven to check first in the local repository. Here the Local repository is the (.m2) folder. Maven checks this folder whether the required JAR is available or not. If it is not available here, it will then go to the remote repository and download the file. Then it will place this JAR into the local repository. Afterward, it will add the dependency automatically to it. Next when executed, it will check with the Local repository rather than going further to the remote repository. This operation is based on JARs. Since WAR is a mixture of multiple JAR files, it is an extensive process to convert multiple JARs into the (.war) file. In general, all the jars will be placed explicitly using a certain procedure. On the other hand, Maven makes it much simpler.

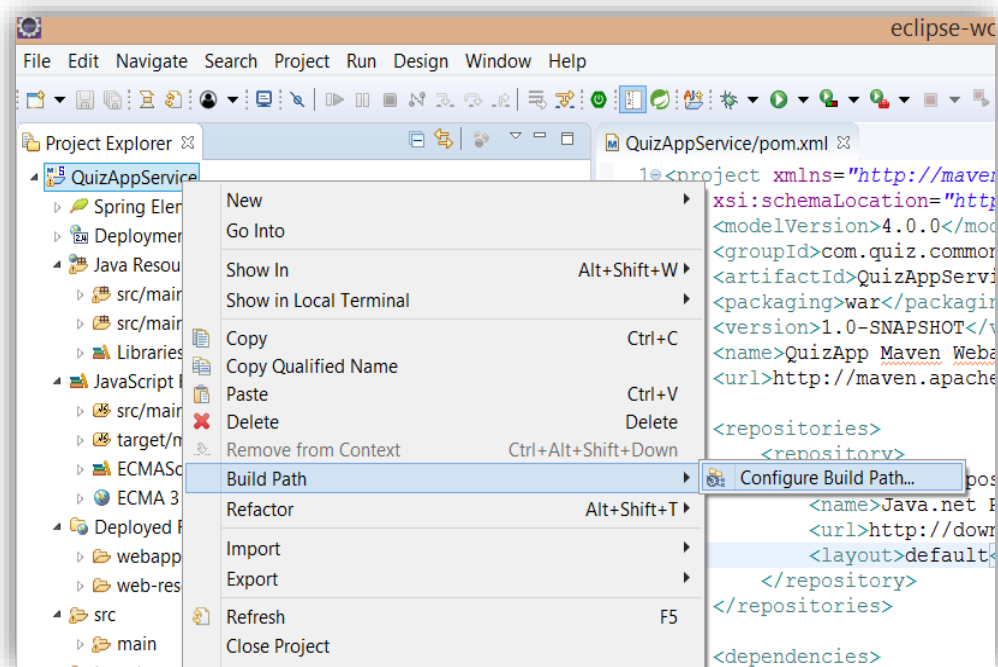


Figure 11: System screenshot of adding Jars explicitly in EclipseIDE.

This process of adding JARs in Eclipse is a time-taking procedure. Sometimes while working with several hundred JARs, it requires the procedure to download from one location, and then further it is included at another location. In the process, some of the files get ignored. So, to avoid this complicated process, Maven includes Maven dependency. Whenever we run the command “mvn clean install” all the JARs will be combined, and the .war file gets generated. This executable .war file can deploy anywhere.

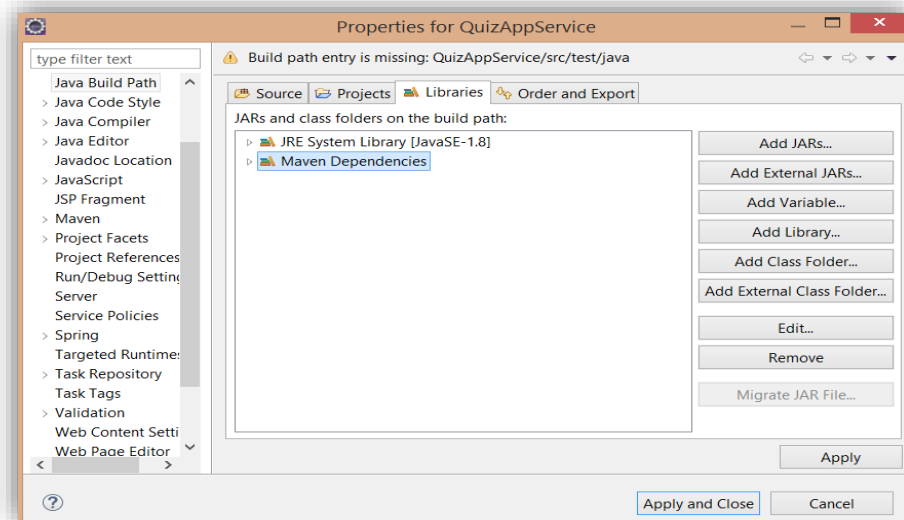


Figure 12: System Screenshot of Maven dependencies instead adding JARs explicitly.

At an industrial level, most companies use Maven. If maven is not available, “Gradle” used instead. 90% of the companies use Maven. There's a slight variation with the commands for Gradle. The ultimate concept of both Maven and Gradle is same. Maven constructs the dependency tree. In case of large projects, Maven is

the best option. The main reason is to bring all the associated JARs and dependencies. Thus, Maven helps to organize and install the dependencies.

4.2 Languages and Tools:

4.2.1 J2EE7 (J2EE - Java 2 Enterprise Edition)

The server side programming language [10] mainly used to implement business logic. For the Backend operations, J2EE7 used. The main reason for the usage of Java as a Backend Language is due to its properties of being open-source and platform independent. It leads to the reusability and flexibility to the system.

4.2.2 Maven illustration

Maven is an open-source build tool customarily utilized as a part of Java and Java EE projects to arrange source documents, execute unit tests and gather appropriation artifacts. Maven works primarily on Java projects and artifacts.

QuizApp built using the build tool Maven, and Maven assembles all the artifacts required. It also takes care of creating final a Web App (. war) file which can deploy on the server.

4.2.3 Eclipse IDE (Neon):

Eclipse [11] is an integrated development tool which is mainly used to develop Java applications.

4.2.4 Jersey Restful web services

The Jersey [12] usage gives a library to actualize Restful web services in a Java, servlet holder. Jersey generates a servlet execution, which examines predefined classes to recognize RESTful resources. The Jersey execution likewise gives a stage of the client library to communicate with a RESTful web service. There are many rest endpoints created in QuizApp using the Jersey web services framework. QuizApp uses POST and GET methods, and it consumes and produces only JSON data.

4.2.5 Angular JS:

AngularJS [13] is a JavaScript Framework. This framework was a part of Single Page Application (SPA) projects. It expands HTML DOM with additional characteristics and makes it more receptive to client activities.

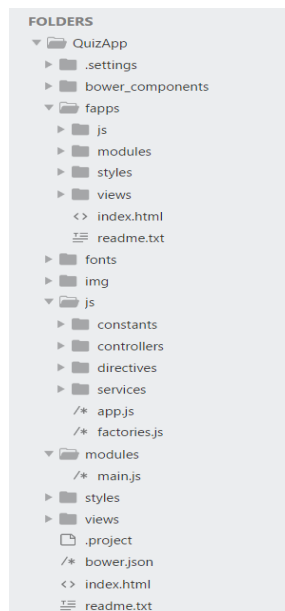


Figure 13: System screenshot of MVC pattern in Angular JS

The QuizApp UI layer is written in Angular JS. It is a single page application which uses the services provided by QuizApp services module and renders the data. UI Layer is designed based on the MVC pattern. It includes all the directives, services, filters and the controllers, etc. For the UI development, we use the Sublime Text tool. For example, if we consider the leaderCtrl.js file, “show leader” is the function. We define the function in the front-end, whereas we define the method at the Back-end. Here “var examined = \$scope.Selected;” defines the examId, where the user is selected. Also “\$scope.Selected;” usually comes from the associating UI(leaderboard.html). Whenever the user selects here, the controller usually comes in the controller's section.

```
2    });
3  }
4
5  $scope.showLeader = function(){
6    var examId = $scope.selected;
7    var promise = $http.get('http://localhost:8080/QuizAppService/rest/v1/quiz/getleaderboard?examid='+examId);
8    promise.then(
9      function mySuccess(response){
10        if(response.data){
11          $scope.leaderBoard = false;
12          $scope.names = response.data;
13        }
14      },function myError(error) {
15        alert("Oops! data issue");
16      });
17  }
```

Figure 14: System screenshot of controller's role in Angular JS

Whenever REST call requested, the resultant request response will be seen the UI Layer. In this part of the code, we call the ExamId. The promise function responds whether the data exists or not. Likewise, the basic flow exists in every section.

4.2.6 MySQL (5.1)

The QuizApp database resides in MySQL, and the data model contains tables and views to store the application data. The system uses an RDBMS (Relational Database Management System), to store and retrieve the data from the database. This system works based on the RDBMS structure and the queries embedded with it.

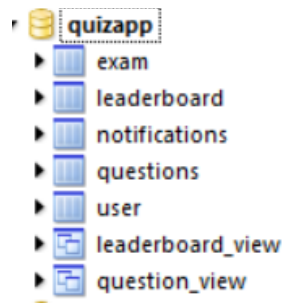


Figure 15: System screenshot of Quizapp Database

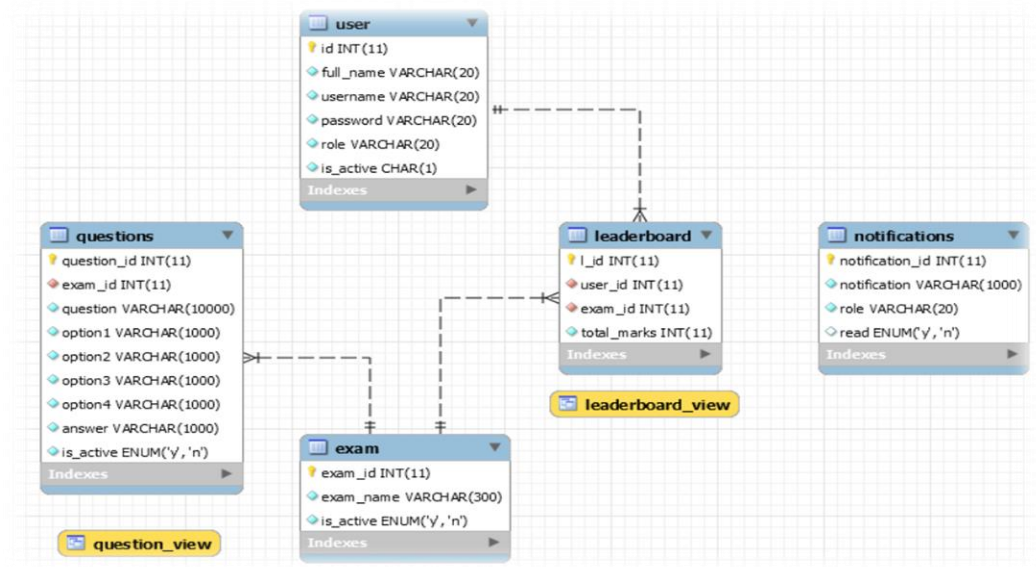


Figure 16: System screenshot of QuizApp EER Diagram from existing DB MySQL

4.3 Schedule and Approach

I followed the Agile methodology [14] since it is an iterative approach to deliver the software in an incremental process. I used it to deliver my project in stages rather than all at once. Each stage of the project took two weeks. Each stage of this project includes user functionalities, also known as user stories. It required prioritizing and delivering these sections during a period. The entire Application is sub-divided into two phases. One is to write the services, and other is for the UI.

The entire application was further divided into stories, and these stories were then grouped into different sprints, (ideally each sprint is for two weeks). The six sprints for the development of QuizApp are of equal-sized duration, each with the time span of 20 days. Thus, according to the Scrum planning,

No. of days for the project = No. of Sprints X No. of days for each Sprint

No. of days of the project = 6 X 20 = **120 days**

Thus, the estimated duration of this project is 120 days, which are further evenly divided among the six sprints. The advantage of having equal-size sprints is that it becomes easier to plan for the upcoming sprints, and thus the stakeholders can be notified in advance.

Table1 shows each sprint and its time span. Each sprint is about 20 days in length, which is almost three weeks in duration. The sprint planning will last four to eight hours before the start of each sprint.

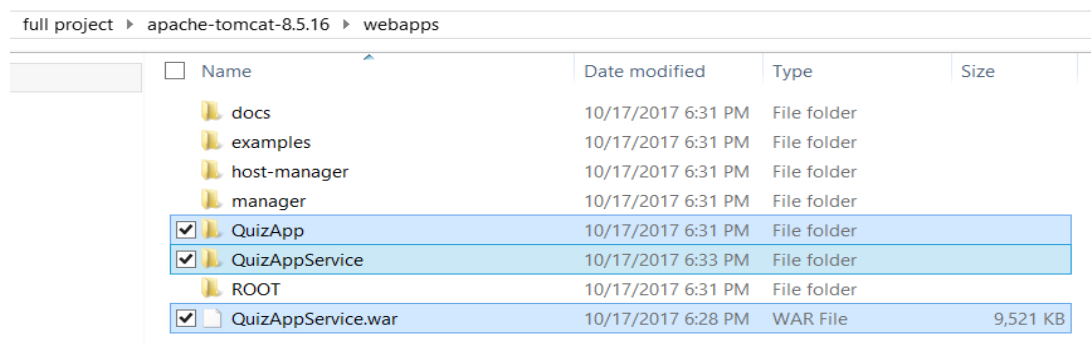
Sprint no.	Start date	End date
1	01/25/2017	02/14/2017
2	02/15/2015	03/05/2017
3	03/06/2017	03/26/2017
4	03/27/2017	04/16/2017
5	04/17/2017	05/06/2017
6	05/07/2017	05/18/2017

Table1: System Sprint Table

4.4 Deployment

The Apache Tomcat® software [15] is an open source implementation of the Java Servlet, Java Server Pages, Java Expression Language and Java Web Socket technologies. This obtained executable .war file is then deployed in Apache-Tomcat-8.5.16. In general, the Tomcat structure has different folders. Either with a Windows, Linux or IOS operating system, the entire structure remains the same. In Tomcat, the “bin” folder contains all the services such as startup.bat, shutdown.bat.... etc. This “bin” folder helps in starting and stopping the server. The “Config” folder contains all the configuration files such as Tomcat users. The Application admin will decide on which port the server should be executed. All

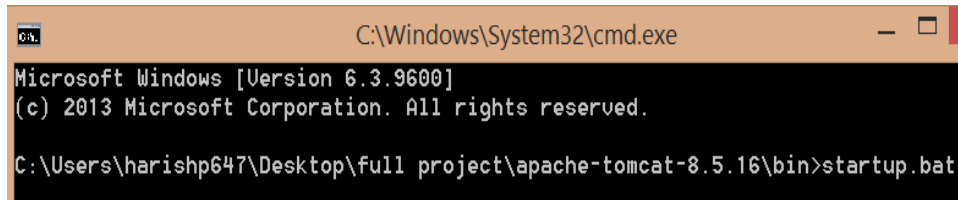
this configuration is configured through the GUI. Feasibilities such as username, password, roles configured. Also, the “Catalina” log file can assist in deciding on which port the dependency JAR is configured. Tomcat is also a web application. Also called a web container. All the libraries regarding the Tomcat are placed in the library folder. This library folder includes all the JARs. The Tomcat server has a predefined web code written by the developers. The “catalina.log” file creates the log files internally, each less than 100kb. The executable QuizAppService.war file is placed in the “Web apps” folder of the Tomcat server. Whenever the server gets started, this .war file will generate the QuizAppService folder. In this QuizAppService folder, all the code I have written will be present. All the classes will be present. All the code that we have written in Java format, the JVM converts to .class files with a bytecode format. This bytecode is machine-understandable. Independent of the machine, this bytecode can be understood. Now the QuizApp (UI folder) will also be placed here. This QuizApp folder does not require any deployment. The QuizApp folder explicitly is placed here in the “web apps” folder and then is deployed.



full project ▸ apache-tomcat-8.5.16 ▸ webapps

<input type="checkbox"/>	Name	Date modified	Type	Size
<input type="checkbox"/>	docs	10/17/2017 6:31 PM	File folder	
<input type="checkbox"/>	examples	10/17/2017 6:31 PM	File folder	
<input type="checkbox"/>	host-manager	10/17/2017 6:31 PM	File folder	
<input type="checkbox"/>	manager	10/17/2017 6:31 PM	File folder	
<input checked="" type="checkbox"/>	QuizApp	10/17/2017 6:31 PM	File folder	
<input checked="" type="checkbox"/>	QuizAppService	10/17/2017 6:33 PM	File folder	
<input type="checkbox"/>	ROOT	10/17/2017 6:31 PM	File folder	
<input checked="" type="checkbox"/>	QuizAppService.war	10/17/2017 6:28 PM	WAR File	9,521 KB

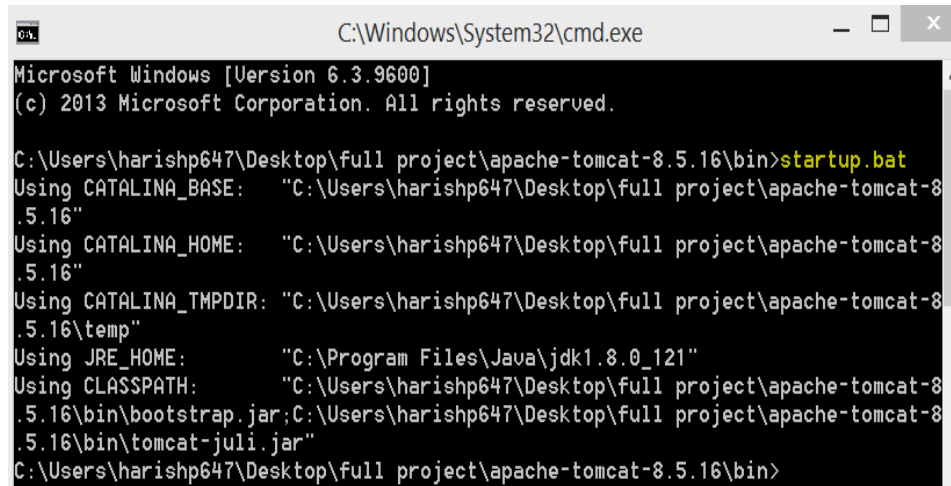
Figure 17: System screenshot of Quiz (App +Service) folder in web apps Tomcat for Deployment



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

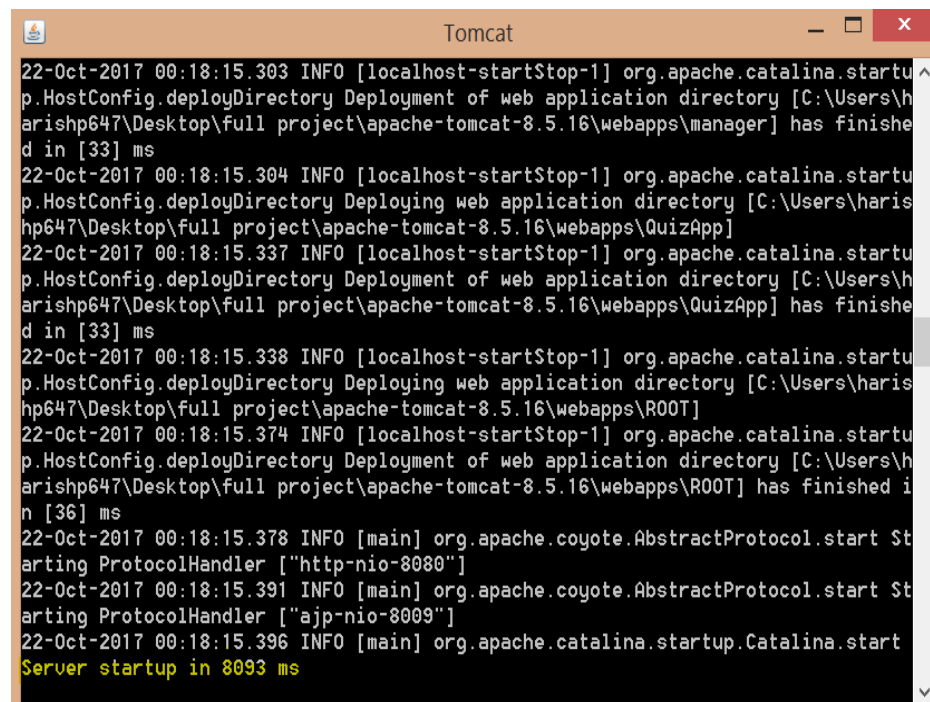
C:\Users\harishp647\Desktop\full project\apache-tomcat-8.5.16\bin>startup.bat
```

Figure 18: System screenshot of Tomcat server startup.bat for Deployment



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\harishp647\Desktop\full project\apache-tomcat-8.5.16\bin>startup.bat
Using CATALINA_BASE:  "C:\Users\harishp647\Desktop\full project\apache-tomcat-8.5.16"
Using CATALINA_HOME:  "C:\Users\harishp647\Desktop\full project\apache-tomcat-8.5.16"
Using CATALINA_TMPDIR: "C:\Users\harishp647\Desktop\full project\apache-tomcat-8.5.16\temp"
Using JRE_HOME:       "C:\Program Files\Java\jdk1.8.0_121"
Using CLASSPATH:      "C:\Users\harishp647\Desktop\full project\apache-tomcat-8.5.16\bin\bootstrap.jar;C:\Users\harishp647\Desktop\full project\apache-tomcat-8.5.16\bin\tomcat-juli.jar"
C:\Users\harishp647\Desktop\full project\apache-tomcat-8.5.16\bin>
```



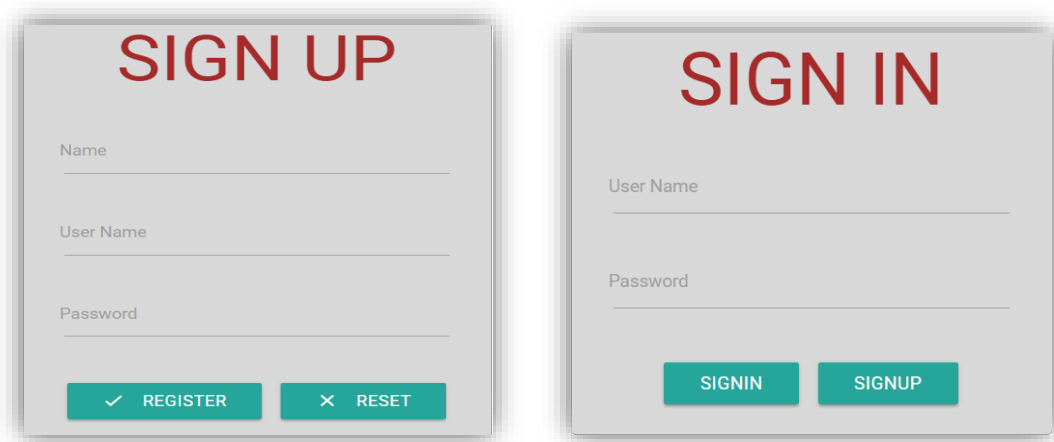
```
Tomcat
22-Oct-2017 00:18:15.303 INFO [localhost-startStop-1] org.apache.catalina.startup
p.HostConfig.deployDirectory Deployment of web application directory [C:\Users\h
arishp647\Desktop\full project\apache-tomcat-8.5.16\webapps\manager] has finishe
d in [33] ms
22-Oct-2017 00:18:15.304 INFO [localhost-startStop-1] org.apache.catalina.startup
p.HostConfig.deployDirectory Deploying web application directory [C:\Users\haris
hp647\Desktop\full project\apache-tomcat-8.5.16\webapps\QuizApp]
22-Oct-2017 00:18:15.337 INFO [localhost-startStop-1] org.apache.catalina.startup
p.HostConfig.deployDirectory Deployment of web application directory [C:\Users\h
arishp647\Desktop\full project\apache-tomcat-8.5.16\webapps\QuizApp] has finishe
d in [33] ms
22-Oct-2017 00:18:15.338 INFO [localhost-startStop-1] org.apache.catalina.startup
p.HostConfig.deployDirectory Deploying web application directory [C:\Users\haris
hp647\Desktop\full project\apache-tomcat-8.5.16\webapps\ROOT]
22-Oct-2017 00:18:15.374 INFO [localhost-startStop-1] org.apache.catalina.startup
p.HostConfig.deployDirectory Deployment of web application directory [C:\Users\h
arishp647\Desktop\full project\apache-tomcat-8.5.16\webapps\ROOT] has finished i
n [36] ms
22-Oct-2017 00:18:15.378 INFO [main] org.apache.coyote.AbstractProtocol.start St
arting ProtocolHandler ["http-nio-8080"]
22-Oct-2017 00:18:15.391 INFO [main] org.apache.coyote.AbstractProtocol.start St
arting ProtocolHandler ["ajp-nio-8009"]
22-Oct-2017 00:18:15.396 INFO [main] org.apache.catalina.startup.Catalina.start
Server startup in 8093 ms
```

Figure 19: System screenshot of entire Tomcat server startup for Deployment

4.5 User Interface

4.5.1 Sign-In Screens

The Sign-up has three user details including Name, Username, and password. The Sign-In page only asks for the username and password details.



The image shows two side-by-side login screens. The left screen is titled 'SIGN UP' in large red letters. It has three input fields: 'Name', 'User Name', and 'Password'. Below the fields are two buttons: a green button with a checkmark and the text 'REGISTER', and a grey button with an 'X' and the text 'RESET'. The right screen is titled 'SIGN IN' in large red letters. It has two input fields: 'User Name' and 'Password'. Below the fields are two green buttons: 'SIGNIN' and 'SIGNUP'.

Figure 20: Login Screens

4.5.2 User Home Page

Whenever the user logs in to the page, he/she is greeted with a welcome notice.



Figure 21: System screenshot of User homepage

4.5.3 User Dashboard Screen

The user Dashboard screen has Navigation bars of Home, Quiz & Leaderboard.



Figure 22: System screenshot of the User Dashboard

4.5.4 Please “select the exam” Screen

The user can get “Please select the exam” option by clicking the Quiz Nav-bar.

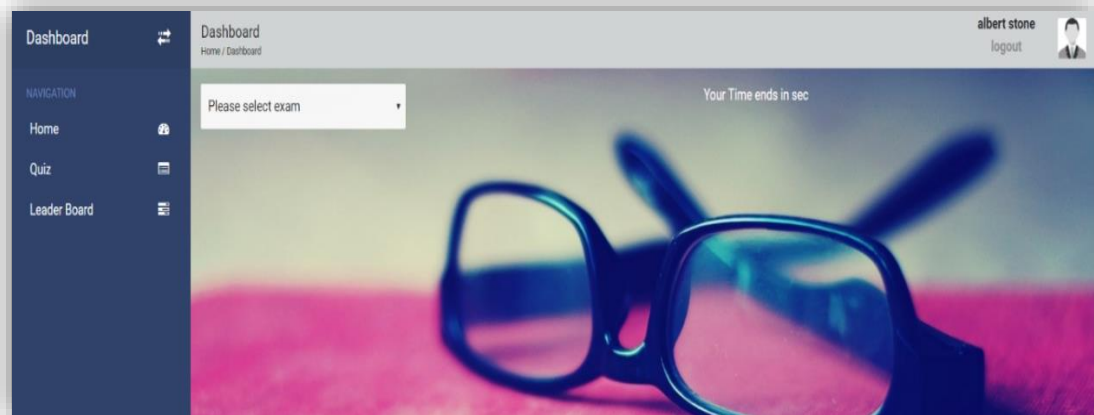


Figure 23: System screenshot of user “Please select Exam” option

4.5.5 Exam Drop down list Screen:

The user can enlist all the exams assigned by the admin and also, he/she can review them later on.

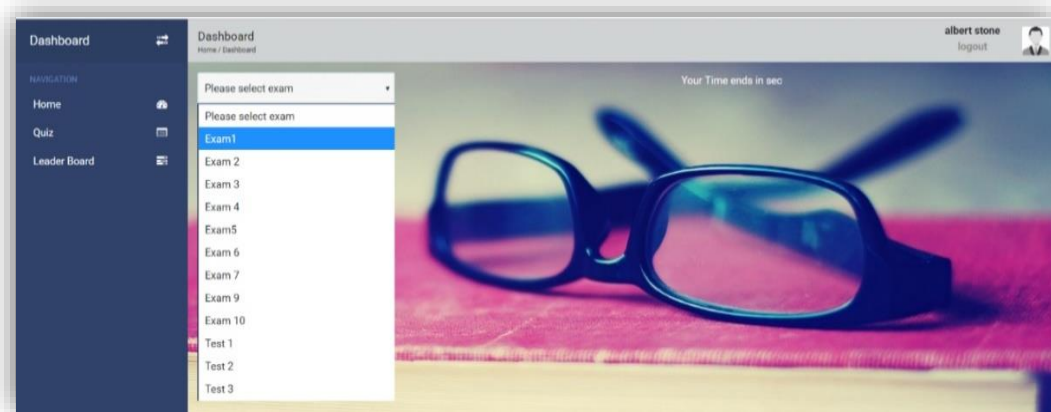


Figure 24: System screenshot of the user's view of Selecting the exam option

4.5.6 Start Exam Screen:

Once the user selects the exam, he/she is then redirected to the exam page with timely information.



Figure 25: System screenshot of the user's view of "Start Exam" button

4.5.7 Exam Screen

After the Selection of the “Start Exam” button, the time limit for the exam starts. Here the user has the option to skip certain questions and continue further. Each exam has negative marking.

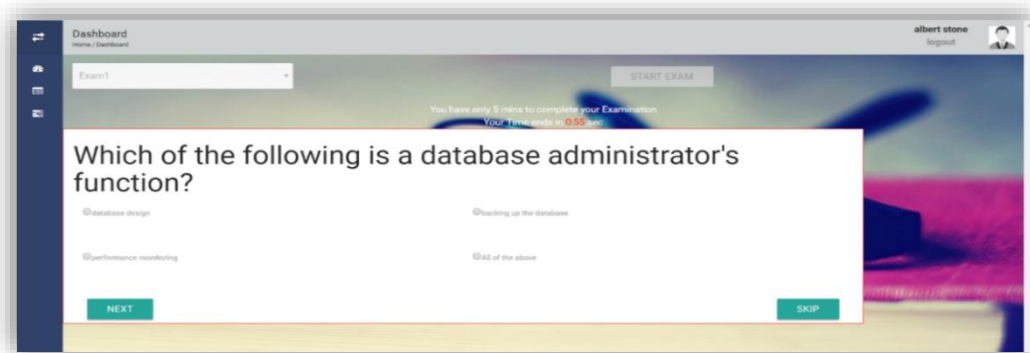


Figure 26: System screenshot of the user Exam screen

4.5.8 Score Screen

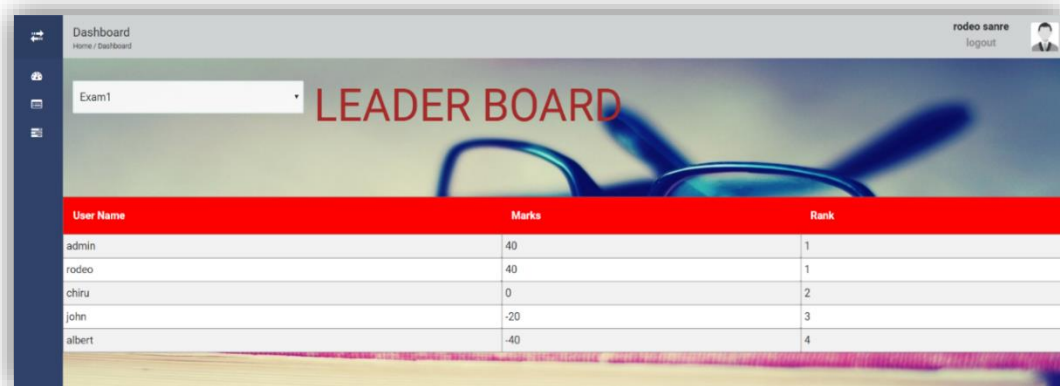
Once he/she finishes the quiz, they can view the score of his/her exam.



Figure 27: System screenshot of the user's resultant Score display

4.5.9 Leader Board Screen

When the user clicks the “Go to leaderboard” button, he/she is then redirected to the Leader Board screen, which displays the username, marks, and position of the user.



User Name	Marks	Rank
admin	40	1
rodeo	40	1
chiru	0	2
john	-20	3
albert	-40	4

Figure 28: System screenshot of user’s resultant Score and position through Leaderboard

4.5.10 Admin Home/Dashboard page

The Admin page has the options of Quiz, Leaderboard and creates exam nav-bars.

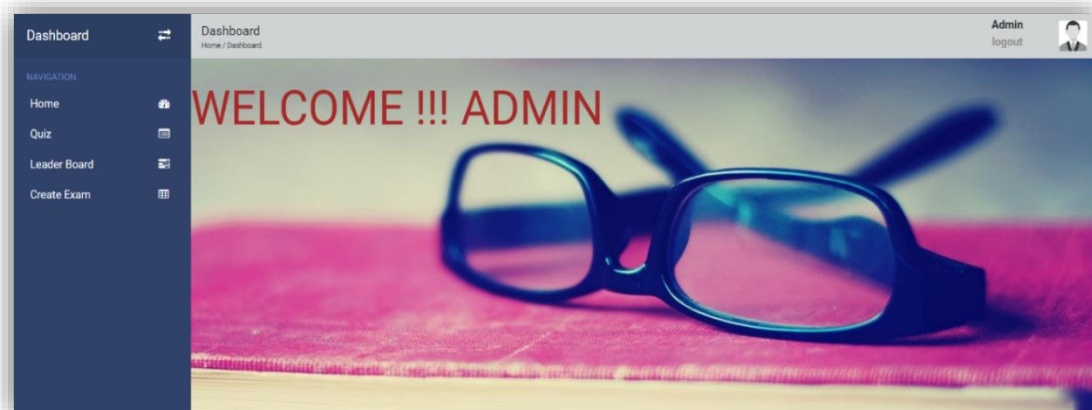


Figure 29: System screenshot of the Admin’s Home/dashboard page

4.5.11 Create Exam screen

Once, he/she clicks the “create exam” navigation bar, he/she is then redirected with Add Questions/Exams.



Figure 30: System screenshot of Admin's Create Exam options page

4.5.12 Add Questions screen

Once he/she clicks the exam, he/she is redirected to add questions for that exam.

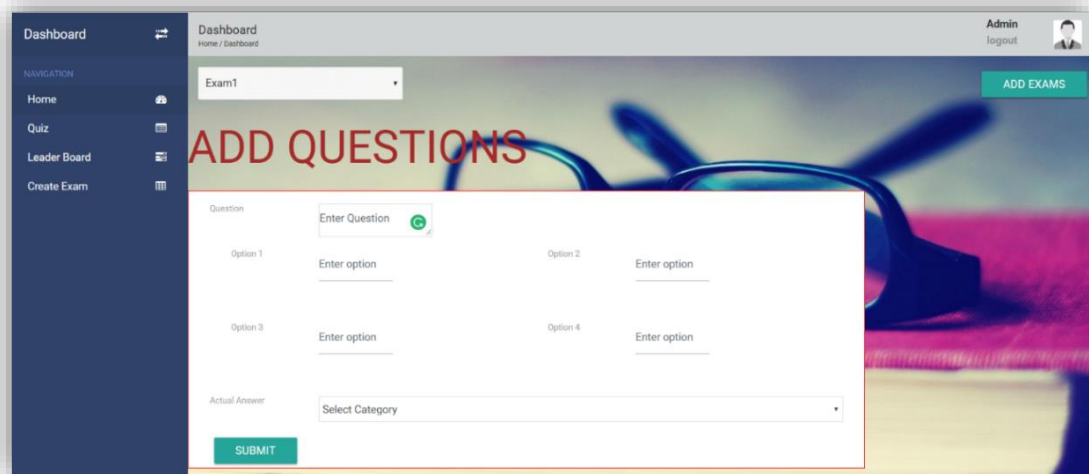


Figure 31: System screenshot of Admin's Add questions page

4.5.13 Create Exam Modal class

If he/she clicks the Add exam button, then he/she is redirected to this screen.

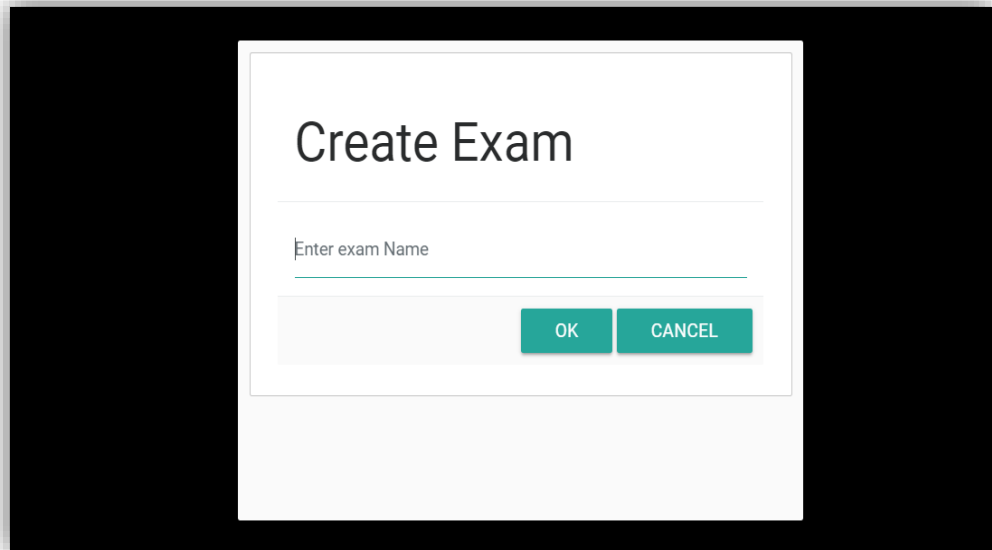


Figure 32: System screenshot of Admin's Create Exam Modal

4.6 Source Code

Quiz Application works by integrating AngularJs with the front-end and Web Services, DB as the backend. Here the Quiz app folder (UI Layer) acts as a front-end. Quizappservice acts as the web services folder.

4.6.1 Enterprise QuizAppservice (Web services)

The Quizappservice is the web service folder, which is developed using Eclipse IDE. In general, the servlet is also called as a web container.

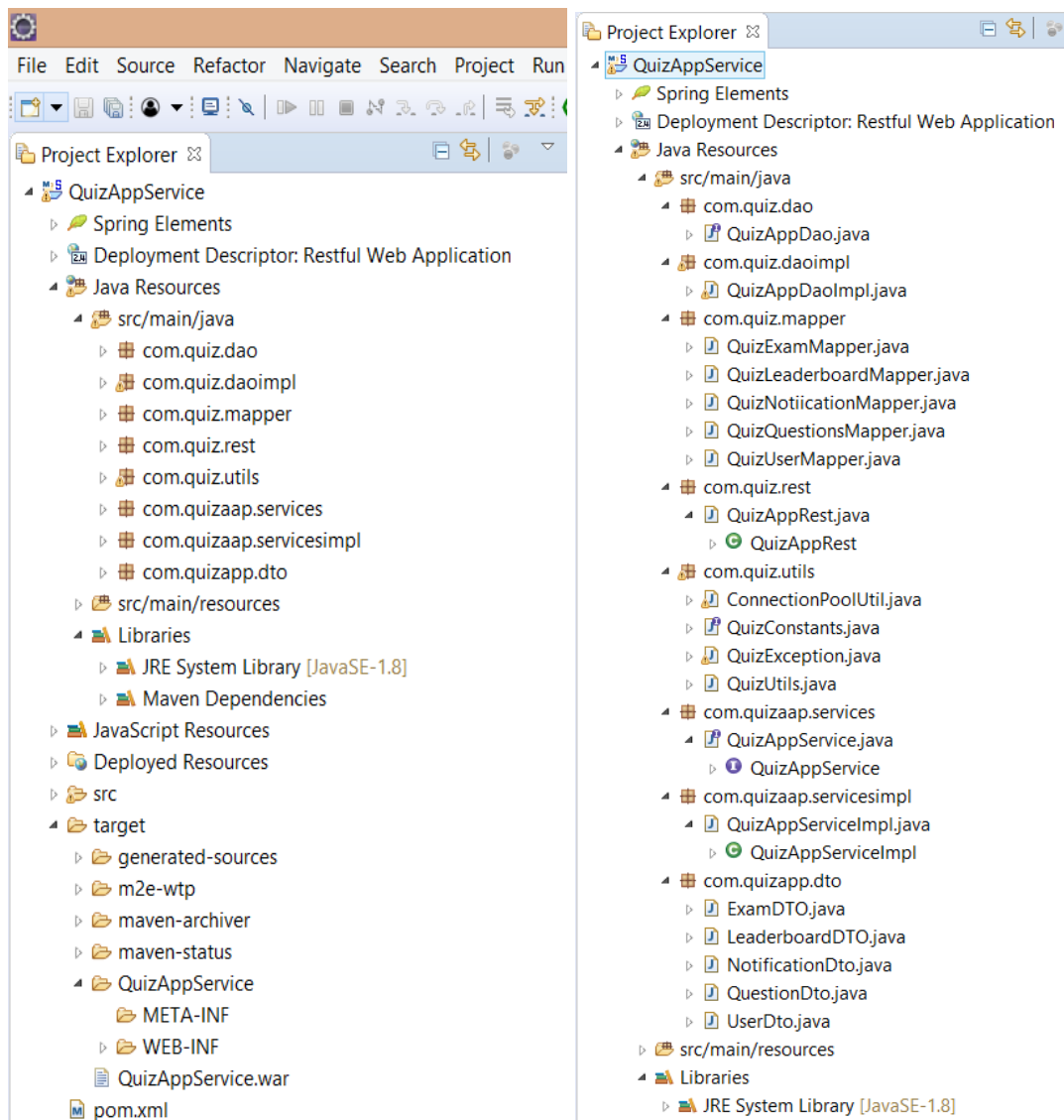


Figure 33: System screenshot of Maven project directory structure

Here in the above structure, QuizAppService is our project folder. It consists of all the files shown above. The first screen is all about the outline of the project folder. The further inner details can be shown on the above second screen of the page. For any web application, whatever the request the application faces the servlet handles it. In general, during the later stages, every servlet action details are specified by default. During later stages, Spring or REST handles whatever the request present is in ([<url-](#)

pattern>/rest/*</url-pattern>). This pattern will asks to request the servlet class(<servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-class>).



```
1<web-app id="WebApp_ID" version="2.4"
2  xmlns="http://java.sun.com/xml/ns/j2ee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
4  http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
5  <display-name>Restful Web Application</display-name>
6
7  <servlet>
8    <servlet-name>Jersey REST Service</servlet-name>
9    <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-class>
10   <init-param>
11     <param-name>jersey.config.server.provider.packages</param-name>
12     <param-value>com.quiz.rest</param-value>
13   </init-param>
14   <load-on-startup>1</load-on-startup>
15 </servlet>
16
17 <servlet-mapping>
18   <servlet-name>Jersey REST Service</servlet-name>
19   <url-pattern>/rest/*</url-pattern>
20 </servlet-mapping>
21
22
23 </web-app>
```

Figure 34: System screenshot of Deployment descriptor

This pattern “org.glassfish.Jersey” refers to the REST. This Line of code asks the controller to go for QuizAppRest.java Source File.

4.6.2 QuizAppRest.java

It has its implementation. It redefines the path structure “/rest.” Later, whatever the calls such as (/get the user), (/create a user), Rest.java gets, there it will be present. Whenever the client requests the server, REST API makes the connection between them. To access the data, we need to use this API to authenticate the client requests. The Servlet handles the client request and renders the data in JSON (JavaScript Object Notation) format. Whenever the data is returned from the server, it has a certain structure. Quiz App has the Java object created on the server side. So whenever, the client requests the data, it

sends the values of the object. Here it sends the state of the object through REST calls. Thus, whenever the user requests the web application, the controller first goes to the “web.XML”, from here to the QuizAppRest.java source file. From here it moves to the Quiz app service source file. The controller here plays a key role in the execution. REST is mainly responsible for generating Client-Server interactions. REST is helpful in making the communications much simpler. It is quite efficient to perform CRUD (Create, Read, Update, Delete) operations using Http methods. These Http methods (POST, GET, PUT, DELETE) perform the CRUD operations. Nowadays, most of the organizations use REST operations. To perform REST API calls the Jersey Rest Web service is implemented.

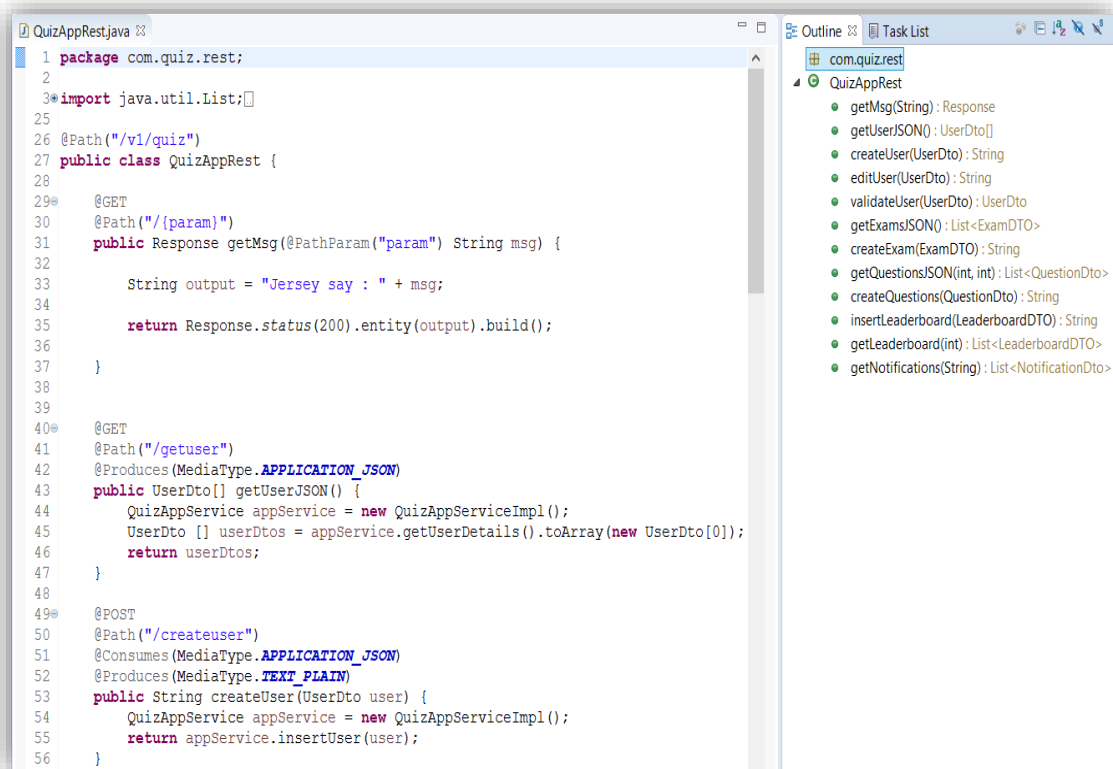


Figure 35: GET and POST methods in QuizAppRest.java

4.6.3 Quiz App Service internal functionality

These are the main architecture. First, we write all the interfaces here. Here we define the interface, because during the later periods we may want to change the features or remove certain things or update something. For that scenario, instead of changing the whole code, we define them here. We define certain basic structures for an interface without writing any implementation. For instance, Quizapp service is a kind of service.

A screenshot of a code editor showing the QuizAppService.java file. The code defines a package, imports List, and declares a public interface QuizAppService with various methods for user management, exam management, and notifications.

```
1 package com.quizaap.services;
2
3 import java.util.List;
4
11
12 public interface QuizAppService {
13
14     public List<UserDto> getUserDetails();
15     public String insertUser(UserDto user);
16     public UserDto validateUser(String userName,String password);
17     public List<ExamDTO> getActiveExams();
18     public String insertExam(ExamDTO examDTO);
19     public List<QuestionDto> getQuestions(int examId,int userId) throws QuizException;
20     public String insertQuestion(QuestionDto questionDto);
21     public String insertLeaderBoard(LeaderboardDTO leaderboardDTO);
22     public List<LeaderboardDTO> getLeaderBoard(int examId);
23     public void insertNotification(NotificationDto notificationDto);
24     public List<NotificationDto> getNotifications(String role);
25     public String editUser(UserDto user);
26
27 }
28
```

Figure 36: System screenshot of QuizAppService Interface

This service has many methods like “get user details,” “insert user details,” “validate user” Etc., For all these methods I am not writing any implementation. The other way we handle this by defining a class and then writing the implementation method. The reason behind not writing any implementation here is that the interface has some concept, which has object creation. The line of code “QuizAppService app service = new QuizAppServiceImpl();” means this class will implement the Quiz app service.

```

QuizAppService.java  QuizAppServiceImpl.java
1 package com.quizaap.servicesimpl;
2
3 import java.util.List;
4
14 public class QuizAppServiceImpl implements QuizAppService {
15
16     @Override
17     public List<UserDto> getUserDetails() {
18         QuizAppDao quizAppDao = new QuizAppDaoImpl();
19         return quizAppDao.getUser();
20     }
21
22     @Override
23     public String insertUser(UserDto user) {
24         QuizAppDao quizAppDao = new QuizAppDaoImpl();
25         return quizAppDao.insertUser(user);
26     }
27
28     @Override
29     public UserDto validateUser(String userName, String password) {
30         QuizAppDao quizAppDao = new QuizAppDaoImpl();
31         return quizAppDao.validateUser(userName, password);
32     }
33
34     @Override
35     public List<ExamDTO> getActiveExams() {
36         QuizAppDao quizAppDao = new QuizAppDaoImpl();
37         return quizAppDao.getActiveExams();
38     }
39
40     @Override
41     public String insertExam(ExamDTO examDTO) {
42         QuizAppDao quizAppDao = new QuizAppDaoImpl();
43         String notificationMessage = "Admin has created an exam, Please find exam with name "
44             + examDTO.getExamName();
45         String role = "user";
46         String status = quizAppDao.insertExam(examDTO);
47         buildNotification(quizAppDao,
48             notificationMessage, role);
49         return status;
50     }
51
52     private void buildNotification(QuizAppDao quizAppDao,
53         String notificationMessage, String role) {
54         NotificationDto dto = new NotificationDto();
55         dto.setNotification(notificationMessage);
56         dto.setRole(role);
57         quizAppDao.insertNotification(dto);
58     }
59
60     @Override
61     public List<QuestionDto> getQuestions(int examId, int userId) throws QuizException {
62         QuizAppDao quizAppDao = new QuizAppDaoImpl();
63         if(quizAppDao.isUserAlreadyAttemptedExam(userId)) {
64             throw new QuizException("user already attempted exam");
65         }
66         return quizAppDao.getQuestions(examId);
67     }
68
69     @Override
70     public String insertQuestion(QuestionDto questionDto) {
71         QuizAppDao quizAppDao = new QuizAppDaoImpl();
72         return quizAppDao.insertQuestion(questionDto);
73     }
74
75     @Override
76     public String insertLeaderBoard(LeaderboardDTO leaderboardDTO) {
77         QuizAppDao quizAppDao = new QuizAppDaoImpl();
78         String status = quizAppDao.insertLeaderBoard(leaderboardDTO);
79         String userName = quizAppDao.getUserById(leaderboardDTO.getUserId()).getFullName();
80         String examName = quizAppDao.getExamById(leaderboardDTO.getExamId()).getExamName();
81         String notificationMessage = "user " + userName + " has attempted exam " + examName;
82         buildNotification(quizAppDao, notificationMessage, "admin");
83         return status;
84     }
85

```

Figure 37: System screenshot of Implementation of QuizAppService

The implementation here means whatever methods present in the QuizAppService.java file, inside (QuizAppServiceImpl.java) this each method will be overridden, and for that, we need to do some implementation for them.

```
@Override
public List<UserDto> getUserDetails() {
    QuizAppDao quizAppDao = new QuizAppDaoImpl();
    return quizAppDao.getUser();
}
```

For example, if I have written my code using MySQL details, then during later stages, if I want to use Oracle DB, then instead of changing the whole code, we write QuizAppOracleImpl, and we use this interface for the implementation. While writing the implementation, we include all the methods present in the interface, but simply change at the object creation. In the REST folder, while creating the source file for QuizAppRest.java, there have been multiple times since the creation of the object. It is not quite recommended for an industry level standard, but for the student level, this is quite reliable. The process flow is from the controller to the service. In the service Web.XML, the request goes from here to any of the controller's. Here we call from the controller to the service. The service here plays a key role in handling the "dao-data access object" layer. This means here, there is a connection established to access and extract the data from the MySQL DB. All this code is written in the Dao source file. The service layer is establishing a communication between the controller and the data layer. The service layer here makes the data to be made available in the right form for the user methods.

4.6.4 QuizApp (AngularJS) Source code

AngularJS is a JavaScript framework, which is used to build web applications. Google is the company, which developed AngularJS. It is an open source project. It means it can be freely used, changed and shared by anyone. It is an excellent framework for building both single page applications and line of business applications. Most of the companies use AngularJS today. There are many public facing websites based on this framework. It includes the concept of Dependency Injection, two-way Data-Binding (Keeps the Model and View) intact all the time. That means a change to the Model updates the View, similarly change to the View updates the Model. It includes both Unit testing and End-End testing from the beginning. AngularJS makes the applications to build a clean MVC structure. It includes many benefits of controlling DOM elements using directives, filters, etc. To build AngularJS, we need only one script file that is Angular.JS

AngularJs operations were performed in two simple steps (i) To add a reference to the Angular script (ii) To include ng-app attribute somewhere in the HTML file. Here ng-app is a directive. There are many directives included in AngularJS. Among the directives ng-App is the starting point of AngularJS application. The angular framework will first check for this ng-App directive somewhere in the HTML page. If this directive is included, the Angular bootstraps itself. It starts to manage the section of the page that has a ng-app directive.

AngularJS further includes modules. It is just like a Main () method in many other applications. The module acts as a container for different parts such as services, directives,

filters, etc. It specifies how the AngularJS is bootstrapped. Creating a module is a straightforward. A module can depend upon other modules.

In angular, a controller is a JS constructor function. The role of it is to build a model for the view to display. The model here is termed as data. In the real world, the controller may call to the web service, which retrieves the data from the database. Here in the application, there are various controllers such as Departmentctrl, Homectrl, Leaderctrl, Loginctrl, Resultant, Signupctrl, Userctrl.

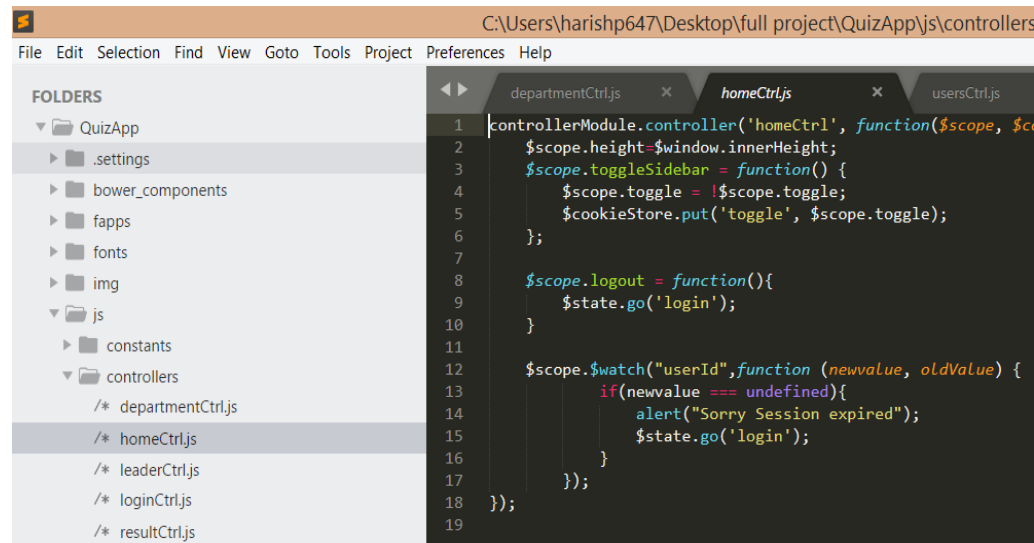


Figure 38: System screenshot of homeCtrl.js

In general, most of the web applications use the services. Most of the objects use the services. A service in angular is simply an object that provides service, which can be reused within the angular application. The Angular service object has properties and methods just like any other JavaScript object. These services can be reused in the application. It has a lot of built-in services. There is two builds in services such as \$http

and \$log. \$http service is used to make Ajax calls. \$ log service is used to log objects to the console. Services are very useful for debugging the applications.

4.6.5 REST API calls

Authentic state exchange (REST) or RESTful [16] web administrations are a method for giving interoperability between PC frameworks on the Internet. REST-consistent Web administrations permit asking for frameworks to get to, and the control printed portrayals of Web assets utilizing a uniform and predefined set of stateless operations. "Web assets" were first characterized on the World Wide Web as archives or records distinguished by their URLs. However, today they have a substantially more non-exclusive and conceptual definition enveloping everything or element that can be recognized, named, tended to or took care of, in any capacity at all, on the Web. In a RESTful Web benefit, demands made for an asset's URI will evoke a reaction that might be in XML, HTML, JSON or some other characterized design.

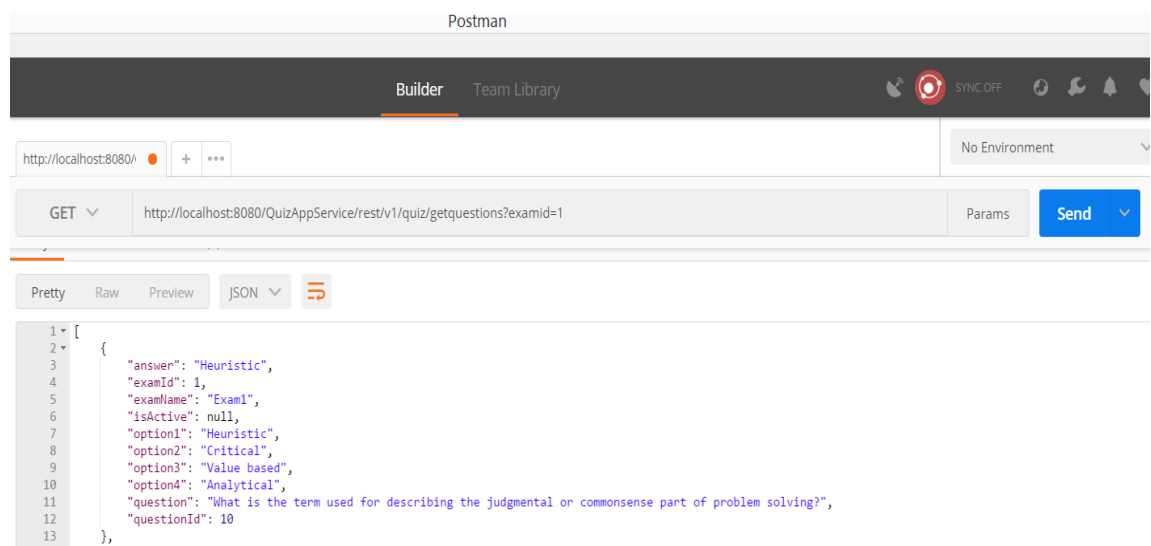


Figure 39: System screenshot of GET call (getquestions) through Postman App

The reaction may affirm that some adjustment has been made to the “put away” asset, and it might give hypertext connects to other related assets or accumulations of assets. Utilizing HTTP, as is more normal, the sort of operations, accessible, incorporates those predefined by the HTTP strategies GET, POST, PUT, DELETE, etc. By utilizing a stateless convention and standard operations, REST framework goes for quick execution, unwavering quality, and the capacity to develop, by re-utilizing segments that can be overseen and refreshed without influencing the framework all in all, even while it is running.

```
http://localhost:8080/QuizAppService/rest/v1/quiz/validateuser
{
  "fullName": "Harish",
  "id": 1,
  "isActive": "y",
  "password": "harish",
  "role": "user",
  "userName": "harish"
}
|
form data username:harish
password:harish

http://localhost:8080/QuizAppService/rest/v1/quiz/createuser

Body:
{
  "fullName": "Admin",

  "isActive": "y",
  "password": "admin",
  "role": "admin",
  "userName": "admin"
}

Output : sucess/failed

http://localhost:8080/QuizAppService/rest/v1/quiz/activeexams
[
  {
    "examId": 1,
    "examName": "Exam1",
    "isActive": "y"
  }
]
```

Figure 40: Screen shot of POST call (validate, create) user & GET call (activeexams)

```

http://localhost:8080/QuizAppService/rest/v1/quiz/getquestions?
examid=1
[
  {
    "answer": "Heuristic",
    "examId": 1,
    "examName": "Exam1",
    "isActive": null,
    "option1": "Heuristic",
    "option2": "Critical",
    "option3": "Value based",
    "option4": "Analytical",
    "question": "What is the term used for describing the
judgmental or commonsense part of problem solving?",
    "questionId": 10
  }
]

http://localhost:8080/QuizAppService/rest/v1/quiz/createquestions

{
  "answer": "10",
  "examId": 1,
  "examName": "Exam1",
  "isActive": "y",
  "option1": "10",
  "option2": "7",
  "option3": "12",
  "option4": "44",
  "question": "Tendulker Tshirt number"
}

http://localhost:8080/QuizAppService/rest/v1/quiz/getleaderb
oard?examId=1

[
  {
    "examId": 1,
    "lId": 0,
    "totalMarks": 10,
    "userId": 1
  }
]

http://localhost:8080/QuizAppService/rest/v1/quiz/getnotification
s?role=admin

[

```

Figure 41: Screen shot of GET call (getleaderboard, getnotifications) and POST call(createquestions)

```

http://localhost:8080/QuizAppService/rest/v1/quiz/getleaderboard?examId=1

[
  {
    "examId": 1,
    "lId": 0,
    "totalMarks": 10,
    "userId": 1
  }
]

http://localhost:8080/QuizAppService/rest/v1/quiz/getnotifications?role=admin

[
  {
    "notificationId": 1,
    "notification": "user Srikanth has attempted exam Exam1",
    "read": null,
    "role": "admin"
  }
]

http://localhost:8080/QuizAppService/rest/v1/quiz/createleaderboard

{
  "examId": 1,
  "totalMarks": 15,
  "userId": 6
}

```

Figure 42: Screen shot of GET call (get leaderboard, get notification) and POST call (create leaderboard).

Chapter 5: Conclusion

5.1 Summary

In this project, Quiz Application with a new approach of the Leader Board console is successfully developed. This application has been designed to have a tiered architecture. The real strength of this application lies in its robust and flexible data model. However, not all users want to have to know this data model inside and out. The API layer allows a developer to know the REST endpoints and read/save to them. This layer is used in different channels like Windows, IOS, Android, etc., This application allows the user to know not only about his score, but also his position among several other attempted users of a Quiz exam. This application, thus brings a new perspective of competitive spirit among the users through an organizational level.

5.2 Development Experience

This project helped me to learn many of the new technologies that I am not familiar within. My advisor assigned me tasks that made intact with these new technologies. This experience motivated me for my future endeavors to become a Full Stack Developer. From this project, I gained experience with the Maven framework which I was not familiar with before. This framework helped me learn about the adding the dependencies by itself rather than adding manually. I came to learn about trends of AWS services and their impact on the future. The Front-end technology (Angular Js) made me experienced in developing Single page application and came to know about its flexible nature. I came to experience REST calls, REST web services. These web services can be connected to any mobile platform. My main intention is to work on the collaboration of Angular Js and web services through which I came up with a Leader Board application.

5.3 Challenges and Future Scope

I experienced several challenges throughout this project. As I was confronted with these challenges, it was a learning experience for me. Regarding this project, I completed the application. During the initial stages of the project, I came up with several features. Each of these features was prioritized based on their importance through agile development. Some of the most important ones were enlisted and would like to work more for the future enhancements.

While the admin adds the questions in the exam, he/she should have two screens. The application here does not show the previous questions, while the admin adds the questions. All the back-end actions were present, but through the UI layer, the connections need to be more specified. So, it is beneficial to have two screens to view the previous ones.

The admin should have the feasibility to enable or disable the user. So, he/she could disable the unwanted users and can confine the application to only specified users.

I have implemented the notification feature through the back-end of the database. Here every action gets displayed through MySQL browser. I wanted to make this notification visible to the admin through the panel. So, he/she can know all the actions performed in the application.

For future developments in the app, instead of signing in using the traditional method, we can add authentication using Google, Facebook, Twitter, etc. making it easier

to log in for the users. This idea can bring sustainable change to the present way of learning and can improve the learning skills much better for modern society.

For scalability of many web applications, Google firebase provides all the tools required to build an app. It provides scalability, data-analytics features to visualize the reports from the application

References

1. "Best Tools to measure employee performance," Retrieved on Feb22, 2017.
<https://www.business.com/articles/14-best-tools-to-measure-employee-performance/>
2. "Deloitte Global Human Capital trend 2014" Retrieved on Oct24, 2014.
<https://www2.deloitte.com/global/en/pages/human-capital/articles/human-capital-trends-2014.html>
3. "Web-Framework," Retrieved on August 25th,2016 from
https://en.wikipedia.org/wiki/Web_framework.
4. "Secure Three-Tier Architecture Pattern," Retrieved on August 22nd,2008 from
<http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4606734&isnumber=4606618>
5. "Digital in 2017 Global Overview" Retrieved on January 24th,2017 from
<https://wearesocial.com/special-reports/digital-in-2017-global-overview>
6. "MVC Framework" Retrieved on December 16th,2016 from
<http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7943348&isnumber=7943265>
7. "Maven" Retrieved on March 30th, 2002 from
<https://maven.apache.org/>

8. “Maven illustration” Retrieved on June 12th,2015 from
<https://www.tutorialspoint.com/maven/>
9. “JAR” Retrieved on November 19th,2015 from
[https://en.wikipedia.org/wiki/JAR_\(file_format\)](https://en.wikipedia.org/wiki/JAR_(file_format))
10. “J2EE7” Retrieved on December 12th,1999 from
<https://docs.oracle.com/javaee/7/tutorial/>
11. “Eclipse IDE” Retrieved on November 4th, 2001 from
<https://eclipse.org/ide/>
12. “Jersey Restful web service” Retrieved on November 20th,2012 from
<http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6419518&isnumber=6419130>
13. “Angular JS” Retrieved on November 20th, 2012 from
<https://angularjs.org/>
14. “Essential Scrum: A practical guide to the Most Popular Agile process” Retrieved on January 19th, 1993 from
<http://ptgmedia.pearsoncmg.com/images/9780137043293/samplepages/0137043295.pdf>

15. “Apache Tomcat,” Retrieved on December 15th,1993 from

<http://tomcat.apache.org/>

16. “REST API calls,” Retrieved on June 9th,2017 from

https://en.wikipedia.org/wiki/Representational_state_transfer