|  |  |
| --- | --- |
| Project  **JAVA ADVANCED** | Abstract  Final Project (Quiz Management) documentation of JAVA Advanced.  Bhrigu Mahajan  M.E (SDM) |

Table of Contents

[Subject description - 3 -](#_Toc507093259)

[Subject analysis - 3 -](#_Toc507093260)

[Major features - 3 -](#_Toc507093261)

Expected results - 3 -

Scope of the application……………………………………………………………………………………………………………..- 3 -

**Limitations…………………………………………………………………………………………………………………………………- 3 -**

Evolutions…………………………………………………………………………………………………………………………………..- 3 -

[Conception.](#_Toc507093264).........................................................................................................................- 4 -

[Identities.](#_Toc507093266).................................................................................................................................- 4 -

[Users.](#_Toc507093267).......................................................................................................................................- 4 -

[Global Application Flow - 10 -](#_Toc507093268)

**Database Design …………………………………………………………………………………………..…………………………. - 14 -**

[Configuration - 15 -](#_Toc507093273)

**Bibliography…………………………………………………………………………………………………………………….- 15 -**

**Website……………………………………………………………………………………………………………………………………**- 15 -

# **Subject description**

This project is the implementation of a Quiz Management Software. The project was developed during the semester’s class and has been completed during the last week during the end of the course.

The implementation of this project has been done by using Frontend (HTML, Angular JS), Backend (REST, Java 8), Database (Derby), Business Logic Framework (Hibernate, Spring) and Server (Tomcat).

# **Subject analysis**

## **Major features**

The IAMProject has 4 major features.

* *Login Authentication*: A User should be authenticated before been able to ….
* *Manage Identities*: The Admin should be able to
  + Add Student
  + Update Student
  + Delete Student
  + Search Student
* *Manage Questions*: The Admin should be able to
  + Add Questions
  + Update Questions
  + Delete Questions
  + Search Questions
* *Give Quiz*: The student should be able to
  + View Questions
  + Give Answers

**Frameworks**

Maven

**Maven** is a [build automation](https://en.wikipedia.org/wiki/Build_automation) tool. It helps handle jar dependencies. Maven can manage a project's build, reporting and documentation from a central piece of information. Maven helps handle jar dependencies easily through the pom.xml file.

Spring

**Spring** Framework is a Java platform that provides comprehensive infrastructure support for developing Java applications. Spring handles the infrastructure so you can focus on your application. Spring enables you to build applications from “plain old Java objects” (POJOs) and to apply enterprise services non-invasively to POJOs. This capability applies to the Java SE programming model and to full and partial Java EE.

Hibernate

**Hibernate** ORM(Hibernate in short) is an [object-relational mapping](https://en.wikipedia.org/wiki/Object-relational_mapping) tool for the [Java](https://en.wikipedia.org/wiki/Java_(programming_language)) programming language. It provides a [framework](https://en.wikipedia.org/wiki/Software_framework) for mapping an [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming) domain model to a [relational database](https://en.wikipedia.org/wiki/Relational_database). Hibernate handles [object-relational impedance mismatch](https://en.wikipedia.org/wiki/Object-relational_impedance_mismatch) problems by replacing direct, [persistent](https://en.wikipedia.org/wiki/Persistence_(computer_science)) database accesses with high-level object handling functions

Junit

JUnit is a [unit testing](https://en.wikipedia.org/wiki/Unit_testing) [framework](https://en.wikipedia.org/wiki/Software_framework) for the [Java programming language](https://en.wikipedia.org/wiki/Java_(programming_language)). JUnit has been important in the development of [test-driven development](https://en.wikipedia.org/wiki/Test-driven_development), and is one of a family of [unit testing](https://en.wikipedia.org/wiki/Unit_testing) frameworks which is collectively known as [xUnit](https://en.wikipedia.org/wiki/XUnit" \o "XUnit) that originated with [SUnit](https://en.wikipedia.org/wiki/SUnit" \o "SUnit).

Log4j2

It is used for logging events. In Log4J2, an appender is simply a destination for log events; it can be as simple as a console and can be complex like any RDBMS. Layouts determine how the logs will be presented and filters filter the data according to the various criterion.

**Expected results**

Users with administrative privileges can create, delete, modify and view users and questions.

Users without administrative can view questions and give quizzes which are stored in database by the admin.

**Scope of the application**

**Limitations**

* Services are not secured.
* Password is exposed in payload sent for authentication.

**Evolutions**

* Multiple type of quizzes in same quiz.
* Implementing Security features.
* Better UI.
* Implement the Enabled Flag

**Data structures**

**User**

Users are basically the entities that are going to be created so that they can either create a quiz or give a quiz depending on the administrative rights. All users are going to be stored in a Derby database.

As per the requirements an identity can have:

* UId: User\_Role\_Id, Primary Key in the Users table.
* Email: represented as a string.
* Enabled: Flag
* Password: represented as a string.
* Role: Defines administrative rights
* Username: Unique

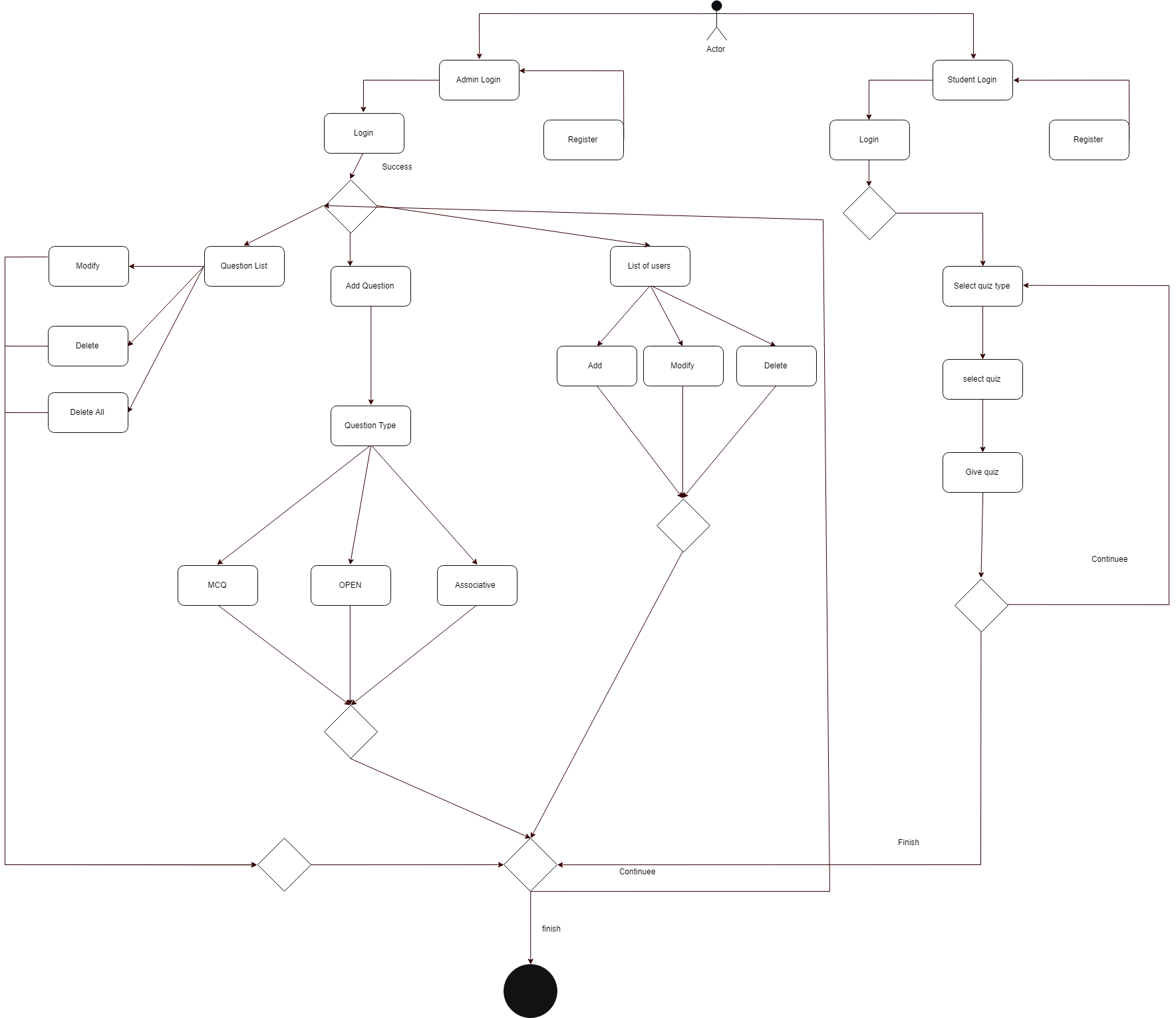
**Question**

Question table contains all the questions with their answers that will be displayed for students. The user with administrative rights can create questions and the student gets to view all the questions

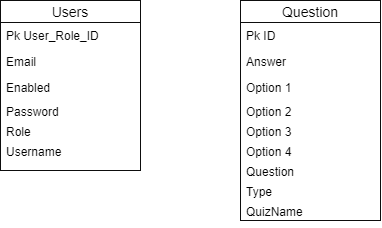
As per the requirements question tables has:

* Id: Question id, Primary Key in the Question table.
* Answer: represented as a string.
* Option 1: represented as a string.
* Option 2: represented as a string.
* Option 3: represented as a string.
* Option 4: represented as a string.
* Question: represented as a string.
* Type: Defines the type of the question
* QuizNmae: Defines the name of the quiz which is a part of the quiz

**Global Application Flow**



**Database Diagram**



**Configuration Description**

**Bibliography**

**Websites:**

<http://thomas-broussard.fr/work/java/courses/project/fundamental.xhtml>