Marcus Ekon

Advanced Java - Project Report
(2018-11-23)

Spring 2018

Table of Contents

Subject description

Examinations are a major part of every education system. Most of those examinations are given as quizzes to the students.

The aim of this project is to provide an automated Quiz Management application to facilitate quiz-based assessment. The system should allow Lecturers to constitute and manage different quiz assessment. It should also automatically correct the appropriate questions *(MCQ, True/False)* in the assessment.

The final application will be built using a Micro-service architecture with a Java (Spring Based) API and a front-end Single Page Application.

Subject analysis

Major features

This application has 2 major features: **Quiz Creation and Quiz Execution.**

Quiz Creation:

(Create Question) Lecturers should be able to create different types of questions in the application.

(Create Quiz) The questions can then be organized into assessment for students.

This functionality would require a user account to track the assessment created by each lecturer and allow students to take the quiz.

Quiz Execution:

Students should be a to take quizzes organized by their lecturers through this web application. They should also be able to check their score at the end of the assessment.

Application Feasibility

Operational & Technical Feasibility

Backend (Java API)

Java Enterprise Edition (JEE) provides several specifications for developing web applications including the RESTful Web Services specification which provides support for building web services according to the REST architectural pattern. This application will be built using the Spring Boot Framework. Spring is an application framework and inversion of control container for the Java platform (Pankaj, 2017). It provides extensions for building web applications on top of Java EE and Spring Boot helps developers build spring applications with minimal configuration. More information on Spring and Spring Boot can be found on this link.

JEE also provides specification for dealing with Data Persistence in a Java application (JPA) and interacting with databases. Hibernate will be used for this project. More information on the hibernate implementation can be found here: http://courses.coreservlets.com/Course-Materials/hibernate.html

Front-end (JavaScript SPA)

The user interface of this application will be built using JavaScript, HTML and CSS. There are many JavaScript frameworks for building SPA web user interfaces with great client-side interactions. ReactJS will be used to build the front-end of this application. ReactJS is an open-source JavaScript library specifically designed for building Single Page Applications. It provides functionalities to build modular, scalable, and fast user interfaces for web applications which can change data without reloading the web page (Pandit, 2018).

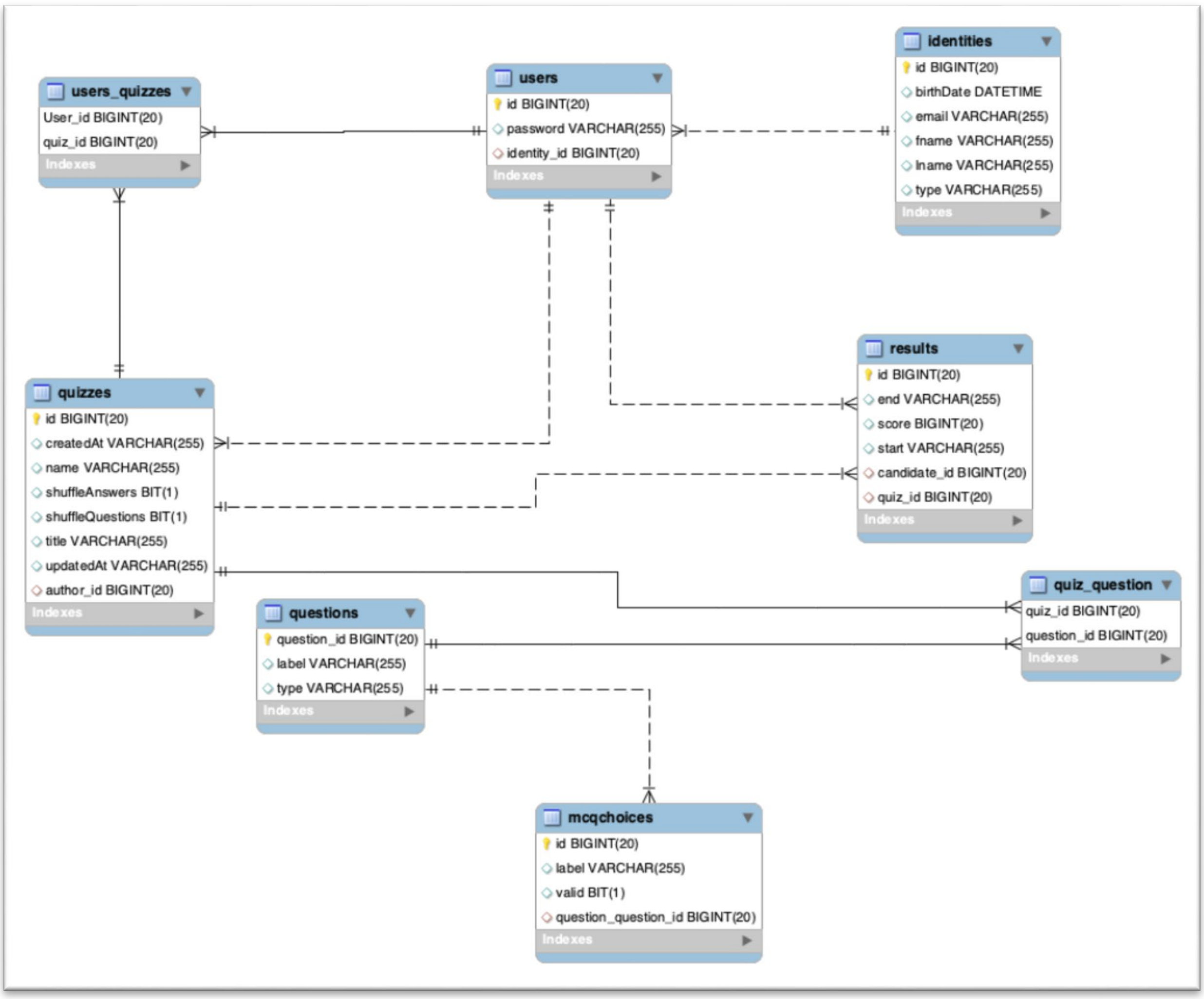
Legal Feasibility

Due to the invasive nature of modern technology, various regulations have been put in place to ensure data controller respect the privacy rights of the users' whose information they possess.

In France, these regulations are set and enforced by the *Commission National de l'Informatique et Des Libertés* (CNIL).

The functionalities offered by this application should respect the regulations outlined in the data protection act. The personal information stored using this application should be adequately protected and kept up-to-date where applicable.

Data description



Application UML Diagram

This application will contain 8 entities: *Identity, User, Quiz, User_Quiz, Question, Quiz_Question, Result, MCQchoice.*

Expected results

At the end of this project, the application should be usable through a JavaScript Single Page Application (SPA) built with the ReactJS framework.

Scope of the application (limits, evolutions)

Scope

The main purpose of this application is to allow Lecturers to create online assessment for their students.

Quizzes can have different types of questions: **Multiple Choice Questions (MCQ), True/False questions, and Text questions.**

MCQ

These are questions which can have multiple answers or single answers.

True/False

These are questions which can be answered by either True of False by the students.

Text

These questions require the student to type out their answer and can be opinion based. As such, it is not in the scope of this system to provide automatic marks for these.

Managing Quizzes

The system should allow the users (Lecturers) to create new questions, organize them into quizzes and view the results of the candidates (students) who take the quiz.

Evolutions

Required Upgrades

Due to time constraints on this project, some required functionalities such as *User Signup* and *Authentication* which would normally be the first functionalities were not implemented, and there aren't as many features as there could be.

Improvements

Baring those, there are other upgrade that the system could have.

Currently, there is a sperate view to create questions and quizzes. The user's workflow could be further improved by allowing them to create questions and quizzes using the same form. They could both choose existing questions and create new ones as part of the quiz creation.

Regarding the questions, Rich text support could be added to allow better formatting for the questions, include support for things such as code highlighting, in questions, the suggested answers, and the candidates' answers.

Although the system provides support for 3 types of questions *(MCQ, Text, and True/False)*, only *MCQ* and *True/False* are correctly handled. The text part is not automated and needs to be manually corrected by the lecturers, without which, a complete quiz report cannot be generated. This process could be improved by allowing the lecturer to specify some necessary keywords in the answers for each *Text* question. The system could then grade the candidates' answers based on the keywords specified for the question.

Furthermore, the lecturer could be given more control over the grading of *Text* questions, granting or not marks based on his discretion in cases where the automated grading does not perform as expected.

Conception

Design & Architectural Patterns

In Software engineering, design patterns are code structing and organization recommendations that represent some of the best Object-Oriented practices for solving recurring problems (Powell-Morse, 2017).

Some design patterns where used in the implementation of this software to avoid code duplication and facilitate the maintenance of the code base in the future. They are briefly outlined below.

Singleton

The Singleton pattern restricts the number of instances of a class to one at a time. This pattern is commonly used when the access to an object is required across the entire application (Powell-Morse, 2017).

Data Access Object (DAO) Pattern

The DAO pattern is a structural pattern that allows developers to isolate the application/business layer from the persistence layer with the help of an abstract API (baeldung, 2018). This way the underlying complexities of performing CRUD operations on the persistence layer, whatever it may be, are abstracted on the application layer which remains database agnostic.

The Façade Pattern

This pattern is used to hide the complexities in a subsystem by providing it with a unified and simplified interface or set of interfaces (Java Point, 2018).

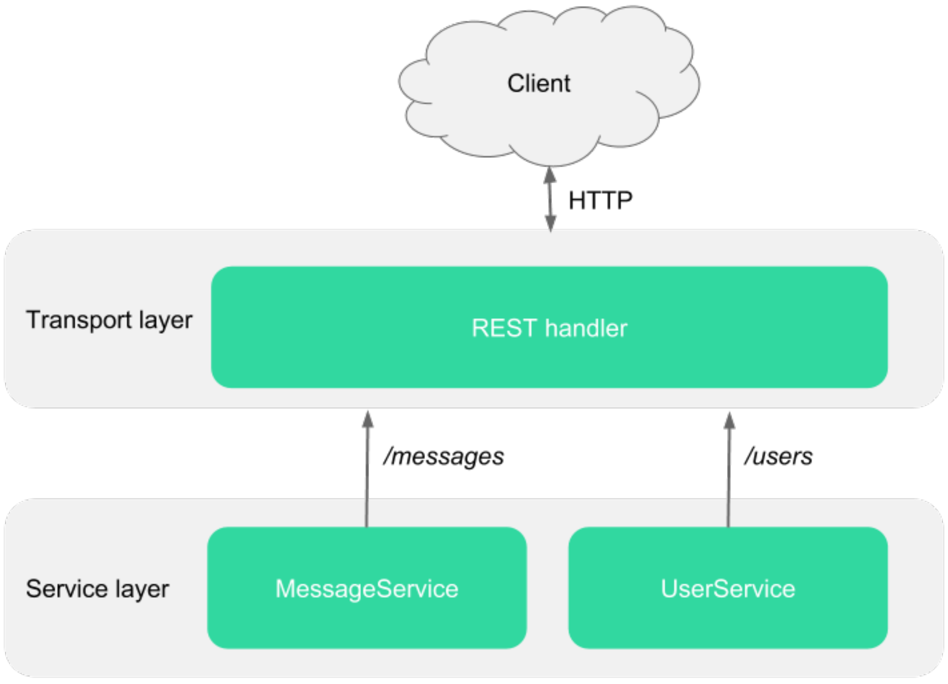
Service Layer

The service layer represents a common interface to the application logic that different clients can use. It implements potentially complex business rules that may require transactions spanning several resources (Stafford, 2018). This architectural pattern was used in this project to avoid the code duplication implementing the business operations directly on each of the applications interfaces may cause. This also affords a more maintainable application code base. This also allows us to more accurately test the components and business rules of the application.

RESTful Service Layer

Representational State Transfer (REST) is an architectural pattern that defines a set of constraints for building web APIs (Luecke, 2018). In REST applications, the communication between client and server is stateless, meaning every request is self-sufficient. It provides a uniform interface to web services, allowing any client to easily interact with the Java API for the quiz application.

By using a RESTful service, we get all the advantages afforded by a service layer architecture as described above, including the ability to easily swap out or add to our REST service, handlers for other protocols such as SOAP.



RESTful Services and HTTP Transport (Luecke, 2018).

Data Modelling

As mentioned above, this application's domain was modelled through 8 entities: *Identity, User, Quiz, User_Quiz, Question, Quiz_Question, Result, MCQchoice.*

Identity
This contains details of the user's account to supplement the *User* entity.

User
Models a typical user account and contains an id and password.

Quiz
Contains details about the quizzes, such as timestamps, name, and author id.

User_Quiz
Maps users (lecturers) to their (created) quizzes.

Question
Contains details about the questions such as the text, the id, and the question type.

MCQChoice
Contains the set of possible answers for MCQ type questions, with an attribute *(valid)* to identify the correct option(s).

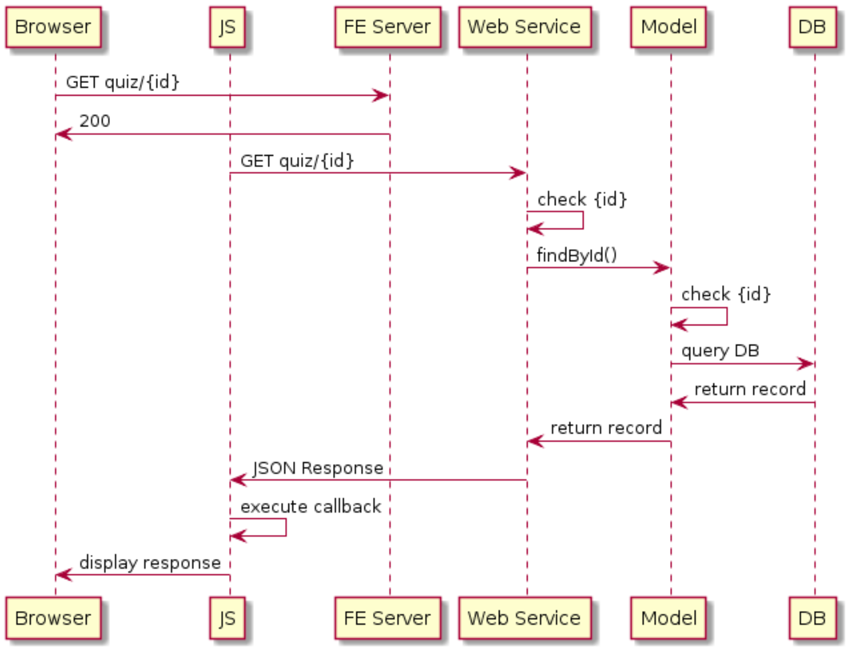
Quiz_Question
Maps quizzes to the questions they contain.

Result
Contains details about the candidates' (students) examinations such as their score, the time of they completed the examination, their id, and the quiz id.

Global Application Flow & Major Features Schema

This section will document the interactions between the different layers of the application for some common operations.

Get Request (Sequent Diagram)



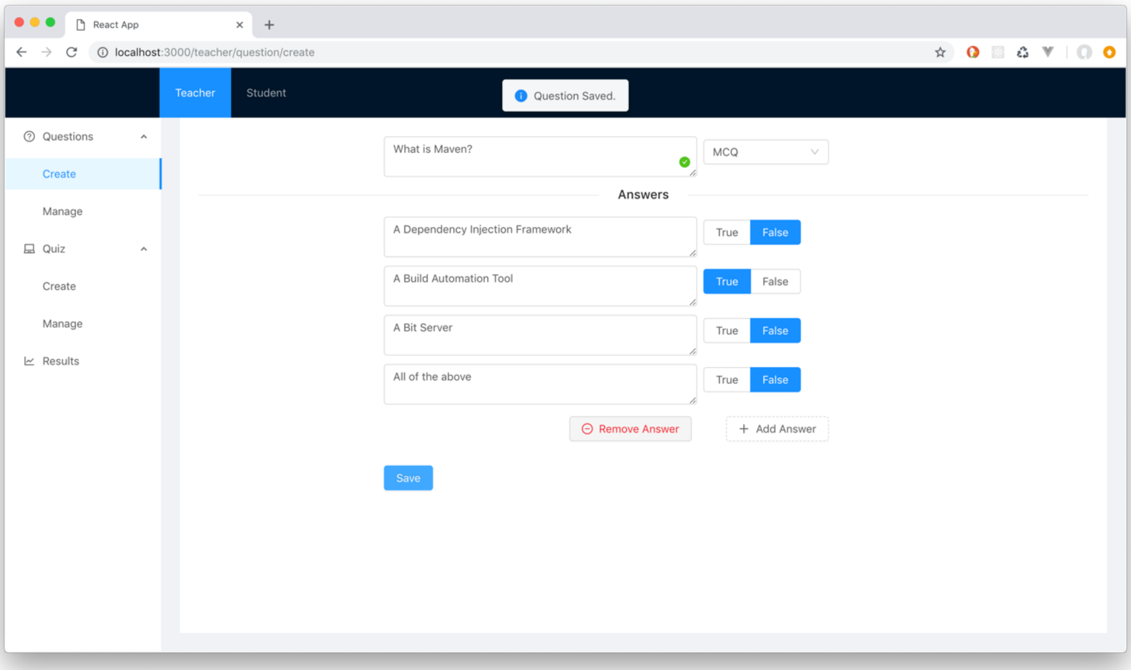
Get Quiz Sequence Diagram

The image above represents a sequence diagram of a Get request for a quiz resource. It shows all the layers in the application between the Client Browser and the Persistance Layer.

Operations description

Create Question

Click on Teacher, then in the left side bar, expand the “Questions” accordion and click on “Create”. Enter the question, select the type, and provide the choices for the student along with their validity.



localhost:3000/teacher/question/create

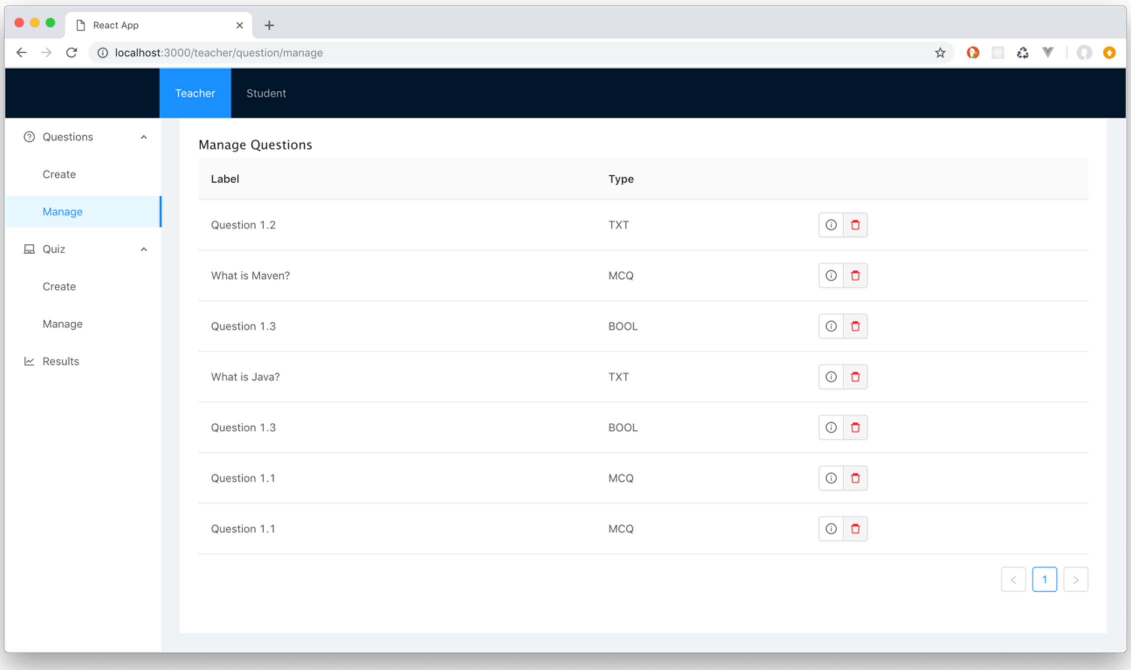
Question Saved.

What is Maven? — MCQ

A Dependency Injection Framework	True [✓ False]
A Build Automation Tool	[✓ True] False
A Bit Server	True [✓ False]
All of the above	True [✓ False]

Create Question View

A message is displayed when a question is successfully saved.



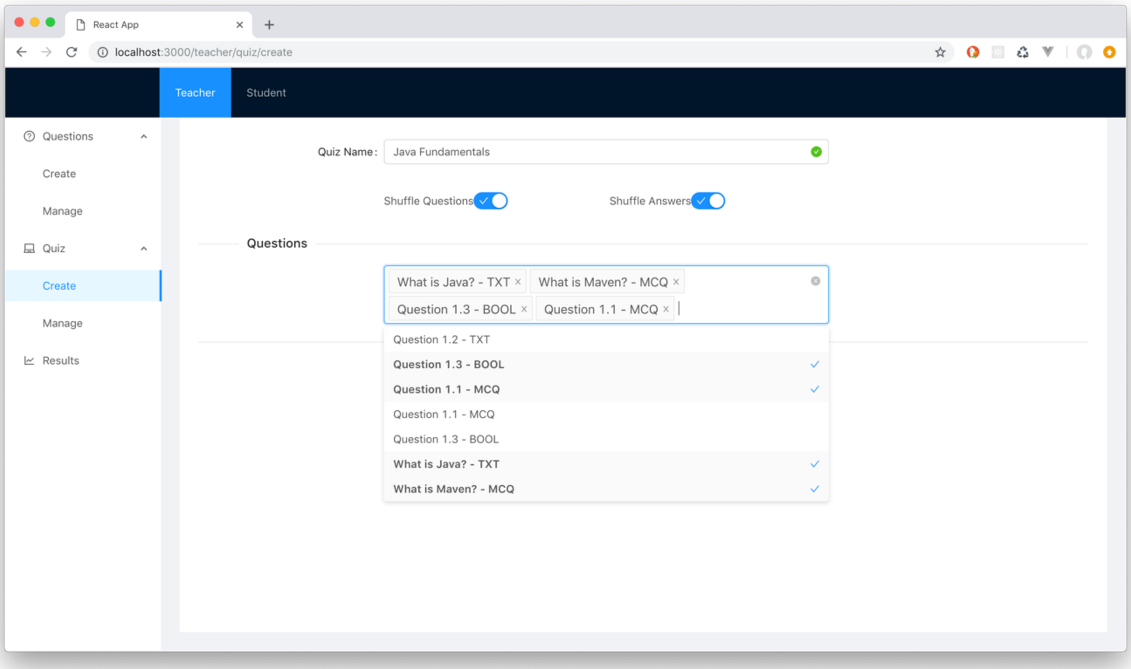
Manage Questions

Label	Type
Question 1.2	TXT
What is Maven?	MCQ
Question 1.3	BOOL
What is Java?	TXT
Question 1.3	BOOL
Question 1.1	MCQ
Question 1.1	MCQ

Manage Question View

Create Quiz

To create a Quiz, click on “Teacher”, expend the “Quiz” accordion and choose “Create”. Fill and submit the form. A message is displayed when the quiz was successfully created.



Quiz Name : Java Fundamentals

Shuffle Questions ✓

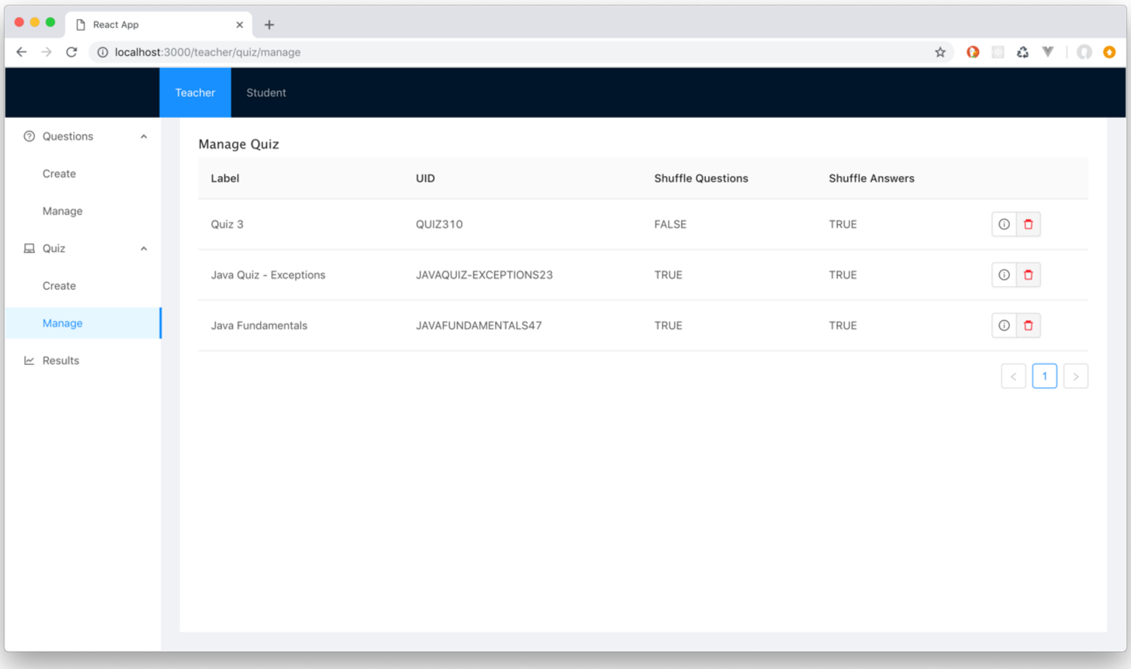
Shuffle Answers ✓

Questions

What is Java? - TXT × What is Maven? - MCQ × Question 1.3 - BOOL × Question 1.1 - MCQ ×

- Question 1.2 - TXT
- Question 1.3 - BOOL ✓
- Question 1.1 - MCQ ✓
- Question 1.1 - MCQ
- Question 1.3 - BOOL
- What is Java? - TXT ✓
- What is Maven? - MCQ ✓

Create Quiz View



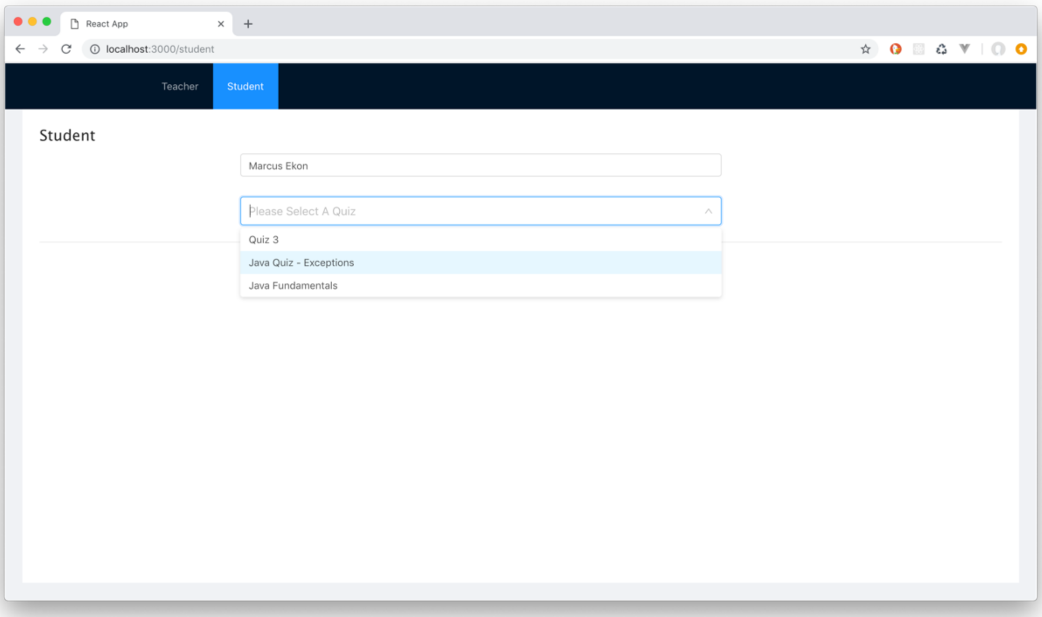
Manage Quiz

Label	UID	Shuffle Questions	Shuffle Answers
Quiz 3	QUIZ310	FALSE	TRUE
Java Quiz - Exceptions	JAVAQUIZ-EXCEPTIONS23	TRUE	TRUE
Java Fundamentals	JAVAFUNDAMENTALS47	TRUE	TRUE

Manage Quiz View

Run Quiz

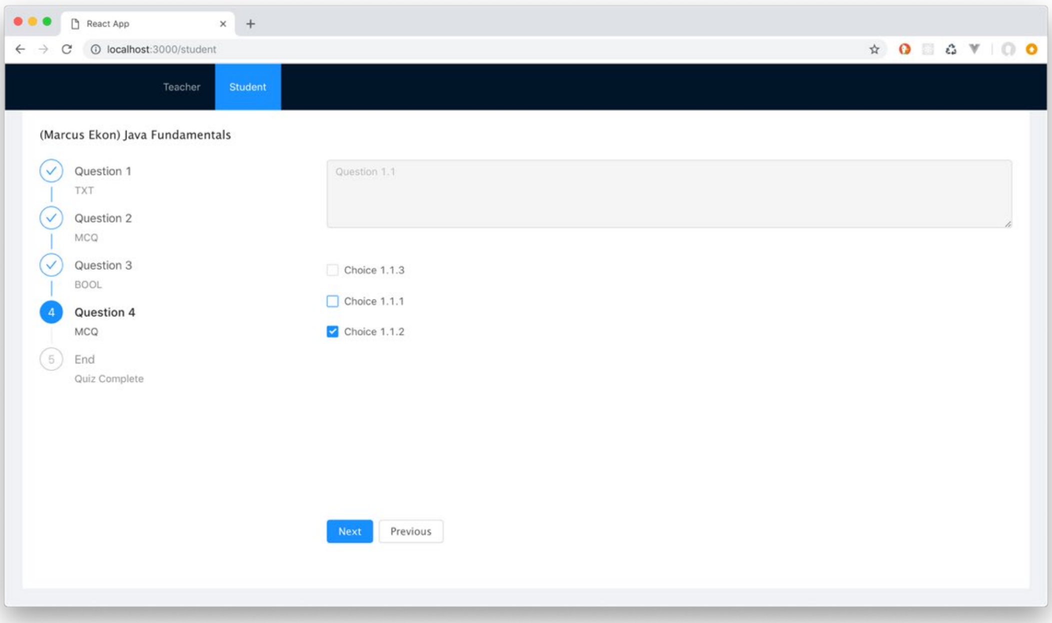
To run a quiz, click on “Student” in the navigation bar, enter your name, and select a quiz to run.



Select Quiz View

After selecting a quiz, students must then click on start and answer the questions.

The screenshot below shows a student in the middle of completing a quiz.



Run Quiz View

Configuration Instructions

To facilitate the development and setup of this application, the Git version control system was used, and the source code was subsequently uploaded to GitHub.

Below, you can find a detailed step by step guide to setting up and configuring the application (This is also available in the *README.md* of the repository).

JAVA Backend API

1. Clone repository *`git clone https://github.com/denyoe/quiz.git`*
2. Move into the quiz-rest-marcek directory: *`cd quiz-rest-marcek`*
3. Then: *`./mvnw spring-boot:run`* to start the server

API BASE URL:	*`http://127.0.0.1:8080`*
QUIZ:	*`http://127.0.0.1:8080/quiz`*
QUESTION:	*`http://localhost:8080/questions`*

ReactJS SPA

4. Open another terminal window and move into the client directory which contains the front-end SPA: *`cd client`*
5. Install the project dependencies with: *`yarn install`*
6. Launch the client SPA using: *`yarn start`*

The application is accessible @:

Teacher Views: *`http://localhost:3000/teacher/question/manage`*
Student Views: *`http://localhost:3000/student`*

Bibliography

Agile Data, 2018. *Relational Databases 101: Looking at the Whole Picture.* [Online]
Available at:
http://www.agiledata.org/essays/relationalDatabases.html#BeyondRelationalDatabases
[Accessed 2 July 2018].
baeldung, 2018. *The DAO Pattern in Java.* [Online]
Available at: https://www.baeldung.com/java-dao-pattern
[Accessed 20 November 2019].
Commission Nationale de l'Informatique et des Libertés, 2018. *Les principes clés de la protection des données personnelles | CNIL.* [Online]
Available at: https://www.cnil.fr/fr/comprendre-vos-obligations/les-principes-cles
[Accessed 2018].
Java Point, 2018. [Online]
Available at: https://www.javatpoint.com/facade-pattern
[Accessed 20 November 2018].
Luecke, D., 2018. [Online]
Available at: https://blog.feathersjs.com/design-patterns-for-modern-web-apis-1f046635215
[Accessed 20 November 2018].
Morwood, D., 2015. *Get Real: Why Identity Management is Too Important to Trust to a Generalist.* [Online]
Available at: https://blog.centrify.com/identity-access-management-solution/
[Accessed 23 Jun 2018].
Pandit, N., 2018. *What Is ReactJS and Why Should We Use It?.* [Online]
Available at: https://www.c-sharpcorner.com/article/what-and-why-reactjs/
[Accessed 19 November 2018].
Pankaj, 2017. *Spring Framework.* [Online]
Available at: https://www.journaldev.com/16922/spring-framework
[Accessed 15 November 2018].
Powell-Morse, A., 2017. *Software Design Patterns: A Guide.* [Online]
Available at: https://airbrake.io/blog/design-patterns/software-design-patterns-guide
[Accessed 15 Jun 2018].
Prompt, M., 2016. *Why Identity Access and Management is More Important than Ever to Information Security.* [Online]
Available at: https://www.informationsecuritybuzz.com/articles/why-identity-access-and-management-is-more-important-than-ever-to-information-security/
[Accessed 23 Jun 2018].
Stafford, R., 2018. *P of EAA: Service Layer.* [Online]
Available at: https://martinfowler.com/eaaCatalog/serviceLayer.html
[Accessed 20 November 2018].
Veness, C., 2017. *Movable Type Scripts.* [Online]
Available at: https://www.movable-type.co.uk/scripts/sha256.html
[Accessed 26 Jun 2018].