ÉCOLE D'INGÉNIEURS EN INFORMATIQUE

Advance Java Project : Quiz Manager

Technical Specification Document

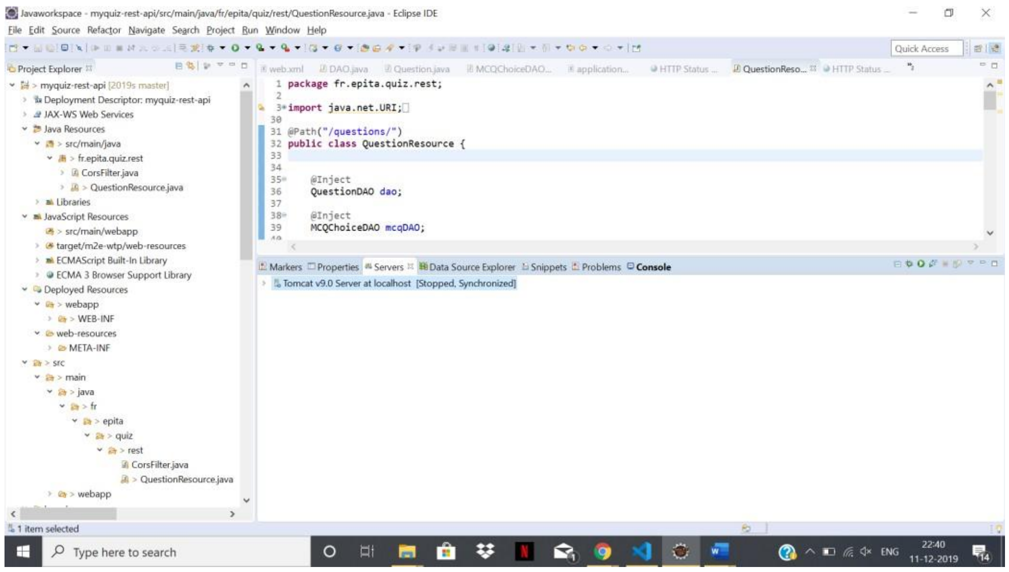
-Simpy Surbhi

Technical Description:

The purpose of this project is to create an application for the management of quiz preparation and execution using Angular 8 (API, web-based, oriented).

Introduction:

This project is a web-based application that helps perform CRUD operations on open questions and MCQ – preparation and execution using Angular 8.This document is prepared for the Advanced Java Project by Simpy Surbhi for her computer science master's program at L'École Pour l'Informatique et les Techniques Avancées (EPITA).


Javaworkspace - myquiz-rest-api/src/main/java/fr/epita/quiz/rest/QuestionResource.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
myquiz-rest-api [2019s master]
Deployment Descriptor: myquiz-rest-api
JAX-WS Web Services
Java Resources
src/main/java
fr.epita.quiz.rest
CorsFilter.java
QuestionResource.java
Libraries
JavaScript Resources
Deployed Resources
Markers Properties Servers Data Source Explorer Snippets Problems Console
Type here to search

```
package fr.epita.quiz.rest;

import java.net.URI;

@Path("/questions/")
public class QuestionResource {


    @Inject
    QuestionDAO dao;

    @Inject
    MCQChoiceDAO mcqDAO;
```

Tomcat v9.0 Server at localhost [Stopped, Synchronized]

Project Overview:

There are two modules in this project: the admin module and the student module. In the student module, the user can only attend the quiz, whereas in the admin module, the administrator can add questions, delete questions, update questions, search questions, search / display list of questions, attend a quiz, evaluate the quiz.

Project Scope:

- Automatically assemble quizzes using open questions, Boolean and multiple choices.
- Auto-grading for multiple-choice questions.
- Creation of data access objects using CRUD Methods.
- Create a configuration file for the application's properties.
- The API is made with an Angular 8.

Software Requirements:

Technologies Used:

BACKEND/WEBSERVER	Java 8, Spring-Hibernate, Apache Tomcat 9, JPA
FRONTEND	Angular 8
DATABASE	H2
VERSION CONTROL	GIT

Project Dependencies:

- Install Java 8
- Install H2 JDBC
- Install Spring-Hibernate
- Angular 8
- Create table in database

Database:

This project uses the H2 database And the values that are sent through the angular are stored in the database.

Acronyms and Abbreviations:

ACRONYMS	ABBREVIATIONS
CRUD	(Create, Read, Update, Delete) ❼ Relational database application functions.
DAO	(Data Access Object) ❼ Service class to communicate with the H2 database.
JPA	(Java Persistence API) ❼ is a Java application programming interface specification that describes the management of relational data in applications using Java Platform.
MCQ	(Multiple Choice Question) ❼ Questions with possible responses listed in our implementation using radial buttons.
JDBC	(Java Database Connectivity) ❼ Application Programming Interface(API) for Java Programming Language.

Project Requirements:

1. Create a quiz based on the requirements of the user.
2. Use questions stored in database (using h2 database).
3. Allows the user to take the quiz.
4. At the end of the assessment, correct open & MCQ questions and display results.
5. Export the quiz in a plain text format.
6. Use CRUD operations on a question.
7. Display the output using Angular 8.

System Specification:

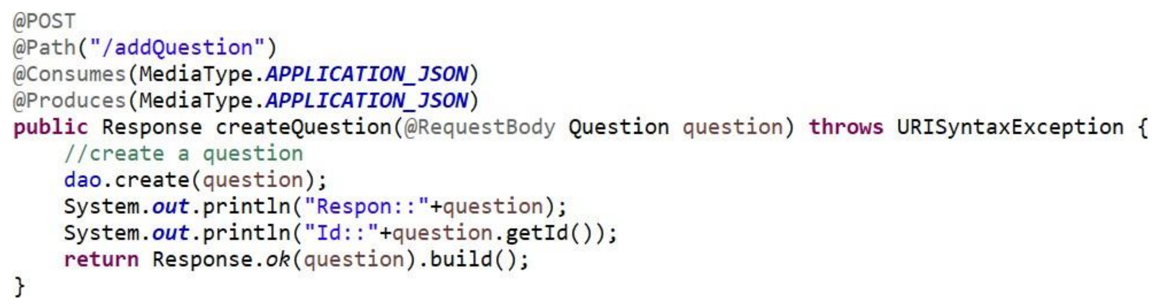
API's:

Generated APIs to connect from front end to backend are listed as follows,

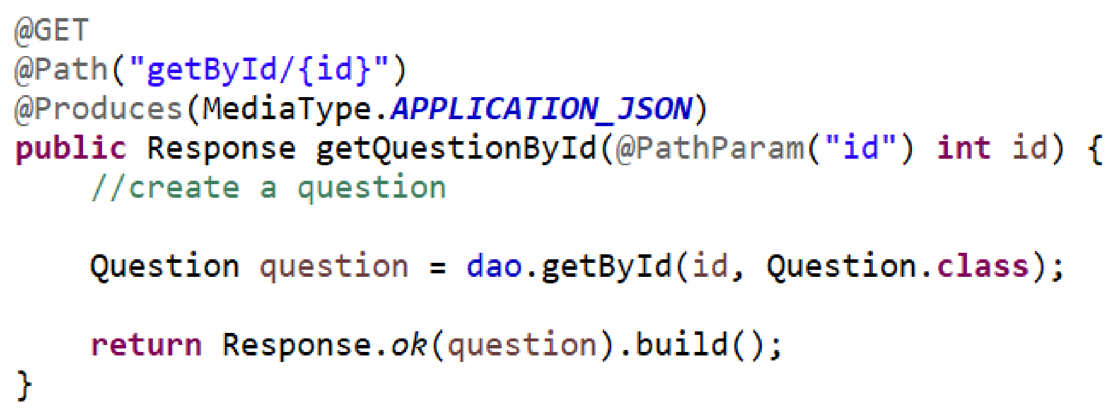
Add Exam ❼ http://localhost:4200/formEditAdd
Exam List ❼ http://localhost:4200/adminhome
Question List ❼ http://localhost:4200/adminhome
Update Exam List
❼ http://localhost:4200/formEditAdd?special=%7B%22id%22:48,%22title%22:%22DevOps%20Exam%22%7D
Add Question ❼ http://localhost:4200/formQuestion
Add Option ❼ http://localhost:4200/formQuestion

Functional Requirements:

- Can be able to Add Questions connecting to the database(h2), i.e., The questions already used and stored in the database can be used by users.



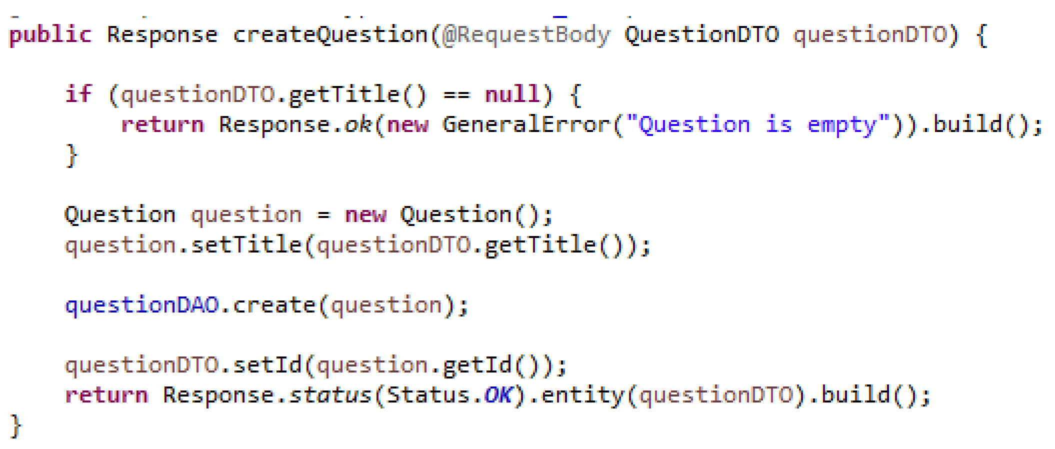
```
@POST
@Path("/addQuestion")
@Consumes(MediaType.APPLICATION_JSON)
@Produces(MediaType.APPLICATION_JSON)
public Response createQuestion(@RequestBody Question question) throws URISyntaxException {
    //create a question
    dao.create(question);
    System.out.println("Respon::"+question);
    System.out.println("Id::"+question.getId());
    return Response.ok(question).build();
}
```



```
@GET
@Path("getById/{id}")
@Produces(MediaType.APPLICATION_JSON)
public Response getQuestionById(@PathParam("id") int id) {
    //create a question

    Question question = dao.getById(id, Question.class);

    return Response.ok(question).build();
}
```

API For ADD QUESTION ❼

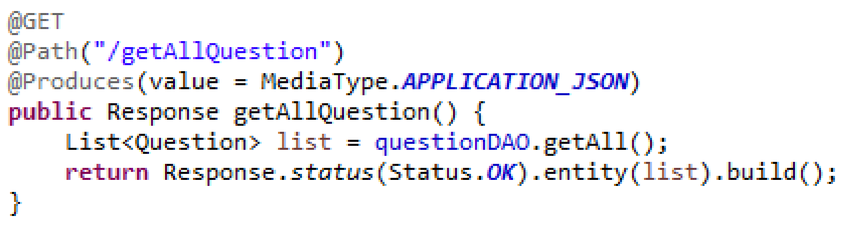
- Can be able to add questions connecting to the database(h2)



```
public Response createQuestion(@RequestBody QuestionDTO questionDTO) {

    if (questionDTO.getTitle() == null) {
        return Response.ok(new GeneralError("Question is empty")).build();
    }

    Question question = new Question();
    question.setTitle(questionDTO.getTitle());

    questionDAO.create(question);

    questionDTO.setId(question.getId());
    return Response.status(Status.OK).entity(questionDTO).build();
}
```

API For List of Question

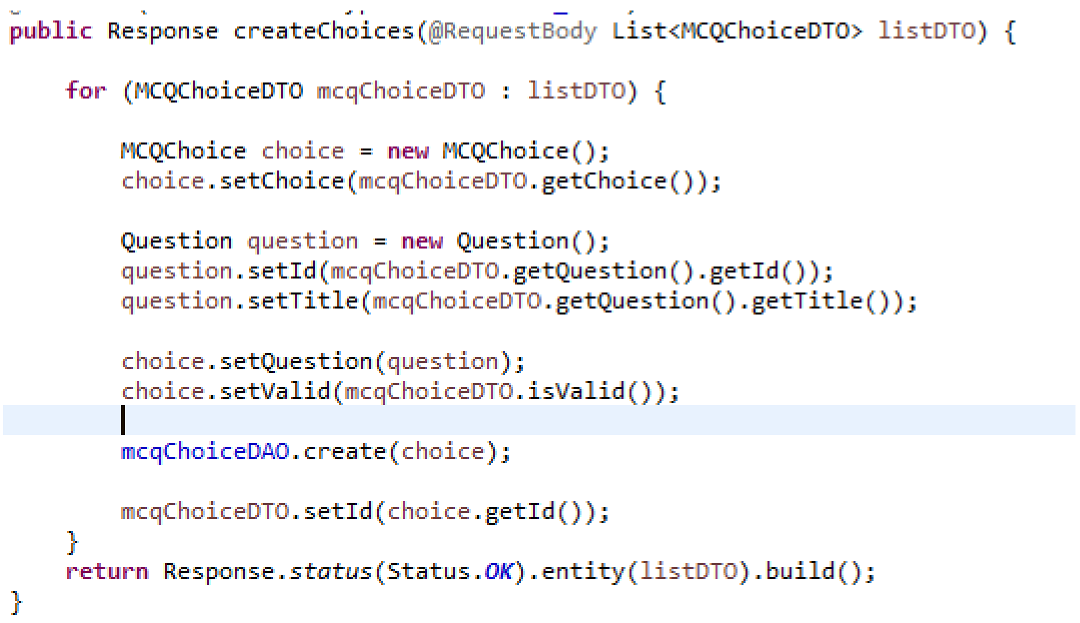
- Can be able to see list of question connecting to the database(h2)



```
@GET
@Path("/getAllQuestion")
@Produces(value = MediaType.APPLICATION_JSON)
public Response getAllQuestion() {
    List<Question> list = questionDAO.getAll();
    return Response.status(Status.OK).entity(list).build();
}
```

API For Create Option ❼

Can be able to create Option connecting to the database(h2)



```
public Response createChoices(@RequestBody List<MCQChoiceDTO> listDTO) {

    for (MCQChoiceDTO mcqChoiceDTO : listDTO) {

        MCQChoice choice = new MCQChoice();
        choice.setChoice(mcqChoiceDTO.getChoice());

        Question question = new Question();
        question.setId(mcqChoiceDTO.getQuestion().getId());
        question.setTitle(mcqChoiceDTO.getQuestion().getTitle());

        choice.setQuestion(question);
        choice.setValid(mcqChoiceDTO.isValid());

        mcqChoiceDAO.create(choice);

        mcqChoiceDTO.setId(choice.getId());
    }
    return Response.status(Status.OK).entity(listDTO).build();
}
```

API For Get Answer :

- Can be able to get answer connecting to the database(h2)

```
@Produces(value = MediaType.APPLICATION_JSON)
public Response getAnswer(@PathParam("id") long questionID) {
    List<MCQChoice> choices = mcqChoiceDAO.getByOtherColumnId(questionID, "question");
    Response response = Response.ok(choices).build();
    return response;
}
```

API For Create Exam❼

- Can be able to create an exam in the database(h2)

```
@POST
@Path("/createExam")
@Produces(value = MediaType.APPLICATION_JSON)
public Response createExam(@RequestBody ExamDTO examDTO) {

    if(examDTO == null || examDTO.getTitle() == null) {
        return Response.ok(new GeneralError("Exam can not create")).build();
    }

    Exam exam = new Exam();
    exam.setTitle(examDTO.getTitle());

    examDAO.create(exam);

    examDTO.setId(exam.getId());
    Response response = Response.ok(examDTO).build();
    return response;
}
```

-

API For LIST of Exam

- Can be able to see list of exam in the database(h2)

```
@GET
@Path("/getAllExam")
@Produces(value = MediaType.APPLICATION_JSON)
public Response getAnswer() {
    Response response = Response.ok(examDAO.getAll()).build();
    return response;
}
```

API For Delete Exam

- Can be able to delete exam in the database(h2)

```java
@POST
@Path("/deleteExam")
@Produces(value = MediaType.APPLICATION_JSON)
public Response deleteExam(@RequestBody ExamDTO examDTO) {
    if(examDTO == null) {
        return Response.ok(new GeneralError("Exam can not deleted")).build();
    }

    examDAO.delete(examDTO.getId());

    Response response = Response.ok(examDTO).build();
    return response;
}
```

API For update Exam

- Can be able to update an exam in the database(h2)

```java
@POST
@Path("/updateExam")
@Produces(value = MediaType.APPLICATION_JSON)
public Response updateExam(@RequestBody ExamDTO examDTO) {
    if(examDTO == null) {
        return Response.ok(new GeneralError("Exam can not update")).build();
    }

    Exam exam = new Exam();
    exam.setId(examDTO.getId());
    exam.setTitle(examDTO.getTitle());

    examDAO.update(exam);

    Response response = Response.ok(examDTO).build();
    return response;
}
```

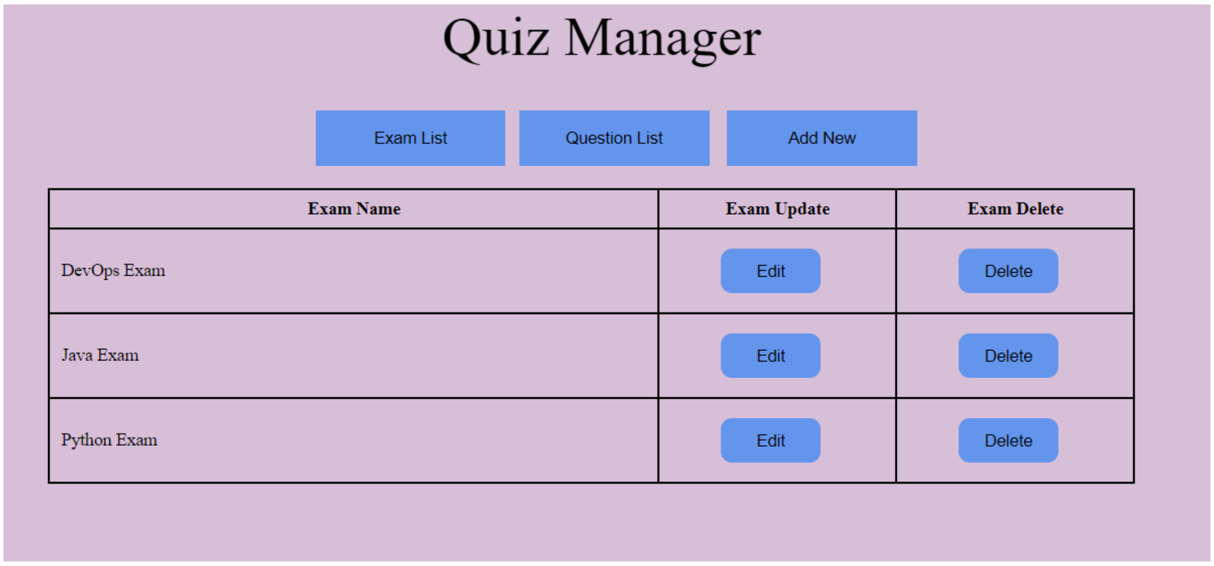
API For get Answer

- Can be able to see answer in the database(h2)

```java
@GET
@Path("/answer/{id}")
@Produces(value = MediaType.APPLICATION_JSON)
public Response getAnswer(@PathParam("id") long answerId) {
    Answer answer = answerDAO.getById(answerId);
    Response response = Response.ok(answer).build();
    return response;
}
```

Project Frontend :

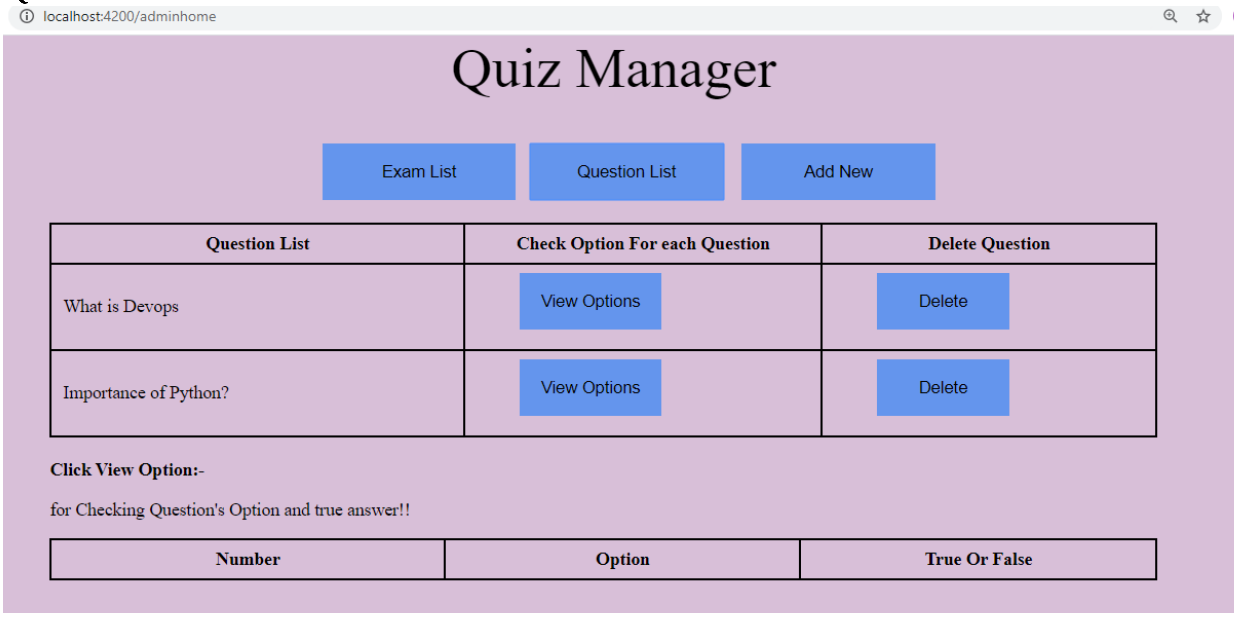
Exam List:



Quiz Manager

Exam Name	Exam Update	Exam Delete
DevOps Exam	Edit	Delete
Java Exam	Edit	Delete
Python Exam	Edit	Delete

Question List:



localhost:4200/adminhome

Quiz Manager

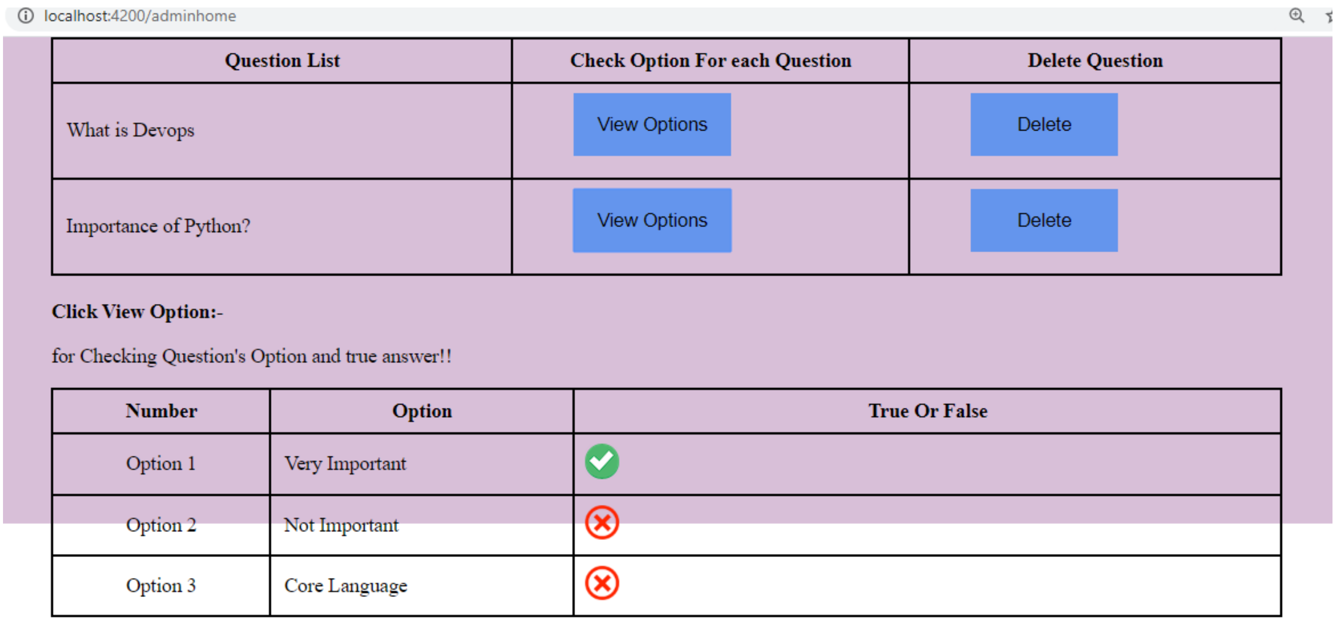
Question List	Check Option For each Question	Delete Question
What is Devops	View Options	Delete
Importance of Python?	View Options	Delete

Click View Option:-

for Checking Question's Option and true answer!!

Number	Option	True Or False

View Option List:


localhost:4200/adminhome

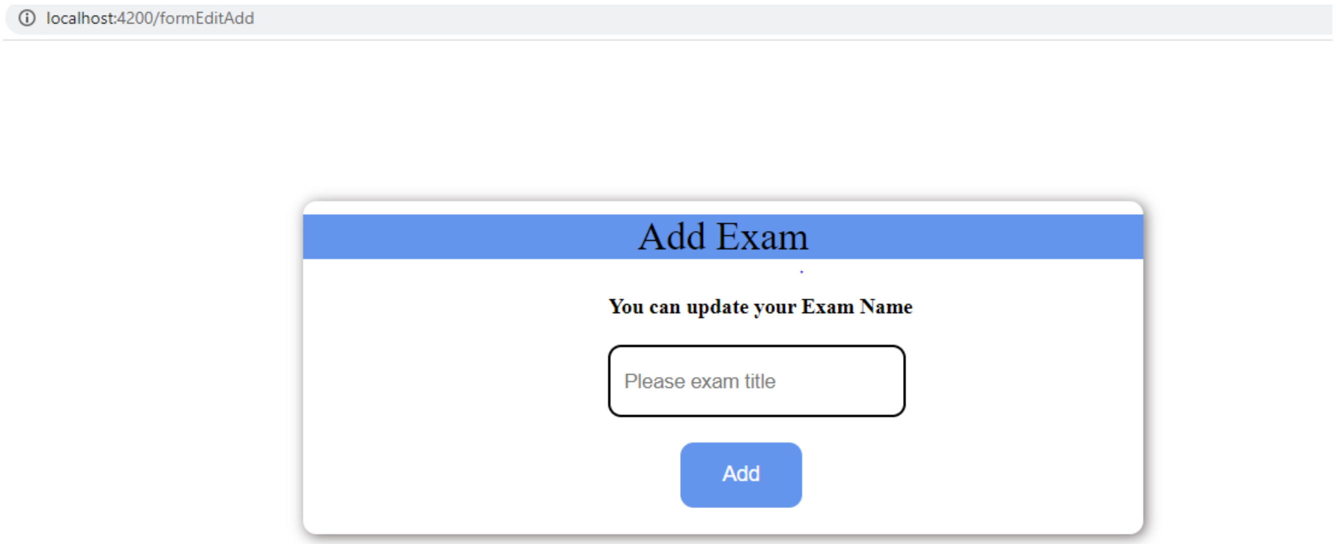
Question List	Check Option For each Question	Delete Question
What is Devops	View Options	Delete
Importance of Python?	View Options	Delete

Click View Option:-

for Checking Question's Option and true answer!!

Number	Option	True Or False
Option 1	Very Important	✓
Option 2	Not Important	✗
Option 3	Core Language	✗

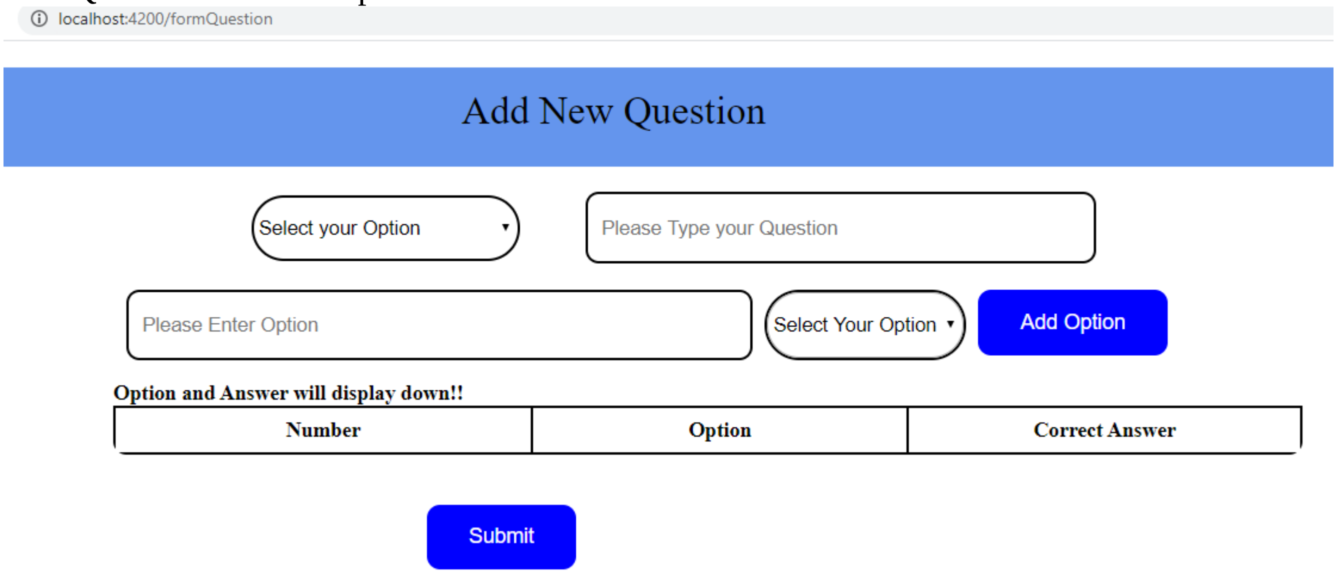
Add New Exam:


localhost:4200/formEditAdd

Add Exam

You can update your Exam Name

Add Question and Add Option


localhost:4200/formQuestion

Add New Question

Option and Answer will display down!!

Number	Option	Correct Answer

User Interface Design:

Refer to the user guide.

Hardware Interfaces

- ✓ Operating System : Windows xp or more, MAC or UNIX
- ✓ Processor : Pentium 3.0 GHz or higher
- ✓ RAM : 256 MB or more
- ✓ Hard Disk : 10 GB or more

Bibliography:

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