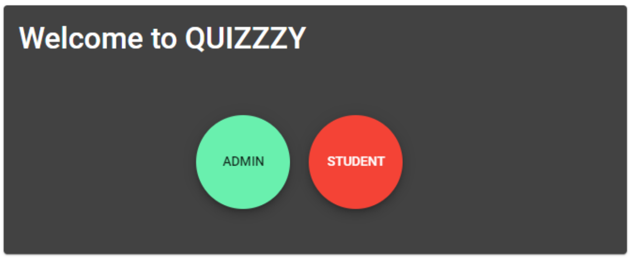
Advanced Java Report

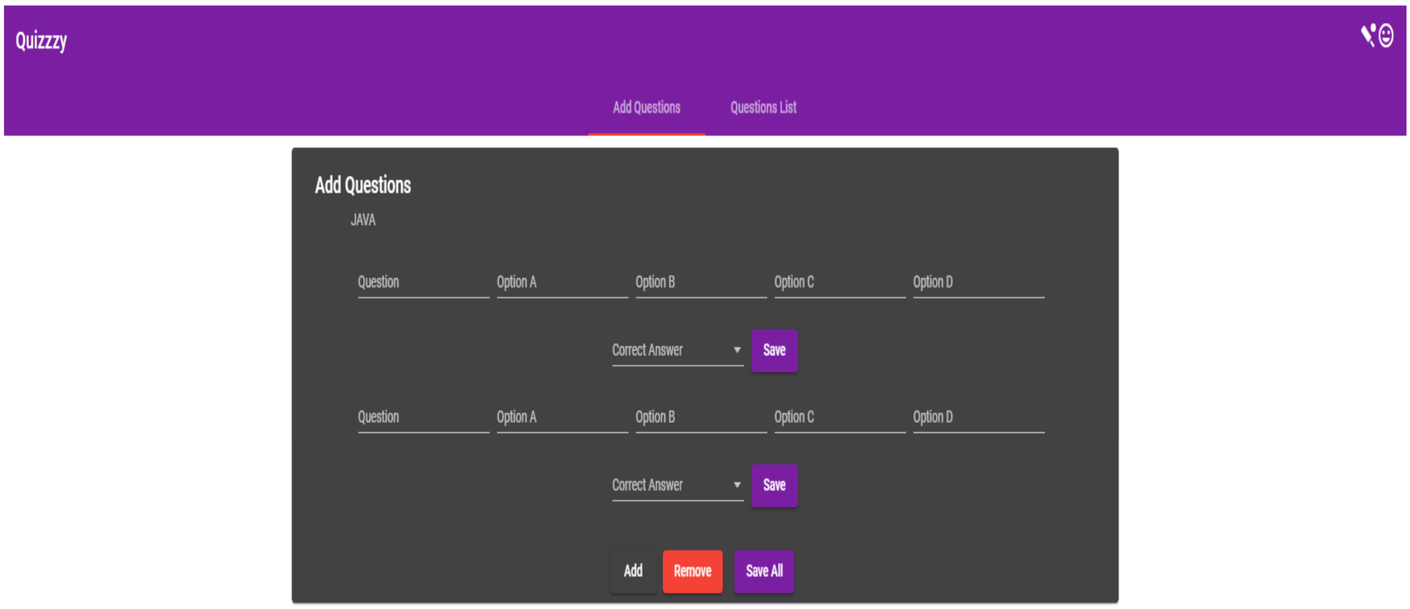
Submitted By: Shivank Mittal

Submitted To: Thomas Broussard Sir

Starting from Frontend, User is welcome by the login page where he have a choice to be a creator of exam seeker.

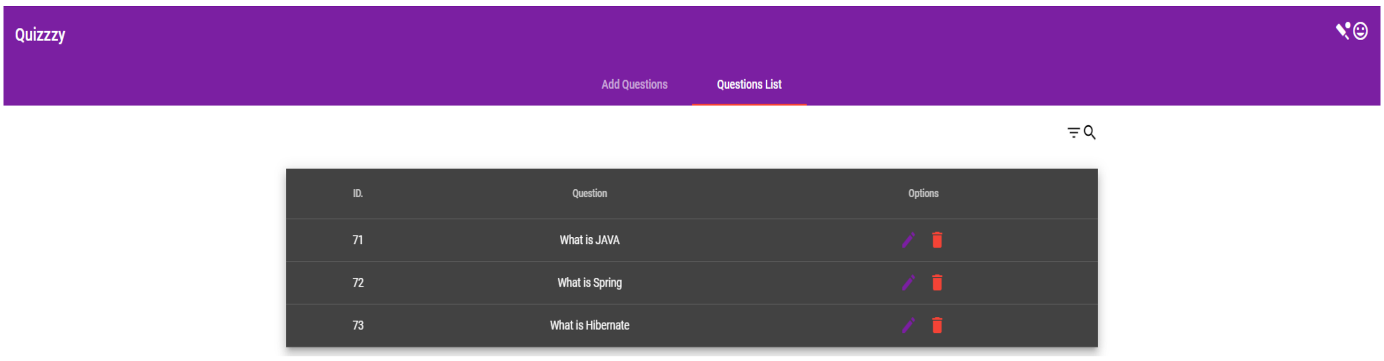


If a user chose to be admin, he will be redirected to another page.



Admin can save single question or multiple questions at same time. There is no limit to add questions, user just need to click add button and one more option for question will appear.

After creating the questions, user can check the questions just by shifting the tabs.



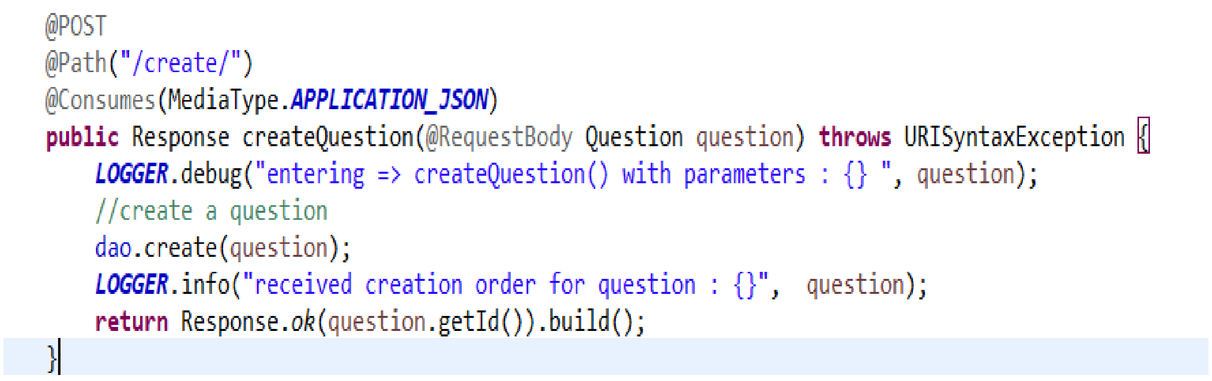
ID.	Question	Options
71	What is JAVA	
72	What is Spring	
73	What is Hibernate	

Here a user can edit, delete and filter the questions and the table get populated with refreshed data at real time. This way we are complete our all CRUD operations.

For the Backend, I have used JAVA , SPRING , JPA and H2 for database.

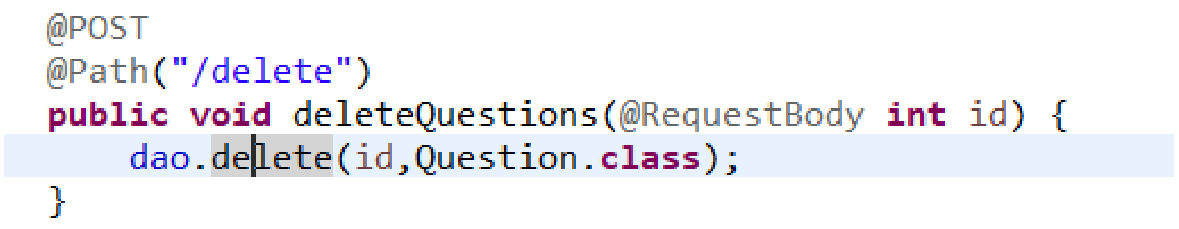
API, s used are:

Create Questing =>



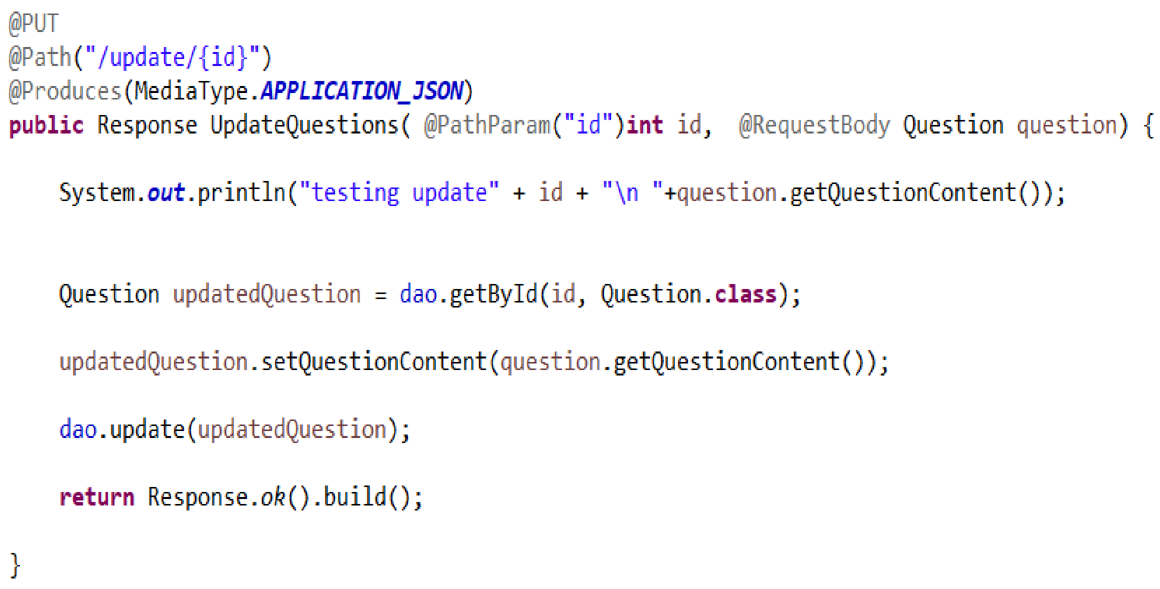
```
@POST
@Path("/create/")
@Consumes(MediaType.APPLICATION_JSON)
public Response createQuestion(@RequestBody Question question) throws URISyntaxException {
    LOGGER.debug("entering => createQuestion() with parameters : {} ", question);
    //create a question
    dao.create(question);
    LOGGER.info("received creation order for question : {}",  question);
    return Response.ok(question.getId()).build();
}
```

Delete Question =>



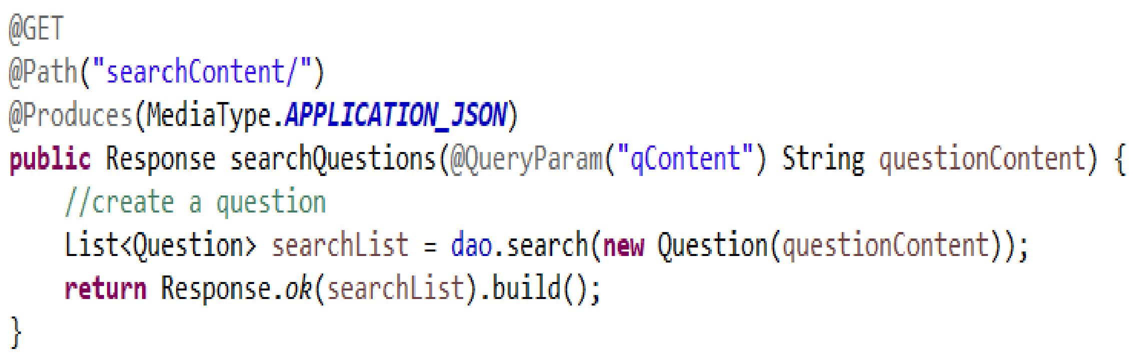
```
@POST
@Path("/delete")
public void deleteQuestions(@RequestBody int id) {
    dao.delete(id,Question.class);
}
```

Update Question =>



```
@PUT
@Path("/update/{id}")
@Produces(MediaType.APPLICATION_JSON)
public Response UpdateQuestions( @PathParam("id")int id,  @RequestBody Question question) {

    System.out.println("testing update" + id + "\n "+question.getQuestionContent());


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

    updatedQuestion.setQuestionContent(question.getQuestionContent());

    dao.update(updatedQuestion);

    return Response.ok().build();

}
```

Search By Content =>



```
@GET
@Path("searchContent/")
@Produces(MediaType.APPLICATION_JSON)
public Response searchQuestions(@QueryParam("qContent") String questionContent) {
    //create a question
    List<Question> searchList = dao.search(new Question(questionContent));
    return Response.ok(searchList).build();
}
```

Search by Id =>

```java
@GET
@Path("/{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();
}
```

Getting all questions =>

```java
@GET
@Path("getAllQuestions/")
@Produces(MediaType.APPLICATION_JSON)
public Response getAllQuestions() {
    //create a question
    List<Question> searchList = dao.getAll(new Question());
    return Response.ok(searchList).build();
}
```