TECHNICAL DOCUMENTATION

Quiz Manager

By

Sarupriya Tharanipathy

Introduction:

1.1 Project Overview:

Quiz is a console application that has question related to many technologies like JAVA, PYTHON and Networking. The quiz consist of MCQ questions and Open questions. There are two logins, one is admin login and the other one is student login. Student will be provided with the username and password to take the quiz. The scores of the student will be displayed to them at the end of the quiz.

1.2 Project Dependencies:

In order to run the program, the following steps are necessary:

1. Install h2 JDBC
2. Create the following tables in the database.
    - USER
    - Quiz
    - Question
    - Answer
    - MCQChoice
    - Score
    - Studentanswer
3. Java Eclipse

Queries for creating the above table:

1. CREATE TABLE ANSWER (ID INT AUTO_INCREMENT,
   QID INTEGER,
   TEXT VARCHAR (MAX));
2. CREATE TABLE MCQCHOICE (ID INT AUTO_INCREMENT,
   CHOICES VARCHAR (214748364),
   QID INT);
3. CREATE TABLE QUESTIONS (QID INT AUTO_INCREMENT,
   QUESTION VARCHAR (MAX),
   DIFFICULTY INT,
   QUIZ VARCHAR (100));
4. CREATE TABLE QUIZ
   (NAME VARCHAR (500),
   ID INT AUTO_INCREMENT);
5. CREATE TABLE SCORE (STUDENT VARCHAR (MAX),
   SCORE INTEGER);
6. CREATE TABLE STUDENTANSWER (ID INTEGER (10) AUTO_INCREMENT,

```
    ANSWER VARCHAR (MAX),
    QUESTION VARCHAR (MAX), STUDENT VARCHAR (MAX));
7. CREATE TABLE USER(USERNAME VARCHAR(100),
    PASSWORD VARCHAR (100),
    ISADMIN BOOLEAN ,
    ID INT AUTO_INCREMENT);
```

1.3 project scope:

1. Assemble the questions based on the topics.
2. Able to CRUD on open questions and multiple choice question(Stores in the database)
3. To be able to search questions based on topics
4. Export the quiz under a plain text format.
5. Run the evaluation and provide automatic mark at the end of the execution.

2. BUSINESS REQUIREMENTS:

Sl.no	**REQUIREMENT DESCRIPTION**
1	Create a quiz based on user's specifications(Topics)
2	Use H2 database to store the topics and questions based on difficulties
3	Allow user(student) to take quiz
4	Validate MCQ Questions and display the result at the end of the quiz
5	Export the Quiz under plain text format

6	Search the questions based on the topics
7	Use Create, update and delete operations for Quiz and the questions

3. SYSTEM SPECIFICATION:

Functional Requirements:

Sl No	Implementation

1	Admin and Student can log into their logins:
	
	public boolean[] validateLogin(String userName, String password) {
	
	boolean[] result = new boolean[3];
	
	String query = ConfigurationService.getInstance()
	.getConfigurationValue(ConfigEntry.DB_QUERIES_VALIDATE_LOGIN,"");
	
	try (Connection connection = getConnection();
	
	PreparedStatement pstmt = connection.prepareStatement(query)) {
	ResultSet rs = pstmt.executeQuery();
	
	while (rs.next()) {
	String user = rs.getString("USERNAME");
	String pw = rs.getString("PASSWORD");
	
	if(userName.equals(user) && password.equals(pw))
	{
	result[0] = true;
	if(result[0])
	{
	result[1] = rs.getBoolean("isAdmin");
	}
	break;
	}
	else
	continue;
	}
	rs.close();
	} catch (SQLException e) {
	
	//throw new SearchFailedException(quizCriterion);
	System.out.println(e.getStackTrace());
	}

2	Admin can create, update and delete the quiz.
	
	//Create the quiz in the Database
	public void create(Quiz quiz) throws CreateFailedException {
	try (Connection connection = getConnection();
	PreparedStatement pstmt =
	connection.prepareStatement(INSERTQUIZ_QUERY);) {
	pstmt.setString(1, quiz.getTitle());
	pstmt.execute();
	} catch (SQLException sqle) {
	System.out.println("Something unexpected happened !! Please restart the application !!");
	
	}
	}
	//Updates a quiz in the Database
	
	public void update(Quiz quiz, Scanner scan) throws DataAccessException {
	
	String newValue = "";
	
	while(newValue.equals("")) {
	newValue = "";
	System.out.println("Enter the quiz name to be updated !!");
	newValue = scan.nextLine();
	}
	
	try (Connection connection = getConnection();
	PreparedStatement pstmt =
	connection.prepareStatement(UPDATEQUIZ_QUERY);
	PreparedStatement pstmt1 =
	connection.prepareStatement(UPDATEQUES_QUERY);) {
	pstmt.setString(1, newValue);
	pstmt.setString(2, quiz.getTitle());
	pstmt1.setString(1, newValue);
	pstmt1.setString(2, quiz.getTitle());
	pstmt.execute();
	pstmt1.execute();
	System.out.println("Quiz updated Successfully !!");
	
	} catch (SQLException sqle) {
	System.out.println("Something unexpected happened !! Please restart the application !!");
	}
	
	}
	
	//Deletes a quiz in the Database
	public void delete(String topic) throws DataAccessException {

	try (Connection connection = getConnection();
	PreparedStatement pstmt =
	connection.prepareStatement(DELETEQUIZ_QUERY);
	PreparedStatement pstmt1 =
	connection.prepareStatement(DELETEQUES_QUERY);){
	pstmt.setString(1, topic);
	pstmt.execute();
	pstmt1.setString(1, topic);
	pstmt1.execute();
	System.out.println("The quiz and the related questions were deleted successfully !!");
	
	} catch (SQLException sqle) {
	System.out.println("Something unexpected happened !! Please restart the application !!");
	}
	}

3	Admin can Create, update and delete the questions as per the topic and difficulties.
	
	public int createQuestion(Question ques, String difficulty) throws CreateFailedException, SQLException {
	int idColVar = 0;
	try (Connection connection = getConnection();
	
	PreparedStatement pstmt1 =
	connection.prepareStatement(INSERTQUES_QUERY, new String[]{"QID"});) {
	pstmt1.setString(1, ques.getquestion());
	pstmt1.setString(2, ques.getTopics());
	pstmt1.setString(3, difficulty);
	pstmt1.executeUpdate();
	ResultSet rs = pstmt1.getGeneratedKeys();
	
	while (rs.next()) {
	idColVar = rs.getInt(1);
	ques.setID(idColVar);
	}
	
	} catch (SQLException sqle) {
	System.out.println("Something unexpected happened !! PLease restart the application !!");
	
	}
	return idColVar;
	}
	//Update Questions based on Difficulties
	public void update(Question ques, Scanner scan) {
	String newValue = "";
	String newAns = "";
	String[] choice = new String[3];

```
int id = getQuestionID(ques.getquestion());

while(newValue.equals("")) {
newValue = "";
System.out.println("Enter the updated question below!!");
newValue = scan.nextLine();
}

try (Connection connection = getConnection();
PreparedStatement pstmt =
connection.prepareStatement(UPDATEQUES_QUERY);
PreparedStatement pstmt1 =
connection.prepareStatement(UPDATEANSWER_QUERY);
PreparedStatement pstmt2 =
connection.prepareStatement(SELECT_MCQCHOICE);
PreparedStatement pstmt3 =
connection.prepareStatement(DELETEQUESCHOICES_QUERY);
PreparedStatement pstmt4 =
connection.prepareStatement(INSERT_MCQ_CHOICES);) {
pstmt.setString(1, newValue);
pstmt.setInt(2, id);
pstmt.execute();
System.out.println("Question updated Successfully !!");


while(newAns.equals("")) {
newAns = "";
System.out.println("Enter the answer for the updated question !!");
newAns = scan.nextLine();
}

pstmt1.setString(1, newAns);
pstmt1.setInt(2, id);
pstmt1.execute();
System.out.println("Answer updated Successfully !!");

pstmt2.setInt(1, id);
ResultSet rs = pstmt2.executeQuery();
pstmt3.setInt(1, id);
if(rs.next()) {
pstmt3.execute();
System.out.println("Enter Updated Choices");
System.out.println("Enter Choice 1");
choice[0] = scan.nextLine();
while(choice[0].equals("")) {
System.out.println("Enter Choice 1");
choice[0] = scan.nextLine();
}
System.out.println("Enter Choice 2");
choice[1] = scan.nextLine();
while(choice[1].equals("")) {
```

	System.out.println("Enter Choice 2");
	choice[1] = scan.nextLine();
	}
	
	System.out.println("Enter Choice 3");
	choice[2] = scan.nextLine();
	while(choice[2].equals("")) {
	System.out.println("Enter Choice 3");
	choice[2] = scan.nextLine();
	}
	
	List<String> list = Arrays.asList(choice);
	while(!list.contains(newAns)) {
	System.out.println("Please enter the given answer as one of the option");
	choice[2]= scan.nextLine();
	}
	
	for(int i=0; i<3; i++) {
	pstmt4.setString(1, choice[i]);
	pstmt4.setInt(2, id);
	pstmt4.execute();
	}
	}
	
	System.out.println("Quiz Updated Successfully !!");
	
	} catch (SQLException sqle) {
	System.out.println("Something unexpected happened !! PLease restart the application !!");
	}
	}
	//Delete Questions
	public void delete(String ques) {
	int id = getQuestionID(ques);
	try (Connection connection = getConnection();
	PreparedStatement pstmt =
	connection.prepareStatement(DELETEQUES_QUERY);
	PreparedStatement pstmt1 =
	connection.prepareStatement(DELETEQUESCHOICES_QUERY);){
	pstmt.setInt(1, id);
	pstmt.execute();
	pstmt1.setInt(1, id);
	pstmt1.execute();
	System.out.println("The question was deleted successfully !!");
	
	} catch (SQLException sqle) {
	System.out.println("Something unexpected happened !! PLease restart the application !!");
	}
	}

4	User can search questions from the chosen topics

	public ArrayList<String> getQuestion(String title, String diff) {
	ArrayList<String> value = new ArrayList<String>();
	
	try (Connection connection = getConnection();
	PreparedStatement pstmt = connection.prepareStatement(SELECT_QUES);) {
	pstmt.setString(1, title);
	pstmt.setString(2, diff);
	ResultSet rs = pstmt.executeQuery();
	
	while(rs.next()) {
	String ques = rs.getString("Question");
	value.add(ques);
	System.out.println(ques);
	System.out.println("");
	}
	
	} catch (SQLException sqle) {
	System.out.println("Something unexpected happened !! PLease restart the application !!");
	}
	return value;
	}

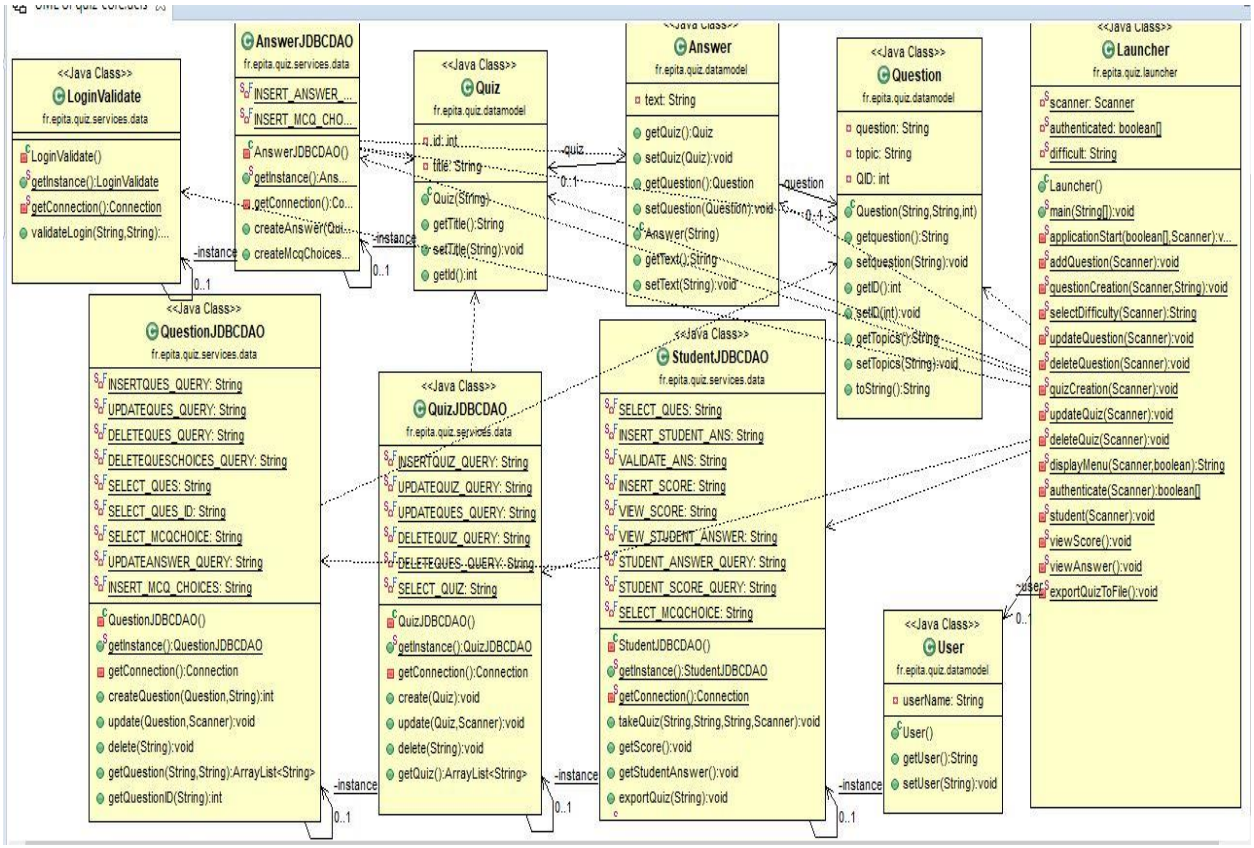
5	Marks are displayed to the students at the end of the Quiz
	
	public void getScore() {
	
	try (Connection connection = getConnection();
	PreparedStatement pstmt = connection.prepareStatement(VIEW_SCORE);) {
	ResultSet rs = pstmt.executeQuery();
	System.out.println("STUDENT" + " " +"Score");
	while(rs.next()) {
	String student = rs.getString("STUDENT");
	int score = rs.getInt("score");
	System.out.println(student +" " +score);
	}
	
	} catch (SQLException sqle) {
	System.out.println("Something unexpected happened !! Please restart the application !!");
	}
	
	}

7	Export Quiz to plain text Format
	
	public void exportQuiz(String userName) throws IOException {
	File file = initializeFile();
	
	try (Connection connection = getConnection();

```
FileWriter fw = new FileWriter(file);
PreparedStatement pstmt1 =
connection.prepareStatement(STUDENT_ANSWER_QUERY);
PreparedStatement pstmt2 =
connection.prepareStatement(STUDENT_SCORE_QUERY);) {
pstmt1.setString(1, userName );
pstmt2.setString(1, userName );
ResultSet rs = pstmt1.executeQuery();
ResultSet rs1 = pstmt2.executeQuery();
while (rs.next()) {

while (rs.next()) {

              fw.append("\n Question: " +rs.getString("Question"));
              fw.append('\n');
              fw.append("\n Answer Entered: " +rs.getString("Answer"));
              fw.append('\n');
             }
while(rs1.next()) {
fw.append("\n" +" The score is "+rs1.getInt("SCORE"));
}

          fw.flush();
          fw.close();
          System.out.println("Text File is created successfully.");
}
}
     catch (Exception e) {
          System.out.println("Something unexpected happened!! Please contact
administrator");
      }
}

private static File initializeFile() throws IOException {
File file = new File("data.txt");
if (!file.exists()) {
File parentFile = file.getAbsoluteFile().getParentFile();
parentFile.mkdirs();
file.createNewFile();
}
return file;
}
```

4. APPENDIX

Uml Class Diagram:


LoginValidate
fr.epita.quiz.services.data
LoginValidate()
getInstance():LoginValidate
getConnection():Connection
validateLogin(String,String):...
AnswerJDBCDAO
fr.epita.quiz.services.data
INSERT_ANSWER_...
INSERT_MCQ_CHO...
AnswerJDBCDAO()
getInstance():Ans...
getConnection():Co...
createAnswer(Qui...
createMcqChoices...
Quiz
fr.epita.quiz.datamodel
id: int
title: String
Quiz(String)
getTitle():String
setTitle(String):void
getId():int
Answer
fr.epita.quiz.datamodel
text: String
getQuiz():Quiz
setQuiz(Quiz):void
getQuestion():Question
setQuestion(Question):void
Answer(String)
getText():String
setText(String):void
Question
fr.epita.quiz.datamodel
question: String
topic: String
QID: int
Question(String,String,int)
getquestion():String
setquestion(String):void
getID():int
setID(int):void
getTopics():String
setTopics(String):void
toString():String
Launcher
fr.epita.quiz.launcher
scanner: Scanner
authenticated: boolean[]
difficult: String
Launcher()
main(String[]):void
applicationStart(boolean[],Scanner):v...
addQuestion(Scanner):void
questionCreation(Scanner,String):void
selectDifficulty(Scanner):String
updateQuestion(Scanner):void
deleteQuestion(Scanner):void
quizCreation(Scanner):void
updateQuiz(Scanner):void
deleteQuiz(Scanner):void
displayMenu(Scanner,boolean):String
authenticate(Scanner):boolean[]
student(Scanner):void
viewScore():void
viewAnswer():void
exportQuizToFile():void
QuestionJDBCDAO
fr.epita.quiz.services.data
INSERTQUES_QUERY: String
UPDATEQUES_QUERY: String
DELETEQUES_QUERY: String
DELETEQUESCHOICES_QUERY: String
SELECT_QUES: String
SELECT_QUES_ID: String
SELECT_MCQCHOICE: String
UPDATEANSWER_QUERY: String
INSERT_MCQ_CHOICES: String
QuestionJDBCDAO()
getInstance():QuestionJDBCDAO
getConnection():Connection
createQuestion(Question,String):int
update(Question,Scanner):void
delete(String):void
getQuestion(String,String):ArrayList<String>
getQuestionID(String):int
QuizJDBCDAO
fr.epita.quiz.services.data
INSERTQUIZ_QUERY: String
UPDATEQUIZ_QUERY: String
UPDATEQUES_QUERY: String
DELETEQUIZ_QUERY: String
DELETEQUES_QUERY: String
SELECT_QUIZ: String
QuizJDBCDAO()
getInstance():QuizJDBCDAO
getConnection():Connection
create(Quiz):void
update(Quiz,Scanner):void
delete(String):void
getQuiz():ArrayList<String>
StudentJDBCDAO
fr.epita.quiz.services.data
SELECT_QUES: String
INSERT_STUDENT_ANS: String
VALIDATE_ANS: String
INSERT_SCORE: String
VIEW_SCORE: String
VIEW_STUDENT_ANSWER: String
STUDENT_ANSWER_QUERY: String
STUDENT_SCORE_QUERY: String
SELECT_MCQCHOICE: String
StudentJDBCDAO()
getInstance():StudentJDBCDAO
getConnection():Connection
takeQuiz(String,String,String,Scanner):void
getScore():void
getStudentAnswer():void
exportQuiz(String):void
User
fr.epita.quiz.datamodel
userName: String
User()
getUser():String
setUser(String):void
-instance
0..1
quiz
question
user