import java.awt.\*;

import java.awt.event.\*;

import java.sql.\*;

import java.util.\*;

import java.util.List;

import javax.swing.\*;

import java.io.\*;

public class question{

String content;

String marks;

String type;

int id;

}

public class multiAnswer{

String answer1;

String answer2;

String answer3;

int id;

}

public class PhysicsQuiz extends Frame implements WindowListener, ActionListener{

private Label errorlabel;

public Connection getConnection() throws SQLException{ //This method connects the program to the database

String user = "root";

String pass = "usbw";

Connection c = null;

c = DriverManager.getConnection("jdbc:mysql://localhost:3307/pastpapers", user, pass);

System.out.println("Connected to database");

return c;

}

private Random rand = new Random();

private Label menuheader;

private Choice topiclist;

private Button test;

private Button scores;

private Button quit;

private Button submit;

private Label questionheader;

private Choice multiquestion;

private JTextField calculation;

private JTextField equation;

private JTextField name;

private Label incorrectQuestion;

private Button finish;

private boolean answered = false;

String topic;

List<Integer> primaryKeys = new LinkedList<Integer>();

int questionNo = 0;

int score = 0;

int random;

int multirandom;

String username;

question[] questions = new question[10];

String[] answers = new String[10];

String[] multianswers = new String[10];

String[] calcanswers = new String[10];

String[] eqanswers = new String[10];

multiAnswer[] answerOptions = new multiAnswer[10];

List<String> incorrectAnswers = new LinkedList<String>();

public PhysicsQuiz(){

mainMenu();

}

public void mainMenu(){ //What the user will see first, where they choose which topic to test on

this.removeAll();

repaint();

setLayout(new GridLayout(10,1));

menuheader = new Label("Choose the topic you wish to revise.");

add(menuheader);

topiclist = new Choice();

topiclist.add("Mechanics");

topiclist.add("Materials");

topiclist.add("Waves");

topiclist.add("DC Electricity");

topiclist.add("Nature of Light");

add(topiclist);

test = new Button("Test!");

add(test);

test.addActionListener(this);

scores = new Button("View previous scores on this topic");

add(scores);

scores.addActionListener(this);

quit = new Button("Quit");

add(quit);

quit.addActionListener(this);

addWindowListener(this);

setTitle("Physics Quiz");

setSize(1600, 600);

setVisible(true);

repaint();

}

public void getTopicQuestions() throws SQLException{ //This gets the questionIDs from the database and stores them in an array. A list of primary keys is used to randomly select 10 questions.

try{

Connection c = getConnection();

Statement st = c.createStatement();

String sqlcontent = "SELECT QuestionID FROM question WHERE QuestionTopic = '" + topic + "'";

ResultSet rs0 = st.executeQuery(sqlcontent);

while(rs0.next()){

System.out.println(rs0.getInt(1));

primaryKeys.add(rs0.getInt(1));

}

c.close();

}catch(SQLException ei){

errorlabel = new Label(ei.toString());

add(errorlabel);

setVisible(true);

}

for(int i = 0;i<questions.length;i++){

try{

random = rand.nextInt(primaryKeys.size());

System.out.println(primaryKeys.get(i));

questions[i] = new question();

questions[i].id = primaryKeys.get(random);

primaryKeys.remove(random);

}catch(NullPointerException er){System.out.println(er + " " + i);}

}

}

public void getTopicAnswers() throws SQLException{ //This gets the answers for each question and stores them in an array.

for(int f=0;f<questions.length;f++){

try{

Connection c = getConnection();

Statement st = c.createStatement();

String sqlcontent = "SELECT AnswerContent FROM answer WHERE AnswerID = '" + questions[f].id + "'";

ResultSet rsA = st.executeQuery(sqlcontent);

try{

while (rsA.next()){

answers[f] = rsA.getString(1);

}

} finally {

rsA.close();

}

c.close();

}catch(SQLException ex){

System.out.println(ex);

}

}

}

public void displayNextQuestion(){ //This gets the next question from the array, then displays it appropriately, depending on whether it is a multiple choice or calculation question.

for(int y=0;y<answerOptions.length;y++){

answerOptions[y] = new multiAnswer();

}

if(questions[questionNo].type.equals("Multiple Choice")){

for(int f=0;f<questions.length;f++){

for(int o=1;o<4;o++){

try{

Connection c = getConnection();

Statement st = c.createStatement();

String sqlcontent2 = "SELECT AnswerOption" + o + " FROM answer WHERE AnswerID = '" +questions[f].id + "'";

ResultSet rsB = st.executeQuery(sqlcontent2);

try{

while(rsB.next()){

if(o == 1){

answerOptions[f].answer1 = rsB.getString(1);

}

else if(o == 2){

answerOptions[f].answer2 = rsB.getString(1);

}

else if(o == 3){

answerOptions[f].answer3 = rsB.getString(1);

}

}

} finally {

rsB.close();

}

c.close();

}catch(SQLException ex){

System.out.println(ex);

}

}

}

questionheader = new Label("Question " + questionNo + " : " + questions[questionNo].content);

add(questionheader);

multiquestion = new Choice();

add(multiquestion);

multirandom = rand.nextInt(3);

if(multirandom == 0){

multiquestion.add(answers[questionNo]);

multiquestion.add(answerOptions[questionNo].answer1);

multiquestion.add(answerOptions[questionNo].answer2);

multiquestion.add(answerOptions[questionNo].answer3);

}

else if(multirandom == 1){

multiquestion.add(answerOptions[questionNo].answer3);

multiquestion.add(answers[questionNo]);

multiquestion.add(answerOptions[questionNo].answer1);

multiquestion.add(answerOptions[questionNo].answer2);

}

else if(multirandom == 2){

multiquestion.add(answerOptions[questionNo].answer1);

multiquestion.add(answerOptions[questionNo].answer2);

multiquestion.add(answers[questionNo]);

multiquestion.add(answerOptions[questionNo].answer3);

}

else if(multirandom == 3){

multiquestion.add(answerOptions[questionNo].answer1);

multiquestion.add(answerOptions[questionNo].answer2);

multiquestion.add(answerOptions[questionNo].answer3);

multiquestion.add(answers[questionNo]);

}

submit = new Button("Submit");

add(submit);

submit.addActionListener(this);

setVisible(true);

repaint();

}

else if(questions[questionNo].type.equals("Calculation")){

questionheader = new Label("Question " + questionNo + " : " + questions[questionNo].content);

add(questionheader);

calculation = new JTextField("Enter your final answer. (The number you get)");

add(calculation);

equation = new JTextField("Enter the final equation you used. (All values except constants should be full words, with capital letters and separated by a space e.g. Force = Mass x Acceleration. Constants using standard form write as \_x10^\_)");

add(equation);

//add JTextField for Units.

submit = new Button("Submit");

add(submit);

submit.addActionListener(this);

setVisible(true);

repaint();

}

else{

System.out.println("Error. Cannot get results from query");

System.out.println(questions[questionNo].content);

}

}

public void Answers(){ //This reads the user's answer and checks which they got right and counts their score. Incorrect questions are stored in an array which will be used to feedback to them.

if(questions[questionNo].type.equals("Multiple Choice")){

multianswers[questionNo] = multiquestion.getSelectedItem();

if(multianswers[questionNo].equals(answers[questionNo])){

score++;

}

else {

incorrectAnswers.add(questions[questionNo].content);

}

}

else if(questions[questionNo].type.equals("Calculation")){

calcanswers[questionNo] = calculation.getText();

eqanswers[questionNo] = equation.getText();

if(calcanswers[questionNo].equals(answers[questionNo]) && eqanswers[questionNo].equals(answers[questionNo])){

score++;

score++;

}

else if(eqanswers[questionNo].equals(answers[questionNo]) && !calcanswers[questionNo].equals(answers[questionNo])){

score++;

}

else if(calcanswers[questionNo].equals(answers[questionNo]) && !eqanswers[questionNo].equals(answers[questionNo])){

score++;

}

else{

incorrectAnswers.add(questions[questionNo].content);

}

}

else{

System.out.println("Error. Cannot get results from query");

System.out.println(questions[questionNo].content);

}

}

public void Results(){ //This displays the results of the test. Each question that was answered incorrectly is displayed. The user enters their name to be stored in the scores file.

questionheader = new Label("Your final score is: " + score + ". Here are the questions that you got wrong:");

add(questionheader);

setVisible(true);

for(int d=0;d < incorrectAnswers.size();d++){

incorrectQuestion = new Label(incorrectAnswers.get(d));

add(incorrectQuestion);

setVisible(true);

repaint();

}

finish = new Button("Back to menu");

add(finish);

finish.addActionListener(this);

name = new JTextField("Enter your name");

add(name);

setVisible(true);

repaint();

}

public void Test() throws SQLException{ //This is where the question content is fetched from the database and stored in an array.

remove(topiclist);

remove(test);

remove(scores);

remove(quit);

remove(menuheader);

for(int j=0;j<questions.length;j++){

try{

Connection c = getConnection();

Statement st = c.createStatement();

String sqlcontent = "SELECT QuestionContent FROM question WHERE QuestionID = " + questions[j].id;

ResultSet rs1 = st.executeQuery(sqlcontent);

try{

while (rs1.next()){

questions[j].content = rs1.getString(1);

}

} finally {

rs1.close();

}

String sqltype = "SELECT QuestionType FROM question WHERE QuestionID = " + questions[j].id;

ResultSet rs2 = st.executeQuery(sqltype);

try{

while (rs2.next()){

questions[j].type = rs2.getString(1);

}

} finally {

rs2.close();

}

st.close();

c.close();

}catch(SQLException ex){

System.out.println(ex);

}

}

displayNextQuestion();

}

public static void main(String[] args){

PhysicsQuiz app = new PhysicsQuiz();

}

public void actionPerformed(ActionEvent e){

String action = e.getActionCommand();

if(action.equals("Test!")){ //When the test button is clicked, gets the topic and then runs the methods to get the questions and answers and run the test.

questionNo = 0;

topic = topiclist.getSelectedItem();

try{

getTopicQuestions();

getTopicAnswers();

Test();

}

catch(SQLException ex){

System.out.println(ex);

}

}

if(action.equals("View previous scores on this topic")){ //Just a link to tell the user where they can find their scores.

this.removeAll();

questionheader = new Label("To find your previous scores, open the text file located in this folder with the title '\*topic\* scores'.");

add(questionheader);

finish = new Button("Back");

add(finish);

finish.addActionListener(this);

setVisible(true);

repaint();

}

if(action.equals("Submit")){ //Runs the method that shows the answers. Also displays all the incorrect questions.

Answers();

this.removeAll();

answered = true;

questionNo++;

if(questionNo < questions.length){

displayNextQuestion();

}

else{

Results();

}

}

if(action.equals("Quit")){

System.exit(0);

}

if(action.equals("Back")){

mainMenu();

}

if(action.equals("Back to menu")){ //After the feedback, when back is clicked this will store the name of the user and their score in the appropriate text file then return to the menu.

username = name.getText();

if(username.equals("")){

username = "Student";

}

else{

username = username;

}

try{

if(topic == "Mechanics"){

BufferedWriter out = new BufferedWriter(new FileWriter("Mechanics Scores.txt", true));

out.newLine();

out.write(username + " got " + score + ".");

out.close();

}

else if(topic == "DC Electricity"){

BufferedWriter out = new BufferedWriter(new FileWriter("DC Electricity Scores.txt", true));

out.write(username + " got " + score + ".");

out.close();

}

else if(topic == "Materials"){

BufferedWriter out = new BufferedWriter(new FileWriter("Materials Scores.txt", true));

out.write(username + " got " + score + ".");

out.close();

}

else if(topic == "Waves"){

BufferedWriter out = new BufferedWriter(new FileWriter("Waves Scores.txt", true));

out.write(username + " got " + score + ".");

out.close();

}

else if(topic == "Nature of Light"){

BufferedWriter out = new BufferedWriter(new FileWriter("Nature of Light Scores.txt", true));

out.write(username + " got " + score + ".");

out.close();

}

} catch (IOException ioe){System.out.println(ioe);}

mainMenu();

}

}

public void windowClosing(WindowEvent e) {

System.exit(0);

}

public void windowOpened(WindowEvent e) { }

public void windowClosed(WindowEvent e) { }

public void windowIconified(WindowEvent e) { }

public void windowDeiconified(WindowEvent e) { }

public void windowActivated(WindowEvent e) { }

public void windowDeactivated(WindowEvent e) { }

}