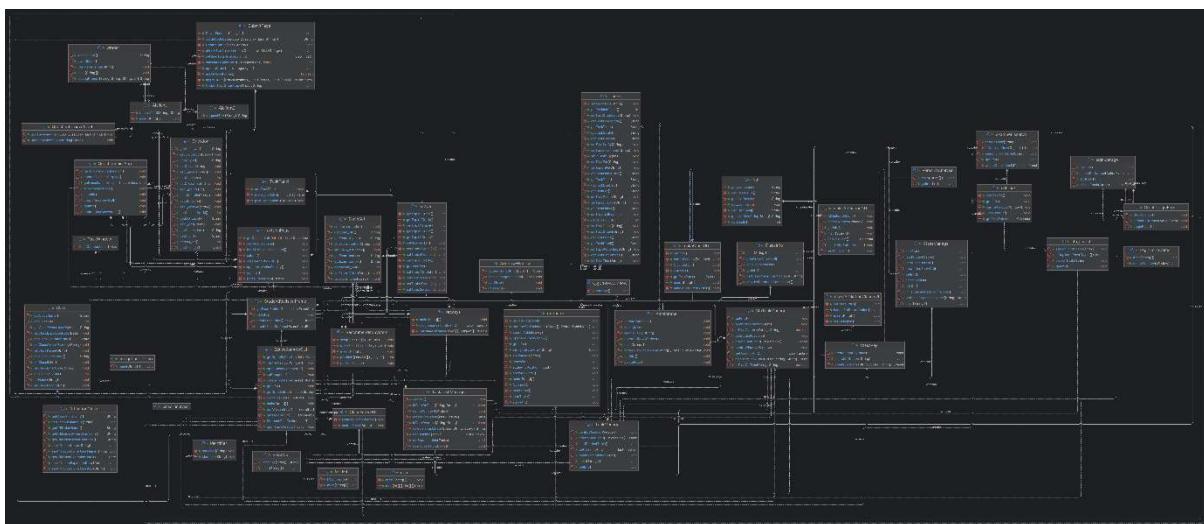


Criterion C: Development

UML diagram



Full image: <https://imgur.com/ng2gl50>



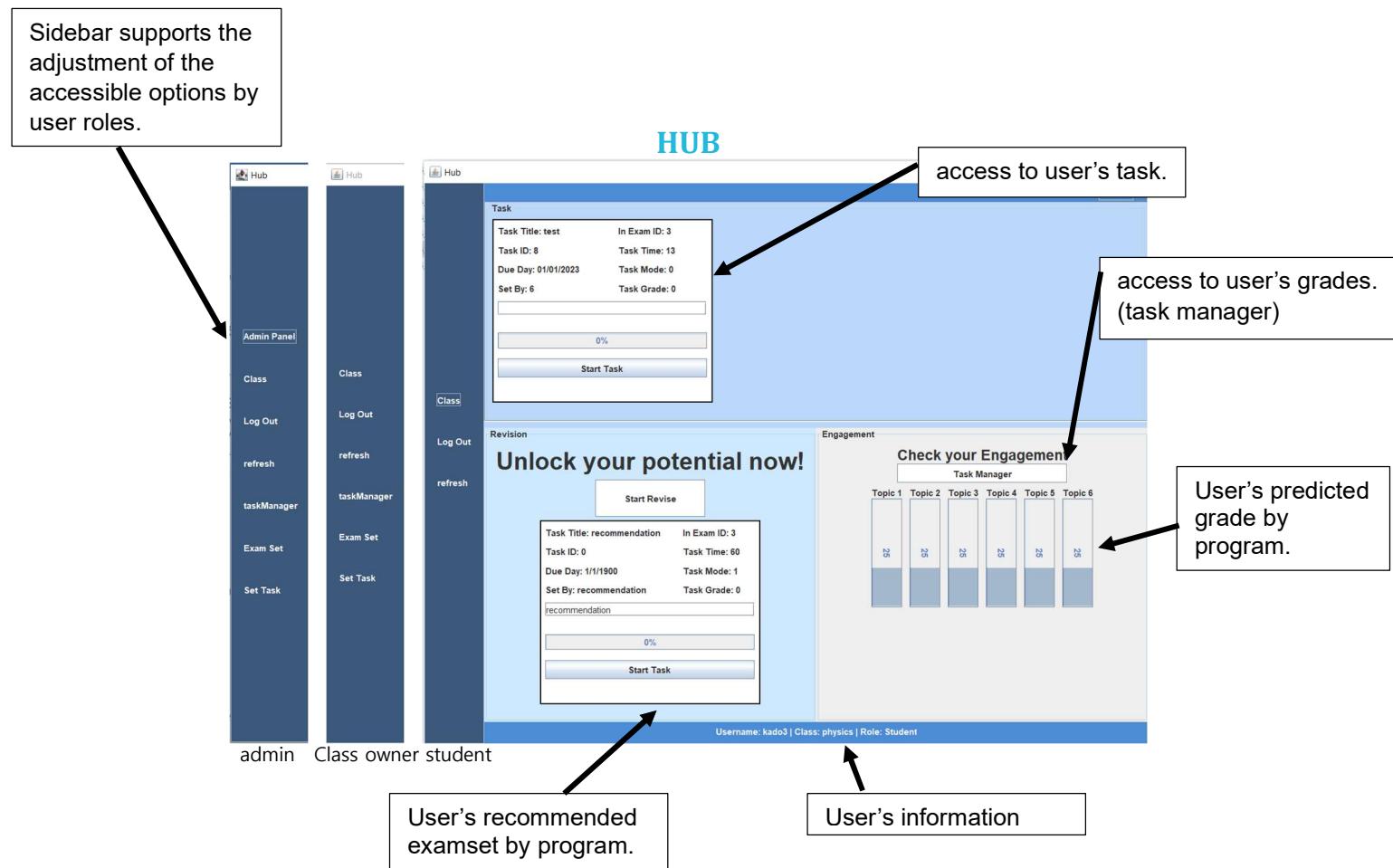
Full image: <https://imgur.com/pAE4Dl>

Contents

Graphical User Interface (GUI).....	4
HUB.....	4
Task manager.....	4
Task app.....	5
Revision Page (student), Review Page (Class Owner)	5
External Libraries.....	7
Web API and auto grading.....	8
Encapsulation.....	10
Inheritance.....	11
Complex Methods to customise JTable.....	12
Security (Hashing password)	14
Web database server.....	15
2D Array.....	17
KNN algorithm for recommend system.....	18
Hash table.....	20
Recursion.....	21
Error handling.....	21
Lambda expression.....	22

Graphical User Interface (GUI)

Users get access to their work, grades, and participation metrics and may engage with this GUI. data are dynamically imported and shown via the GUI. This also supports the adjustment of the accessible options by user roles such as students, instructors, administrators, etc. The program provides a distinct separation between user interface components and fundamental functionality by using the Model-View-Controller (MVC) design pattern.



Task manager

The screenshot shows a Windows application window titled "TaskManager". At the top, it displays the student information: "Student name: admin | Class: IGCSE physics revision club | Email: admin@admin.com". Below this is a table titled "Tasks" with columns: Task ID, Task Title, Set By, Task Time, In Exam ID, Due Day, result %, State, review, and Result Bar. There are six rows of task data. A callout box on the left points to the table area with the text: "filled with task information imported from the Online MySQL database." Another callout box on the right points to the "review" column with the text: "Button to review all questions."

Task ID	Task Title	Set By	Task Time	In Exam ID	Due Day	result %	State	review	Result Bar
1	Entrance test	1	20	1	02/08/2023	5	Finish overdue	Review	
3	as	1	10	1	01/01/2023	0	Overdue	Review	
6	topic1 test	1	15	3	01/01/2023	25	Finish overdue	Review	

The screenshot shows a Windows application window titled "Task app". At the top, it displays the student information: "Username: admin | Class: IGCSE physics revision club | Role: Admin". Below this is a question box with the text: "Q 2 :A student investigates a pendulum. He measures the time for the pendulum to complete 20 oscillations. He repeats the experiment three more times. The readings are shown. experiment time for 20 oscillations / s 1 17.6 2 19.8 3 17.6 4 18.6 What is the average period of the pendulum? A 0.88 s B 0.92 s C 17.6 s D 18.4 s". A callout box on the right points to this text with the text: "Show question (text)".

Below the question is a text area with the following steps:
1 A student investigates a pendulum.
 He measures the time for the pendulum to complete 20 oscillations.
 He repeats the experiment three more times.
 The readings are shown.

A callout box on the right points to this area with the text: "Show question (image)".

Below the text is a table:

experiment	time for 20 oscillations / s
1	17.6
2	19.8
3	17.6
4	18.6

Below the table is the question: "What is the average period of the pendulum?". The options are: A 0.88 s, B 0.92 s, C 17.6 s, D 18.4 s. A callout box on the right points to the options with the text: "Button to choose answer.".

At the bottom of the screen are navigation buttons: "Previous" and "Next". A callout box on the right points to these buttons with the text: "Button to change multiple question.".

Revision Page (student), Review Page (Class Owner)

Q 2 :A student investigates a pendulum. He measures the time for the pendulum to complete 20 oscillations. He repeats the experiment three more times. The readings are shown. experiment time for 20 oscillations / s 1 17.6 2 19.8 3 17.6 4 18.6 What is the average period of the pendulum? A 0.88 s B 0.92 s C 17.6 s D 18.4 s

He measures the time for the pendulum to complete 20 oscillations.

He repeats the experiment three more times.

The readings are shown.

experiment	time for 20 oscillations / s
1	17.6
2	19.8
3	17.6
4	18.6

What is the average period of the pendulum?

- A 0.88 s B 0.92 s C 17.6 s D 18.4 s

Your Answer:

Correct Answer:

Commentary:

Mark: 0/2 0.0%

total Mark: 2/4 50.0%

D

B

Multiple choice question type does not support AI commentary

Difficulty: 1

User's answer.

correct answer.

Chat GPT's justification.

Previous Next

Username: admin | Class: IGCSE physics revision club | Role: Admin

Time Remaining: 58:53

Q 1 :Coal-fired power stations provide electricity for homes and industry. A government decides to replace a coal-fired power station with a hydroelectric power station. (a) Describe how electrical energy may be obtained from the gravitational potential energy of the water behind a hydroelectric dam.

- 4 Coal-fired power stations provide electricity for homes and industry.

A government decides to replace a coal-fired power station with a hydroelectric power station.

- (a) Describe how electrical energy may be obtained from the gravitational potential energy of the water behind a hydroelectric dam.

.....

 [3]

water fall and turn turbine. the turbine generate the electricity

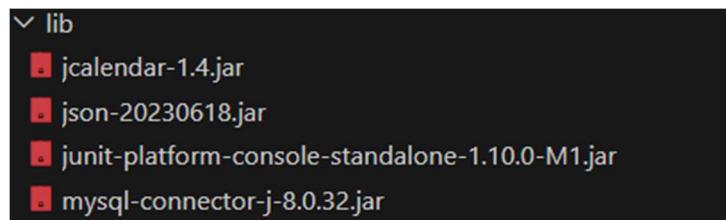
Text field to answer writing question.

Next

Username: admin | Class: IGCSE physics revision club | Role: Admin

External Libraries

Jcalander.jar provides the capability to conveniently process date selection and display. json.jar is used to handle JSON data, mysql-connector.jar supports efficient communication between Java applications and MySQL databases, and junit-platform-console-standalone.jar is used to verify the accuracy and quality of code.



```
● ● ●
1 import org.json.JSONObject;
```

```
● ● ●
1 import com.toedter.calendar.JCalendar;
2 import com.toedter.calendar.JDateChooser;
```

```
● ● ●
1 import java.sql.Connection;
2 import java.sql.DriverManager;
3 import java.sql.PreparedStatement;
4 import java.sql.ResultSet;
5 import java.sql.SQLException;
6 import java.util.ArrayList;
7 import java.util.List;
```

Web API and auto grading

this HTTP client class('AIclient2') allows us to mark the essay question with ChatGPT by using Web API that is "ChatGPT Best Price" provided by "rapid Api". This returns a response in a JSON format. This plays an important role in reducing the workload of running an IGCSE physics club, which the client already has a heavy workload with his job as a teacher.

```
Set AI model as 'gpt-3.5-turbo' → Line 11: private String model = "gpt-3.5-turbo";  
Make a request header by using 'url' from class parameter API-KEY, host name. → Lines 25-30: HttpClient httpClient = HttpClient.newHttpClient();  
HttpRequest request = HttpRequest.newBuilder()  
.uri(new URI(url))  
.header("Content-Type", "application/json")  
.header("X-RapidAPI-Key", "API-KEY FROM RapidAPI")  
.header("X-RapidAPI-Host", "chatgpt-best-price.p.rapidapi.com")  
.POST(HttpRequest.BodyPublishers.ofString("{\"model\": \"" + model  
+ "\", \"messages\": [{\"role\": \"user\", \"content\": \"" + content + "\"}]}"))  
.build();  
Use post method to build a JSON that has model name and content from function parameter. → Lines 31-34: HttpResponse<String> response = httpClient.send(request, BodyHandlers.ofString());  
responseString = response.body();  
} catch (URISyntaxException | IOException | InterruptedException e) {  
e.printStackTrace();  
}  
Send request with http protocol to webserver. → Line 36: return responseString;  
Error handling → Line 36: return responseString;
```

The diagram illustrates the AIclient2.java code structure with various annotations:

- Set AI model as 'gpt-3.5-turbo'**: Points to the line where the AI model is set to "gpt-3.5-turbo".
- Make a request header by using 'url' from class parameter API-KEY, host name.**: Points to the code block that constructs the HTTP request, including headers for Content-Type, X-RapidAPI-Key, and X-RapidAPI-Host.
- Use post method to build a JSON that has model name and content from function parameter.**: Points to the code block that uses the POST method to send a JSON payload containing the model name and content.
- Send request with http protocol to webserver.**: Points to the return statement that sends the request back to the webserver.
- Error handling**: Points to the catch block that handles URISyntaxException, IOException, and InterruptedException.

Use the 'AIclient2' class:

```
1 String generatedPrompt = settingPrompt(questionString, markSheetAnswer, userAnswer, maxMark);  
2 AIclient2 AIclient = new AIclient2("https://chatgpt-best-price.p.rapidapi.com/v1/chat/completions");  
3 String content = generatedPrompt;  
4 String response = AIclient.requestToAI(content);  
5 ResponseParser(response);  
Create prompt for ChatGPT. → Line 1: settingPrompt(questionString, markSheetAnswer, userAnswer, maxMark);  
Create an 'AIclient2' object. → Line 2: AIclient2 AIclient = new AIclient2("https://chatgpt-best-price.p.rapidapi.com/v1/chat/completions");  
call ResponseParser() to convert JSON object to useful information. → Line 5: ResponseParser(response);
```

The diagram illustrates the usage of the AIclient2 class with annotations explaining each step:

- Create prompt for ChatGPT.**: Points to the line where the generated prompt is created using the settingPrompt method.
- Create an 'AIclient2' object.**: Points to the line where the AIclient2 object is created using the constructor.
- call ResponseParser() to convert JSON object to useful information.**: Points to the line where the ResponseParser method is called to convert the JSON response into useful information.

A prompt is created using the 'settingPrompt()' method.

```
1 public static String settingPrompt(String question, String markSheet, String userAnswer, int maxMark) {  
2     prompt = String.format(  
3         "you are grader, the IGCSE student answered determin awardmark and justification      " +  
4         "Question is " + question + "    " +  
5         "marksheet is " + markSheet + "    " +  
6         "useranswer is " + userAnswer + "    " +  
7         "maxMark is " + maxMark + "    " + "    " +  
8         "Please answer in the following format with the above information.      " +  
9         "    " +  
10        "Award Mark:           " +  
11        "Justification:"  
12  
13    );  
14    return prompt;  
15 }
```

Use ResponseParser() method to parse the JSON object.

```
1 public static void ResponseParser(String jsonResponse) {  
2     JSONObject jsonObject = new JSONObject(jsonResponse);  
3     JSONObject messageObject = jsonObject.getJSONArray("choices")  
4         .getJSONObject(0)  
5         .getJSONObject("message");  
6  
7     String content = messageObject.getString("content");  
8     int startIndex = content.indexOf("Award Mark: ") + 12;  
9     int endIndex = content.indexOf("\n", startIndex);  
10  
11     awardMark = Integer.parseInt(content.substring(startIndex, endIndex).trim());  
12  
13     startIndex = content.indexOf("Justification: ") + 15;  
14     justification = content.substring(startIndex).trim();  
15  
16     System.out.println(awardMark);  
17     System.out.println("\n");  
18     System.out.println(justification);  
19 }
```

Declare JSON object.

In the content, get useful information such as award Mark, justification.

Encapsulation

I utilized encapsulation throughout the program to ensure that some variables could not be accessed directly without the use of getters and setters. I had to utilize these methods to access these variables consistently, which simplified and improved the code. This is also a common OOP feature, which makes my software simpler to comprehend and maintain. Additionally, by afterwards placing these objects in the array list after creating them in this manner, the code became even clearer.

```
1 public class Class {  
2     private int classID;  
3     private String className;  
4     private String classOwnerUserKey;  
5     private String classEnterCode;  
6     private int classMaxStu;  
7     private Date classExpirationDate;  
8  
9     public Class(int classID, String className, String classOwnerUserKey,  
10                 String classEnterCode, int classMaxStu, Date classExpirationDate) {  
11         this.classID = classID;  
12         this.className = className;  
13         this.classOwnerUserKey = classOwnerUserKey;  
14  
15         this.classEnterCode = classEnterCode;  
16         this.classMaxStu = classMaxStu;  
17         this.classExpirationDate = classExpirationDate;  
18     }  
19  
20     public int getClassID() {  
21         return classID;  
22     }  
23  
24     public void setClassID(int classID) {  
25         this.classID = classID;  
26     }  
}
```

Encapsulated variables

Getter method

Setter method

Inheritance

I utilized inheritance to reduce time since, in the end, the makeAffiliationClassGUI basically adds different components to the makeClassGUI and adds pertinent features to the Affiliation.

```
1 public class makeAffiliationClassGUI extends makeClassGUI
```

Inheritance was also used in the class for encapsulation. This is the case with Affiliation Class and Class. AffiliationClass's object is suitable for using inheritance because it only needs to add a few variables such as 'affiliationName' along with the variables in Class.

```
1 public class Class {  
2     private int classID;  
3     private String className;  
4     private String classOwnerUserKey;  
5     private String classEnterCode;  
6     private int classMaxStu;  
7     private Date classExpirationDate;  
8  
9     public Class(int classID, String className, String classOwnerUserKey,  
10                  String classEnterCode, int classMaxStu, Date classExpirationDate) {  
11         this.classID = classID;  
12         this.className = className;  
13         this.classOwnerUserKey = classOwnerUserKey;  
14  
15         this.classEnterCode = classEnterCode;  
16         this.classMaxStu = classMaxStu;  
17         this.classExpirationDate = classExpirationDate;  
18     }  
}
```

Encapsulated variables in 'Class' class

```
1 public class AffiliationClass extends Class {  
2     private String classAffiliationID;  
3     private String affiliationName;  
4     private Date affiliationExpirationDate;  
5     private String affiliationOwnerUserKey;  
6     private String affiliationOwnerUserName;  
7  
8     public AffiliationClass(int classID, String className, String classOwnerUserKey, String classEnterCode,  
9                             int classMaxStu, Date classExpirationDate, String classAffiliationID, String affiliationName,  
10                            Date affiliationExpirationDate, String affiliationOwnerUserKey, String affiliationOwnerUserName) {  
11         super(classID, className, classOwnerUserKey, classEnterCode, classMaxStu, classExpirationDate);  
12         this.classAffiliationID = classAffiliationID;  
13         this.affiliationName = affiliationName;  
14         this.affiliationExpirationDate = affiliationExpirationDate;  
15         this.affiliationOwnerUserKey = affiliationOwnerUserKey;  
16         this.affiliationOwnerUserName = affiliationOwnerUserName;  
17     }  
}
```

Encapsulated variables in 'AffiliationClass' class

inherited variables from superclass ('Class' class)

Complex Methods to customise JTable

The code extends the JTable class to create unique cell renderers and editors. Where the JTable is inherited, and the method is overridden to vary the behavior for each column. It allows the modification of the JTable as purpose. This is required to increase user usability by changing JTable for build Task manager (see Graphical User Interface (GUI) - Task manager in page 2).

```

1  private JTable createTaskTable(List<Tasks> tasksList) {
2
3      String[] columnNames = {
4          "Task ID", "For", "Finish time", "result %",
5          "State", "review"
6      };
7
8      DefaultTableModel tableModel = new DefaultTableModel(columnNames, 0);
9
10     for (Tasks task : tasksList) {
11         sql sql = new sql();
12         String TaskFor = task.getTaskFor();
13         sql.getUserInfo(Integer.valueOf(TaskFor));
14
15         // Call the getUserInfo method and retrieve the values
16         String StudentName = sql.getUsername();
17
18         Object[] rowData = {
19             task.getIdTaskSQL(), StudentName, task.getTaskFinishTime(), task.getTaskPercentage()
20         };
21         tableModel.addRow(rowData);
22     }
23
24     JTable table = new JTable(tableModel) {
25
26         @Override
27         public TableCellRenderer getCellRenderer(int row, int column) {
28             if (column == 5) { // Button column
29                 return new ButtonRenderer();
30             }
31
32             else if (column == 4) { // Task state column (13th column)
33                 return new DefaultTableCellRenderer() {
34                     @Override
35                     public Component getTableCellRendererComponent(JTable table, Object value,
36                         boolean isSelected, boolean hasFocus, int row, int column) {
37
38                         Tasks task = tasksList.get(row);
39
40                         super.getTableCellRendererComponent(table, value, isSelected, hasFocus, row,
41                             column);
42
43                         // Calculate task state based on TaskDueDay, TaskFinish, and TaskFinishTime
44                         String taskState = calculateTaskState(task.getTaskDueDay(), task.getTaskFinish(),
45                             task.getTaskFinishTime());
46                         setText(taskState); // Set the task state text
47                         return this;
48                     }
49                 };
50             }
51
52             return super.getCellRenderer(row, column);
53         }
54
55         @Override
56         public TableCellEditor getCellEditor(int row, int column) {
57             if (column == 5) { // Button column
58                 return new ButtonEditor(new JCheckBox());
59             }
60             return super.getCellEditor(row, column);
61         }
62
63         table.setPreferredScrollableViewportSize(new Dimension(800, 400)); // Adjust the size as needed
64
65         TableColumn gaugeColumn = new TableColumn(3, 100); // Column index and width
66         gaugeColumn.setHeaderValue("Result Bar");
67         table.addColumn(gaugeColumn);
68         gaugeColumn.setCellRenderer(new GaugeBarRenderer());
69
70     }
71
72     return table;
73 }

```

Define column names.

Creates a DefaultTableModel with the specified column names and zero rows.

Iterates through the tasks list, get information such as task ID, student name, task finish time, task percentage from Tasks object and the database and add to the table

Override getCellRenderer() method to customize cell render.

Return a custom button renderer.

Return a custom cell renderer with overriding method to customize column to show calculated task state.

Override getCellEditor() method to implicate Jcheckbox() to allow the buttons to be clicked.

Add gaugeColumn to show result bar (show the Correct percentage as bar).

```

1 class GaugeBarRenderer extends JProgressBar implements TableCellRenderer {
2     public GaugeBarRenderer() {
3         setBorderPainted(false);
4         setStringPainted(true);
5         setString(""); // Empty string to show only the gauge bar
6     }
7
8     @Override
9     public Component getTableCellRendererComponent(JTable table, Object value,
10         boolean isSelected, boolean hasFocus, int row, int column) {
11         int percentage = Integer.parseInt(value.toString());
12         setValue(percentage);
13         return this;
14     }
15 }
16
17 class ButtonRenderer extends JButton implements TableCellRenderer {
18     public ButtonRenderer() {
19         setOpaque(true);
20     }
21
22     @Override
23     public Component getTableCellRendererComponent(JTable table, Object value,
24         boolean isSelected, boolean hasFocus, int row, int column) {
25         setText("Review"); // Set button text
26         return this;
27     }
28 }
29
30 class ButtonEditor extends DefaultCellEditor {
31     protected JButton button;
32     private int currentRow;
33
34     public ButtonEditor(JCheckBox checkBox) {
35         super(checkBox);
36         button = new JButton();
37         button.setOpaque(true);
38         setClickCountToStart(1);
39         button.addActionListener(e -> {
40             // Open a new window based on the button click
41             Tasks task = tasksList.get(currentRow);
42
43             if (task.getTaskFinish().equals("1")) {
44                 // Task is finished, open revision page
45                 revisionPage revisionPage = new revisionPage(userKey, task);
46                 revisionPage.setVisible(true);
47             } else if (task.getTaskFinish().equals("0")) {
48                 // Task is not finished, show a message
49                 JOptionPane.showMessageDialog(null, "Task is not finished.");
50             }
51         });
52     }
53
54     @Override
55     public Component getTableCellEditorComponent(JTable table, Object value,
56         boolean isSelected, int row, int column) {
57         currentRow = row; // Store the current row for reference
58         button.setText("Review"); // Set button text
59         return button;
60     }
61 }
62

```

Initialize gaugeBarRenderer with not showing border and showing empty string.

Overriding the method to customize the rendering component for the result bar column and set percentage from the parameter object.

Initialize ButtonRenderer with making the button opaque.

Overriding the method to customize the rendering component for the review button column and set the text of the button to "Review".

Initializes the button editor with specific settings and action listener to handle button clicks. When the button is clicked, open the revisionPage(review page) [see GUI – revision page (student) review page (class owner)]

Overriding the method to customize the editor component for the review button column

Security (Hashing password)

By using SHA-256 algorithm, the password encrypted when the password upload to database. The login system hashes user input with SHA-256 algorithm and compares with data from database. It improves the level of security. Even when the database gets hacked the password will not be able to decrypt.

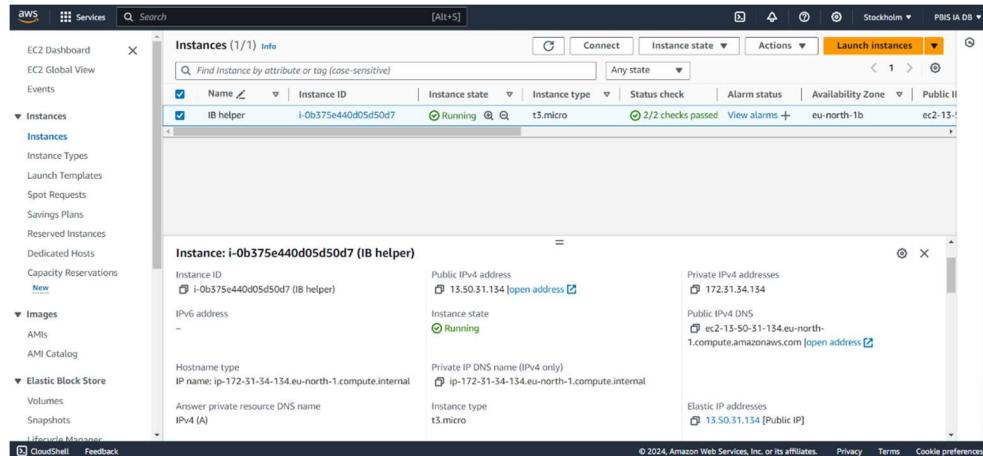
```
● ● ●
1 Connection con = DriverManager.getConnection("jdbc:mysql://13.50.31.134/IB_CS_IA",
2           "kado",
3           "-----");
4 String query = "SELECT * FROM user_info WHERE Email = ? AND pass = ?";
5 PreparedStatement statement = con.prepareStatement(query);
6 statement.setString(1, userValue);
7 passenc = SHA256.enqrSHA(passValue); ← Data encrypting.
8 statement.setString(2, passenc);
9 System.out.println(passenc);
10 ResultSet resultSet = statement.executeQuery();
```

```
● ● ●
1 public static String enqrSHA(String input) {
2     try {
3         // Create MessageDigest instance for SHA-256
4         MessageDigest sha256 = MessageDigest.getInstance("SHA-256");
5
6         // Apply SHA-256 encryption to the input
7         byte[] hashBytes = sha256.digest(input.getBytes());
8
9         // Convert byte array to a string of hexadecimal values
10        StringBuilder hexString = new StringBuilder();
11        for (byte hashByte : hashBytes) {
12            String hex = Integer.toHexString(0xff & hashByte);
13            if (hex.length() == 1) {
14                hexString.append('0');
15            }
16            hexString.append(hex);
17        }
18        return hexString.toString();
19
20    } catch (NoSuchAlgorithmException e) {
21        e.printStackTrace();
22        return null;
23    }
24 }
```

Web database server

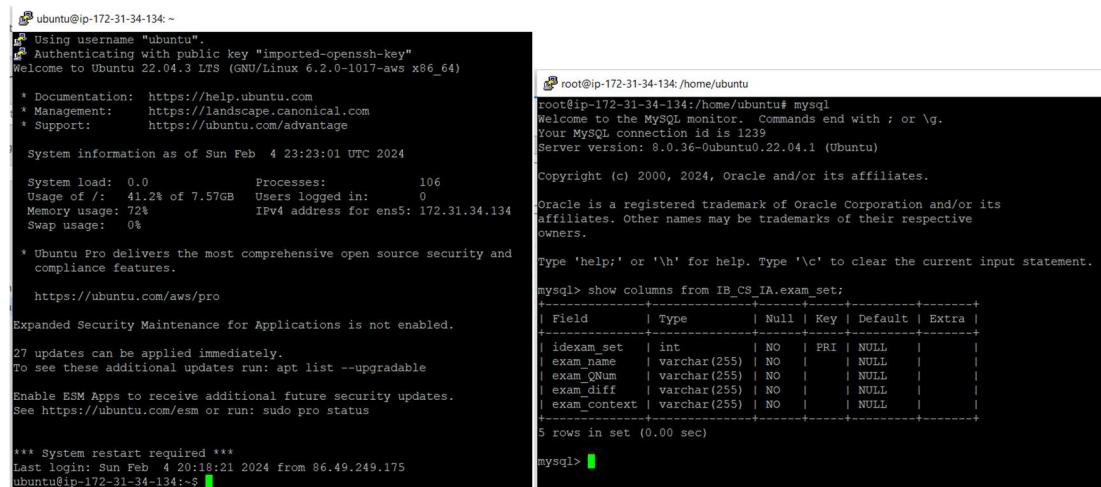
This program requires students to check the task set by the teacher on the teacher's computer in real-time and take the test. All the data in the program must interact with the online database organically and in real-time. Therefore, building a web online database server is desirable. This means that students and teachers don't have to work on the same computer, while also giving students the ability to go home and do their homework.

AWS EC2 service that lends server computer has ubuntu OS:



The screenshot shows the AWS EC2 Instances page. A single instance named "IB helper" is listed, which is currently running. The instance is of type t3.micro and is located in the eu-north-1b availability zone. It has a public IP address of 13.50.31.134 and a private IP address of 172.31.34.134. The instance was launched on 2024-02-04 at 13:50:57.

SSL connection to my web database server that has Nginx, Node.js, MySQL:



```
ubuntu@ip-172-31-34-134:~  
Using username "ubuntu".  
Authenticating with public key "imported-openssh-key"  
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 6.2.0-1017-aws x86_64)  
  
* Documentation: https://help.ubuntu.com  
* Management: https://landscape.canonical.com  
* Support: https://ubuntu.com/advantage  
  
System information as of Sun Feb  4 23:23:01 UTC 2024  
  
System load: 0.0 Processes: 106  
Usage of /: 41.2% of 7.57GB Users logged in: 0  
Memory usage: 72% IPv4 address for ens5: 172.31.34.134  
Swap usage: 0%  
  
* Ubuntu Pro delivers the most comprehensive open source security and  
compliance features.  
  
https://ubuntu.com/aws/pro  
  
Expanded Security Maintenance for Applications is not enabled.  
27 updates can be applied immediately.  
To see these additional updates run: apt list --upgradable  
Enable ESM Apps to receive additional future security updates.  
See https://ubuntu.com/esm or run: sudo pro status  
  
*** System restart required ***  
Last login: Sun Feb  4 20:18:21 2024 from 86.49.249.175  
ubuntu@ip-172-31-34-134:~$  
  
root@ip-172-31-34-134:/home/ubuntu  
root@ip-172-31-34-134:/home/ubuntu# mysql  
Welcome to the MySQL monitor. Commands end with ; or \g.  
Your MySQL connection id is 1239  
Server version: 8.0.36-0ubuntu0.22.04.1 (Ubuntu)  
  
Copyright (c) 2000, 2024, Oracle and/or its affiliates.  
Oracle is a registered trademark of Oracle Corporation and/or its  
affiliates. Other names may be trademarks of their respective  
owners.  
  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
  
mysql> show columns from IB_CS_IA.exam_set;  
+-----+-----+-----+-----+-----+  
| Field | Type | Null | Key | Default | Extra |  
+-----+-----+-----+-----+-----+  
| indexam_set | int | NO | PRI | NULL |  
| exam_name | varchar(255) | NO | | NULL |  
| exam_QNum | varchar(255) | NO | | NULL |  
| exam_diff | varchar(255) | NO | | NULL |  
| exam_context | varchar(255) | NO | | NULL |  
+-----+-----+-----+-----+-----+  
5 rows in set (0.00 sec)  
  
mysql>
```

Connect to webserver with chrome:



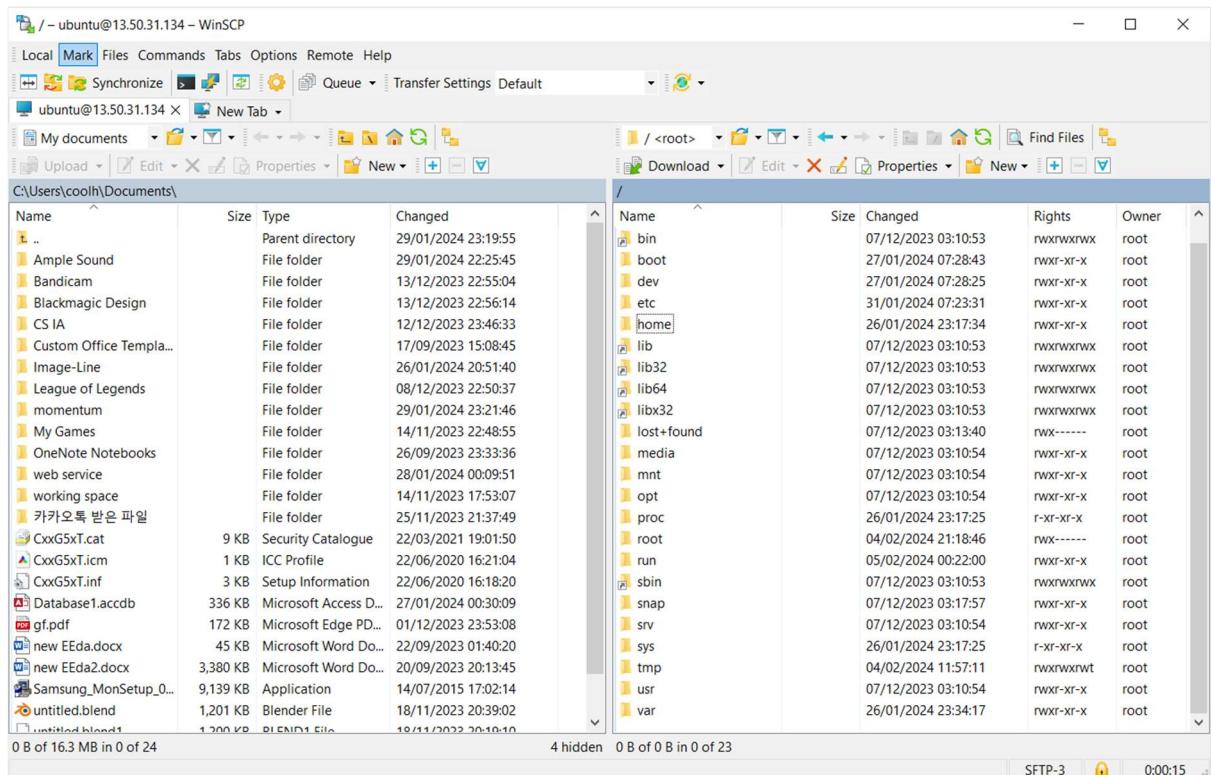
Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org. Commercial support is available at nginx.com.

Thank you for using nginx.

File view of the webserver:



Implication:

```

 1  public void getUserInfo(int userKey) {
 2      try {
 3          Connection con = DriverManager.getConnection("jdbc:mysql://13.50.31.134/IB_CS_IA",
 4          "kado","-----");
 5          String query = "SELECT * FROM user_info WHERE userKey = ?";
 6          PreparedStatement statement = con.prepareStatement(query);
 7          statement.setInt(1, userKey);
 8          ResultSet resultSet = statement.executeQuery();
 9
10          if (resultSet.next()) {
11              // Retrieve values from the result set
12              username = resultSet.getString("name");
13              email = resultSet.getString("Email");
14              Class = resultSet.getString("class");
15              userRole = resultSet.getString("userRole");
16              classId = resultSet.getInt("classID");
17
18              userRoleL = getUserRoleString(userRole);
19
20          }
21
22          con.close();
23      } catch (SQLException e) {
24          e.printStackTrace();
25      }
26  }

```

2D Array

The use of a 2D Array made the program clearer. Using a 2d array was a logical choice of data structure as it allowed the program to not only store the exam and questions. This enabled the program to have an appropriate structure to store the questions as they read from the database. This was the efficient way to handle and transfer data from the database to the KNN algorithm.

```
1 examsetData = new Object[examSetList.length][10]; ← initializes a 2D array named examsetData with the first dimension's length is the length of the examSetList array and the second dimension's length is 10.
2 try { ← Iterates over the examSetList and the examSet Object is used to next lines.
3     Connection con = DriverManager.getConnection("jdbc:mysql://13.50.31.134/IB_CS_IA",
4     "kado","-----");
5     String query = "SELECT * FROM q_set WHERE Q_examSet_id = ?";
6     int flag_examSet = 0;
7     for (ExamSet examSet : examSetList) {
8
9         PreparedStatement statement = con.prepareStatement(query);
10        int examId = examSet.getIdexam_set();
11        statement.setInt(1, examId);
12        ResultSet resultSet = statement.executeQuery();
13        int exam_QNum = Integer.valueOf(examSet.getExam_QNum());
14        Object[] setInfo = new Object[exam_QNum + 1];
15        setInfo[0] = examSet;
16        int flag_setInfo = 1;
17
18        while (resultSet.next()) {
19
20            // Your existing code for fetching question details
21            int idQ_sets = resultSet.getInt("idQ_set");
22            String Q_questions = resultSet.getString("Q_question");
23            int Q_examSet_ids = resultSet.getInt("Q_examSet_id");
24            String Q_diffs = resultSet.getString("Q_diff");
25            byte[] Q_images = resultSet.getBytes("Q_image");
26            String Q_answers = resultSet.getString("Q_answer");
27
27            String Q_type = resultSet.getString("Q_type");
28            String Q_points = resultSet.getString("Q_points");
29            String Q_topic = resultSet.getString("Q_topic");
30
31            Question question = new Question(idQ_sets, Q_questions, Q_examSet_ids,
32                                              Q_diffs, Q_images, Q_type,
33                                              Q_answers,
34                                              Q_points, Q_topic);
35
36            // Now set the value of x in the setInfo array
37
38            setInfo[flag_setInfo] = question; // -1 because arrays are 0-based ← iterates over the result from the database. The detailed information of each question is used to construct a Question object.
39            flag_setInfo++;
40
41        }
42
43    }
44
45    examsetData[flag_examSet] = setInfo; ← assigns the Question object to the setInfo array at the position indicated by flag_setInfo, then increments flag_setInfo to prepare for the next iteration.
46    flag_examSet++;
47
48
49 }
```

initializes a 2D array named examsetData with the first dimension's length is the length of the examSetList array and the second dimension's length is 10.

Iterates over the examSetList and the examSet Object is used to next lines.

Get the exam ID from exam set object in exam set list and set query with this exam ID.

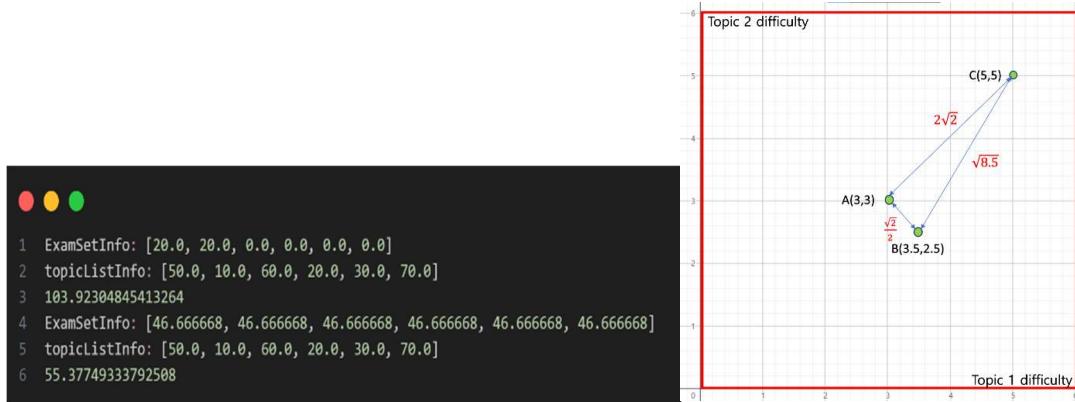
iterates over the result from the database. The detailed information of each question is used to construct a Question object.

assigns the Question object to the setInfo array at the position indicated by flag_setInfo, then increments flag_setInfo to prepare for the next iteration.

assigns the setInfo array to the examsetData array at the position indicated by flag_examSet, then increments flag_examSet to prepare for the next iteration.

KNN algorithm for recommend system

K-Nearest Neighbors Algorithm *is a type of supervised learning algorithm used for both regression and classification [6]*. It calculates similarity between user's topical predicted grades and topical difficulties exam set. It recommends the exam set with the highest similarity.



The similarity corresponds to the distance between two data sets which can be calculated follow.

For points with data coordinates $(p_1, p_2, p_3, \dots, p_n)$ and $(q_1, q_2, q_3, \dots, q_n)$, the distance is calculated as:

$$d(p, q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \dots + (p_n - q_n)^2}$$

It is used because a mathematically simple model and relatively easy to code compared to other machine learning algorithms, does not require training steps, and is suitable for both classification and regression tasks. For input, a 2D array is used. This is used to recommend exam sets that are appropriate for each student's level, helping them to improve their performance.

```

1 private List<Integer> recommendExamSet(int k) {
2     Map<Integer, Double> distances = new HashMap<>();
3
4     // Iterate through each setInfo in examsetData and calculate distances
5     for (Object[] setInfo : examsetData) {
6         ExamSet examSet = (ExamSet) setInfo[0];
7         int examSetID = examSet.getIdexam_set();
8
9         double distance = calculateDistance(topicList, setInfo);
10        distances.put(examSetID, distance);
11    }
12
13    // Sort distances and find k-nearest neighbors
14    List<Integer> recommendedExamSetIDs = new ArrayList<>();
15    distances.entrySet().stream()
16        .sorted(Map.Entry.comparingByValue())
17        .limit(k)
18        .forEach(entry -> recommendedExamSetIDs.add(entry.getKey()));
19    System.out.println(recommendedExamSetIDs);
20
21    return recommendedExamSetIDs;
22 }

```

Return list containing k number of the exam set IDs.

Declare a hash table.

Use setInfo array in examsetData 2D array to compare itself and topicList and calculate distance between two data sets and add distance to hash table using examsetID as key.

User's predicted grade list

Sort hash table by distance, limit k number of nearest neighbors, add their exam ID to recommendedExamSetIDs list.

```

1 private static double calculateDistance(float[][] topicList, Object[] setInfo) {
2     // Calculate ExamSetInfo and topicListInfo
3     float[] ExamSetInfo = new float[6];
4     int[] topicQuestionCount = new int[6];
5
6     for (int i = 1; i < setInfo.length; i++) {
7         Question question = (Question) setInfo[i];
8         int topic = Integer.parseInt(question.getIdQ_topic());
9         float difficulty = 10 * Float.parseFloat(question.getQ_diff());
10
11        ExamSetInfo[topic - 1] += difficulty;
12        topicQuestionCount[topic - 1]++;
13    }
14
15    // Calculate average difficulty for each topic in topicList
16    float[] topicListInfo = new float[6];
17
18    for (float[] topicInfo : topicList) {
19        int topic = (int) topicInfo[0];
20        float difficulty = topicInfo[1];
21
22        topicListInfo[topic - 1] = difficulty;
23    }
24
25    // Calculate average difficulty for each topic in ExamSetInfo
26    for (int i = 0; i < ExamSetInfo.length; i++) {
27        if (topicQuestionCount[i] > 0) {
28            ExamSetInfo[i] /= topicQuestionCount[i];
29        }
30    }
31
32    // Calculate Euclidean distance
33    double distance = 0.0;
34    for (int i = 0; i < ExamSetInfo.length; i++) {
35        distance += Math.pow(ExamSetInfo[i] - topicListInfo[i], 2);
36    }
37    distance = Math.sqrt(distance);
38
39    // Print the calculated arrays
40    System.out.println("ExamSetInfo: " + Arrays.toString(ExamSetInfo));
41    System.out.println("topicListInfo: " + Arrays.toString(topicListInfo));
42    System.out.println(distance);
43
44    return distance;
45 }

```

Declare list to store aggregated difficulty value.

Declare list to track of the number of questions for each topic.

iterates through each question in the exam set and aggregate difficulty and total question number.

iterates through topic list and get info from this array such as each topic's number and difficulty and store those in topicListInfo list.

Get average difficulty of each topic by dividing aggregated difficulty by number of questions.

By using the distance formula, calculate distance between two data sets.

Hash table

The hash table stores ExamSet ID and the distances between the exam set topical difficulties and user's predicted grades for the KNN algorithm. The hash function has an average time complexity of $O(1)$ in search and insertion-deletion for a large number of data. It takes advantage of when the program has massive amount of examset data.

Time Complexity

Average Case	Add	Remove	Search
Array	$O(1)$	$O(n)$	$O(n)$
Sorted Array	$O(n)$	$O(\lg n)$	$O(\lg n)$
Linked List	$O(1)$	$O(n)$	$O(n)$
BST	$O(\lg n)$	$O(\lg n)$	$O(\lg n)$
Hash Table	$\sim O(1)$	$\sim O(1)$	$\sim O(1)$

Note: For sorted array and BST, keys have to be ordered.

5

[5]



```
1 Map<Integer, Double> distances = new HashMap<>();
```



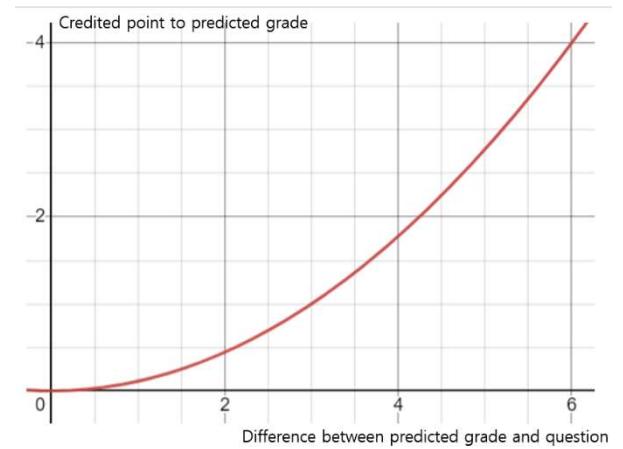
```
1 distances.entrySet().stream()
2         .sorted(Map.Entry.comparingByValue())
3         .limit(k)
4         .forEach(entry -> recommendedExamSetIDs.add(entry.getKey()))
```

Recursion

The square function was needed to create a system that would give students more credit if they did more difficult questions than their predicted score, and less credit if they did less difficult questions, and factor that into their predicted score. This code allows the users to see their predicted grade rise should they do well with the more difficult questions. This also serves to motivate the user to keep trying harder questions.

```
● ● ●
1 public float PowerFunction(float base, int power) {
2
3     if (power == 1) {
4         return 1;
5     } else if (power > 0) {
6         return base * PowerFunction(base, power - 1);
7     } else if (power < 0) {
8         return 1 / PowerFunction(base, -power);
9     } else {
10        return 0.0f;
11    }
12 }
```

```
● ● ●
1 float base = (diffgrade / 3);
2 adjustPlusGrade = PowerFunction(base, 2);
```



Error handling

To handle failures (such as SQL errors and input errors) that occur in the program, I employed try-catch methods throughout the program. This helps avoid a disastrous situation in which the application must terminate owing to a mistake or the data in the database is destroyed.

```
17 public String requestToAI(String content) {
18     String responseString = "";
19
20     try {
21         HttpClient httpClient = HttpClient.newHttpClient();
22         HttpRequest request = HttpRequest.newBuilder()
23             .uri(new URI(url))
24             .header("Content-Type", "application/json")
25             .header("X-RapidAPI-Key", "API-KEY FROM RapidAPI")
26             .header("X-RapidAPI-Host", "chatgpt-best-price.p.rapidapi.com")
27             .POST(HttpRequest.BodyPublishers.ofString("{\"model\": \""
28                 + "\", \"messages\": [{\"role\": \"user\", \"content\": \""
29                 + content + "\"}] })")
30             .build();
31
32         HttpResponse<String> response = httpClient.send(request, BodyHandlers.ofString());
33         responseString = response.body();
34     } catch (URISyntaxException | IOException | InterruptedException e) {
35         e.printStackTrace();
36     }
37
38     return responseString;
39 }
40 }
```

To make sure that the incoming e-mail follows the normal format, the program validates it with the following code:

```
1 private boolean isValidEmail(String email) {
2     String emailRegex = "^[a-zA-Z0-9_+&*-]+(?:\\.[a-zA-Z0-9_-+&*-]+)*@(?:[a-zA-Z0-9-]+\\.)+[a-zA-Z]{2,7}$";
3     return email.matches(emailRegex);
4 }
```

Lambda expression

It has streamlined the program's structure by assigning tasks to distinct functions. This approach provides a better organized and effective code structure in addition to making development easier.

```
● ● ●  
1 logoutButton.addActionListener(e -> {  
2     try {  
3         CreateLoginForm form = new CreateLoginForm();  
4         form.setSize(900, 600);  
5         centreWindow(form);  
6         form.setVisible(true);  
7     } catch (Exception E) {  
8         JOptionPane.showMessageDialog(null, E.getMessage());  
9     }  
10    this.dispose();  
11});  
1 Timer timer = new Timer(2000, e -> {  
2     try {  
3         getTaskInfo(userKey);  
4         showTask(tasksList);  
5         getGradeInfo(userKey);  
6         setGauge(userKey);  
7         getInfo();  
8     } catch (SQLException e1) {  
9         // TODO Auto-generated catch block  
10        e1.printStackTrace();  
11    }  
12});
```

```
● ● ●  
1 String[] options = {"Class", "Logout" };  
2 comboBox = new JComboBox<>(options);  
3 comboBox.addActionListener(e -> {  
4     String selectedOption = (String) comboBox.getSelectedItem();  
5     if (selectedOption.equals("Class")) {  
6         openClassWindow(userKey);  
7     } else if (selectedOption.equals("Logout")) {  
8         try {  
9             CreateLoginForm form = new CreateLoginForm();  
10            form.setSize(900, 600);  
11            centreWindow(form);  
12            form.setVisible(true);  
13        } catch (Exception E) {  
14            JOptionPane.showMessageDialog(null, E.getMessage());  
15        }  
16        this.dispose();  
17    }  
18});
```

Bibliography

- [1] Bhargava, a. y. (2017). hello coding 그림으로 개념을 이해하는 알고리즘(An Algorithm for Understanding Concepts with Hello Coding Figures). Seoul: 한빛미디어(Hanbit Media).
- [2] PlanetScale Documentation. (n.d.). PlanetScale Documentation | PlanetScale Documentation. <https://planetscale.com/docs>
- [3] Java Tutorial. (n.d.). Java Tutorial. <https://www.w3schools.com/java/default.asp>
- [4] ChatGPT Best Price API Documentation (truongvuhung102) | RapidAPI. (n.d.). ChatGPT Best Price API Documentation (Truongvuhung102) | RapidAPI. <https://rapidapi.com/truongvuhung102/api/chatgpt-best-price>
- [5] hash table. (n.d.). Hash Table. <https://velog.io/@gringrape/hash-table>
- [6] Christopher, A. (2021, December 29). K-Nearest Neighbor - The Startup - Medium. Medium. <https://medium.com/swlh/k-nearest-neighbor-ca2593d7a3c4>
- [7] OpenAI. ChatGPT, Version 3.5, 9 Mar. 2024, <https://chat.openai.com>

Word count: 985 (exclude heading, table of context, equation, labels, screenshot, bibliography)