

MIT WORLD PEACE UNIVERSITY

Object Oriented Programming with Java and C++  
Second Year B. Tech, Semester 1

---

---

MINI PROJECT WITH JAVA - PRICE GUESSING  
GAME  
*"How Much?"*

---

---

PROJECT REPORT

Prepared By

Krishnaraj Thadesar  
Cyber Security and Forensics  
Batch A2, PA 20

November 25, 2022

## **Contents**

<b>1 Introduction</b>	<b>1</b>
<b>2 Methodology</b>	<b>1</b>
<b>3 Platform</b>	<b>1</b>
<b>4 Requirements</b>	<b>1</b>
<b>5 Installation and Running</b>	<b>2</b>
<b>6 Database Screenshots</b>	<b>2</b>
6.1 MongoDB . . . . .	2
6.2 Local CSV Files . . . . .	3
<b>7 Unique Features</b>	<b>3</b>
7.1 Dark Mode . . . . .	3
7.2 Data Backup . . . . .	4
7.3 Web Scrapping . . . . .	4
7.4 Working Login and Account Creation . . . . .	5
<b>8 Color Schemes Used</b>	<b>5</b>
<b>9 Screenshots of the Project</b>	<b>6</b>
9.1 The Login Page . . . . .	6
9.2 The Menu Screen . . . . .	6
9.3 The Topic Selection Screen . . . . .	7
9.4 The Highscore Screen . . . . .	7
9.5 The Help and About . . . . .	8
9.6 The Game Over Screen . . . . .	8
<b>10 Walk-Through of the Files</b>	<b>9</b>
10.1 Project Structure . . . . .	9
10.2 TopicsFrame.java . . . . .	11
10.3 MongoManager.java . . . . .	11
10.4 MenuFrame.java . . . . .	11
10.5 Main.java . . . . .	11
10.6 LoginFrame.java . . . . .	11
10.7 HighscoreFrame.java . . . . .	11
10.8 HelpFrame.java . . . . .	11
10.9 GameOverFrame.java . . . . .	11
10.10GameFrame.java . . . . .	12
10.11DataBaseManager.java . . . . .	12
10.12Colors.java . . . . .	12
10.13BackgroundPanel.java . . . . .	12
10.14AmazonScrapper.java . . . . .	12
<b>11 Conclusion and Topics Learnt</b>	<b>12</b>

<b>12 Dependencies</b>	<b>13</b>
<b>13 References</b>	<b>14</b>
<b>14 Code Files</b>	<b>14</b>

## **1 Introduction**

This project was made for Submission to Object Oriented Programming with Java and C++ as the End Semester Report. The Motivation behind selecting this topic was that Online shopping has become rather prevalent now a days after COVID, and that has made the average consumer more aware about prices of everyday items, as well as Items out of everyday scope rather well. This game tests that theory, while trying to make it fun and learning concepts of Java along the way.

*The Concept is simple. You are shown a few topics to select from, and then an image along with the title of the Product is shown. There are 4 Choices for its Price which you are supposed to guess within 10 Seconds. For guessing correctly, the time remaining gets added to your score, and you can try again upon guessing incorrectly.*

## **2 Methodology**

The Working Methodology of the Game is Discussed below in a few points and elaborated further in the Report.

- There are 2 Active databases Maintained throughout the execution of the Program, MongoDB and CSV. CSV support is added in case the User does not have MongoDB installed in his or her System.
- Upon Starting the Game, it checks for the last time its database was updated, if it was not within a day, it updates it.
- The databases are updated by quering directly to Amazon and Scraping data. Several Web-pages of Amazon are visited, and their pictures and prices are scrapped. They are then stored in the Database.
- The GUI is written entirely in Java Swing and awt.

## **3 Platform**

**Operating System:** Arch Linux x86-64

**IDEs or Text Editors Used:** IntelliJ Idea Ultimate Edition for Java

**Compilers :** javac, with JDK 18.0.2 for Java

**Database :** MongoDB 6.0.3.1

## **4 Requirements**

- **Java 8**
- **Any 32 or 64 bit Operating System**
- **1 GB RAM**
- **Active Internet Connection**

Management

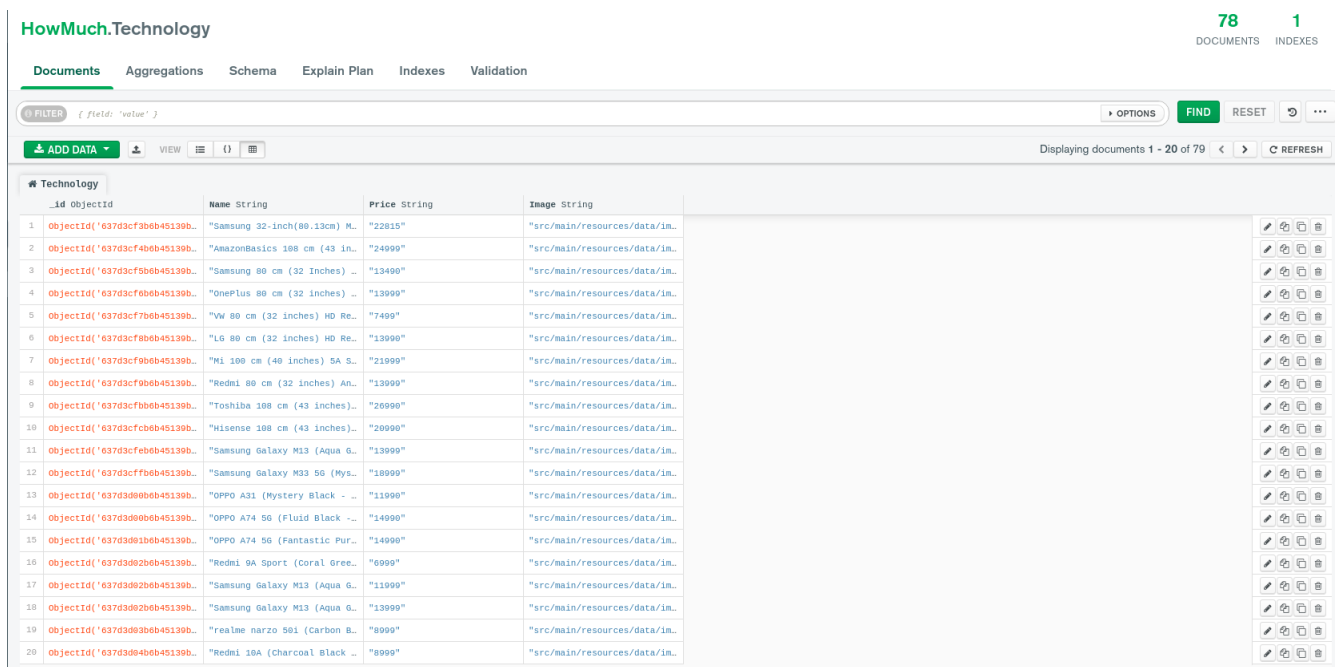
## 5 Installation and Running

- Navigate to <https://github.com/KrishnarajT/How-Much/releases>
- Download the .jar file from the releases when it is released that is.
- Navigate there from your terminal and do

```
java -jar ./How_Much.jar
```

## 6 Database Screenshots

### 6.1 MongoDB



The screenshot shows the MongoDB Compass interface. At the top, the title is 'HowMuch.Technology'. On the right, there are counters for '78 DOCUMENTS' and '1 INDEXES'. Below the title bar, there are tabs for 'Documents', 'Aggregations', 'Schema', 'Explain Plan', 'Indexes', and 'Validation'. The 'Documents' tab is selected. A filter bar shows '{ field: "value" }'. Below the filter bar, there are buttons for 'ADD DATA', 'VIEW', and a table icon. On the right, there are buttons for 'OPTIONS', 'FIND', 'RESET', and a refresh icon. Below these buttons, it says 'Displaying documents 1 - 20 of 79'. The main area displays a table of 20 documents. Each document has four fields: '\_id Objectid', 'Name String', 'Price String', and 'Image String'. The table is numbered 1 to 20 on the left. To the right of each row, there are icons for editing, deleting, and other actions.

#	Technology	_id Objectid	Name String	Price String	Image String
1		ObjectId('637d3cf3b6b45139b...')	"Samsung 32-inch(80.13cm) M...	"22815"	"src/main/resources/data/1m...
2		ObjectId('637d3cf4b6b45139b...')	"AmazonBasics 108 cm (43 in...	"24999"	"src/main/resources/data/1m...
3		ObjectId('637d3cf9b6b45139b...')	"Samsung 80 cm (32 Inches) ...	"13499"	"src/main/resources/data/1m...
4		ObjectId('637d3cf6b6b45139b...')	"OnePlus 80 cm (32 inches) ...	"13999"	"src/main/resources/data/1m...
5		ObjectId('637d3cf7b6b45139b...')	"VW 80 cm (32 inches) HD Re...	"7499"	"src/main/resources/data/1m...
6		ObjectId('637d3cf8b6b45139b...')	"LG 80 cm (32 inches) HD Re...	"13999"	"src/main/resources/data/1m...
7		ObjectId('637d3cf9b6b45139b...')	"Mi 100 cm (40 inches) 5A S...	"21999"	"src/main/resources/data/1m...
8		ObjectId('637d3cf9b6b45139b...')	"Redmi 80 cm (32 inches) An...	"13999"	"src/main/resources/data/1m...
9		ObjectId('637d3cf6b6b45139b...')	"Toshiba 108 cm (43 inches)...	"26999"	"src/main/resources/data/1m...
10		ObjectId('637d3cfcb6b45139b...')	"Hisense 108 cm (43 inches)...	"20999"	"src/main/resources/data/1m...
11		ObjectId('637d3cf6b6b45139b...')	"Samsung Galaxy M13 (Aqua G...	"13999"	"src/main/resources/data/1m...
12		ObjectId('637d3cf7b6b45139b...')	"Samsung Galaxy M13 5G (Mys...	"18999"	"src/main/resources/data/1m...
13		ObjectId('637d3d9b6b45139b...')	"OPPO A31 (Mystery Black - ...	"11999"	"src/main/resources/data/1m...
14		ObjectId('637d3d9b6b45139b...')	"OPPO A74 5G (Fluid Black -...	"14999"	"src/main/resources/data/1m...
15		ObjectId('637d3d91b6b45139b...')	"OPPO A74 5G (Fantastic Pur...	"14999"	"src/main/resources/data/1m...
16		ObjectId('637d3d92b6b45139b...')	"Redmi 9A Sport (Coral Gree...	"6999"	"src/main/resources/data/1m...
17		ObjectId('637d3d92b6b45139b...')	"Samsung Galaxy M13 (Aqua G...	"11999"	"src/main/resources/data/1m...
18		ObjectId('637d3d92b6b45139b...')	"Samsung Galaxy M13 (Aqua G...	"13999"	"src/main/resources/data/1m...
19		ObjectId('637d3d93b6b45139b...')	"realme narzo 56i (Carbon B...	"8999"	"src/main/resources/data/1m...
20		ObjectId('637d3d94b6b45139b...')	"Redmi 10A (Charcoal Black ...	"8999"	"src/main/resources/data/1m...

Figure 1: A Screenshot of the MongoDB Compass Showing Records Stored in the Teachnology Schema

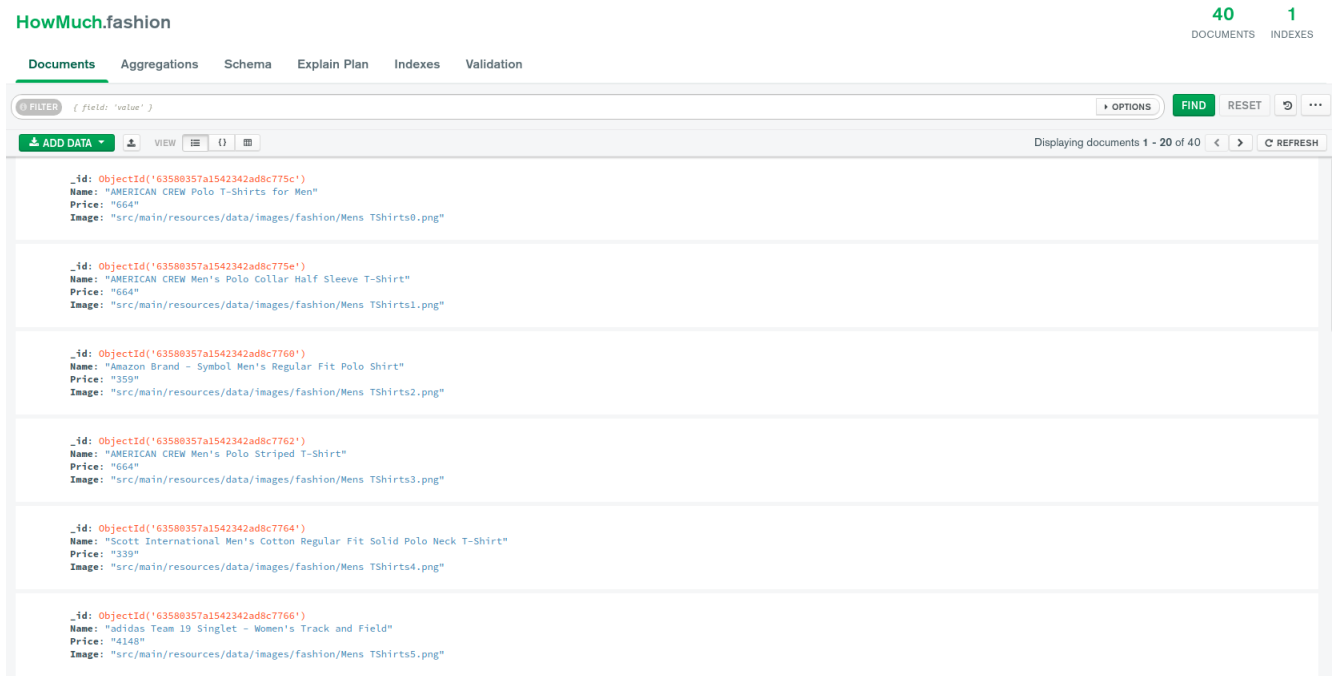


Figure 2: Record Showing the Fashion Schema Documents

## 6.2 Local CSV Files

C1	C2	C3
1 EPPE Men's Combo of Round Neck Half Sleeve Dryfit...	630	src/main/resources/data/images/fashion/Mens TShirts0.png
2 Allen Solly Men's Regular Fit T-Shirt	433	src/main/resources/data/images/fashion/Mens TShirts1.png
3 BLUELANDER Printed Round Neck Half Sleeve T-Shirt...	299	src/main/resources/data/images/fashion/Mens TShirts2.png
4 Allen Solly Men's Regular Fit T-Shirt	539	src/main/resources/data/images/fashion/Mens TShirts3.png
5 Allen Solly Men Polo	758	src/main/resources/data/images/fashion/Mens TShirts4.png
6 U.S. POLO ASSN. Men T-Shirt	352	src/main/resources/data/images/fashion/Mens TShirts5.png
7 Amazon Brand - Symbol Men T-Shirt	919	src/main/resources/data/images/fashion/Mens TShirts6.png
8 AELOMART Men's T Shirt	497	src/main/resources/data/images/fashion/Mens TShirts7.png
9 Scott International Men's Regular Fit T-Shirt (Pa...	474	src/main/resources/data/images/fashion/Mens TShirts8.png
10 Amazon Brand - Symbol Men's Regular T-Shirt	859	src/main/resources/data/images/fashion/Mens TShirts9.png
11 Park Avenue Full Sleeve Shawl Collar Dark Brown S...	4399	src/main/resources/data/images/fashion/Formal Suits0.png
12 Park Avenue Solid Rayon Blend Dark Blue Regular F...	4599	src/main/resources/data/images/fashion/Formal Suits1.png
13 Park Avenue Dark Grey Suits	4499	src/main/resources/data/images/fashion/Formal Suits2.png
14 Park Avenue Medium Grey Suits	5849	src/main/resources/data/images/fashion/Formal Suits3.png
15 Razab Enterprises (SAAYA) 5 Button Bandhgala/Jodh...	3045	src/main/resources/data/images/fashion/Formal Suits4.png
16 Arrow Men's Polyester Blend Formal Business Suit ...	4979	src/main/resources/data/images/fashion/Formal Suits5.png

Figure 3: Screenshot of the Local CSV File

## 7 Unique Features

### 7.1 Dark Mode

Dark mode is toggled by a switch. It simply flips a boolean variable statically defined in Colors.java. Other classes will then set Colors on their screens depending on this variable, for each Swing element in their Panel or Frame.



Figure 4: Dark mode Turned on



Figure 5: Dark mode Turned Off

## 7.2 Data Backup

Data backup is an important feature that ensures the user never has to face a situation where there is no product to be loaded on the Screen.

- There are 3 Databases maintained.
- When updating, the program updates MongoDB and CSV if they have not been updated.
- Each of the database have their own text file to maintain when the last time it was that they got updated.
- They are updated only once a day, as updating takes time.
- Updating is done in separate threads running in the Background.
- There is a 3rd backup database of CSV files that just duplicates the current state of the CSV database each time the user exits the program.
- The Game can update the Database in the Background while the user is playing the game, and at this point the backup database can be used.

## 7.3 Web Scrapping

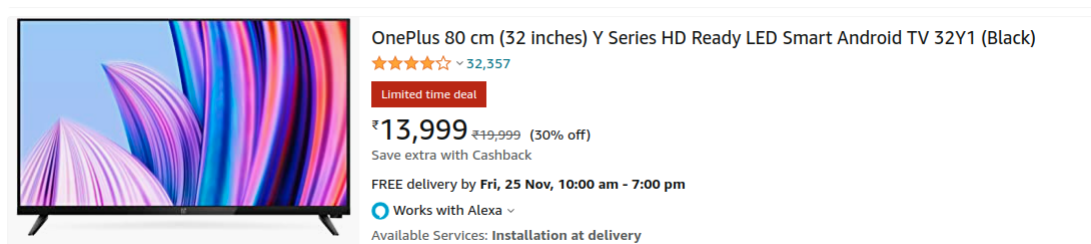
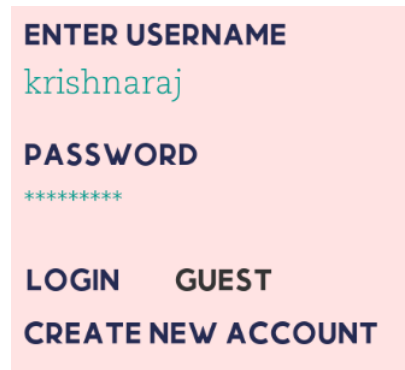


Figure 6: Product on Amazon

- Every Webpage on amazon with some product has products that look like the one above.

- The HTML page is scrapped, and the respective divs are searched in it to find each product and its price.
- It is then stored in the database after downloading and parsing the HTML file.

### 7.4 Working Login and Account Creation



ENTER USERNAME  
krishnaraj

PASSWORD  
\*\*\*\*\*

LOGIN GUEST

CREATE NEW ACCOUNT

Figure 7:

- All the Requirements of a simple login are satisfied here.
- After the user inputs the username, it is validated in the local CSV file.
- If found, the password is expected, checked and login is permitted.
- If not found, password is validated, and a new user account creation is permitted.

## 8 Color Schemes Used

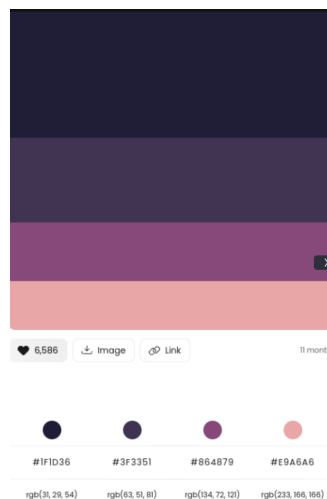


Figure 8: Color Palette

The Above Colors where used and are defined in the Colors.java.



## 9 Screenshots of the Project

### 9.1 The Login Page

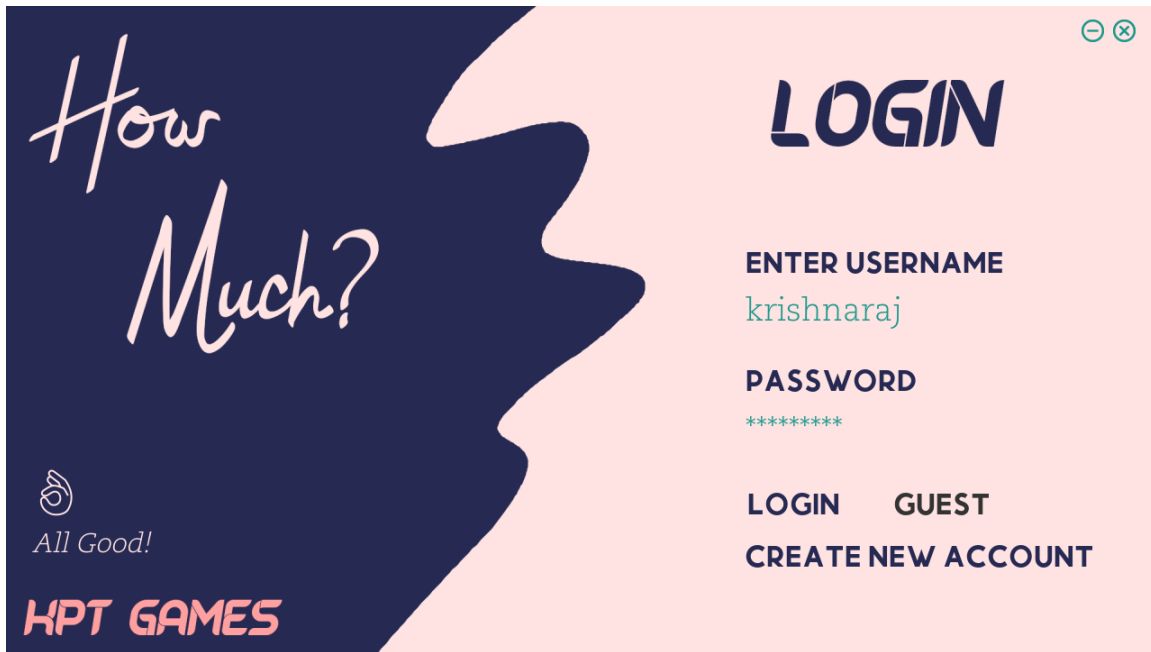


Figure 9: The Login page after a successful login

### 9.2 The Menu Screen



Figure 10:

### 9.3 The Topic Selection Screen

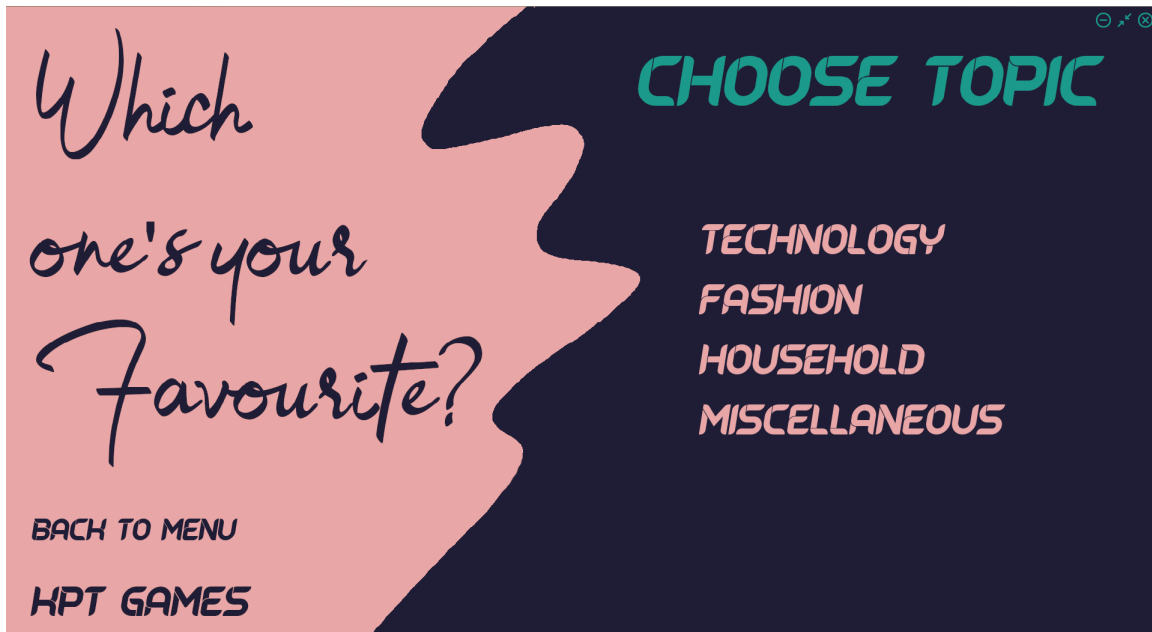


Figure 11: The Login page after a successful login

### 9.4 The Highscore Screen



Figure 12: The Login page after a successful login

## 9.5 The Help and About



Figure 13: The Login page after a successful login

## 9.6 The Game Over Screen



Figure 14: The Login page after a successful login

## 10 Walk-Through of the Files

### 10.1 Project Structure

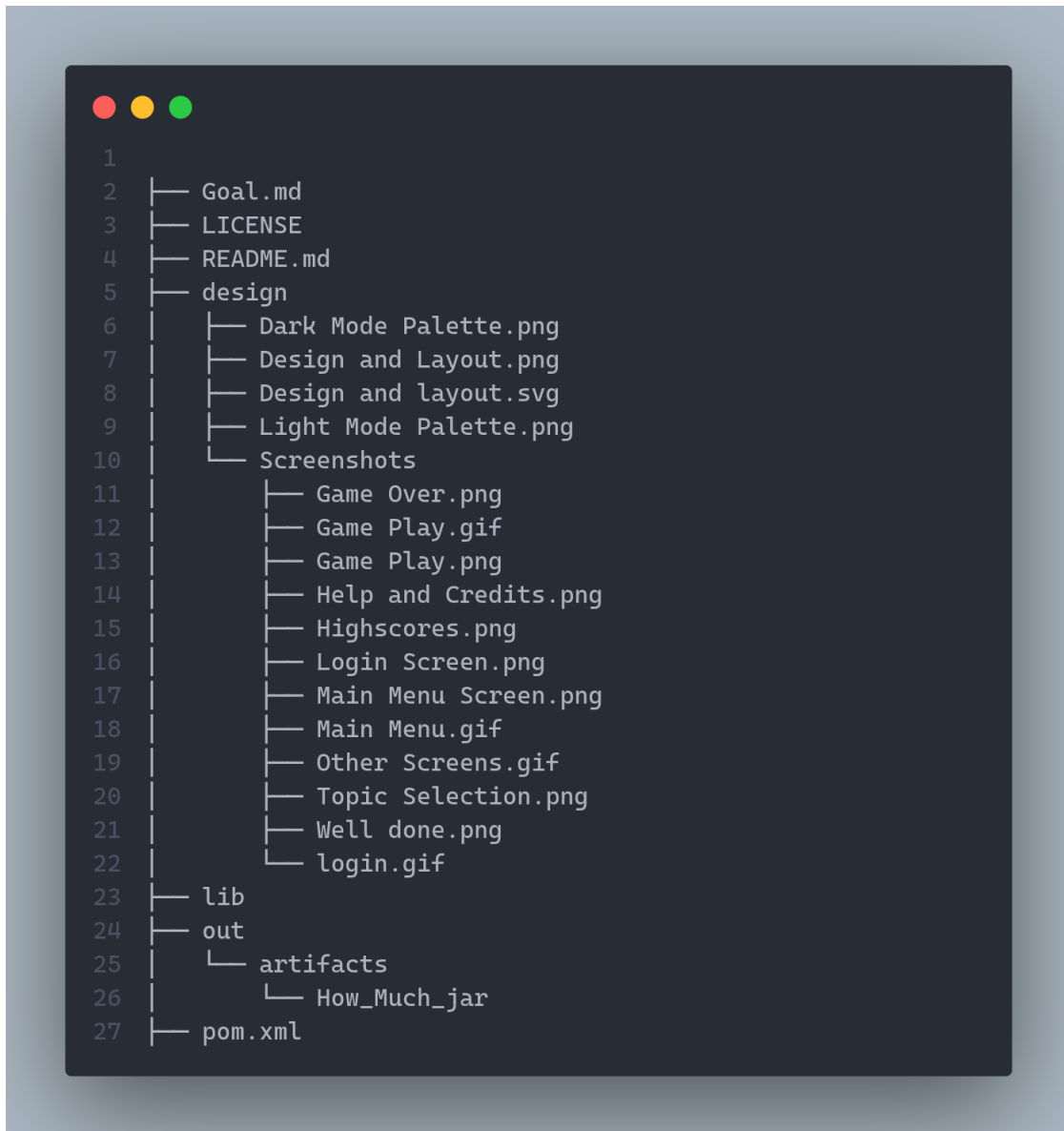


Figure 15:

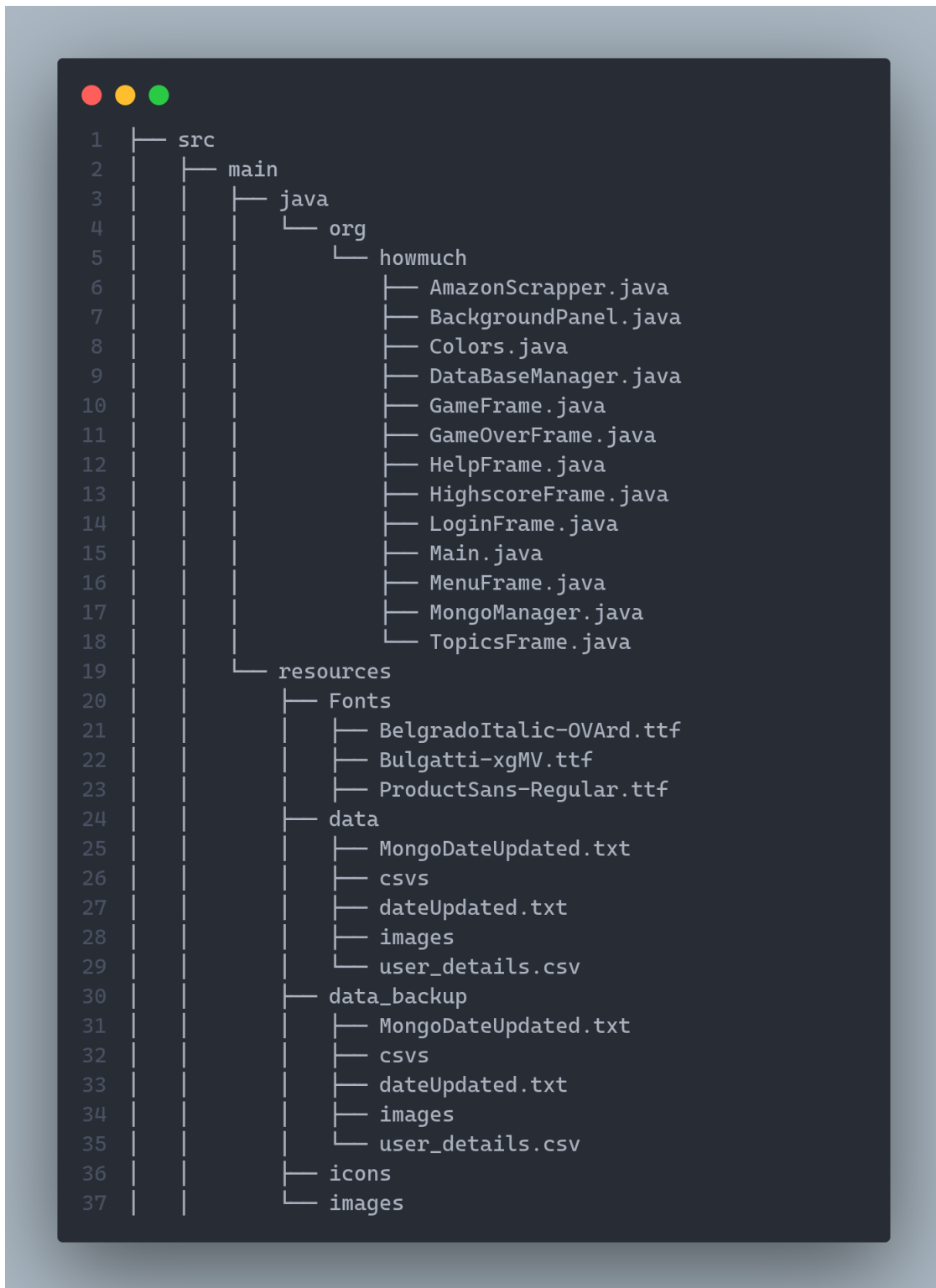


Figure 16:

### **10.2 TopicsFrame.java**

This file manages the entire topic selection screen. It has various functions regarding showing the topics on the screen. It then sets static variables defined in the Main class with respect to the selected Topic.

### **10.3 MongoManager.java**

Important Class, which manages all interactions with MongoDB. It establishes connection with it, and flips a statically defined variable in Main called usingMongo to true or false depending on the success of the connection. It also has functions to update, clear, and retrieve values to and from the Mongo Database.

### **10.4 MenuFrame.java**

This file manages the entire Menu selection screen. It has various functions regarding showing the topics on the screen. It then sets static variables defined in the Main class with respect to the selected Topic. It also has the Dark mode toggle, which flips a boolean called usingDarkMode defined in Colors.java.

### **10.5 Main.java**

This class calls all the other classes. It also has the main function. It uses multithreading to update the database at the same time as displaying the GUI. It has a function that manages the interactions between all the other classes. It also has several statically defined variables.

### **10.6 LoginFrame.java**

A Class that manages everything defined in the Login Screen. It has functions to check if the password fits the given criteria, and it queries and updates the database using functions defined in other classes.

### **10.7 HighscoreFrame.java**

A Simple class that just displays the High scores of the User. It retrieves that data using database functions, and shows the top 5 Highscoring Users along with their scores.

### **10.8 HelpFrame.java**

A Simple Class that simply displays what to do in the game, how to play and the credits.

### **10.9 GameOverFrame.java**

A Simple class that just shows the score and gives an option to the user to go back to the main menu to try again if the game was lost, or won.

### **10.10 GameFrame.java**

This is the Main class, in that it shows the actual game. It has functions to check if the databases are working properly, and what to refer in case some of them dont work. It retrieves data using functions definid in other classes. Calculates 4 suitable options depending on the Correct price, and displayes everything on the Screen.

### **10.11 DataBaseManager.java**

Important class that manages the Local CSV files. It updates, retrieves, and clears it. It also has functions to check the database for login functions, like password matching, username matching, adding username etc.

### **10.12 Colors.java**

Another important class that has all the colors defined in it. It has a function to reassign colors, which is called every time the Dark Mode switch is flipped.

### **10.13 BackgroundPanel.java**

Important class, as it is the panel that is used by all the other screens to display the background. It has a function to set the background as the Swing JPanel background by taking arguement of the location of the image to be inserted, while maintaining aspect ratio of the image.

### **10.14 AmazonScrapper.java**

A very important class, as it has functions to actually scrap the data from amazon, and parse it. It then verifies the data, checks for invalid characters, and if everything is fine it calls functions from teh database class to insert new data into the databases.

## **11 Conclusion and Topics Learnt**

A lot of topics were learnt in the process of making this Project. It was very useful to make this Game, and it got me a lot more fluent in writing Java code.

1. Multithreading was Understood in a greater depth, and implemented many times.
2. Web Scrapping was Learnt and in the process various java Libraries were used and understood.
3. Swing in java was learnt in a higher detail.
4. Database Management was understood.
5. JDBC Drivers, Connections of java with MongoDB and MySQL were learnt in detail.
6. Several Bugs were Resolved and as a Result of that programming skills were improved.
7. Designing skills were also improved.

## 12 Dependencies

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache
5     .org/xsd/maven-4.0.0.xsd">
6
7     <groupId>org.example</groupId>
8     <artifactId>How-Much</artifactId>
9     <version>1.0-SNAPSHOT</version>
10
11     <properties>
12         <maven.compiler.source>18</maven.compiler.source>
13         <maven.compiler.target>18</maven.compiler.target>
14         <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
15     </properties>
16
17     <repositories>
18         <repository>
19             <id>groupdocs-artifacts-repository</id>
20             <name>GroupDocs Artifacts Repository</name>
21             <url>https://releases.groupdocs.com/java/repo/</url>
22         </repository>
23     </repositories>
24
25     <dependencies>
26         <dependency>
27
28             <groupId>net.sourceforge.htmlunit</groupId>
29             <artifactId>htmlunit</artifactId>
30             <version>2.60.0</version>
31         </dependency>
32         <dependency>
33             <groupId>com.fasterxml.jackson.core</groupId>
34             <artifactId>jackson-databind</artifactId>
35             <version>2.13.2.1</version>
36         </dependency>
37
38         <dependency>
39             <groupId>org.mongodb</groupId>
40             <artifactId>mongo-java-driver</artifactId>
41             <version>3.12.2</version>
42         </dependency>
43
44         <dependency>
45             <groupId>com.opencsv</groupId>
46             <artifactId>opencsv</artifactId>
47             <version>4.1</version>
48         </dependency>
49
50         <dependency>
51             <groupId>com.groupdocs</groupId>
52             <artifactId>groupdocs-conversion</artifactId>
53             <version>22.8.1</version>
54         </dependency>
55     </dependencies>
```



```
56     </dependencies>
57
58 </project>
```

Listing 1: Maven Dependency File

## 13 References

- <https://www.notion.so/045f2a28abfe4a82a3ba0e8251c0e196?v=1ddbea2937324f79af4eae827f8a3132chrome://newtab/>
- <https://classroom.google.com/u/0/w/NTUxMDUyMzM4MDk4/t/all>
- <https://webscraping.pro/scraping-amazon-webdriver-java/>
- <https://www.geeksforgeeks.org/scraping-amazon-product-information-using-beautiful-soup/>
- <https://www.scrapingbee.com/blog/web-scraping-amazon/>
- [https://docs.oracle.com/cd/E50453\\_01/doc.80/e50452/run\\_java\\_guis.htm#:~:text=In%20Java%20applications%2C%20the%20components,GUI%20forms%20can%20be%20built.](https://docs.oracle.com/cd/E50453_01/doc.80/e50452/run_java_guis.htm#:~:text=In%20Java%20applications%2C%20the%20components,GUI%20forms%20can%20be%20built.)
- <https://www.javatpoint.com/multithreading-in-java>
- <https://www.digitalocean.com/community/tutorials/multithreading-in-java>
- <https://www.guru99.com/multithreading-java.html>
- <https://one.trustedstream.life/space-robot/?pl=uy-e9pFAkEqFCifo9sDBZw&sm=space-robot&hash=6IlUsQZL8I67jbxjdCa5gg&exp=1669365585>
- <https://www.mongodb.com/languages/java>
- <https://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html#:~:text=The%20Action%20interface%20provides%20a,be%20accessed%20by%20several%20controls.&text=Defines%20the%20data%20model%20used,Slider%20s%20and%20ProgressBar%20s.>
- Credits to Teachers and Friends for their constant help and support.

## 14 Code Files

```
1 package org.howmuch;
2
3 import java.awt.*;
4
5 public class Colors {
6
7     static Boolean DarkMode = false;
8
9     // Light mode Colors
10    static Color light_bgColor = new Color(255, 227, 227);
11    static Color light_primaryColor = new Color(38, 42, 83);
12    static Color light_secondaryColor = new Color(255, 160, 160);
13    static Color light_accentColor = new Color(27, 153, 139);
14}
```

```
15 // Dark Mode Colors
16 static Color dark_bg_color = new Color(31, 29, 54);
17 static Color dark_primaryColor = new Color(233, 166, 166);
18 static Color dark_secondaryColor = new Color(175, 89, 159);
19 static Color dark_accentColor = new Color(27, 153, 139);
20
21 static Color bgColor = DarkMode ? dark_bg_color : light_bgColor;
22 static Color primaryColor = DarkMode ? dark_primaryColor : light_primaryColor;
23 static Color secondaryColor = DarkMode ? dark_secondaryColor :
light_secondaryColor;
24 static Color accentColor = DarkMode ? dark_accentColor : light_accentColor;
25
26 public static void reassignColors() {
27     bgColor = DarkMode ? dark_bg_color : light_bgColor;
28     primaryColor = DarkMode ? dark_primaryColor : light_primaryColor;
29     secondaryColor = DarkMode ? dark_secondaryColor : light_secondaryColor;
30     accentColor = DarkMode ? dark_accentColor : light_accentColor;
31 }
32 }
```

Listing 2: Main Java File

```
1 package org.howmuch;
2
3 import javax.swing.*;
4 import java.awt.*;
5
6 public class BackgroundPanel extends JPanel {
7     private Image background;
8
9     public void paintComponent(Graphics g) {
10
11         super.paintComponent(g);
12         int width = this.getSize().width;
13         int height = this.getSize().height;
14
15         if (this.background != null) {
16             // Add the size of the window in drawImage method()
17             g.drawImage(this.background, 0, 0, width, height, null);
18         }
19     }
20
21     public void setBackground(String imagePath) {
22         // Simply sets the background as the one that you have provided. It needs
to be
23         // a png file (I think)
24         this.background = new ImageIcon(imagePath).getImage();
25         repaint();
26     }
27
28 }
```

Listing 3: Main Java file

```
1 package org.howmuch;
2
3 import org.xml.sax.SAXException;
4 import javax.swing.*;
5 import javax.xml.parsers.ParserConfigurationException;
```

```
6 import java.awt.*;
7 import java.io.*;
8 import java.time.LocalDate;
9
10 // You have to extend the thread class to create a new thread for running the
    databases.
11 public class Main extends Thread {
12
13     // Statically defining important variables used throughout the game. They are
14     // statically defined coz they are used by other classes very often.
15     public static String[] Topics = new String[] { "Technology", "Fashion", "
Household", "Miscellaneous" };
16     public static String currentTopic = Topics[0];
17     static final int WIDTH = 1280, HEIGHT = 720;
18     static boolean maximized = false, isGuest = true, grantAccess = false,
isLocalDatabaseUpToDate = false,
19         isMongoUpToDate = false, usingMongo = false;
20
21     // Declaring Objects of other classes that we are going to call from main.
22     static LoginFrame loginFrame;
23     static MenuFrame menuFrame;
24     static HelpFrame helpFrame;
25     static HighscoreFrame highscoreFrame;
26     static TopicsFrame topicsFrame;
27     static GameFrame gameFrame;
28     static GameOverFrame gameOverFrame;
29
30     static Font buttonFont, textFont, password_font, options_font, emoji_font;
31     static JButton exit_btn, resize_btn, minimize_btn;
32     static JPanel basicButtons_pnl;
33
34     // These are the icons from where we get the resize, exit and the minimize
35     // button. They are custom made coz they look better,
36     // eliminate the need for the titlebar making the UI look cleaner, albeit less
37     // useful.
38     // They also let you have full control over what you want to do when they are
39     // pressed, and what you wanna call, which you cant do without them.
40     // You can also control now exactly the resizing behaviour of your software.
41     static ImageIcon exit = new ImageIcon("src/main/resources/icons/circle_delete.
png");
42     static Image exit_image = exit.getImage().getScaledInstance(25, 25, Image.
SCALE_SMOOTH);
43     static ImageIcon minimize = new ImageIcon("src/main/resources/icons/
circle_minus.png");
44     static Image minimize_image = minimize.getImage().getScaledInstance(25, 25,
Image.SCALE_SMOOTH);
45     static ImageIcon resizeUp = new ImageIcon("src/main/resources/icons/resize_3.
png");
46     static Image resizeUp_image = resizeUp.getImage().getScaledInstance(25, 25,
Image.SCALE_SMOOTH);
47     static ImageIcon resizeDown = new ImageIcon("src/main/resources/icons/resize_4
.png");
48     static Image resizeDown_image = resizeDown.getImage().getScaledInstance(25,
25, Image.SCALE_SMOOTH);
49
50     /**
51      * Creates fonts by instantiating the font objects with their respective fonts
52      * stored locally. Static and used everywhere. Its an important function and
53      * gets called in almost every class constructor.
```

```
54     */
55     public static void createFonts() {
56         try {
57             GraphicsEnvironment ge = GraphicsEnvironment.
getLocalGraphicsEnvironment();
58
59             // Used for Buttons Almost everywhere.
60             buttonFont = Font
61                 .createFont(Font.TRUETYPE_FONT, new File("/run/media/
krishnaraj/Programs/Java/How Much/src/main/resources/Fonts/BelgradoItalic-OVard
.ttf"))
62                 .deriveFont(50f);
63             // Used Mostly on the Login Page.
64             textFont = Font.createFont(Font.TRUETYPE_FONT, new File("/run/media/
krishnaraj/Programs/Java/How Much/src/main/resources/Fonts/MomcakeBold-WyonA.
otf"))
65                 .deriveFont(50f);
66             // Used for password Entering
67             password_font = Font
68                 .createFont(Font.TRUETYPE_FONT, new File("/run/media/
krishnaraj/Programs/Java/How Much/src/main/resources/Fonts/CaeciliaLTPro45Light
.TTF"))
69                 .deriveFont(35f);
70             // Used only for Emojis
71             emoji_font = Font.createFont(Font.TRUETYPE_FONT,
72                 new File("/run/media/krishnaraj/Programs/Java/How Much/src/
main/resources/Fonts/NotoEmoji-VariableFont_wght.ttf")).deriveFont(35f);
73             // Used to show the Price, needs to contain the Rupee symbol
74             options_font = Font
75                 .createFont(Font.TRUETYPE_FONT, new File("/run/media/
krishnaraj/Programs/Java/How Much/src/main/resources/Fonts/ProductSans-Regular.
ttf"))
76                 .deriveFont(35f);
77
78             // registering them locally, not required.
79             ge.registerFont(textFont);
80             ge.registerFont(buttonFont);
81             ge.registerFont(password_font);
82             ge.registerFont(emoji_font);
83             ge.registerFont(options_font);
84
85             } catch (FontFormatException | IOException e) {
86                 e.printStackTrace();
87                 System.out.println("Couldnt create the fonts. ");
88             }
89         }
90
91         /*
92         * Function to Create the resize, minimize and the exit button, they are all
93         * placed in a panel, so that you can move them around easily without the
hassle
94         * of moving each thing. Just move the panel. Here we define them.
95         */
96         public static void createBasicButtonPanel() {
97             basicButtons_pnl = new JPanel();
98             FlowLayout fl = new FlowLayout(FlowLayout.LEFT, 10, 0);
99             basicButtons_pnl.setLayout(fl);
100
101             exit_btn = new JButton();
```

```
102     exit_btn.setIcon(new ImageIcon(exit_image));
103     exit_btn.setAlignmentY(Box.CENTER_ALIGNMENT);
104     exit_btn.setAlignmentX(Box.CENTER_ALIGNMENT);
105     exit_btn.setBounds(new Rectangle(25, 25));
106     exit_btn.setFont(buttonFont.deriveFont(44f));
107     exit_btn.setFocusPainted(false);
108     exit_btn.setContentAreaFilled(false);
109     exit_btn.setOpaque(true);
110     exit_btn.setBorder(null);
111
112     resize_btn = new JButton();
113     if (Main.maximized) {
114         resize_btn.setIcon(new ImageIcon(resizeDown_image));
115     } else {
116         resize_btn.setIcon(new ImageIcon(resizeUp_image));
117     }
118     resize_btn.setAlignmentY(Box.CENTER_ALIGNMENT);
119     resize_btn.setAlignmentX(Box.CENTER_ALIGNMENT);
120     resize_btn.setBounds(new Rectangle(25, 25));
121     resize_btn.setFont(buttonFont.deriveFont(44f));
122     resize_btn.setFocusPainted(false);
123     resize_btn.setContentAreaFilled(false);
124     resize_btn.setOpaque(true);
125     resize_btn.setBorder(null);
126
127     minimize_btn = new JButton();
128     minimize_btn.setIcon(new ImageIcon(minimize_image));
129     minimize_btn.setAlignmentY(Box.CENTER_ALIGNMENT);
130     minimize_btn.setAlignmentX(Box.CENTER_ALIGNMENT);
131     minimize_btn.setBounds(new Rectangle(25, 25));
132     minimize_btn.setFont(buttonFont.deriveFont(44f));
133     minimize_btn.setFocusPainted(false);
134     minimize_btn.setContentAreaFilled(false);
135     minimize_btn.setOpaque(true);
136     minimize_btn.setBorder(null);
137
138     // Adding them to the panel here.
139     basicButtons_pnl.add(minimize_btn);
140     basicButtons_pnl.add(resize_btn);
141     basicButtons_pnl.add(exit_btn);
142 }
143
144 /**
145  * @param status = 1: Call Main Menu <br>
146  *                status = 2: Call Topic Selection<br>
147  *                status = 3: Call Help and Credits<br>
148  *                status = 4: View Highscores<br>
149  *                status = 5: Update Database<br>
150  *                status = 6: Start Game<br>
151  *                status = 7: Game over Screen<br>
152  *                status = 0: Exit Game<br>
153  *
154  *                Important Function as it decides to change to another frame,
155  *                provides some security with grantedAccess boolean,
156  *                and Also does the mandatory things that need to be done if
157  *                the
158  *                close button is pressed.
159  */
160 public static void changeFrame(int status) {
```

```
160     // Status is 0 when you wanna quit, so we gotta do some stuff before you
quit,
161     // like creating the backup.
162     if (status == 0) {
163
164         // Create a local backup of the users file irrespective of what was
done during
165         // gameplay.
166         DataBaseManager.createLocalDatabaseBackupOfUsers();
167
168         // If the user is a guest, then the index is less than 0, in which
case dont
169         // update anything.
170         if (DataBaseManager.USER_INDEX < 0) {
171             System.out.println("You are a guest, so not updating anything. \n"
);
172         } else {
173             // If its a user, then update the user score. The index is known
already in a
174             // static variable.
175             DataBaseManager.updateUserScore();
176         }
177
178         // This is what keeps track of when the last time was that the
database was
179         // updated. You dont need to update it every time you run the game.
String lastUpdateDate = "";
180
181         // Opening the Backup Date file and checking the last time it was
backed up.
182         File dateFile = new File(DataBaseManager.LOCAL_BACKUP_DATEFILE);
183         if (dateFile.exists()) {
184             try (BufferedReader br = new BufferedReader(new FileReader(
dateFile))) {
185                 lastUpdateDate = br.readLine();
186
187                 // If the database was backed up last today, then dont do it.
188                 if (lastUpdateDate.equals(String.valueOf(LocalDate.now()))) {
189                     System.out.println("Backup DataBases are Up to Date!");
190                 } else {
191                     // Else update it.
192                     DataBaseManager.createLocalDatabaseBackup();
193                 }
194             } catch (IOException e) {
195                 throw new RuntimeException(e);
196             } catch (NullPointerException exception) {
197                 System.out.println("Nothing in the Date File. ");
198             }
199         } else {
200             // If the file itself doesnt exist, then clearly there doesnt
exist any backup,
201             // so we better back up at that point.
202             try {
203                 DataBaseManager.createLocalDatabaseBackup();
204             } catch (Exception e) {
205                 System.out.println("You havent really created the database yet
, so not creating backup either. ");
206             }
207         }
208     }
```

```
209         // Exit game
210         System.out.println("Thanks for Playing! ");
211         System.exit(0);
212     }
213     if (grantAccess) {
214         System.out.println("Access Granted!");
215         switch (status) {
216             case 1 -> {
217                 // Showing Main Menu
218                 grantAccess = false;
219                 menuFrame = new MenuFrame();
220             }
221             case 2 -> {
222                 // Showing the TopicsFrame
223                 grantAccess = false;
224                 topicsFrame = new TopicsFrame();
225             }
226             case 3 -> {
227                 // Showing the Help Screen
228                 grantAccess = false;
229                 helpFrame = new HelpFrame();
230             }
231             case 4 -> {
232                 // Showing Highscores
233                 grantAccess = false;
234                 highscoreFrame = new HighscoreFrame();
235             }
236             case 5 -> {
237                 System.out.println("Updating Database");
238
239                 // Instead of overwriting the files, or appending to them, as
they contain old
240                 // data,
241                 // we will just erase them altogether and create them again.
242                 DataBaseManager.clearLocalDatabase();
243                 MongoManager.clearMongoDb();
244
245                 // Scrap everything and Start Saving
246                 AmazonScrapper obj = new AmazonScrapper();
247                 try {
248                     AmazonScrapper.scrapAndSave();
249                 } catch (Exception e) {
250                     System.out.println("Couldnt update the database, there was
some problem. It was");
251                     System.out.println(e.getMessage());
252                 }
253                 // Just copy everything to the backup either way.
254                 DataBaseManager.createLocalDatabaseBackup();
255
256                 File dateFile;
257
258                 // Updating the Mongo and Local Database File.
259                 dateFile = new File(DataBaseManager.LOCAL_DATEFILE);
260                 try (FileWriter f = new FileWriter(dateFile, false)) {
261                     f.write(String.valueOf(LocalDate.now()));
262                 } catch (IOException e) {
263                     throw new RuntimeException(e);
264                 }
265                 dateFile = new File(DataBaseManager.LOCAL_MONGODATEFILE);
```

```
266         try (FileWriter f = new FileWriter(dateFile, false)) {
267             f.write(String.valueOf(LocalDate.now()));
268         } catch (IOException e) {
269             throw new RuntimeException(e);
270         }
271     }
272     case 6 -> {
273         // Showing Game Screen
274         grantAccess = false;
275         gameFrame = new GameFrame();
276     }
277     case 7 -> {
278         // This is only called by the gameframe, which has a timer,
which is what calls
279         // this function, and as its in a different class,
280         // you have to close the things from here coz that timer cant
access its parent
281         // class properties.
282         gameFrame.setVisible(false);
283         gameFrame.dispose();
284
285         // Show GameOverScreen
286         gameOverFrame = new GameOverFrame();
287     }
288     default -> {
289         // In Case something goes really wrong, just backup and exit.
290         DataBaseManager.createLocalDatabaseBackup();
291
292         // Exit game
293         System.out.println("Thanks for Playing! ");
294         System.exit(0);
295     }
296 }
297 } else {
298     System.out.println("Access Denied Who are you? What are you trynna do
here? ");
299     System.exit(0);
300 }
301 }
302
303 /*
304  * This function is overridden from the Thread class, coz its empty there, and
305  * thread.start calls this function.
306  * And this is where you put loops or something in case you wanna do something
307  * for ever as a game Loop and access data members stored somewhere else and
308  * written to by some other classes.
309  * The Job of this function here is important in that its the first function
310  * that is real multithread. It checks the database, and if they are not up to
311  * date, it updates them.
312  */
313 public void run() {
314
315     // Just establish the connection, and if thats not possible, then we
clearly
316     // arent gonna be using mongo.
317     usingMongo = MongoManager.establishConnectionWithMongo();
318
319     // Same logic as demod in changeFrame()
320     String lastUpdateDate = "";
```



```
321
322 // Checking the Local CSV Files
323 File dateFile = new File(DataBaseManager.LOCAL_DATEFILE);
324 if (dateFile.exists()) {
325     try (BufferedReader br = new BufferedReader(new FileReader(dateFile)))
326     {
327         lastUpdateDate = br.readLine();
328         System.out.println(lastUpdateDate);
329         if (lastUpdateDate.equals(String.valueOf(LocalDate.now()))) {
330             System.out.println("Local DataBases are Up to Date!");
331             isLocalDatabaseUpToDate = true;
332         }
333     } catch (IOException e) {
334         throw new RuntimeException(e);
335     } catch (NullPointerException exception) {
336         System.out.println("Nothing in the Local Date File. ");
337     }
338 }
339 // Now check the mongodb database date file to check when was the last
340 time it
341 // was updated. Same Logic tho.
342 dateFile = new File(DataBaseManager.LOCAL_MONGODATEFILE);
343 if (dateFile.exists()) {
344     try (BufferedReader br = new BufferedReader(new FileReader(dateFile)))
345     {
346         lastUpdateDate = br.readLine();
347         System.out.println(lastUpdateDate);
348         if (lastUpdateDate.equals(String.valueOf(LocalDate.now()))) {
349             System.out.println("Mongo DataBases are Up to Date!");
350             isMongoUpToDate = true;
351         }
352     } catch (IOException e) {
353         throw new RuntimeException(e);
354     } catch (NullPointerException exception) {
355         System.out.println("Nothing in the mongo Date File. ");
356     }
357 }
358 // If say one of them is not updated, then we gotta scrap amazon.
359 if (!isLocalDatabaseUpToDate || (usingMongo && !isMongoUpToDate)) {
360     System.out.println("Beginning to Scrap Data From Amazon, as one of the
361     DataBases isnt updated. ");
362     if (!isLocalDatabaseUpToDate) {
363         // As an edge case, if mongo isnt up to date, we dont wanna clear
364         the local one.
365         DataBaseManager.clearLocalDatabase();
366     }
367     if (usingMongo && !isMongoUpToDate) {
368         // If the local one isnt up to date we dont wanna clear mongo.
369         MongoManager.clearMongoDb();
370     }
371     // Scrap and save, as at this point we already know what works and
372     what doesnt,
373     // and what is updated and what isnt,
374     // AmazonScrapper class can figure out where to save stuff. After that
375     // everything would have to be updated.
```

```
374     AmazonScrapper obj = new AmazonScrapper();
375     try {
376         AmazonScrapper.scrapAndSave();
377         isLocalDatabaseUpToDate = true;
378
379         // writing to the date file coz we must have updated at this point
380         dateFile = new File(DataBaseManager.LOCAL_DATEFILE);
381         try (FileWriter f = new FileWriter(dateFile, false)) {
382             f.write(String.valueOf(LocalDate.now()));
383         } catch (IOException e) {
384             throw new RuntimeException(e);
385         }
386         System.out.println("Updated the local database, no need to depend
on the backup anymore");
387
388         if (usingMongo) {
389             // Coz at this point it has to be, as we just scrapped and
didn't get any errors.
390             isMongoUpToDate = true;
391
392             // writing to the date file coz we must have updated at this
point
393             dateFile = new File(DataBaseManager.LOCAL_MONGODATEFILE);
394             try (FileWriter f = new FileWriter(dateFile, false)) {
395                 f.write(String.valueOf(LocalDate.now()));
396             } catch (IOException e) {
397                 throw new RuntimeException(e);
398             }
399             System.out.println("Updated the Mongo database, no need to
depend on the local one anymore");
400         }
401         } catch (Exception e) {
402             System.out.print("Couldnt update one of the databases, in the case
that one of them wasn't updated. ");
403             System.out.println(e.getMessage());
404         }
405
406         // This has to happen at this point as a forced minimum.
407         isLocalDatabaseUpToDate = true;
408     }
409 }
410
411 public static void main(String[] args) {
412
413     // This is so that the fonts are rendered correctly in Swing gui.
414     System.setProperty("awt.useSystemAAFontSettings", "on");
415     System.setProperty("swing.aatext", "true");
416
417     // This is to call the thread, so we can check the databases.
418     Main t1 = new Main();
419     t1.start();
420
421     // As the thread starts, we start the game. Usually it has to read from
the
422     // backup file if the database isn't updated yet. After which it would
start
423     // reading from there. As downloading the images and putting them in the
424     // database takes time and we can't wait that long, that job is
multithreaded.
```

```
425         // The use of the backup database is :
426         // 1. It has some basic images that are shipped with the jar file so in
         case
427         // someone doesnt have internet, atleast they have something.
428         // 2. It is the fallback in case something goes wrong while doing or
         reading
429         // something from one of the files.
430         // 3. It serves as the Primary database when we are updating the local
         database,
431         // and we still need to show stuff to the user so they can play the game.
         This
432         // is the most important one.
433         loginFrame = new LoginFrame();
434     }
435 }
```

Listing 4: Main Java file

```
1
2  /*
3   * This is the loginFrame file, which is one of the first classes that comes into
         picture, pun intended.
4   * It does everything it can to make it look like the login screen of a modern
         website like google. Even goes as far as to use the same fonts.
5   * It checks the username and the password entered by the user, and matches it
         with the csv file it has. And reports the situation on screen.
6   * Once the user is justified and logged in, or has created a new account, it adds
         them, assigns some basic variables, and then Calls the Main Menu class by
         calling the changeFrame function in the Main class.
7   */
8
9  package org.howmuch;
10
11  import javax.swing.*;
12  import java.awt.*;
13  import java.awt.event.ActionEvent;
14  import java.awt.event.ActionListener;
15  import java.util.Arrays;
16
17  // Basically importing every static thing from Main coz its used so often
18  // not a very good practice.
19  import static org.howmuch.Main.*;
20
21  public class LoginFrame extends JFrame implements Runnable {
22
23      public static boolean running = true, userExists = false, incorrectPassword =
         false, newUser = false;
24      JLabel username_lbl, password_lbl, background_lbl, status_lbl,
         status_emoji_lbl;
25      JButton login_btn, guest_btn, newAccount_btn, exit_btn, resize_btn,
         minimize_btn;
26      JTextField username_txt_fld;
27      JPasswordField password_txt_fld;
28      Thread loginThread;
29
30      /*
31       * This is the standard implmentation of a constructor in this game. There are
32       * some basic attributes of the Frame calsses that it extends from
33       * And then sets them. It then creates the things you are supposed to create
```

```
in
34  * the GUI, and then adds them to the frame. This could be done in a panel,
yes,
35  * but then you couldnt use some things like the ComponentListener class that
36  * only listens to the Frame mainly, and that helps in calling certain
functions
37  * when you resize the screen.
38  */
39  LoginFrame() {
40      this.setTitle("How Much? ");
41      this.setResizable(false);
42      this.setUndecorated(true);
43      this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
44
45      createFonts();
46      createButtons();
47      createLabels();
48      createTextFields();
49
50      this.add(status_lbl);
51      this.add(status_emoji_lbl);
52      this.add(login_btn);
53      this.add(username_lbl);
54      this.add(password_lbl);
55      this.add(guest_btn);
56      this.add(newAccount_btn);
57      this.add(exit_btn);
58      this.add(minimize_btn);
59      this.add(username_txt_fld);
60      this.add(password_txt_fld);
61      this.add(background_lbl);
62
63      // Thread to check the password entered by the user every 2 seconds is
invoked
64      // by this thread's start method.
65      startThread();
66
67      this.pack();
68      this.setVisible(true);
69      this.setLocationRelativeTo(null); // put in the center.
70  }
71
72  /*
73  * Standard function to create buttons and assign their attributes. As some
74  * lines are dupliated, they certainly can be extracted as separete methods
75  * themselves.
76  */
77  public void createButtons() {
78      login_btn = new JButton();
79      login_btn.setText("Login");
80      login_btn.setAlignmentY(Box.CENTER_ALIGNMENT);
81      login_btn.setAlignmentX(Box.CENTER_ALIGNMENT);
82      login_btn.setBounds(775, 532, 200, 40);
83      login_btn.setFont(textFont.deriveFont(44f));
84      login_btn.setFocusPainted(false);
85      login_btn.setContentAreaFilled(false);
86      login_btn.setOpaque(true);
87      login_btn.setBorder(null);
88
```

```
89     login_btn.setBackground(Colors.bgColor);
90
91     // To control what happens when the mouse interacts with this button.
92     login_btn.addChangeListener(evt -> {
93         if (login_btn.getModel().isPressed()) {
94             login_btn.setForeground(Colors.primaryColor);
95         } else if (login_btn.getModel().isRollover()) {
96             login_btn.setForeground(Colors.secondaryColor);
97         } else {
98             login_btn.setForeground(Colors.primaryColor);
99         }
100     });
101
102     // Stuff to do when it is pressed. Applicable to all buttons here.
103     login_btn.addActionListener(e -> {
104         if (DataBaseManager.doesUsernameExist(username_txt_fld.getText())) {
105
106             // Now the user is trying to login, so we check if the password
107             matches
108             // if it does, then we assign some basic variables, and close this
109             screen, open
110             // the menu screen.
111             if (DataBaseManager.doesPasswordMatch(username_txt_fld.getText(),
112                 String.valueOf(password_txt_fld.getPassword()))) {
113                 DataBaseManager.currentUsername = username_txt_fld.getText();
114                 DataBaseManager.currentPassword = String.valueOf(
115                 password_txt_fld.getPassword());
116                 this.setVisible(false);
117                 this.dispose();
118                 running = false;
119                 grantAccess = true;
120                 Main.changeFrame(1);
121             } else {
122                 grantAccess = false;
123                 incorrectPassword = true;
124             }
125         } else {
126             // If the user has entered a username and a password, and he
127             doesnt exist, then
128             // clearly is a new user.
129             newUser = true;
130         }
131     });
132     login_btn.setEnabled(false);
133
134     guest_btn = new JButton();
135     guest_btn.setText("Guest");
136     guest_btn.setAlignmentY(Box.CENTER_ALIGNMENT);
137     guest_btn.setAlignmentX(Box.CENTER_ALIGNMENT);
138     guest_btn.setBounds(940, 532, 200, 40);
139     guest_btn.setFont(textFont.deriveFont(44f));
140     guest_btn.setFocusPainted(false);
141     guest_btn.setContentAreaFilled(false);
142     guest_btn.setOpaque(true);
143     guest_btn.setBorder(null);
144     guest_btn.setBackground(Colors.bgColor);
145     guest_btn.addChangeListener(evt -> {
146         if (guest_btn.getModel().isPressed()) {
```

```
144         guest_btn.setForeground(Colors.primaryColor);
145     } else if (guest_btn.getModel().isRollover()) {
146         guest_btn.setForeground(Colors.secondaryColor);
147     } else {
148         guest_btn.setForeground(Colors.primaryColor);
149     }
150 });
151 guest_btn.addActionListener(e -> {
152     Main.isGuest = true;
153     DataBaseManager.currentPassword = "guest";
154     DataBaseManager.currentUsername = "guest";
155     this.setVisible(false);
156     this.dispose();
157     running = false;
158     grantAccess = true;
159     Main.changeFrame(1);
160 });
161
162 newAccount_btn = new JButton();
163 newAccount_btn.setText("Create New Account");
164 newAccount_btn.setAlignmentY(Box.CENTER_ALIGNMENT);
165 newAccount_btn.setAlignmentX(Box.CENTER_ALIGNMENT);
166 newAccount_btn.setBounds(615, 590, 800, 40);
167 newAccount_btn.setFont(textFont.deriveFont(44f));
168 newAccount_btn.setFocusPainted(false);
169 newAccount_btn.setContentAreaFilled(false);
170 newAccount_btn.setOpaque(true);
171 newAccount_btn.setBorder(null);
172 newAccount_btn.setBackground(Colors.bgColor);
173 newAccount_btn.addChangeListener(evt -> {
174     if (newAccount_btn.getModel().isPressed()) {
175         newAccount_btn.setForeground(Colors.primaryColor);
176     } else if (newAccount_btn.getModel().isRollover()) {
177         newAccount_btn.setForeground(Colors.secondaryColor);
178     } else {
179         newAccount_btn.setForeground(Colors.primaryColor);
180     }
181 });
182
183 newAccount_btn.addActionListener(e -> {
184     Main.isGuest = false;
185     if (!DataBaseManager.doesUsernameExist(username_txt_fld.getText())) {
186         DataBaseManager.currentUsername = username_txt_fld.getText();
187         DataBaseManager.currentPassword = String.valueOf(password_txt_fld.
getPassword());
188         DataBaseManager.currentScore = 0;
189         DataBaseManager.addNewUser();
190         running = false;
191         this.setVisible(false);
192         this.dispose();
193         grantAccess = true;
194         Main.changeFrame(1);
195     } else {
196         System.out.println("User Already Exists");
197         userExists = true;
198     }
199 });
200 newAccount_btn.setEnabled(false);
201
```

```
202     ImageIcon exit = new ImageIcon("/run/media/krishnaraj/Programs/Java/How
Much/src/main/resources/icons/circle_delete.png");
203     Image exit_image = exit.getImage().getScaledInstance(25, 25, Image.
SCALE_SMOOTH);
204     ImageIcon minimize = new ImageIcon("/run/media/krishnaraj/Programs/Java/
How Much/src/main/resources/icons/circle_minus.png");
205     Image minimize_image = minimize.getImage().getScaledInstance(25, 25, Image
.SCALE_SMOOTH);
206     ImageIcon resize = new ImageIcon("/run/media/krishnaraj/Programs/Java/How
Much/src/main/resources/icons/screen_expand_3.png");
207     Image resize_image = resize.getImage().getScaledInstance(25, 25, Image.
SCALE_SMOOTH);

208
209     exit_btn = new JButton();
210     // exit_btn.setText("-");
211     exit_btn.setIcon(new ImageIcon(exit_image));
212     exit_btn.setAlignmentY(Box.CENTER_ALIGNMENT);
213     exit_btn.setAlignmentX(Box.CENTER_ALIGNMENT);
214     exit_btn.setBounds(1230, 15, 25, 25);
215     exit_btn.setFont(textFont.deriveFont(44f));
216     exit_btn.setFocusPainted(false);
217     exit_btn.setContentAreaFilled(false);
218     exit_btn.setOpaque(true);
219     exit_btn.setBorder(null);
220     exit_btn.setBackground(Colors.bgColor);
221     exit_btn.addChangeListener(evt -> {
222         if (exit_btn.getModel().isPressed()) {
223             exit_btn.setForeground(Colors.primaryColor);
224             this.setVisible(false);
225             this.dispose();
226             running = false;
227             Main.changeFrame(0);
228         } else if (exit_btn.getModel().isRollover()) {
229             exit_btn.setForeground(Colors.secondaryColor);
230         } else {
231             exit_btn.setForeground(Colors.primaryColor);
232         }
233     });
234
235     resize_btn = new JButton();
236     resize_btn.setIcon(new ImageIcon(resize_image));
237     resize_btn.setAlignmentY(Box.CENTER_ALIGNMENT);
238     resize_btn.setAlignmentX(Box.CENTER_ALIGNMENT);
239     resize_btn.setBounds(1195, 15, 25, 25);
240     resize_btn.setFont(textFont.deriveFont(44f));
241     resize_btn.setFocusPainted(false);
242     resize_btn.setContentAreaFilled(false);
243     resize_btn.setOpaque(true);
244     resize_btn.setBorder(null);
245     resize_btn.setBackground(Colors.bgColor);
246     resize_btn.addChangeListener(evt -> {
247         if (exit_btn.getModel().isPressed()) {
248             this.setExtendedState(JFrame.MAXIMIZED_BOTH);
249             exit_btn.setForeground(Colors.primaryColor);
250         } else if (exit_btn.getModel().isRollover()) {
251             exit_btn.setForeground(Colors.secondaryColor);
252         } else {
253             exit_btn.setForeground(Colors.primaryColor);
254         }
255     });
```

```
255     });
256
257     minimize_btn = new JButton();
258     // minimize_btn.setText("-");
259     minimize_btn.setIcon(new ImageIcon(minimize_image));
260     minimize_btn.setAlignmentY(Box.CENTER_ALIGNMENT);
261     minimize_btn.setAlignmentX(Box.CENTER_ALIGNMENT);
262     minimize_btn.setBounds(1195, 15, 25, 25);
263     minimize_btn.setFont(textFont.deriveFont(44f));
264     minimize_btn.setFocusPainted(false);
265     minimize_btn.setContentAreaFilled(false);
266     minimize_btn.setOpaque(true);
267     minimize_btn.setBorder(null);
268     minimize_btn.setBackground(Colors.bgColor);
269     minimize_btn.addChangeListener(evt -> {
270         if (minimize_btn.getModel().isPressed()) {
271             this.setState(JFrame.ICONIFIED);
272             minimize_btn.setForeground(Colors.primaryColor);
273         } else if (minimize_btn.getModel().isRollover()) {
274             minimize_btn.setForeground(Colors.secondaryColor);
275         } else {
276             minimize_btn.setForeground(Colors.primaryColor);
277         }
278     });
279
280 }
281
282 /*
283  * Standard function to create labels used in this frame.
284  */
285 public void createLabels() {
286     ImageIcon icon = new ImageIcon("src/main/resources/images/Login_bg.png");
287     Image bg_image = icon.getImage().getScaledInstance(1280, 720, Image.
SCALE_SMOOTH);
288
289     background_lbl = new JLabel();
290     background_lbl.setIcon(new ImageIcon(bg_image));
291
292     username_lbl = new JLabel();
293     username_lbl.setText("Enter Username");
294     username_lbl.setFont(textFont.deriveFont(44f));
295     username_lbl.setAlignmentX(Box.CENTER_ALIGNMENT);
296     username_lbl.setBounds(822, 244, 800, 80);
297     username_lbl.setForeground(Colors.primaryColor);
298
299     password_lbl = new JLabel();
300     password_lbl.setText("Password");
301     password_lbl.setFont(textFont.deriveFont(44f));
302     password_lbl.setAlignmentX(Box.CENTER_ALIGNMENT);
303     password_lbl.setBounds(822, 340, 800, 150);
304     password_lbl.setForeground(Colors.primaryColor);
305
306     status_lbl = new JLabel();
307     status_lbl.setText("");
308     status_lbl.setFont(password_font.deriveFont(30f).deriveFont(Font.ITALIC));
309     status_lbl.setAlignmentX(Box.CENTER_ALIGNMENT);
310     status_lbl.setBounds(30, 580, 800, 40);
311     status_lbl.setForeground(Colors.bgColor);
312
```



```
313     status_emoji_lbl = new JLabel();
314     status_emoji_lbl.setText("");
315     status_emoji_lbl.setFont(emoji_font.deriveFont(50f));
316     status_emoji_lbl.setAlignmentX(Box.CENTER_ALIGNMENT);
317     status_emoji_lbl.setBounds(25, 500, 80, 80);
318     status_emoji_lbl.setForeground(Colors.bgColor);
319 }
320
321 /*
322  * Standard function to create Text Fields.
323  */
324 public void createTextFields() {
325     username_txt_fld = new JTextField("");
326     username_txt_fld.setFont(password_font);
327     username_txt_fld.setBounds(822, 319, 300, 50);
328     username_txt_fld.setBackground(Colors.bgColor);
329     username_txt_fld.setOpaque(true);
330     username_txt_fld.setBorder(null);
331     username_txt_fld.setForeground(Colors.accentColor);
332     username_txt_fld.addActionListener(new ActionListener() {
333         @Override
334         public void actionPerformed(ActionEvent e) {
335             System.out.println(username_txt_fld.getText());
336         }
337     });
338
339     password_txt_fld = new JPasswordField("");
340     password_txt_fld.setFont(password_font);
341     password_txt_fld.setBounds(822, 452, 300, 50);
342     password_txt_fld.setBackground(Colors.bgColor);
343     password_txt_fld.setOpaque(true);
344     password_txt_fld.setBorder(null);
345     password_txt_fld.setEchoChar('*');
346     password_txt_fld.setForeground(Colors.accentColor);
347     password_txt_fld.setAlignmentY(Box.CENTER_ALIGNMENT);
348 }
349
350 /*
351  * This function is what checks the password, and so naturally has a lot of if
352  * statements.
353  * Most of them are self explanatory.
354  */
355 @Override
356 public void run() {
357     long lastTime = System.nanoTime();
358     double amountOfTicks = 5.00;
359     double ns = 1000000000 / amountOfTicks;
360     double delta = 0;
361
362     while (running) {
363         // basic game loop logic to ensure 60 fps, dont think too much about
it, it
364         // makes sense.
365         // you can reuse it for consistency, or make a new one.
366
367         long now = System.nanoTime();
368         delta += (now - lastTime) / ns;
369         lastTime = now;
```

```
371         if (delta >= 1) {
372             // System.out.println(username_txt_fld.getText());
373             // System.out.println(password_txt_fld.getPassword());
374             if (username_txt_fld.getText().length() == 0) {
375                 newAccount_btn.setEnabled(false);
376                 status_lbl.setText("Enter Username & Password");
377                 status_emoji_lbl.setText("\uD83E\uDEE3");
378                 // status_emoji_lbl.setText("");
379             } else if (newUser) {
380                 status_lbl.setText("Welcome! Create New Account");
381                 status_emoji_lbl.setText("\uD83D\uDE4F");
382                 // status_emoji_lbl.setText("\uD83D\uDE1E");
383                 try {
384                     Thread.sleep(2000);
385                 } catch (InterruptedException e) {
386                     throw new RuntimeException(e);
387                 }
388                 newUser = false;
389             } else if (incorrectPassword) {
390                 status_lbl.setText("Password doesn't Match!");
391                 status_emoji_lbl.setText("\uD83D\uDE16");
392                 // status_emoji_lbl.setText("\uD83D\uDE1E");
393                 try {
394                     Thread.sleep(2000);
395                 } catch (InterruptedException e) {
396                     throw new RuntimeException(e);
397                 }
398                 incorrectPassword = false;
399             } else if (userExists) {
400                 status_lbl.setText("User Already Exists, Try to Login");
401                 status_emoji_lbl.setText("\uD83D\uDE15");
402                 try {
403                     Thread.sleep(2000);
404                 } catch (InterruptedException e) {
405                     throw new RuntimeException(e);
406                 }
407                 userExists = false;
408             } else if (password_txt_fld.getPassword().length < 8) {
409                 newAccount_btn.setEnabled(false);
410                 status_lbl.setText("Nope, Password is too Short");
411                 status_emoji_lbl.setText("\uD83D\uDE0F");
412                 // status_emoji_lbl.setText("\uD83E\uDD0F");
413                 // status_emoji_lbl.setText("\uD83D\uDE15");
414             } else if (Arrays.equals(password_txt_fld.getPassword(), "abcdefgh
".toCharArray())) {
415                 newAccount_btn.setEnabled(false);
416                 status_lbl.setText("Anyone can guess that bruh");
417                 status_emoji_lbl.setText("\uD83D\uDC80");
418             } else if (Arrays.equals(password_txt_fld.getPassword(), "12345678
".toCharArray())) {
419                 newAccount_btn.setEnabled(false);
420                 status_lbl.setText("12345678? Really?");
421                 status_emoji_lbl.setText("\uD83E\uDEE0");
422             } else if (Arrays.equals(password_txt_fld.getPassword(), "asdfghjk
".toCharArray())) {
423                 newAccount_btn.setEnabled(false);
424                 status_lbl.setText("Be Lazy, but not thaaat lazy");
425                 // status_emoji_lbl.setText("\uD83D\uDE42");
426                 status_emoji_lbl.setText("\uD83D\uDC80");
```

```
427         } else if (Arrays.equals(password_txt_fld.getPassword(), "asdfasdf
    ".toCharArray())) {
428             newAccount_btn.setEnabled(false);
429             status_lbl.setText("Even Krishnaraj isnt this lazy");
430             // status_emoji_lbl.setText("\uD83E\uDD21");
431             status_emoji_lbl.setText("\uD83D\uDC80");
432         } else if (password_txt_fld.getPassword().length > 30) {
433             newAccount_btn.setEnabled(false);
434             status_lbl.setText("Woah, Its too big");
435             status_emoji_lbl.setText("\uD83D\uDE0F");
436         } else {
437             status_lbl.setText("All Good!");
438             status_emoji_lbl.setText("\uD83D\uDC4C");
439             // status_emoji_lbl.setText("\uD83D\uDE0E" );
440             // status_emoji_lbl.setText("");
441             newAccount_btn.setEnabled(true);
442             login_btn.setEnabled(true);
443         }
444         delta--;
445     }
446 }
447 }
448 }
449
450 public void startThread() {
451
452     // Creating the Game Thread
453     loginThread = new Thread(this);
454     loginThread.start();
455
456 }
457 }
```

Listing 5: Main Java file

```
1 package org.howmuch;
2
3 import javax.swing.*;
4 import javax.swing.event.ChangeEvent;
5 import javax.swing.event.ChangeListener;
6 import java.awt.*;
7 import java.awt.event.*;
8 import java.util.ArrayList;
9 import java.util.Arrays;
10 import java.util.Random;
11 import java.util.TimerTask;
12 import java.util.Timer;
13
14 import static java.lang.Math.round;
15 import static org.howmuch.Main.*;
16
17 public class GameFrame extends JFrame {
18     static Timer timer;
19     public static int time_left = 9;
20     public static boolean gameWon = false;
21     static boolean[] whichOptionCorrect;
22     BackgroundPanel backgroundPanel;
23     BackgroundPanel productImagePanel;
24     static JButton option_1_btn;
```

```
25 static JButton option_2_btn;
26 static JButton option_3_btn;
27 static JButton option_4_btn;
28 JPanel options_panel;
29 static JLabel time_lbl;
30 JTextArea productName_txtArea;
31 public static int randomIndex = 0;
32 static String[] currentData;
33 static int correctPrice;
34 static JLabel confetti;
35
36 GameFrame() {
37     randomIndex = 0;
38     time_left = 9;
39     backgroundPanel = new BackgroundPanel();
40     gameWon = false;
41     this.setTitle("How Much?");
42     if (maximized) {
43         this.setExtendedState(MAXIMIZED_BOTH);
44     } else {
45         this.setPreferredSize(new Dimension(Main.WIDTH, Main.HEIGHT));
46     }
47     this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
48     this.setResizable(true);
49     this.setUndecorated(true);
50     this.setMinimumSize(new Dimension(1280, 720));
51
52     createFonts();
53     createBasicButtonPanel();
54     createButtons();
55     createPanels();
56     createLabels();
57
58
59     // Main stuff
60     findRandomIndex();
61     assignCurrentData();
62
63     // Important Updates
64     reassignColors();
65     reassignBounds();
66
67     startTimer();
68
69
70     this.addComponentListener(new ComponentAdapter() {
71         @Override
72         public void componentResized(ComponentEvent e) {
73             reassignBounds();
74             repaint();
75         }
76     });
77
78     this.add(confetti);
79     this.add(productName_txtArea);
80     this.add(productImagePanel);
81     this.add(time_lbl);
82     this.add(options_panel);
83     this.add(basicButtons_pnl);
```

```
84         this.add(backgroundPanel);
85         this.pack();
86         this.setLocationRelativeTo(null);
87         this.setVisible(true);
88     }
89
90     public static void startTimer() {
91         timer = new Timer();
92         timer.schedule(new TimerTask() {
93             @Override
94             public void run() {
95                 System.out.println(time_left);
96                 if (time_left == 0) {
97                     timer.cancel();
98                     timer.purge();
99                     grantAccess = true;
100                     Main.changeFrame(7);
101                 }
102                 time_left--;
103                 changeTimeOnTimer();
104             }
105         }, 1000, 1000);
106     }
107
108     public static void changeTimeOnTimer() {
109         time_lbl.setText(String.valueOf(time_left));
110     }
111
112     private void assignCurrentData() {
113         currentData = new String[]{"", "", ""};
114         System.out.println("Accessing or atleast trying to access data here");
115         try {
116             if (!usingMongo) {
117                 System.out.println("We are accessing data from the local database
118 as mongo isnt working");
119                 currentData = DataBaseManager.readFromLocalDatabase(currentTopic,
120 randomIndex);
121             } else {
122                 currentData = MongoManager.fetchDataFromMongo(currentTopic,
123 randomIndex);
124                 System.out.println("Reading data from mongo sucessful");
125             }
126         } catch (Exception e) {
127             System.out.println("We got some Issues reading the file from Mongodb");
128         }
129         ;
130         currentData = DataBaseManager.readFromLocalDatabase(currentTopic,
131 randomIndex);
132     }
133
134     private void findRandomIndex() {
135         int max = DataBaseManager.findLength(currentTopic);
136         Random random = new Random();
137         // Generates random integers 0 to 49
138         randomIndex = random.nextInt(max);
139     }
140
141     private void loadGameDataOnScreen() {
142         System.out.println(Arrays.toString(currentData));
143     }
```

```
138     Image productImage = new ImageIcon(currentData[2]).getImage();
139     int maxWidth = (int) (0.4 * this.getWidth());
140     int maxHeight = (int) (0.4 * this.getHeight());
141
142     int[] imageSize = calculateImageSize(maxWidth, maxHeight, productImage.
getWidth(productImagePanel), productImage.getHeight(productImagePanel));
143     System.out.println(Arrays.toString(imageSize));
144     productImagePanel.setBounds((int) (0.07 * this.getWidth()) + maxWidth / 2
- imageSize[0] / 2, (int) (0.07 * this.getHeight()) + maxHeight / 2 - imageSize
[1] / 2, imageSize[0], imageSize[1]);
145     productImagePanel.setBackground(currentData[2]);
146
147     productName_txtArea.setBounds((int) (0.065 * this.getWidth()), (int) (0.83
* this.getHeight()), (int) (0.5 * this.getWidth()), (int) (0.2 * this.
getHeight()));
148     productName_txtArea.setText(currentData[0]);
149
150     // setting price
151     setPrices();
152 }
153
154 public static void setPrices() {
155     Random random = new Random();
156     int[] wrongPrices = new int[]{0, 0, 0};
157     correctPrice = Math.round(Integer.parseInt(currentData[1])) + 2;
158     System.out.println("Correct price is: ");
159     System.out.println(correctPrice);
160
161     double randomMultiplier = 0.3 + random.nextDouble(2.7);
162     System.out.println(randomMultiplier);
163     wrongPrices[0] = (int) (correctPrice * randomMultiplier);
164
165     randomMultiplier = 0.3 + random.nextDouble(2.7);
166     System.out.println(randomMultiplier);
167     wrongPrices[1] = (int) (correctPrice * randomMultiplier);
168
169     randomMultiplier = 0.3 + random.nextDouble(2.7);
170     System.out.println(randomMultiplier);
171     wrongPrices[2] = (int) (correctPrice * randomMultiplier);
172
173     ArrayList<Integer> list = new ArrayList<Integer>(4);
174     list.add(correctPrice);
175     for (int i = 0; i < 3; i++) {
176         list.add(wrongPrices[i]);
177     }
178     System.out.println(list);
179
180     // option_1_btn.setText("R" + String.format("%.0f", (double)
181 //         round((double) list.remove(random.nextInt(list.size())) / 100) *
100
182 //     ));
183 // option_2_btn.setText("R" + String.format("%.0f", (double)
184 //         round((double) list.remove(random.nextInt(list.size())) / 100) *
100
185 //     ));
186 // option_3_btn.setText("R" + String.format("%.0f", (double)
187 //         round((double) list.remove(random.nextInt(list.size())) / 100) *
100
188 //     ));
```

```
189 //         option_4_btn.setText("R" + String.format("%.0f", (double)
190 //             round((double) list.remove(random.nextInt(list.size())) / 100) *
191 //         100
192 //     ));
193     whichOptionCorrect = new boolean[]{false, false, false, false};
194     int optionValue = 0;
195
196     optionValue = list.remove(random.nextInt(list.size()));
197     whichOptionCorrect[0] = optionValue == correctPrice;
198     option_1_btn.setText("R" + String.format("%.0f", (double) optionValue));
199
200     optionValue = list.remove(random.nextInt(list.size()));
201     whichOptionCorrect[1] = optionValue == correctPrice;
202     option_2_btn.setText("R" + String.format("%.0f", (double) optionValue));
203
204     optionValue = list.remove(random.nextInt(list.size()));
205     whichOptionCorrect[2] = optionValue == correctPrice;
206     option_3_btn.setText("R" + String.format("%.0f", (double) optionValue));
207
208     optionValue = list.remove(random.nextInt(list.size()));
209     whichOptionCorrect[3] = optionValue == correctPrice;
210     option_4_btn.setText("R" + String.format("%.0f", (double) optionValue));
211 }
212
213 private int[] calculateImageSize(int maxWidth, int maxHeight, double width,
214 double height) {
215     int wt = 600, ht = 550;
216     double aspectRatio = width / height;
217     System.out.println(aspectRatio);
218     if (aspectRatio > 1) {
219         // landscape
220         wt = maxWidth;
221         ht = (int) (wt / aspectRatio);
222     } else {
223         // portrait image
224         ht = maxHeight;
225         wt = (int) (ht * aspectRatio);
226     }
227     return new int[]{wt, ht};
228 }
229
230 private void createLabels() {
231     time_lbl = new JLabel();
232     time_lbl.setAlignmentY(Box.CENTER_ALIGNMENT);
233     time_lbl.setAlignmentX(Box.CENTER_ALIGNMENT);
234     time_lbl.setOpaque(true);
235     time_lbl.setBorder(null);
236     time_lbl.setText(String.valueOf(time_left));
237
238     productName_txtArea = new JTextArea();
239     productName_txtArea.setAlignmentY(Box.CENTER_ALIGNMENT);
240     productName_txtArea.setAlignmentX(Box.LEFT_ALIGNMENT);
241     productName_txtArea.setOpaque(true);
242     productName_txtArea.setBorder(null);
243     productName_txtArea.setLineWrap(true);
244
245     ImageIcon imageIcon = new ImageIcon("src/main/resources/images/confetti.
```

```
gif");
246     confetti = new JLabel(imageIcon);
247     confetti.setVisible(false);
248 }
249
250 private void reassignBounds() {
251     Dimension screenSize = this.getSize();
252
253     // The Entire basic button panel for closing minimizing and stuff
254     basicButtons_pnl.setBounds(this.getWidth() - (exit_btn.getWidth() * 3) -
255 40, 10, exit_btn.getWidth() * 3 + 35, exit_btn.getHeight());
256
257     // Options panel
258     options_panel.setBounds((int) (0.70 * screenSize.getWidth()), (int) (0.58
259 * screenSize.getHeight()), (int) (0.60 * screenSize.getWidth()), 700);
260
261     // Buttons in the Options Panel
262     option_1_btn.setBounds(new Rectangle((int) (0.45 * screenSize.getWidth()),
263 80));
264     option_1_btn.setFont(options_font.deriveFont((float) (0.05 * getHeight()))
265 );
266
267     option_2_btn.setBounds(new Rectangle((int) (0.45 * screenSize.getWidth()),
268 80));
269     option_2_btn.setFont(options_font.deriveFont((float) (0.05 * getHeight()))
270 );
271
272     option_3_btn.setBounds(new Rectangle((int) (0.45 * screenSize.getWidth()),
273 70));
274     option_3_btn.setFont(options_font.deriveFont((float) (0.05 * getHeight()))
275 );
276
277     option_4_btn.setBounds(new Rectangle((int) (0.45 * screenSize.getWidth()),
278 70));
279     option_4_btn.setFont(options_font.deriveFont((float) (0.05 * getHeight()))
280 );
281
282     productName_txtArea.setFont(password_font.deriveFont((float) (0.04 *
283 getHeight())));
284
285     // The Score label
286     time_lbl.setBounds((int) (0.890 * screenSize.getWidth()), (int) (0.14 *
287 screenSize.getHeight()), (int) (0.05 * screenSize.getWidth()), (int) (0.11 *
288 screenSize.getHeight()));
289     time_lbl.setFont(buttonFont.deriveFont((float) (0.09 * getHeight())));
290
291     confetti.setBounds((int) (-0.3 * screenSize.getWidth()), (int) (-0.27 *
292 screenSize.getHeight()), this.getWidth(), this.getHeight());
293     loadGameDataOnScreen();
294 }
295
296 private void reassignColors() {
297
298     if (Colors.DarkMode) {
299         backgroundPanel.setBackground("src/main/resources/images/gamescreen.
300 png");
301     } else {
302         backgroundPanel.setBackground("src/main/resources/images/gamescreen.
303 png");
304     }
305 }
```



```
288     }
289     Colors.reassignColors();
290     basicButtons_pnl.setBackground(Colors.light_primaryColor);
291     exit_btn.setBackground(Colors.light_primaryColor);
292     resize_btn.setBackground(Colors.light_primaryColor);
293     minimize_btn.setBackground(Colors.light_primaryColor);
294     option_1_btn.setBackground(Colors.light_primaryColor);
295     option_1_btn.setForeground(Colors.light_bgColor);
296     option_2_btn.setBackground(Colors.light_primaryColor);
297     option_2_btn.setForeground(Colors.light_bgColor);
298     option_4_btn.setBackground(Colors.light_primaryColor);
299     option_4_btn.setForeground(Colors.light_bgColor);
300     option_3_btn.setBackground(Colors.light_primaryColor);
301     option_3_btn.setForeground(Colors.light_bgColor);
302     productName_txtArea.setBackground(Color.WHITE);
303     productName_txtArea.setForeground(Colors.light_primaryColor);
304     if (Colors.DarkMode) {
305         time_lbl.setBackground(Colors.light_secondaryColor);
306     } else {
307         time_lbl.setBackground(Colors.light_secondaryColor);
308     }
309     time_lbl.setForeground(Colors.light_primaryColor);
310 }
311
312 private void createPanels() {
313     options_panel = new JPanel();
314     BoxLayout bl = new BoxLayout(options_panel, BoxLayout.Y_AXIS);
315     options_panel.setLayout(bl);
316     options_panel.add(option_1_btn);
317     options_panel.add(Box.createRigidArea(new Dimension(0, 25)));
318     options_panel.add(option_2_btn);
319     options_panel.add(Box.createRigidArea(new Dimension(0, 25)));
320     options_panel.add(option_3_btn);
321     options_panel.add(Box.createRigidArea(new Dimension(0, 25)));
322     options_panel.add(option_4_btn);
323     options_panel.setBackground(new Color(0, 0, 0, 0));
324
325     productImagePanel = new BackgroundPanel();
326 }
327
328 private void createButtons() {
329
330     // Removing Change and Action Listeners.
331     removeAllChangeAndActionListenersFromBasicButtons();
332     exit_btn.addChangeListener(evt -> {
333         if (exit_btn.getModel().isPressed()) {
334             timer.cancel();
335             timer.purge();
336             exit_btn.setForeground(Colors.primaryColor);
337             this.setVisible(false);
338             this.dispose();
339             Main.changeFrame(0);
340         } else if (exit_btn.getModel().isRollover()) {
341             exit_btn.setForeground(Colors.secondaryColor);
342         } else {
343             exit_btn.setForeground(Colors.primaryColor);
344         }
345     });
346     resize_btn.addActionListener(e -> {
```

```
347         if (!Main.maximized) {
348             this.setExtendedState(MAXIMIZED_BOTH);
349             resize_btn.setIcon(new ImageIcon(resizeDown_image));
350         } else {
351             this.setExtendedState(JFrame.NORMAL);
352             this.setLocationRelativeTo(null);
353             Dimension dimension = Toolkit.getDefaultToolkit().getScreenSize();
354             int x = (int) ((dimension.getWidth() - Main.WIDTH) / 2);
355             int y = (int) ((dimension.getHeight() - Main.HEIGHT) / 2);
356             this.setBounds(x, y, Main.WIDTH, Main.HEIGHT);
357             resize_btn.setIcon(new ImageIcon(resizeUp_image));
358         }
359         Main.maximized = !Main.maximized;
360     });
361
362     minimize_btn.addChangeListener(evt -> {
363         if (minimize_btn.getModel().isPressed()) {
364             this.setState(JFrame.ICONIFIED);
365             minimize_btn.setForeground(Colors.primaryColor);
366         } else if (minimize_btn.getModel().isRollover()) {
367             minimize_btn.setForeground(Colors.secondaryColor);
368         } else {
369             minimize_btn.setForeground(Colors.primaryColor);
370         }
371     });
372
373     option_1_btn = new JButton();
374     assignButtonProperties(option_1_btn);
375     option_2_btn = new JButton();
376     assignButtonProperties(option_2_btn);
377     option_3_btn = new JButton();
378     assignButtonProperties(option_3_btn);
379     option_4_btn = new JButton();
380     assignButtonProperties(option_4_btn);
381 }
382
383 static void removeAllChangeAndActionListenersFromBasicButtons() {
384     for (ActionListener listener : exit_btn.getActionListeners()) {
385         exit_btn.removeActionListener(listener);
386     }
387     for (ChangeListener listener : exit_btn.getChangeListeners()) {
388         exit_btn.removeChangeListener(listener);
389     }
390
391     for (ActionListener listener : resize_btn.getActionListeners()) {
392         resize_btn.removeActionListener(listener);
393     }
394     for (ChangeListener listener : resize_btn.getChangeListeners()) {
395         resize_btn.removeChangeListener(listener);
396     }
397
398     for (ActionListener listener : minimize_btn.getActionListeners()) {
399         minimize_btn.removeActionListener(listener);
400     }
401     for (ChangeListener listener : minimize_btn.getChangeListeners()) {
402         minimize_btn.removeChangeListener(listener);
403     }
404 }
405
```

```
406 static void removeChangeAndActionListenersFromOption_btns() {
407     for (ChangeListener changeListener : option_1_btn.getChangeListeners()) {
408         option_1_btn.removeChangeListener(changeListener);
409     }
410     for (ChangeListener changeListener : option_2_btn.getChangeListeners()) {
411         option_2_btn.removeChangeListener(changeListener);
412     }
413     for (ChangeListener changeListener : option_3_btn.getChangeListeners()) {
414         option_3_btn.removeChangeListener(changeListener);
415     }
416     for (ChangeListener changeListener : option_4_btn.getChangeListeners()) {
417         option_4_btn.removeChangeListener(changeListener);
418     }
419
420     for (ActionListener ActionListener : option_1_btn.getActionListeners()) {
421         option_1_btn.removeActionListener(ActionListener);
422     }
423     for (ActionListener ActionListener : option_2_btn.getActionListeners()) {
424         option_2_btn.removeActionListener(ActionListener);
425     }
426     for (ActionListener ActionListener : option_3_btn.getActionListeners()) {
427         option_3_btn.removeActionListener(ActionListener);
428     }
429     for (ActionListener ActionListener : option_4_btn.getActionListeners()) {
430         option_4_btn.removeActionListener(ActionListener);
431     }
432 }
433
434
435 private void assignButtonProperties(JButton optionButton) {
436     optionButton.setText("");
437     optionButton.setAlignmentY(Box.CENTER_ALIGNMENT);
438     optionButton.setAlignmentX(Box.LEFT_ALIGNMENT);
439     optionButton.setFocusPainted(false);
440     optionButton.setBounds(0, 0, 500, 500);
441     optionButton.setContentAreaFilled(false);
442     optionButton.setOpaque(true);
443     optionButton.setBorder(null);
444     optionButton.addChangeListener(evt -> {
445         if (optionButton.getModel().isPressed()) {
446             optionButton.setForeground(Colors.accentColor);
447         } else if (optionButton.getModel().isRollover()) {
448             optionButton.setForeground(Colors.accentColor);
449         } else {
450             optionButton.setForeground(Colors.light_bgColor);
451         }
452     });
453     optionButton.addActionListener(e -> {
454         int this_btn_price = Integer.parseInt(optionButton.getText().replace("
R", "").replace(",", ""));
455         if (correctPrice == this_btn_price) {
456             runWinningClosingErrands();
457         } else {
458             runLosingClosingErrands(this_btn_price);
459         }
460     });
461 }
462
463 private void runLosingClosingErrands(int this_btn_price) {
```

```
464         removeChangeAndActionListenersFromOption_btns();
465         if (whichOptionCorrect[0]) {
466             option_1_btn.setForeground(new Color(56, 159, 82));
467             // option_1_btn.setFont(options_font.deriveFont((float) (0.05 *
getHeight()).deriveFont(Font.BOLD));
468             option_2_btn.setForeground(new Color(227, 83, 83));
469             option_3_btn.setForeground(new Color(227, 83, 83));
470             option_4_btn.setForeground(new Color(227, 83, 83));
471         } else if (whichOptionCorrect[1]) {
472             option_2_btn.setForeground(new Color(56, 159, 82));
473             // option_2_btn.setFont(options_font.deriveFont((float) (0.05 *
getHeight()).deriveFont(Font.BOLD));
474             option_1_btn.setForeground(new Color(227, 83, 83));
475             option_3_btn.setForeground(new Color(227, 83, 83));
476             option_4_btn.setForeground(new Color(227, 83, 83));
477         } else if (whichOptionCorrect[2]) {
478             option_3_btn.setForeground(new Color(56, 159, 82));
479             // option_3_btn.setFont(options_font.deriveFont((float) (0.05 *
getHeight()).deriveFont(Font.BOLD));
480             option_2_btn.setForeground(new Color(227, 83, 83));
481             option_1_btn.setForeground(new Color(227, 83, 83));
482             option_4_btn.setForeground(new Color(227, 83, 83));
483         } else if (whichOptionCorrect[3]) {
484             option_4_btn.setForeground(new Color(56, 159, 82));
485             // option_4_btn.setFont(options_font.deriveFont((float) (0.05 *
getHeight()).deriveFont(Font.BOLD));
486             option_2_btn.setForeground(new Color(227, 83, 83));
487             option_3_btn.setForeground(new Color(227, 83, 83));
488             option_1_btn.setForeground(new Color(227, 83, 83));
489         }
490         System.out.println("That was an incorrect Guess");
491         System.out.println(this_btn_price);
492         time_left = 3;
493         gameWon = false;
494         grantAccess = true;
495     }
496
497     private void runWinningClosingErrands() {
498         removeChangeAndActionListenersFromOption_btns();
499         System.out.println("You guessed correctly");
500         DataBaseManager.currentScore += time_left;
501         confetti.setVisible(true);
502         time_left = 2;
503         grantAccess = true;
504         gameWon = true;
505     }
506 }
```

Listing 6: Main Java file

```
1 package org.howmuch;
2
3 import javax.swing.*;
4 import java.awt.*;
5 import java.awt.event.*;
6
7 import static org.howmuch.Main.*;
8
9 public class TopicsFrame extends JFrame {
```

```
10
11 BackgroundPanel backgroundPanel;
12 JButton option1_btn, option2_btn, option3_btn, option4_btn;
13 JButton backToMenu_btn;
14 JPanel options_panel;
15
16 TopicsFrame() {
17     backgroundPanel = new BackgroundPanel();
18
19     this.setTitle("How Much?");
20     if (maximized) {
21         this.setExtendedState(MAXIMIZED_BOTH);
22     } else {
23         this.setPreferredSize(new Dimension(Main.WIDTH, Main.HEIGHT));
24     }
25     this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
26     this.setResizable(true);
27     this.setUndecorated(true);
28     this.setMinimumSize(new Dimension(1280, 720));
29
30     createFonts();
31     createBasicButtonPanel();
32     createButtons();
33     createPanels();
34     reassignColors();
35     reassignBounds();
36
37     this.addComponentListener(new ComponentAdapter() {
38         @Override
39         public void componentResized(ComponentEvent e) {
40             reassignBounds();
41             repaint();
42         }
43     });
44
45     this.add(backToMenu_btn);
46     this.add(options_panel);
47     this.add(basicButtons_pnl);
48     this.add(backgroundPanel);
49     this.pack();
50     this.setLocationRelativeTo(null);
51     this.setVisible(true);
52 }
53
54 private void reassignBounds() {
55     Dimension screenSize = this.getSize();
56
57     // The back to menu mode label
58     backToMenu_btn.setBounds((int) (0.015 * screenSize.getWidth()), (int)
59 (0.80 * screenSize.getHeight()),
60 (int) (0.20 * screenSize.getWidth()), (int) (0.07 * screenSize.
61 getHeight()));
62     backToMenu_btn.setFont(buttonFont.deriveFont((float) (0.05 * getHeight()))
63 );
64
65     // The Entire basic button panel for closing minimizing and stuff
66     basicButtons_pnl.setBounds(this.getWidth() - (exit_btn.getWidth() * 3) -
67 40, 10, exit_btn.getWidth() * 3 + 35,
68     exit_btn.getHeight());
```

```
65
66     // Options panel
67     options_panel.setBounds((int) (0.60 * screenSize.getWidth()), (int) (0.34
68 * screenSize.getHeight()),
69         (int) (0.45 * screenSize.getWidth()), 700);
70
71     // Buttons in the Options Panel
72     option1_btn.setBounds(new Rectangle((int) (0.45 * screenSize.getWidth()),
73 80));
74     option1_btn.setFont(buttonFont.deriveFont((float) (0.07 * getHeight())));
75
76     option2_btn.setBounds(new Rectangle((int) (0.45 * screenSize.getWidth()),
77 80));
78     option2_btn.setFont(buttonFont.deriveFont((float) (0.07 * getHeight())));
79
80     option3_btn.setBounds(new Rectangle((int) (0.45 * screenSize.getWidth()),
81 70));
82     option3_btn.setFont(buttonFont.deriveFont((float) (0.07 * getHeight())));
83
84     option4_btn.setBounds(new Rectangle((int) (0.45 * screenSize.getWidth()),
85 70));
86     option4_btn.setFont(buttonFont.deriveFont((float) (0.07 * getHeight())));
87 }
88
89 private void reassignColors() {
90
91     if (Colors.DarkMode) {
92         backgroundPanel.setBackground("/run/media/krishnaraj/Programs/Java/How
93 Much/src/main/resources/images/choose topic dark.png");
94     } else {
95         backgroundPanel.setBackground("/run/media/krishnaraj/Programs/Java/How
96 Much/src/main/resources/images/choose topic.png");
97     }
98     Colors.reassignColors();
99     basicButtons_pnl.setBackground(Colors.bgColor);
100    exit_btn.setBackground(Colors.bgColor);
101    resize_btn.setBackground(Colors.bgColor);
102    backToMenu_btn.setBackground(Colors.primaryColor);
103    backToMenu_btn.setForeground(Colors.bgColor);
104    minimize_btn.setBackground(Colors.bgColor);
105    option1_btn.setBackground(Colors.bgColor);
106    option1_btn.setForeground(Colors.primaryColor);
107    option2_btn.setBackground(Colors.bgColor);
108    option2_btn.setForeground(Colors.primaryColor);
109    option4_btn.setBackground(Colors.bgColor);
110    option4_btn.setForeground(Colors.primaryColor);
111    option3_btn.setBackground(Colors.bgColor);
112    option3_btn.setForeground(Colors.primaryColor);
113 }
114
115 private void createPanels() {
116     options_panel = new JPanel();
117     BorderLayout bl = new BorderLayout(options_panel, BorderLayout.Y_AXIS);
118     options_panel.setLayout(bl);
119     options_panel.add(option1_btn);
120     options_panel.add(Box.createRigidArea(new Dimension(0, 25)));
121     options_panel.add(option2_btn);
122     options_panel.add(Box.createRigidArea(new Dimension(0, 25)));
123     options_panel.add(option3_btn);
```

```
117         options_panel.add(Box.createRigidArea(new Dimension(0, 25)));
118         options_panel.add(option4_btn);
119         options_panel.setBackground(new Color(0, 0, 0, 0));
120
121     }
122
123     private void createButtons() {
124
125         // Removing Change and Action Listeners.
126         GameFrame.removeAllChangeAndActionListenersFromBasicButtons();
127
128         exit_btn.addChangeListener(evt -> {
129             if (exit_btn.getModel().isPressed()) {
130                 exit_btn.setForeground(Colors.primaryColor);
131                 Main.changeFrame(0);
132             } else if (exit_btn.getModel().isRollover()) {
133                 exit_btn.setForeground(Colors.secondaryColor);
134             } else {
135                 exit_btn.setForeground(Colors.primaryColor);
136             }
137         });
138         resize_btn.addActionListener(e -> {
139             if (!Main.maximized) {
140                 this.setExtendedState(MAXIMIZED_BOTH);
141                 resize_btn.setIcon(new ImageIcon(resizeDown_image));
142             } else {
143                 this.setExtendedState(JFrame.NORMAL);
144                 this.setLocationRelativeTo(null);
145                 Dimension dimension = Toolkit.getDefaultToolkit().getScreenSize();
146                 int x = (int) ((dimension.getWidth() - Main.WIDTH) / 2);
147                 int y = (int) ((dimension.getHeight() - Main.HEIGHT) / 2);
148                 this.setBounds(x, y, Main.WIDTH, Main.HEIGHT);
149                 resize_btn.setIcon(new ImageIcon(resizeUp_image));
150             }
151             Main.maximized = !Main.maximized;
152         });
153
154         minimize_btn.addChangeListener(evt -> {
155             if (minimize_btn.getModel().isPressed()) {
156                 this.setState(JFrame.ICONIFIED);
157                 minimize_btn.setForeground(Colors.primaryColor);
158             } else if (minimize_btn.getModel().isRollover()) {
159                 minimize_btn.setForeground(Colors.secondaryColor);
160             } else {
161                 minimize_btn.setForeground(Colors.primaryColor);
162             }
163         });
164
165         backToMenu_btn = new JButton();
166         backToMenu_btn.setText("Back to Menu ");
167         backToMenu_btn.setAlignmentY(Box.CENTER_ALIGNMENT);
168         backToMenu_btn.setAlignmentX(Box.LEFT_ALIGNMENT);
169         backToMenu_btn.setFocusPainted(false);
170         backToMenu_btn.setContentAreaFilled(false);
171         backToMenu_btn.setOpaque(true);
172         backToMenu_btn.setBorder(null);
173         backToMenu_btn.addChangeListener(evt -> {
174             if (backToMenu_btn.getModel().isPressed()) {
175                 backToMenu_btn.setForeground(Colors.bgColor);
```

```
176         } else if (backToMenu_btn.getModel().isRollover()) {
177             backToMenu_btn.setForeground(Colors.accentColor);
178         } else {
179             backToMenu_btn.setForeground(Colors.bgColor);
180         }
181     });
182     backToMenu_btn.addActionListener(e -> {
183         this.setVisible(false);
184         this.dispose();
185         grantAccess = true;
186         Main.changeFrame(1);
187     });
188
189     option1_btn = new JButton();
190     option1_btn.setText(Topics[0] + " ");
191     option1_btn.setAlignmentY(Box.CENTER_ALIGNMENT);
192     option1_btn.setAlignmentX(Box.LEFT_ALIGNMENT);
193     option1_btn.setFocusPainted(false);
194     option1_btn.setBounds(0, 0, 500, 500);
195     option1_btn.setContentAreaFilled(false);
196     option1_btn.setOpaque(true);
197     option1_btn.setBorder(null);
198     option1_btn.addChangeListener(evt -> {
199         if (option1_btn.getModel().isPressed()) {
200             option1_btn.setForeground(Colors.accentColor);
201         } else if (option1_btn.getModel().isRollover()) {
202             option1_btn.setForeground(Colors.accentColor);
203         } else {
204             option1_btn.setForeground(Colors.primaryColor);
205         }
206     });
207
208     option1_btn.addActionListener(e -> {
209         grantAccess = true;
210         currentTopic = Topics[0];
211         this.setVisible(false);
212         this.dispose();
213         grantAccess = true;
214         Main.changeFrame(6);
215     });
216
217     option2_btn = new JButton();
218     option2_btn.setText(Topics[1] + " ");
219     option2_btn.setAlignmentY(Box.CENTER_ALIGNMENT);
220     option2_btn.setAlignmentX(Box.LEFT_ALIGNMENT);
221     option2_btn.setFocusPainted(false);
222     option2_btn.setContentAreaFilled(false);
223     option2_btn.setOpaque(true);
224     option2_btn.setBorder(null);
225     option2_btn.addChangeListener(evt -> {
226         if (option2_btn.getModel().isPressed()) {
227             option2_btn.setForeground(Colors.accentColor);
228         } else if (option2_btn.getModel().isRollover()) {
229             option2_btn.setForeground(Colors.accentColor);
230         } else {
231             option2_btn.setForeground(Colors.primaryColor);
232         }
233     });
234     option2_btn.addActionListener(e -> {
```



```
235         grantAccess = true;
236         currentTopic = Topics[1];
237         this.setVisible(false);
238         this.dispose();
239         grantAccess = true;
240         Main.changeFrame(6);
241     });
242
243     option3_btn = new JButton();
244     option3_btn.setText(Topics[2] + " ");
245     option3_btn.setAlignmentY(Box.CENTER_ALIGNMENT);
246     option3_btn.setAlignmentX(Box.LEFT_ALIGNMENT);
247     option3_btn.setFocusPainted(false);
248     option3_btn.setContentAreaFilled(false);
249     option3_btn.setOpaque(true);
250     option3_btn.setBorder(null);
251     option3_btn.addChangeListener(evt -> {
252         if (option3_btn.getModel().isPressed()) {
253             option3_btn.setForeground(Colors.accentColor);
254         } else if (option3_btn.getModel().isRollover()) {
255             option3_btn.setForeground(Colors.accentColor);
256         } else {
257             option3_btn.setForeground(Colors.primaryColor);
258         }
259     });
260     option3_btn.addActionListener(e -> {
261         grantAccess = true;
262         currentTopic = Topics[2];
263         this.setVisible(false);
264         this.dispose();
265         grantAccess = true;
266         Main.changeFrame(6);
267     });
268
269     option4_btn = new JButton();
270     option4_btn.setText(Topics[3] + " ");
271     option4_btn.setAlignmentY(Box.CENTER_ALIGNMENT);
272     option4_btn.setAlignmentX(Box.LEFT_ALIGNMENT);
273     option4_btn.setFont(buttonFont.deriveFont(44f));
274     option4_btn.setFocusPainted(false);
275     option4_btn.setContentAreaFilled(false);
276     option4_btn.setOpaque(true);
277     option4_btn.setBorder(null);
278     option4_btn.addChangeListener(evt -> {
279         if (option4_btn.getModel().isPressed()) {
280             option4_btn.setForeground(Colors.accentColor);
281         } else if (option4_btn.getModel().isRollover()) {
282             option4_btn.setForeground(Colors.accentColor);
283         } else {
284             option4_btn.setForeground(Colors.primaryColor);
285         }
286     });
287     option4_btn.addActionListener(e -> {
288         grantAccess = true;
289         currentTopic = Topics[3];
290         this.setVisible(false);
291         this.dispose();
292         grantAccess = true;
293         Main.changeFrame(6);
```

```
294     });  
295 }  
296  
297 }
```

Listing 7: Main Java file

```
1 package org.howmuch;  
2  
3 import javax.swing.*;  
4 import java.awt.*;  
5 import java.awt.event.ComponentAdapter;  
6 import java.awt.event.ComponentEvent;  
7  
8 import static org.howmuch.Main.*;  
9  
10 public class GameOverFrame extends JFrame {  
11     BackgroundPanel backgroundPanel;  
12     JButton backtoTopic_btn;  
13     JLabel score_lbl;  
14  
15     GameOverFrame() {  
16         backgroundPanel = new BackgroundPanel();  
17  
18         this.setTitle("How Much? ");  
19         if (maximized) {  
20             this.setExtendedState(MAXIMIZED_BOTH);  
21         } else {  
22             this.setPreferredSize(new Dimension(Main.WIDTH, Main.HEIGHT));  
23         }  
24         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
25         this.setResizable(true);  
26         this.setUndecorated(true);  
27         this.setMinimumSize(new Dimension(1280, 720));  
28  
29         createFonts();  
30         createBasicButtonPanel();  
31         createLabels();  
32         createButtons();  
33         reassignColors();  
34         reassignBounds();  
35  
36         this.addComponentListener(new ComponentAdapter() {  
37             @Override  
38             public void componentResized(ComponentEvent e) {  
39                 reassignBounds();  
40                 repaint();  
41             }  
42         });  
43  
44         this.add(score_lbl);  
45         this.add(backtoTopic_btn);  
46         this.add(basicButtons_pnl);  
47         this.add(backgroundPanel);  
48         this.pack();  
49         this.setLocationRelativeTo(null);  
50         this.setVisible(true);  
51     }  
52 }
```

```
53     private void reassignBounds() {
54         Dimension screenSize = this.getSize();
55
56         // The back to menu mode label
57         backtoTopic_btn.setBounds((int) (0.001 * screenSize.getWidth()), (int)
58         (0.80 * screenSize.getHeight()),
59         (int) (0.20 * screenSize.getWidth()), (int) (0.07 * screenSize.
60         getHeight()));
61         backtoTopic_btn.setFont(buttonFont.deriveFont((float) (0.06 * getHeight()
62         )));
63
64         // The Entire basic button panel for closing minimizing and stuff
65         basicButtons_pnl.setBounds(this.getWidth() - (exit_btn.getWidth() * 3) -
66         40, 10, exit_btn.getWidth() * 3 + 35,
67         exit_btn.getHeight());
68
69         // Score Label
70         score_lbl.setBounds((int) (0.76 * screenSize.getWidth()), (int) (0.80 *
71         screenSize.getHeight()),
72         (int) (0.31 * screenSize.getWidth()), (int) (0.13 * screenSize.
73         getHeight()));
74         score_lbl.setFont(buttonFont.deriveFont((float) (0.14 * getHeight())));
75     }
76
77     private void reassignColors() {
78         Colors.reassignColors();
79         if (GameFrame.gameWon) {
80             if (Colors.DarkMode) {
81                 backgroundPanel.setBackground("/run/media/krishnaraj/Programs/Java
82                 /How Much/src/main/resources/images/game won over dark.png");
83             } else {
84                 backgroundPanel.setBackground("/run/media/krishnaraj/Programs/Java
85                 /How Much/src/main/resources/images/game won over.png");
86             }
87         } else {
88             if (Colors.DarkMode) {
89                 backgroundPanel.setBackground("/run/media/krishnaraj/Programs/Java
90                 /How Much/src/main/resources/images/game over dark.png");
91             } else {
92                 backgroundPanel.setBackground("/run/media/krishnaraj/Programs/Java
93                 /How Much/src/main/resources/images/game over.png");
94             }
95         }
96         backtoTopic_btn.setBackground(Colors.primaryColor);
97         backtoTopic_btn.setForeground(Colors.bgColor);
98
99         score_lbl.setBackground(Colors.bgColor);
100        score_lbl.setForeground(Colors.accentColor);
101
102        basicButtons_pnl.setBackground(Colors.bgColor);
103        exit_btn.setBackground(Colors.bgColor);
104        resize_btn.setBackground(Colors.bgColor);
105        minimize_btn.setBackground(Colors.bgColor);
106    }
107
108    private void createLabels() {
109        score_lbl = new JLabel();
110        score_lbl.setText(String.valueOf(DataBaseManager.currentScore));
111        score_lbl.setAlignmentY(Box.CENTER_ALIGNMENT);
112    }
```

```
102     score_lbl.setAlignmentX(Box.CENTER_ALIGNMENT);
103     score_lbl.setOpaque(true);
104     score_lbl.setBorder(null);
105 }
106
107 private void createButtons() {
108     backtoTopic_btn = new JButton();
109     backtoTopic_btn.setText("Try Again");
110     backtoTopic_btn.setAlignmentY(Box.CENTER_ALIGNMENT);
111     backtoTopic_btn.setAlignmentX(Box.LEFT_ALIGNMENT);
112     backtoTopic_btn.setFocusPainted(false);
113     backtoTopic_btn.setContentAreaFilled(false);
114     backtoTopic_btn.setOpaque(true);
115     backtoTopic_btn.setBorder(null);
116     backtoTopic_btn.addChangeListener(evt -> {
117         if (backtoTopic_btn.getModel().isPressed()) {
118             backtoTopic_btn.setForeground(Colors.bgColor);
119         } else if (backtoTopic_btn.getModel().isRollover()) {
120             backtoTopic_btn.setForeground(Colors.accentColor);
121         } else {
122             backtoTopic_btn.setForeground(Colors.bgColor);
123         }
124     });
125     backtoTopic_btn.addActionListener(e -> {
126         this.setVisible(false);
127         this.dispose();
128         grantAccess = true;
129         Main.changeFrame(2);
130     });
131
132     // Removing Change and Action Listeners.
133     GameFrame.removeAllChangeAndActionListenersFromBasicButtons();
134
135     exit_btn.addChangeListener(evt -> {
136         if (exit_btn.getModel().isPressed()) {
137             exit_btn.setForeground(Colors.primaryColor);
138             Main.changeFrame(0);
139         } else if (exit_btn.getModel().isRollover()) {
140             exit_btn.setForeground(Colors.secondaryColor);
141         } else {
142             exit_btn.setForeground(Colors.primaryColor);
143         }
144     });
145     resize_btn.addActionListener(e -> {
146         if (!Main.maximized) {
147             this.setExtendedState(MAXIMIZED_BOTH);
148             resize_btn.setIcon(new ImageIcon(resizeDown_image));
149         } else {
150             this.setExtendedState(JFrame.NORMAL);
151             this.setLocationRelativeTo(null);
152             Dimension dimension = Toolkit.getDefaultToolkit().getScreenSize();
153             int x = (int) ((dimension.getWidth() - Main.WIDTH) / 2);
154             int y = (int) ((dimension.getHeight() - Main.HEIGHT) / 2);
155             this.setBounds(x, y, Main.WIDTH, Main.HEIGHT);
156             resize_btn.setIcon(new ImageIcon(resizeUp_image));
157         }
158         Main.maximized = !Main.maximized;
159     });
160 }
```

```
161         minimize_btn.addChangeListener(evt -> {
162             if (minimize_btn.getModel().isPressed()) {
163                 this.setState(JFrame.ICONIFIED);
164                 minimize_btn.setForeground(Colors.primaryColor);
165             } else if (minimize_btn.getModel().isRollover()) {
166                 minimize_btn.setForeground(Colors.secondaryColor);
167             } else {
168                 minimize_btn.setForeground(Colors.primaryColor);
169             }
170         });
171     }
172 }
```

Listing 8: Main Java file

```
1  /*
2  * Class that does everything that we wanna do with mongodb. Things like inserting
3  * , deleting, creating and fetching data.
4  */
5  package org.howmuch;
6
7  import com.mongodb.MongoClient;
8  import com.mongodb.client.*;
9  import org.bson.Document;
10
11  public class MongoManager {
12      static MongoDB database;
13      public static String MONGO_DATABASE_NAME = "HowMuch";
14      public static int MONGO_PORT_NO = 27017;
15      public static String MONGO_HOST = "localhost";
16
17      public static String[] fetchDataFromMongo(String currentTopic, int randomIndex
18      ) {
19          MongoCollection<org.bson.Document> collection = database.getCollection(
20          currentTopic);
21          FindIterable<Document> iterDoc = collection.find();
22          int i = 0;
23          for (Document document : iterDoc) {
24              System.out.println(document);
25              if (i == randomIndex) {
26                  return new String[] { (String) document.get("Name"), document.
27                  getString("Price"),
28                      document.getString("Image") };
29              }
30              i++;
31          }
32          return new String[] { "Sadly Not Found", "Sadly Not Found", "Sadly Not
33          Found" };
34      }
35
36      public static boolean establishConnectionWithMongo() {
37          // Creating a MongoDB client
38          try {
39              MongoClient mongoClient = new MongoClient(MONGO_HOST, MONGO_PORT_NO);
40              // Connecting to the database
41              database = mongoClient.getDatabase(MONGO_DATABASE_NAME);
42              System.out.println("Connected Successfully to mongoDb");
43          } catch (Exception e) {
44              e.printStackTrace();
45          }
46      }
47  }
```

```
40         return true;
41
42     } catch (Exception e) {
43         System.out.println("Couldnt establish connection due to some reason");
44         System.out.println(e.getMessage());
45         return false;
46     }
47 }
48
49 public static void addDataToMongo(String Topic, String[] data) {
50     try {
51         MongoClient mongoClient = new MongoClient(MONGO_HOST, MONGO_PORT_NO);
52         // Connecting to the database
53         database = mongoClient.getDatabase(MONGO_DATABASE_NAME);
54         Topic = Topic.substring(0, 1).toUpperCase() + Topic.substring(1);
55         MongoCollection<Document> collection = database.getCollection(Topic);
56         Document dataDocToAdd = new Document();
57         dataDocToAdd.append("Name", data[0]);
58         dataDocToAdd.append("Price", data[1]);
59         dataDocToAdd.append("Image", data[2]);
60         collection.insertOne(dataDocToAdd);
61         System.out.println(data[0]);
62         System.out.println("\n\nAdded record to mongo-----\n\n");
63     } catch (Exception e) {
64         System.out.println("Couldnt add data");
65     }
66 }
67
68 public static void clearMongoDb() {
69     MongoCollection<Document> collection = database.getCollection(Main.Topics
70 [0]);
71     collection.drop();
72     collection = database.getCollection(Main.Topics[1]);
73     collection.drop();
74     collection = database.getCollection(Main.Topics[2]);
75     collection.drop();
76     collection = database.getCollection(Main.Topics[3]);
77     collection.drop();
78 }
```

Listing 9: Main Java file

```
1  /*
2   * An Important class, As it controls a lot of the important variables, and all
3   * interactions with the Local CSV File databases.
4   */
5  package org.howmuch;
6
7  import com.mongodb.client.MongoDatabase;
8  import com.opencsv.CSVReader;
9  import com.opencsv.CSVWriter;
10 import org.apache.commons.io.FileUtils;
11
12 import java.io.File;
13 import java.io.FileReader;
14 import java.io.FileWriter;
15 import java.io.IOException;
```

```
16 import java.util.Arrays;
17 import java.util.List;
18 import java.util.Objects;
19
20 public class DataBaseManager {
21
22     public static String LOCAL_DATAFOLDER = "/run/media/krishnaraj/Programs/Java/
How Much/src/main/resources/data";
23     public static String LOCAL_CSV_FOLDER = "/run/media/krishnaraj/Programs/Java/
How Much/src/main/resources/data/csvs";
24     public static String LOCAL_IMG_FOLDER = "/run/media/krishnaraj/Programs/Java/
How Much/src/main/resources/data/images";
25     public static String LOCAL_BACKUP_DATAFOLDER = "/run/media/krishnaraj/Programs
/Java/How Much/src/main/resources/data_backup";
26     public static String LOCAL_BACKUP_CSV_FOLDER = "/run/media/krishnaraj/Programs
/Java/How Much/src/main/resources/data_backup/csvs";
27     public static String LOCAL_BACKUP_IMG_FOLDER = "/run/media/krishnaraj/Programs
/Java/How Much/src/main/resources/data_backup/images";
28     public static String USERDATA_FILEPATH = "/run/media/krishnaraj/Programs/Java/
How Much/src/main/resources/data/user_details.csv";
29     public static String BACKUP_USERDATA_FILEPATH = "/run/media/krishnaraj/
Programs/Java/How Much/src/main/resources/data_backup/user_details.csv";
30     public static String LOCAL_DATEFILE = "/run/media/krishnaraj/Programs/Java/How
Much/src/main/resources/data/dateUpdated.txt";
31     public static String LOCAL_MONGODATEFILE = "/run/media/krishnaraj/Programs/
Java/How Much/src/main/resources/data/MongoDateUpdated.txt";
32     public static String LOCAL_BACKUP_DATEFILE = "/run/media/krishnaraj/Programs/
Java/How Much/src/main/resources/data_backup/dateUpdated.txt";
33
34     static String currentUsername = "guest";
35     static int USER_INDEX = -1;
36     static String currentPassword = "guest";
37     static int currentScore = 0;
38
39     /**
40      * Brutally Clear the images and csv in the local Database and start fresh
with
41      * only files.
42      */
43     public static void clearLocalDatabase() {
44         try {
45             // Delete all pre existing images
46             File data_deleter = new File(LOCAL_IMG_FOLDER);
47             listFilesForFolder(data_deleter);
48             for (File subfile : Objects.requireNonNull(data_deleter.listFiles()))
49             {
50                 if (subfile.isDirectory()) {
51                     for (File f : Objects.requireNonNull(subfile.listFiles())) {
52                         f.delete();
53                     }
54                 }
55             }
56
57             // Also clear the csv files.
58             data_deleter = new File(LOCAL_CSV_FOLDER);
59             listFilesForFolder(data_deleter);
60             for (File subfile : Objects.requireNonNull(data_deleter.listFiles()))
61             {
62                 subfile.delete();
```

```
61     }
62
63     // Recreate them.
64     File createfiles = new File(LOCAL_BACKUP_CSV_FOLDER + "/" + Main.
Topics[0].toLowerCase() + ".csv");
65     createfiles.createNewFile();
66     createfiles = new File(LOCAL_BACKUP_CSV_FOLDER + "/" + Main.Topics[1].
toLowerCase() + ".csv");
67     createfiles.createNewFile();
68     createfiles = new File(LOCAL_BACKUP_CSV_FOLDER + "/" + Main.Topics[2].
toLowerCase() + ".csv");
69     createfiles.createNewFile();
70     createfiles = new File(LOCAL_BACKUP_CSV_FOLDER + "/" + Main.Topics[3].
toLowerCase() + ".csv");
71     createfiles.createNewFile();
72     } catch (Exception e) {
73         System.out.println("Some io exception occured");
74     }
75 }
76
77 /*
78  * Simply displays every file in a directory
79  */
80 public static void listFilesForFolder(final File folder) {
81     Arrays.stream(folder.listFiles()).forEach(fileEntry -> {
82         if (fileEntry.isDirectory()) {
83             listFilesForFolder(fileEntry);
84         } else {
85             System.out.println(fileEntry.getName());
86             System.out.println(fileEntry.getPath());
87         }
88     });
89 }
90
91 /*
92  * Adds a new user to the local CSV Database. Creates that file if it doesnt
93  * exist.
94  */
95 public static void addNewUser() {
96     System.out.println("gonna add new user");
97     File userDatafile = new File(USERDATA_FILEPATH);
98
99     try (CSVReader reader = new CSVReader(new FileReader(userDatafile), ',', ''))
100 {
101         List<String[]> csvBody = reader.readAll();
102         USER_INDEX = csvBody.size();
103     } catch (IOException e) {
104         throw new RuntimeException(e);
105     }
106
107     // append the new user to the login file.
108     try (FileWriter userDataFileWriter = new FileWriter(userDatafile, true)) {
109         // create CSVWriter object filewriter object as parameter
110         try (CSVWriter writer = new CSVWriter(userDataFileWriter, CSVWriter.
DEFAULT_SEPARATOR,
111             CSVWriter.NO_QUOTE_CHARACTER, CSVWriter.
DEFAULT_ESCAPE_CHARACTER, CSVWriter.DEFAULT_LINE_END)) {
```



```
113         String[] data = { currentUsername, currentPassword, String.valueOf
114         (currentScore) };
115         writer.writeNext(data);
116         System.out.println("added new user");
117     }
118     } catch (IOException e) {
119         System.out.println("Cant open user data file. ");
120     }
121 }
122
123 public static void addDataToCSV(String filePath, String[] data) {
124     File userDatafile = new File(filePath);
125
126     // append the new user to the login file.
127     try (FileWriter userDataFileWriter = new FileWriter(userDatafile, true)) {
128
129         // create CSVWriter object filewriter object as parameter
130         try (CSVWriter writer = new CSVWriter(userDataFileWriter, CSVWriter.
131         DEFAULT_SEPARATOR,
132         CSVWriter.NO_QUOTE_CHARACTER, CSVWriter.
133         DEFAULT_ESCAPE_CHARACTER, CSVWriter.DEFAULT_LINE_END)) {
134             // System.out.println(Arrays.toString(data));
135             writer.writeNext(data);
136         }
137     } catch (IOException e) {
138         System.out.println("Cant open user data file. ");
139     }
140 }
141
142 public static boolean doesUsernameExist(String username) {
143     File inputFile = new File(USERDATA_FILEPATH);
144     try (CSVReader reader = new CSVReader(new FileReader(inputFile), ',')) {
145         List<String[]> csvBody = reader.readAll();
146         for (String[] s : csvBody) {
147             if (s[0].equals(username)) {
148                 System.out.println("User Already Exists");
149                 return true;
150             }
151         }
152     } catch (IOException e) {
153         System.out.println("couldnt create csvreader in username exists
154         checker method. ");
155     }
156     return false;
157 }
158
159 public static boolean doesPasswordMatch(String username, String password) {
160     File inputFile = new File(USERDATA_FILEPATH);
161     try (CSVReader reader = new CSVReader(new FileReader(inputFile), ',')) {
162         List<String[]> csvBody = reader.readAll();
163         for (int i = 0; i < csvBody.size(); i++) {
164             String[] s = csvBody.get(i);
165             if (s[0].equals(username)) {
166                 System.out.println("User Found");
167                 if (s[1].equals(password)) {
168                     System.out.println("Password Matches");
169                     USER_INDEX = i;
170                 }
171             }
172         }
173     }
174 }
```

```
168         return true;
169     } else
170         return false;
171     }
172 }
173 } catch (IOException e) {
174     System.out.println("couldnt create csvreader in password matching
method");
175 }
176 return false;
177 }
178
179 public static List<String[]> getStoredUserScores() {
180     File inputFile = new File(USERDATA_FILEPATH);
181     List<String[]> csvBody = null;
182     try (CSVReader reader = new CSVReader(new FileReader(inputFile), ',')) {
183         csvBody = reader.readAll();
184         return csvBody;
185     } catch (IOException e) {
186         System.out.println("couldnt create csvreader in userscore method");
187     }
188     return csvBody;
189 }
190
191 public static void updateUserScore() {
192
193     File inputFile = new File(USERDATA_FILEPATH);
194
195     List<String[]> csvBody;
196     try (CSVReader reader = new CSVReader(new FileReader(inputFile), ',')) {
197         csvBody = reader.readAll();
198         csvBody.get(USER_INDEX)[2] = String.valueOf(currentScore);
199     } catch (IOException e) {
200         throw new RuntimeException(e);
201     }
202
203     try (CSVWriter writer = new CSVWriter(new FileWriter(inputFile), ',')) {
204         writer.writeAll(csvBody);
205         writer.flush();
206     } catch (IOException e) {
207         throw new RuntimeException(e);
208     }
209 }
210
211 public static void createLocalDatabaseBackupOfUsers() {
212     System.out.println("-----CREATING LOCAL DATABASE BACKUP of the
user file-----");
213     try {
214         File sourceDirectory = new File(USERDATA_FILEPATH);
215         File destinationDirectory = new File(BACKUP_USERDATA_FILEPATH);
216         FileUtils.copyFile(sourceDirectory, destinationDirectory);
217     } catch (IOException e) {
218         throw new RuntimeException(e);
219     }
220 }
221
222 public static void createLocalDatabaseBackup() {
223     try {
224         System.out.println("-----CREATING LOCAL DATABASE BACKUP
```

```
-----");
225     // Delete all pre existing images
226     File data_deleter = new File(LOCAL_BACKUP_IMG_FOLDER);
227     // listFilesForFolder(data_deleter);
228     for (File subfile : Objects.requireNonNull(data_deleter.listFiles()))
{
229         if (subfile.isDirectory()) {
230             for (File f : Objects.requireNonNull(subfile.listFiles())) {
231                 f.delete();
232             }
233         }
234     }
235     data_deleter = new File(LOCAL_BACKUP_CSV_FOLDER);
236     for (File subfile : Objects.requireNonNull(data_deleter.listFiles()))
{
237         subfile.delete();
238     }
239     File createfiles = new File(LOCAL_BACKUP_CSV_FOLDER + "/" + Main.
Topics[0].toLowerCase() + ".csv");
240     createfiles.createNewFile();
241     createfiles = new File(LOCAL_BACKUP_CSV_FOLDER + "/" + Main.Topics[1].
toLowerCase() + ".csv");
242     createfiles.createNewFile();
243     createfiles = new File(LOCAL_BACKUP_CSV_FOLDER + "/" + Main.Topics[2].
toLowerCase() + ".csv");
244     createfiles.createNewFile();
245     createfiles = new File(LOCAL_BACKUP_CSV_FOLDER + "/" + Main.Topics[3].
toLowerCase() + ".csv");
246     createfiles.createNewFile();
247 } catch (Exception e) {
248     System.out.println("Some io exception occured");
249 }
250
251 try {
252     File sourceDirectory = new File(LOCAL_DATAFOLDER);
253     File destinationDirectory = new File(LOCAL_BACKUP_DATAFOLDER);
254     FileUtils.copyDirectory(sourceDirectory, destinationDirectory);
255
256 } catch (IOException e) {
257     throw new RuntimeException(e);
258 }
259 }
260
261 public static String[] readFromLocalDatabase(String Topic, int index) {
262     File inputFile;
263     if (Main.isLocalDatabaseUpToDate) {
264         System.out.println("running from the local database");
265         inputFile = new File(LOCAL_CSV_FOLDER + '/' + Topic.toLowerCase() + ".
csv");
266         try (CSVReader reader = new CSVReader(new FileReader(inputFile), ','))
{
267             List<String[]> csvBody = reader.readAll();
268             if (index > csvBody.size()) {
269                 return csvBody.get(csvBody.size() - 1);
270             }
271             return csvBody.get(index);
272 } catch (IOException e) {
273     throw new RuntimeException(e);
274 }
```

```
275
276     } else {
277         System.out.println("running from the backup local database");
278         inputFile = new File(LOCAL_BACKUP_CSV_FOLDER + '/' + Topic.toLowerCase
() + ".csv");
279         try (CSVReader reader = new CSVReader(new FileReader(inputFile), ','))
{
280             List<String[]> csvBody = reader.readAll();
281             if (index > csvBody.size()) {
282                 String[] s;
283                 s = csvBody.get(csvBody.size() - 1);
284                 s[2] = s[2].replace("/data/", "/data_backup/");
285                 return s;
286             }
287             String[] s;
288             s = csvBody.get(index);
289             s[2] = s[2].replace("/data/", "/data_backup/");
290             return s;
291         } catch (IOException e) {
292             throw new RuntimeException(e);
293         }
294     }
295 }
296
297 public static int findLength(String Topic) {
298     File inputFile;
299     if (Main.isLocalDatabaseUpToDate) {
300         inputFile = new File(LOCAL_CSV_FOLDER + '/' + Topic.toLowerCase() + ".
csv");
301     } else {
302         inputFile = new File(LOCAL_BACKUP_CSV_FOLDER + '/' + Topic.toLowerCase
() + ".csv");
303     }
304     try (CSVReader reader = new CSVReader(new FileReader(inputFile), ',')) {
305         List<String[]> csvBody = reader.readAll();
306         return csvBody.size();
307     } catch (IOException e) {
308         throw new RuntimeException(e);
309     }
310 }
311 }
312 }
```

Listing 10: Main Java file

```
1 package org.howmuch;
2
3 import javax.swing.*;
4 import java.awt.*;
5 import java.awt.event.ComponentAdapter;
6 import java.awt.event.ComponentEvent;
7 import java.util.ArrayList;
8 import java.util.Arrays;
9 import java.util.Collections;
10
11 import static org.howmuch.Main.*;
12
13 public class HighscoreFrame extends JFrame {
14     BackgroundPanel backgroundPanel;
```

```
15 JButton backToMenu_btn;
16 JTextArea highScores_txtArea;
17
18 HighscoreFrame() {
19     backgroundPanel = new BackgroundPanel();
20
21     this.setTitle("How Much? ");
22     if (maximized) {
23         this.setExtendedState(MAXIMIZED_BOTH);
24     } else {
25         this.setPreferredSize(new Dimension(Main.WIDTH, Main.HEIGHT));
26     }
27     this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
28     this.setResizable(true);
29     this.setUndecorated(true);
30     this.setMinimumSize(new Dimension(1280, 720));
31
32     createFonts();
33     createBasicButtonPanel();
34     createButtons();
35     createLabels();
36     reassignColors();
37     reassignBounds();
38
39     this.addComponentListener(new ComponentAdapter() {
40         @Override
41         public void componentResized(ComponentEvent e) {
42             reassignBounds();
43             repaint();
44         }
45     });
46
47     this.add(highScores_txtArea);
48     this.add(backToMenu_btn);
49     this.add(basicButtons_pnl);
50     this.add(backgroundPanel);
51     this.pack();
52     this.setLocationRelativeTo(null);
53     this.setVisible(true);
54 }
55
56 private void createLabels() {
57     highScores_txtArea = new JTextArea();
58     int peopleCount = 0;
59     java.util.List<String[]> userData = DataBaseManager.getStoredUserScores();
60     for (int i = 0; i < userData.size(); i++) {
61         System.out.println(Integer.parseInt(userData.get(i)[2]));
62     }
63     ArrayList<Integer> scores = new ArrayList<>();
64     StringBuilder sb = new StringBuilder();
65     for (int i = 0; i < userData.size(); i++) {
66         System.out.println(Integer.parseInt(userData.get(i)[2]));
67         scores.add(Integer.parseInt(userData.get(i)[2]));
68     }
69     scores.sort(Collections.reverseOrder());
70
71     for (int i = 0; i < scores.size(); i++) {
72         System.out.println(Integer.parseInt(String.valueOf(scores.get(i))));
73     }
```

```
74         System.out.println("fianls");
75         for (int i = 0; i < scores.size(); i++) {
76             for (int j = 0; j < userData.size(); j++) {
77                 if (Integer.valueOf(userData.get(j)[2]).equals(scores.get(i))) {
78                     System.out.println(Arrays.toString(userData.get(j)));
79                     sb.append(userData.get(j)[0] + "    -    " + userData.get(j)[2]
+ "\n");
80                     peopleCount++;
81                     if (peopleCount == 5) {
82                         break;
83                     }
84                 }
85             }
86             if (peopleCount == 5) {
87                 break;
88             }
89         }
90         highScores_txtArea.setText(String.valueOf(sb));
91         highScores_txtArea.setAlignmentY(Box.CENTER_ALIGNMENT);
92         highScores_txtArea.setAlignmentX(Box.LEFT_ALIGNMENT);
93         highScores_txtArea.setOpaque(true);
94         highScores_txtArea.setBorder(null);
95         highScores_txtArea.setLineWrap(true);
96     }
97
98     private void reassignBounds() {
99         Dimension screenSize = this.getSize();
100
101         // The back to menu mode label
102         backToMenu_btn.setBounds((int) (0.015 * screenSize.getWidth()), (int)
(0.80 * screenSize.getHeight()),
103             (int) (0.20 * screenSize.getWidth()), (int) (0.07 * screenSize.
getHeight()));
104         backToMenu_btn.setFont(buttonFont.deriveFont((float) (0.05 * getHeight()))
);
105
106         // The Entire basic button panel for closing minimizing and stuff
107         basicButtons_pnl.setBounds(this.getWidth() - (exit_btn.getWidth() * 3) -
40, 10, exit_btn.getWidth() * 3 + 35,
108             exit_btn.getHeight());
109         highScores_txtArea.setBounds((int) (0.60 * screenSize.getWidth()), (int)
(0.38 * screenSize.getHeight()),
110             (int) (0.60 * screenSize.getWidth()), 700);
111         highScores_txtArea.setFont(textFont.deriveFont(44f));
112
113     }
114
115     private void reassignColors() {
116         Colors.reassignColors();
117         if (Colors.DarkMode) {
118             backgroundPanel.setBackground("/run/media/krishnaraj/Programs/Java/How
Much/src/main/resources/images/highscore dark.png");
119         } else {
120             backgroundPanel.setBackground("/run/media/krishnaraj/Programs/Java/How
Much/src/main/resources/images/highscore.png");
121         }
122         backToMenu_btn.setBackground(Colors.primaryColor);
123         backToMenu_btn.setForeground(Colors.bgColor);
124     }
```

```
125         basicButtons_pnl.setBackground(Colors.bgColor);
126         exit_btn.setBackground(Colors.bgColor);
127         resize_btn.setBackground(Colors.bgColor);
128         minimize_btn.setBackground(Colors.bgColor);
129         highScores_txtArea.setBackground(Colors.bgColor);
130         highScores_txtArea.setForeground(Colors.primaryColor);
131     }
132
133     private void createButtons() {
134         backToMenu_btn = new JButton();
135         backToMenu_btn.setText("Back to Menu");
136         backToMenu_btn.setAlignmentY(Box.CENTER_ALIGNMENT);
137         backToMenu_btn.setAlignmentX(Box.LEFT_ALIGNMENT);
138         backToMenu_btn.setFocusPainted(false);
139         backToMenu_btn.setContentAreaFilled(false);
140         backToMenu_btn.setOpaque(true);
141         backToMenu_btn.setBorder(null);
142         backToMenu_btn.addChangeListener(evt -> {
143             if (backToMenu_btn.getModel().isPressed()) {
144                 backToMenu_btn.setForeground(Colors.bgColor);
145             } else if (backToMenu_btn.getModel().isRollover()) {
146                 backToMenu_btn.setForeground(Colors.accentColor);
147             } else {
148                 backToMenu_btn.setForeground(Colors.bgColor);
149             }
150         });
151         backToMenu_btn.addActionListener(e -> {
152             this.setVisible(false);
153             this.dispose();
154             grantAccess = true;
155             Main.changeFrame(1);
156         });
157
158         // Removing Change and Action Listeners.
159         GameFrame.removeAllChangeAndActionListenersFromBasicButtons();
160
161         exit_btn.addChangeListener(evt -> {
162             if (exit_btn.getModel().isPressed()) {
163                 exit_btn.setForeground(Colors.primaryColor);
164                 Main.changeFrame(0);
165             } else if (exit_btn.getModel().isRollover()) {
166                 exit_btn.setForeground(Colors.secondaryColor);
167             } else {
168                 exit_btn.setForeground(Colors.primaryColor);
169             }
170         });
171         resize_btn.addActionListener(e -> {
172             if (!Main.maximized) {
173                 this.setExtendedState(MAXIMIZED_BOTH);
174                 resize_btn.setIcon(new ImageIcon(resizeDown_image));
175             } else {
176                 this.setExtendedState(JFrame.NORMAL);
177                 this.setLocationRelativeTo(null);
178                 Dimension dimension = Toolkit.getDefaultToolkit().getScreenSize();
179                 int x = (int) ((dimension.getWidth() - Main.WIDTH) / 2);
180                 int y = (int) ((dimension.getHeight() - Main.HEIGHT) / 2);
181                 this.setBounds(x, y, Main.WIDTH, Main.HEIGHT);
182                 resize_btn.setIcon(new ImageIcon(resizeUp_image));
183             }
184         });
185     }
```

```
184         Main.maximized = !Main.maximized;
185     });
186
187     minimize_btn.addChangeListener(evt -> {
188         if (minimize_btn.getModel().isPressed()) {
189             this.setState(JFrame.ICONIFIED);
190             minimize_btn.setForeground(Colors.primaryColor);
191         } else if (minimize_btn.getModel().isRollover()) {
192             minimize_btn.setForeground(Colors.secondaryColor);
193         } else {
194             minimize_btn.setForeground(Colors.primaryColor);
195         }
196     });
197 }
198 }
```

Listing 11: Main Java file

```
1 package org.howmuch;
2
3 import javax.swing.*;
4 import java.awt.*;
5 import java.awt.event.ComponentAdapter;
6 import java.awt.event.ComponentEvent;
7
8 import static org.howmuch.Main.*;
9
10 public class HelpFrame extends JFrame {
11     BackgroundPanel backgroundPanel;
12     JButton backToMenu_btn;
13
14     HelpFrame() {
15         backgroundPanel = new BackgroundPanel();
16
17         this.setTitle("How Much? ");
18         if (maximized) {
19             this.setExtendedState(MAXIMIZED_BOTH);
20         } else {
21             this.setPreferredSize(new Dimension(Main.WIDTH, Main.HEIGHT));
22         }
23         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
24         this.setResizable(true);
25         this.setUndecorated(true);
26         this.setMinimumSize(new Dimension(1280, 720));
27
28         createFonts();
29         createBasicButtonPanel();
30         createButtons();
31         reassignColors();
32         reassignBounds();
33
34         this.addComponentListener(new ComponentAdapter() {
35             @Override
36             public void componentResized(ComponentEvent e) {
37                 reassignBounds();
38                 repaint();
39             }
40         });
41 }
```



```
42         this.add(backToMenu_btn);
43         this.add(basicButtons_pnl);
44         this.add(backgroundPanel);
45         this.pack();
46         this.setLocationRelativeTo(null);
47         this.setVisible(true);
48     }
49
50     private void reassignBounds() {
51         Dimension screenSize = this.getSize();
52
53         // The back to menu mode label
54         backToMenu_btn.setBounds((int) (0.015 * screenSize.getWidth()), (int)
55         (0.80 * screenSize.getHeight()),
56         (int) (0.20 * screenSize.getWidth()), (int) (0.07 * screenSize.
57         getHeight()));
58         backToMenu_btn.setFont(buttonFont.deriveFont((float) (0.05 * getHeight()))
59         );
60
61         // The Entire basic button panel for closing minimizing and stuff
62         basicButtons_pnl.setBounds(this.getWidth() - (exit_btn.getWidth() * 3) -
63         40, 10, exit_btn.getWidth() * 3 + 35,
64         exit_btn.getHeight());
65
66     }
67
68     private void reassignColors() {
69         Colors.reassignColors();
70         if (Colors.DarkMode) {
71             backgroundPanel.setBackground("/run/media/krishnaraj/Programs/Java/How
72             Much/src/main/resources/images/help and credits dark.png");
73         } else {
74             backgroundPanel.setBackground("/run/media/krishnaraj/Programs/Java/How
75             Much/src/main/resources/images/help and credits.png");
76         }
77         backToMenu_btn.setBackground(Colors.primaryColor);
78         backToMenu_btn.setForeground(Colors.bgColor);
79
80         basicButtons_pnl.setBackground(Colors.bgColor);
81         exit_btn.setBackground(Colors.bgColor);
82         resize_btn.setBackground(Colors.bgColor);
83         minimize_btn.setBackground(Colors.bgColor);
84     }
85
86     private void createButtons() {
87         backToMenu_btn = new JButton();
88         backToMenu_btn.setText("Back to Menu");
89         backToMenu_btn.setAlignmentY(Box.CENTER_ALIGNMENT);
90         backToMenu_btn.setAlignmentX(Box.LEFT_ALIGNMENT);
91         backToMenu_btn.setFocusPainted(false);
92         backToMenu_btn.setContentAreaFilled(false);
93         backToMenu_btn.setOpaque(true);
94         backToMenu_btn.setBorder(null);
95         backToMenu_btn.addActionListener(evt -> {
96             if (backToMenu_btn.getModel().isPressed()) {
97                 backToMenu_btn.setForeground(Colors.bgColor);
98             } else if (backToMenu_btn.getModel().isRollover()) {
99                 backToMenu_btn.setForeground(Colors.accentColor);
100             } else {
```

```
95         backToMenu_btn.setForeground(Colors.bgColor);
96     }
97 });
98 backToMenu_btn.addActionListener(e -> {
99     this.setVisible(false);
100    this.dispose();
101    grantAccess = true;
102    Main.changeFrame(1);
103 });
104
105 // Removing Change and Action Listeners.
106 GameFrame.removeAllChangeAndActionListenersFromBasicButtons();
107
108 exit_btn.addChangeListener(evt -> {
109     if (exit_btn.getModel().isPressed()) {
110         exit_btn.setForeground(Colors.primaryColor);
111         Main.changeFrame(0);
112     } else if (exit_btn.getModel().isRollover()) {
113         exit_btn.setForeground(Colors.secondaryColor);
114     } else {
115         exit_btn.setForeground(Colors.primaryColor);
116     }
117 });
118
119 resize_btn.addActionListener(e -> {
120     if (!Main.maximized) {
121         this.setExtendedState(MAXIMIZED_BOTH);
122         resize_btn.setIcon(new ImageIcon(resizeDown_image));
123     } else {
124         this.setExtendedState(JFrame.NORMAL);
125         this.setLocationRelativeTo(null);
126         Dimension dimension = Toolkit.getDefaultToolkit().getScreenSize();
127         int x = (int) ((dimension.getWidth() - Main.WIDTH) / 2);
128         int y = (int) ((dimension.getHeight() - Main.HEIGHT) / 2);
129         this.setBounds(x, y, Main.WIDTH, Main.HEIGHT);
130         resize_btn.setIcon(new ImageIcon(resizeUp_image));
131     }
132     Main.maximized = !Main.maximized;
133 });
134
135 minimize_btn.addChangeListener(evt -> {
136     if (minimize_btn.getModel().isPressed()) {
137         this.setState(JFrame.ICONIFIED);
138         minimize_btn.setForeground(Colors.primaryColor);
139     } else if (minimize_btn.getModel().isRollover()) {
140         minimize_btn.setForeground(Colors.secondaryColor);
141     } else {
142         minimize_btn.setForeground(Colors.primaryColor);
143     }
144 });
145 }
```

Listing 12: Main Java file

```
1  /*
2  * The class that Performs the role of the heart of the code. It is what downloads
   the images from aamzon, and saves them.
3  * It scraps them, and we get the html page from the website. From there we can
   get the parts that we need as xml and then by parsing that xml
```

```
4  * We can then get exactly what we want.
5  */
6
7  package org.howmuch;
8
9  import java.io.IOException;
10 import java.net.MalformedURLException;
11 import java.net.URL;
12 import java.nio.charset.StandardCharsets;
13 import java.nio.file.Files;
14 import java.nio.file.Path;
15 import java.nio.file.Paths;
16 import java.util.*;
17
18 import com.groupdocs.conversion.Converter;
19 import com.groupdocs.conversion.filetypes.ImageFileType;
20 import com.groupdocs.conversion.options.convert.ImageConvertOptions;
21 import org.apache.commons.io.FileExistsException;
22 import org.w3c.dom.*;
23
24 import javax.xml.parsers.*;
25 import java.io.*;
26
27 import com.gargoylesoftware.htmlunit.*;
28 import com.gargoylesoftware.htmlunit.html.*;
29 import org.xml.sax.SAXException;
30
31 public class AmazonScraper {
32     static Converter converter;
33     static ImageConvertOptions options;
34     static WebClient webClient;
35     static DocumentBuilder builder;
36     static DocumentBuilderFactory factory;
37     public static HashMap<Integer, String[]> searchQueries_map = new HashMap<>();
38     public static String AMAZON_PREFIX_URL = "https://www.amazon.in/s?k=";
39
40     AmazonScraper() {
41         options = new ImageConvertOptions();
42         options.setFormat(ImageFileType.Png);
43         fillSearchQueries();
44
45         factory = DocumentBuilderFactory.newInstance();
46         try {
47             builder = factory.newDocumentBuilder();
48         } catch (ParserConfigurationException e) {
49             throw new RuntimeException(e);
50         }
51
52         // Define and declare basic web browser
53         webClient = new WebClient(BrowserVersion.CHROME);
54         webClient.getOptions().setCssEnabled(false);
55         webClient.getOptions().setThrowExceptionOnFailingStatusCode(false);
56         webClient.getOptions().setJavaScriptEnabled(false);
57         webClient.getOptions().setThrowExceptionOnScriptError(false);
58         webClient.getOptions().setPrintContentOnFailingStatusCode(false);
59     }
60
61     public static void fillSearchQueries() {
62         System.out.println(Arrays.toString(Main.Topics));
```

```
63
64 //      This is the final stuff here, but is commented out for quicker
        debugging.
65
66      searchQueries_map.put(0, new String[]{"Televisions", "Mobile Phones",
67      "Laptops", "Iphone", "Macbook", "Refrigerators", "Washing Machines", "
Smart Watches", "Gaming Laptops", "Computer Accessories", "GPUs", "Tablets",
68      "Playstation", "Xbox"});
69      searchQueries_map.put(1, new String[]{"Mens TShirts", "Formal Suits", "
Mens Casual Wear", "Womens Casual Wear", "Womens Formal Wear", "Kids Clothes",
70      "Makeup", "Beauty Products", "Analog Watches", "Earrings", "Necklaces",
71      "Jewellery", "Branded Clothes", "Gold Jewellery", "Shoes"});
72      searchQueries_map.put(2, new String[]{"Furniture", "Tape", "Stationary",
"Cutlery", "Kitchen Products", "Toothpaste", "Chocolates", "Soaps", "Water
Bottles", "Carpets", "Sofa Sets", "Tables and Desks", "Cleaning Products"});
73      searchQueries_map.put(3, new String[]{"Gifts", "Car Appliances", "Diwali
Lights", "Decoration", "Birthday Decor", "Lenses"});
74 //
75 //      searchQueries_map.put(0, new String[] { "8k OLED Televisions" });
76 //      searchQueries_map.put(1, new String[] { "Kurti", "Womens Dresses" });
77 //      searchQueries_map.put(2, new String[] { "Furniture" });
78 //      searchQueries_map.put(3, new String[] { "Gifts" });
79
80      for (Map.Entry<Integer, String[]> m : searchQueries_map.entrySet()) {
81          System.out.println(m.getKey() + " " + Arrays.toString(m.getValue()));
82      }
83  }
84
85  /*
86   * Main function that scraps amazon
87   */
88  public static void scrapAndSave() throws ParserConfigurationException,
IOException, SAXException {
89      for (Map.Entry<Integer, String[]> topic : searchQueries_map.entrySet()) {
90          for (int topic_queries = 0; topic_queries < topic.getValue().length;
topic_queries++) {
91              for (int page = 1; page < 2; page++) {
92                  try {
93                      HtmlPage urlHTML = webClient.getPage(
94                          AMAZON_PREFIX_URL + topic.getValue()[topic_queries
] + "&crid=2JOW4XXQM1KWM&sprefix="
95                          + topic.getValue()[topic_queries] + "%2
Caps%2C220&ref=sr_pg_" + page);
96                      webClient.getCurrentWindow().getJobManager().removeAllJobs
();
97
98                      List<HtmlElement> searchResults_List = urlHTML
99                          .getByXPath("//div[@data-component-type='s-search-
result']");
100                      int max = Math.min(searchResults_List.size(), 10);
101                      for (int searchResult = 0; searchResult < max;
searchResult++) {
102                          HtmlDivision divv = (HtmlDivision) searchResults_List.
get(searchResult);
103
104                          StringBuilder xmlStringBuilder = new StringBuilder();
105                          xmlStringBuilder.append("<?xml version=\"1.0\"?>");
106                          xmlStringBuilder.append(divv.asXml());
107
```

```
108         ByteArrayInputStream input = new ByteArrayInputStream(
109             xmlStringBuilder.toString().getBytes(
110                 StandardCharsets.UTF_8));
111         xmlParser(input,
112             DataBaseManager.LOCAL_IMG_FOLDER + '/' + Main.
113             Topics[topic.getKey()].toLowerCase()
114                 + '/' + topic.getValue()[topic_queries
115 ] + searchResult + ".webp",
116             Main.Topics[topic.getKey()].toLowerCase());
117     }
118     } catch (IOException e) {
119         System.out.println("An error occurred: " + e);
120     }
121 }
122 }
123 }
124
125 /*
126  * Function that uses Xpath to go through the xml code, parse it and then
127  * return
128  * the necessary strings.
129  */
130 public static void xmlParser(ByteArrayInputStream inputFile, String
131 imageFilePath, String Topic)
132     throws ParserConfigurationException, IOException, SAXException {
133
134     String productName = "Sadly Not Found", productPrice = "Sadly Not Found",
135     productImagePath = "Sadly Not Found";
136
137     Document doc = builder.parse(inputFile);
138
139     NodeList nListImages = doc.getElementsByTagName("img");
140     Element imageElement = (Element) nListImages.item(0);
141     String[] allImageURLs = imageElement.getAttribute("srcset").split(",");
142     String hdImageIrl = allImageURLs[allImageURLs.length - 1].split(" ")[1];
143
144     productName = imageElement.getAttribute("alt");
145     productName = productName.replace(",", " -");
146     if (productName.contains("Sponsored Ad - ")) {
147         productName = productName.replace("Sponsored Ad - ", "");
148     }
149     System.out.println(productName);
150
151     saveImage(hdImageIrl, imageFilePath);
152     productImagePath = imageFilePath.replace(".webp", ".png");
153
154     NodeList nList = doc.getElementsByTagName("span");
155     for (int i = 0; i < nList.getLength(); i++) {
156         Element currElement = (Element) nList.item(i);
157         if (currElement.getAttribute("class").equals("a-price-whole")) {
158             System.out.println("Price is: ");
159             productPrice = currElement.getTextContent().replace(".", "");
160             productPrice = productPrice.strip().replace(",", "");
161             System.out.println(productPrice);
162         }
163     }
164 }
```

```
161     String[] data = new String[] { productName, productPrice, productImagePath
162     };
163     if (productName.equalsIgnoreCase("Sadly Not Found") || productPrice.
164     equalsIgnoreCase("Sadly Not Found")
165     || productImagePath.equalsIgnoreCase("Sadly Not Found")) {
166         System.out.println("Not adding this data");
167     } else {
168         if (!Main.isLocalDatabaseUpToDate) {
169             DataBaseManager.addDataToCSV(DataBaseManager.LOCAL_CSV_FOLDER + '/'
170             + Topic + ".csv", data);
171         }
172         if (Main.usingMongo) {
173             if (!Main.isMongoUpToDate) {
174                 MongoManager.addDataToMongo(Topic, data);
175             }
176         }
177     }
178 }
179
180 /*
181  * Simple function save the image, but coz we cant work with webp images, we
182  * gotta convert them to png and save them right away.
183  */
184 public static void saveImage(String URLst, String filepath) {
185     if (new File(filepath).exists()) {
186         System.out.println("File Exists, gonna replace it");
187         new File(filepath).delete();
188     }
189
190     try (InputStream in = new URL(URLst).openStream()) {
191         Files.copy(in, Paths.get(filepath));
192         converter = new Converter(filepath);
193         filepath = filepath.replace(".webp", ".png");
194         converter.convert(filepath, options);
195         Files.delete(Path.of(filepath.replace(".png", ".webp")));
196     } catch (IOException e) {
197         System.out.println("we got some issue here with this file");
198     }
199 }
```

Listing 13: Main Java file