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 24, 2022

Contents

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 prevelant 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 leanring 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 databse 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 Webpages of Amazon and 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

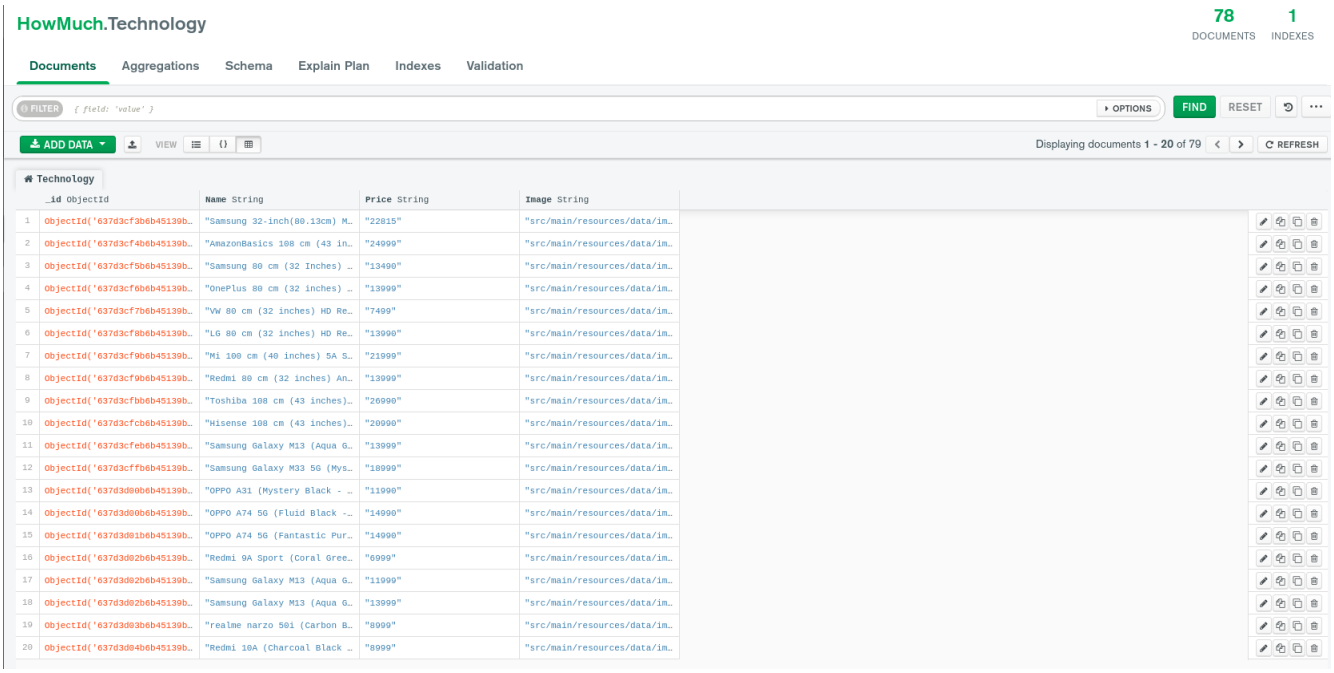

Figure 1: A Screenshot of the MongoDB Compass Showing Records Stored in the Teachnology Schema

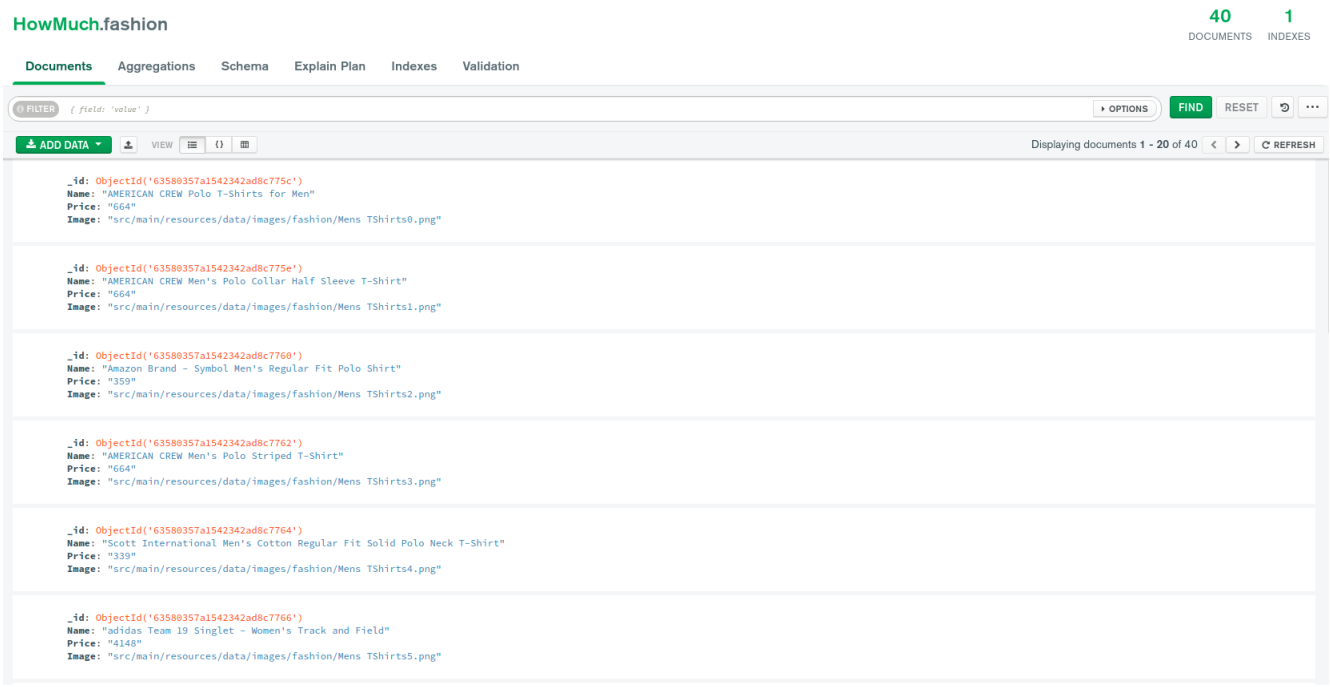

Figure 2: Record Showing the Fashion Schema Documents

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

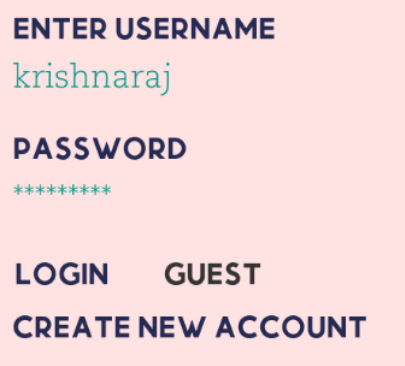

Figure 7:

- All the Requirements of a simple login are satisfied here.
- After the user inputs the username, it is valided 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

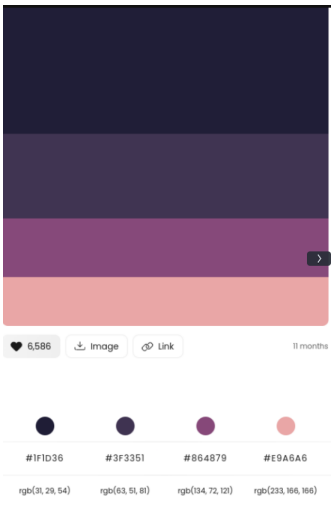

Figure 8: Color Palette

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

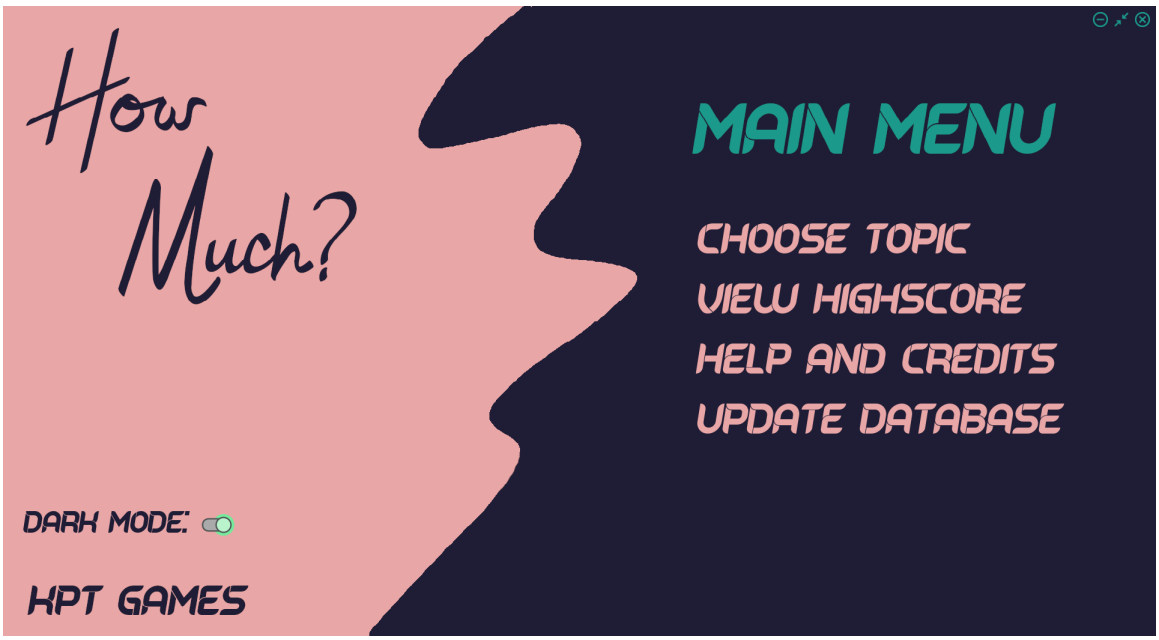

Figure 10:

9.3 The Topic Selection Screen

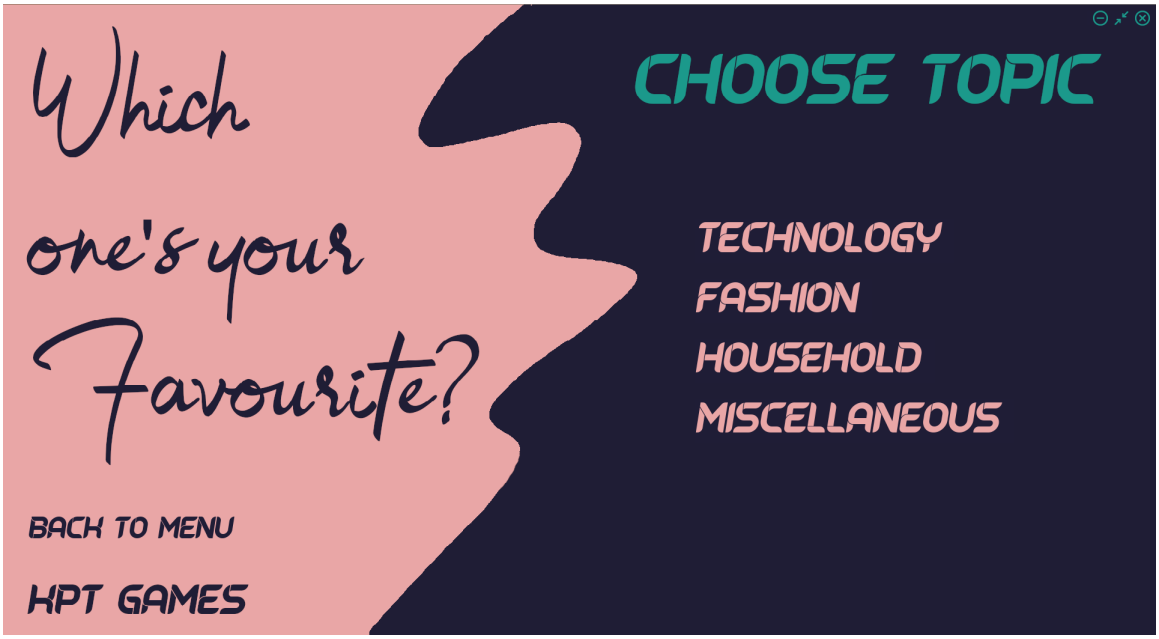

Figure 11: The Login page after a successful login

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:

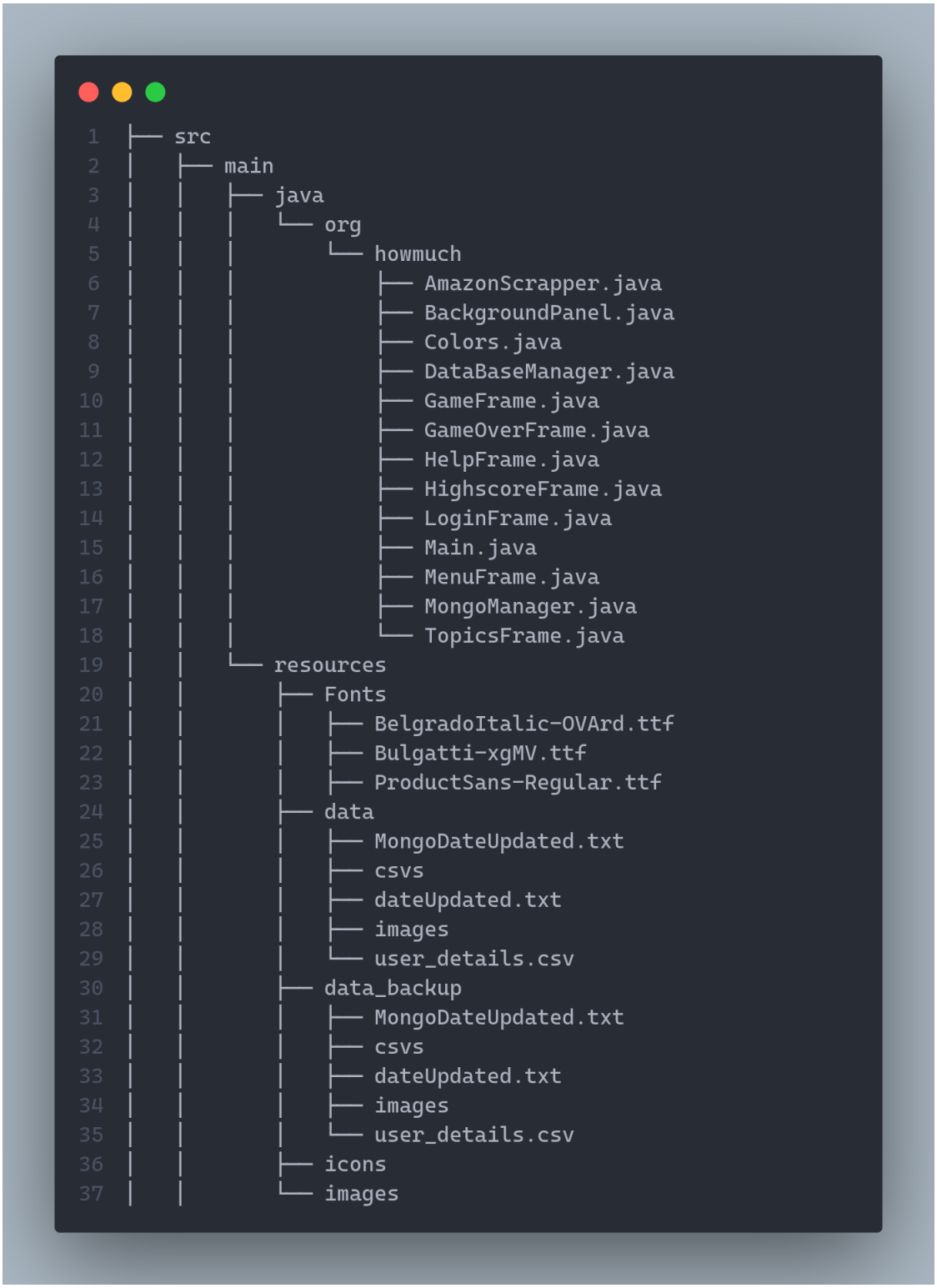
src
main
java
org
howmuch
AmazonScrapper.java
BackgroundPanel.java
Colors.java
DataBaseManager.java
GameFrame.java
GameOverFrame.java
HelpFrame.java
HighscoreFrame.java
LoginFrame.java
Main.java
MenuFrame.java
MongoManager.java
TopicsFrame.java
resources
Fonts
BelgradoItalic-OVArd.ttf
Bulgatti-xgMV.ttf
ProductSans-Regular.ttf
data
MongoDateUpdated.txt
csvs
dateUpdated.txt
images
user_details.csv
data_backup
MongoDateUpdated.txt
csvs
dateUpdated.txt
images
user_details.csv
icons
images

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 defind 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

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache
    .org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>org.example</groupId>
    <artifactId>How-Much</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <maven.compiler.source>18</maven.compiler.source>
        <maven.compiler.target>18</maven.compiler.target>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>

    <repositories>
        <repository>
            <id>groupdocs-artifacts-repository</id>
            <name>GroupDocs Artifacts Repository</name>
            <url>https://releases.groupdocs.com/java/repo/</url>
        </repository>
    </repositories>

    <dependencies>
        <dependency>

            <groupId>net.sourceforge.htmlunit</groupId>
            <artifactId>htmlunit</artifactId>
            <version>2.60.0</version>
        </dependency>
        <dependency>
            <groupId>com.fasterxml.jackson.core</groupId>
            <artifactId>jackson-databind</artifactId>
            <version>2.13.2.1</version>
        </dependency>

        <dependency>
            <groupId>org.mongodb</groupId>
            <artifactId>mongo-java-driver</artifactId>
            <version>3.12.2</version>
        </dependency>

        <dependency>
            <groupId>com.opencsv</groupId>
            <artifactId>opencsv</artifactId>
            <version>4.1</version>
        </dependency>

        <dependency>
            <groupId>com.groupdocs</groupId>
            <artifactId>groupdocs-conversion</artifactId>
            <version>22.8.1</version>
        </dependency>

```

```
    </dependencies>

</project>
```

Listing 1: Maven Dependency File

13 Code Files

```
package org.howmuch;

import java.awt.*;

public class Colors {

    static Boolean DarkMode = false;

    // Light mode Colors
    static Color light_bgColor = new Color(255, 227, 227);
    static Color light_primaryColor = new Color(38, 42, 83);
    static Color light_secondaryColor = new Color(255, 160, 160);
    static Color light_accentColor = new Color(27, 153, 139);

    // Dark Mode Colors
    static Color dark_bg_color = new Color(31, 29, 54);
    static Color dark_primaryColor = new Color(233, 166, 166);
    static Color dark_secondaryColor = new Color(175, 89, 159);
    static Color dark_accentColor = new Color(27, 153, 139);

    static Color bgColor = DarkMode ? dark_bg_color : light_bgColor;
    static Color primaryColor = DarkMode ? dark_primaryColor : light_primaryColor;
    static Color secondaryColor = DarkMode ? dark_secondaryColor :
    light_secondaryColor;
    static Color accentColor = DarkMode ? dark_accentColor : light_accentColor;

    public static void reassignColors() {
        bgColor = DarkMode ? dark_bg_color : light_bgColor;
        primaryColor = DarkMode ? dark_primaryColor : light_primaryColor;
        secondaryColor = DarkMode ? dark_secondaryColor : light_secondaryColor;
        accentColor = DarkMode ? dark_accentColor : light_accentColor;
    }
}
```

Listing 2: Main Java File

```
package org.howmuch;

import javax.swing.*;
import java.awt.*;

public class BackgroundPanel extends JPanel {
    private Image background;

    public void paintComponent(Graphics g) {

        super.paintComponent(g);
        int width = this.getSize().width;
        int height = this.getSize().height;

        if (this.background != null) {
```

```
            // Add the size of the window in drawImage method()
            g.drawImage(this.background, 0, 0, width, height, null);
        }
    }

    public void setBackground(String imagePath) {
        // Simply sets the background as the one that you have provided. It needs to be
        // a png file (I think)
        this.background = new ImageIcon(imagePath).getImage();
        repaint();
    }

}
```

Listing 3: Main Java file

```
package org.howmuch;

import org.xml.sax.SAXException;
import javax.swing.*;
import javax.xml.parsers.ParserConfigurationException;
import java.awt.*;
import java.io.*;
import java.time.LocalDate;

// You have to extend the thread class to create a new thread for running the databases.
public class Main extends Thread {

    // Statically defining important variables used throughout the game. They are
    // statically defined coz they are used by other classes very often.
    public static String[] Topics = new String[] { "Technology", "Fashion", "Household", "Miscellaneous" };
    public static String currentTopic = Topics[0];
    static final int WIDTH = 1280, HEIGHT = 720;
    static boolean maximized = false, isGuest = true, grantAccess = false, isLocalDatabaseUpToDate = false,
            isMongoUpToDate = false, usingMongo = false;

    // Declaring Objects of other classes that we are going to call from main.
    static LoginFrame loginFrame;
    static MenuFrame menuFrame;
    static HelpFrame helpFrame;
    static HighscoreFrame highscoreFrame;
    static TopicsFrame topicsFrame;
    static GameFrame gameFrame;
    static GameOverFrame gameOverFrame;

    static Font buttonFont, textFont, password_font, options_font, emoji_font;
    static JButton exit_btn, resize_btn, minimize_btn;
    static JPanel basicButtons_pnl;

    // These are the icons from where we get the resize, exit and the minimize
    // button. They are custom made coz they look better,
    // eliminate the need for the titlebar making the UI look cleaner, albeit less
    // useful.
    // They also let you have full control over what you want to do when they are
    // 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
    the
157      *                 close button is pressed.
158      **/
159     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.
180             String lastUpdateDate = "";
181 
182             // Opening the Backup Date file and checking the last time it was
    backed up.
183             File dateFile = new File(DataBaseManager.LOCAL_BACKUP_DATEFILE);
184             if (dateFile.exists()) {
185                 try (BufferedReader br = new BufferedReader(new FileReader(
    dateFile))) {
186                     lastUpdateDate = br.readLine();
187 
188                     // If the database was backed up last today, then dont do it.
```

```
189                     if (lastUpdateDate.equals(String.valueOf(LocalDate.now()))) {
190                         System.out.println("Backup DataBases are Up to Date!");
191                     } else {
192                         // Else update it.
193                         DataBaseManager.createLocalDatabaseBackup();
194                     }
195                 } catch (IOException e) {
196                     throw new RuntimeException(e);
197                 } catch (NullPointerException exception) {
198                     System.out.println("Nothing in the Date File. ");
199                 }
200             } else {
201                 // If the file itself doesnt exist, then clearly there doesnt
    exist any backup,
202                 // so we better back up at that point.
203                 try {
204                     DataBaseManager.createLocalDatabaseBackup();
205                 } catch (Exception e) {
206                     System.out.println("You havent really created the database yet
    , so not creating backup either. ");
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                 lastUpdateDate = br.readLine();
327                 System.out.println(lastUpdateDate);
328                 if (lastUpdateDate.equals(String.valueOf(LocalDate.now()))) {
329                     System.out.println("Local DataBases are Up to Date!");
330                     isLocalDatabaseUpToDate = true;
331                 }
332             } catch (IOException e) {
333                 throw new RuntimeException(e);
334             } catch (NullPointerException exception) {
335                 System.out.println("Nothing in the Local Date File. ");
336             }
337         }
338 
339         // Now check the mongodb database date file to check when was the last
    time it
340         // was updated. Same Logic tho.
341         dateFile = new File(DataBaseManager.LOCAL_MONGODATEFILE);
342         if (dateFile.exists()) {
343             try (BufferedReader br = new BufferedReader(new FileReader(dateFile)))
     {
344                 lastUpdateDate = br.readLine();
345                 System.out.println(lastUpdateDate);
346                 if (lastUpdateDate.equals(String.valueOf(LocalDate.now()))) {
347                     System.out.println("Mongo DataBases are Up to Date!");
348                     isMongoUpToDate = true;
349                 }
350             } catch (IOException e) {
351                 throw new RuntimeException(e);
352             } catch (NullPointerException exception) {
353                 System.out.println("Nothing in the mongo Date File. ");
354             }
```

```
355         }
356 
357         // If say one of them is not updated, then we gotta scrap amazon.
358         if (!isLocalDatabaseUpToDate || (usingMongo && !isMongoUpToDate)) {
359 
360             System.out.println("Beginning to Scrap Data From Amazon, as one of the
     DataBases isnt updated. ");
361             if (!isLocalDatabaseUpToDate) {
362                 // As an edge case, if mongo isnt up to date, we dont wanna clear
    the local one.
363                 DataBaseManager.clearLocalDatabase();
364             }
365             if (usingMongo && !isMongoUpToDate) {
366                 // If the local one isnt up to date we dont wanna clear mongo.
367                 MongoManager.clearMongoDb();
368             }
369 
370             // Scrap and save, as at this point we already know what works and
    what doesnt,
371             // and what is updated and what isnt,
372             // AmazonScrapper class can figure out where to save stuff. After that
373             // 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
    didnt get any erros.
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 wasnt 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 isnt 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 cant 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 CompoenentListener 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 dupiliated, 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
    matches
107                 // if it does, then we assign some basic variables, and close this
     screen, open
108                 // the menu screen.
109                 if (DataBaseManager.doesPasswordMatch(username_txt_fld.getText(),
110                         String.valueOf(password_txt_fld.getPassword()))) {
111                     DataBaseManager.currentUsername = username_txt_fld.getText();
112                     DataBaseManager.currentPassword = String.valueOf(
    password_txt_fld.getPassword());
113                     this.setVisible(false);
114                     this.dispose();
115                     running = false;
116                     grantAccess = true;
117                     Main.changeFrame(1);
118                 } else {
119                     grantAccess = false;
120                     incorrectPassword = true;
121                 }
122             } else {
```

```
                // If the user has entered a username and a password, and he doesnt exist, then
                // clearly is a new user.
                newUser = true;
            }

        });
        login_btn.setEnabled(false);

        guest_btn = new JButton();
        guest_btn.setText("Guest");
        guest_btn.setAlignmentY(Box.CENTER_ALIGNMENT);
        guest_btn.setAlignmentX(Box.CENTER_ALIGNMENT);
        guest_btn.setBounds(940, 532, 200, 40);
        guest_btn.setFont(textFont.deriveFont(44f));
        guest_btn.setFocusPainted(false);
        guest_btn.setContentAreaFilled(false);
        guest_btn.setOpaque(true);
        guest_btn.setBorder(null);
        guest_btn.setBackground(Colors.bgColor);
        guest_btn.addChangeListener(evt -> {
            if (guest_btn.getModel().isPressed()) {
                guest_btn.setForeground(Colors.primaryColor);
            } else if (guest_btn.getModel().isRollover()) {
                guest_btn.setForeground(Colors.secondaryColor);
            } else {
                guest_btn.setForeground(Colors.primaryColor);
            }
        });
        guest_btn.addActionListener(e -> {
            Main.isGuest = true;
            DataBaseManager.currentPassword = "guest";
            DataBaseManager.currentUsername = "guest";
            this.setVisible(false);
            this.dispose();
            running = false;
            grantAccess = true;
            Main.changeFrame(1);
        });

        newAccount_btn = new JButton();
        newAccount_btn.setText("Create New Account");
        newAccount_btn.setAlignmentY(Box.CENTER_ALIGNMENT);
        newAccount_btn.setAlignmentX(Box.CENTER_ALIGNMENT);
        newAccount_btn.setBounds(615, 590, 800, 40);
        newAccount_btn.setFont(textFont.deriveFont(44f));
        newAccount_btn.setFocusPainted(false);
        newAccount_btn.setContentAreaFilled(false);
        newAccount_btn.setOpaque(true);
        newAccount_btn.setBorder(null);
        newAccount_btn.setBackground(Colors.bgColor);
        newAccount_btn.addChangeListener(evt -> {
            if (newAccount_btn.getModel().isPressed()) {
                newAccount_btn.setForeground(Colors.primaryColor);
            } else if (newAccount_btn.getModel().isRollover()) {
                newAccount_btn.setForeground(Colors.secondaryColor);
            } else {
                newAccount_btn.setForeground(Colors.primaryColor);
            }
```

```
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         });
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 consistancy, or make a new one.
366 
367             long now = System.nanoTime();
368             delta += (now - lastTime) / ns;
369             lastTime = now;
370 
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;
```

```
                } else if (password_txt_fld.getPassword().length < 8) {
                    newAccount_btn.setEnabled(false);
                    status_lbl.setText("Nope, Password is too Short");
                     status_emoji_lbl.setText("\uD83D\uDE0F");
                    // status_emoji_lbl.setText("\uD83E\uDD0F");
//                    status_emoji_lbl.setText("\uD83D\uDE15");
                } else if (Arrays.equals(password_txt_fld.getPassword(), "abcdefgh
    ".toCharArray())) {
                    newAccount_btn.setEnabled(false);
                    status_lbl.setText("Anyone can guess that bruh");
                    status_emoji_lbl.setText("\uD83D\uDC80");
                } else if (Arrays.equals(password_txt_fld.getPassword(), "12345678
    ".toCharArray())) {
                    newAccount_btn.setEnabled(false);
                    status_lbl.setText("12345678? Really?");
                    status_emoji_lbl.setText("\uD83E\uDEE0");
                } else if (Arrays.equals(password_txt_fld.getPassword(), "asdfghjk
    ".toCharArray())) {
                    newAccount_btn.setEnabled(false);
                    status_lbl.setText("Be Lazy, but not thaaat lazy");
                    // status_emoji_lbl.setText("\uD83D\uDE42");
                    status_emoji_lbl.setText("\uD83D\uDC80");
                } else if (Arrays.equals(password_txt_fld.getPassword(), "asdfasdf
    ".toCharArray())) {
                    newAccount_btn.setEnabled(false);
                    status_lbl.setText("Even Krishnaraj isnt this lazy");
                    // status_emoji_lbl.setText("\uD83E\uDD21");
                    status_emoji_lbl.setText("\uD83D\uDC80");
                } else if (password_txt_fld.getPassword().length > 30) {
                    newAccount_btn.setEnabled(false);
                    status_lbl.setText("Woah, Its too big");
                    status_emoji_lbl.setText("\uD83D\uDE0F");
                } else {
                    status_lbl.setText("All Good!");
                    status_emoji_lbl.setText("\uD83D\uDC4C");
                    // status_emoji_lbl.setText("\uD83D\uDE0E" );
                    // status_emoji_lbl.setText("" );
                    newAccount_btn.setEnabled(true);
                    login_btn.setEnabled(true);
                }
                delta--;
            }

        }
    }

    public void startThread() {

        // Creating the Game Thread
        loginThread = new Thread(this);
        loginThread.start();

    }
}
```

Listing 5: Main Java file

```
package org.howmuch;

```

```
import javax.swing.*;
import javax.swing.event.ChangeEvent;
import javax.swing.event.ChangeListener;
import java.awt.*;
import java.awt.event.*;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Random;
import java.util.TimerTask;
import java.util.Timer;

import static java.lang.Math.round;
import static org.howmuch.Main.*;

public class GameFrame extends JFrame {
    static Timer timer;
    public static int time_left = 9;
    public static boolean gameWon = false;
    static boolean[] whichOptionCorrect;
    BackgroundPanel backgroundPanel;
    BackgroundPanel productImagePanel;
    static JButton option_1_btn;
    static JButton option_2_btn;
    static JButton option_3_btn;
    static JButton option_4_btn;
    JPanel options_panel;
    static JLabel time_lbl;
    JTextArea productName_txtArea;
    public static int randomIndex = 0;
    static String[] currentData;
    static int correctPrice;
    static JLabel confetti;

    GameFrame() {
        randomIndex = 0;
        time_left = 9;
        backgroundPanel = new BackgroundPanel();
        gameWon = false;
        this.setTitle("How Much?");
        if (maximized) {
            this.setExtendedState(MAXIMIZED_BOTH);
        } else {
            this.setPreferredSize(new Dimension(Main.WIDTH, Main.HEIGHT));
        }
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setResizable(true);
        this.setUndecorated(true);
        this.setMinimumSize(new Dimension(1280, 720));

        createFonts();
        createBasicButtonPanel();
        createButtons();
        createPanels();
        createLabels();


        // Main stuff
        findRandomIndex();
        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
    as mongo isnt working");
118                 currentData = DataBaseManager.readFromLocalDatabase(currentTopic,
    randomIndex);
```

```
119             } else {
120                 currentData = MongoManager.fetchDataFromMongo(currentTopic,
    randomIndex);
121                 System.out.println("Reading data from mongo sucessful");
122             }
123         } catch (Exception e) {
124             System.out.println("We got some Issues reading the file from Mongodb")
    ;
125             currentData = DataBaseManager.readFromLocalDatabase(currentTopic,
    randomIndex);
126         }
127     }
128 
129     private void findRandomIndex() {
130         int max = DataBaseManager.findLength(currentTopic);
131         Random random = new Random();
132         // Generates random integers 0 to 49
133         randomIndex = random.nextInt(max);
134     }
135 
136     private void loadGameDataOnScreen() {
137         System.out.println(Arrays.toString(currentData));
138         Image productImage = new ImageIcon(currentData[2]).getImage();
139         int maxWidth = (int) (0.4 * this.getWidth());
140         int maxHeight = (int) (0.4 * this.getWidth());
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) *
        100
191 //        ));
192         whichOptionCorrect = new boolean[]{false, false, false, false};
193         int optionValue = 0;
194 
195         optionValue = list.remove(random.nextInt(list.size()));
196         whichOptionCorrect[0] = optionValue == correctPrice;
197         option_1_btn.setText("R" + String.format("%,.0f", (double) optionValue));
198 
199         optionValue = list.remove(random.nextInt(list.size()));
200         whichOptionCorrect[1] = optionValue == correctPrice;
201         option_2_btn.setText("R" + String.format("%,.0f", (double) optionValue));
202 
203         optionValue = list.remove(random.nextInt(list.size()));
204         whichOptionCorrect[2] = optionValue == correctPrice;
205         option_3_btn.setText("R" + String.format("%,.0f", (double) optionValue));
206 
207         optionValue = list.remove(random.nextInt(list.size()));
208         whichOptionCorrect[3] = optionValue == correctPrice;
209         option_4_btn.setText("R" + String.format("%,.0f", (double) optionValue));
210 
211     }
212 
213     private int[] calculateImageSize(int maxWidth, int maxHeight, double width,
        double height) {
214         int wt = 600, ht = 550;
215         double aspectRatio = width / height;
216         System.out.println(aspectRatio);
217         if (aspectRatio > 1) {
218             // landscape
219             wt = maxWidth;
220             ht = (int) (wt / aspectRatio);
221         } else {
222             // portrait image
223             ht = maxHeight;
```

```
224             wt = (int) (ht * aspectRatio);
225         }
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) -
    40, 10, exit_btn.getWidth() * 3 + 35, exit_btn.getHeight());
255 
256         // Options panel
257         options_panel.setBounds((int) (0.70 * screenSize.getWidth()), (int) (0.58
    * screenSize.getHeight()), (int) (0.60 * screenSize.getWidth()), 700);
258 
259         // Buttons in the Options Panel
260         option_1_btn.setBounds(new Rectangle((int) (0.45 * screenSize.getWidth()),
     80));
261         option_1_btn.setFont(options_font.deriveFont((float) (0.05 * getHeight()))
    );
262 
263         option_2_btn.setBounds(new Rectangle((int) (0.45 * screenSize.getWidth()),
     80));
264         option_2_btn.setFont(options_font.deriveFont((float) (0.05 * getHeight()))
    );
265 
266         option_3_btn.setBounds(new Rectangle((int) (0.45 * screenSize.getWidth()),
     70));
267         option_3_btn.setFont(options_font.deriveFont((float) (0.05 * getHeight()))
    );
268 
269         option_4_btn.setBounds(new Rectangle((int) (0.45 * screenSize.getWidth()),
     70));
270         option_4_btn.setFont(options_font.deriveFont((float) (0.05 * getHeight()))
    );
271 
```

```
272         productName_txtArea.setFont(password_font.deriveFont((float) (0.04 *
    getHeight())));
273 
274         // The Score label
275         time_lbl.setBounds((int) (0.890 * screenSize.getWidth()), (int) (0.14 *
    screenSize.getHeight()), (int) (0.05 * screenSize.getWidth()), (int) (0.11 *
    screenSize.getHeight()));
276         time_lbl.setFont(buttonFont.deriveFont((float) (0.09 * getHeight())));
277 
278         confetti.setBounds((int) (-0.3 * screenSize.getWidth()), (int) (-0.27 *
    screenSize.getHeight()), this.getWidth(), this.getHeight());
279         loadGameDataOnScreen();
280     }
281 
282     private void reassignColors() {
283 
284         if (Colors.DarkMode) {
285             backgroundPanel.setBackground("src/main/resources/images/gamescreen.
    png");
286         } else {
287             backgroundPanel.setBackground("src/main/resources/images/gamescreen.
    png");
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)
    (0.80 * screenSize.getHeight()),
59                  (int) (0.20 * screenSize.getWidth()), (int) (0.07 * screenSize.
    getHeight()));
60          backToMenu_btn.setFont(buttonFont.deriveFont((float) (0.05 * getHeight()))
    );
61
62          // The Entire basic button panel for closing minimizing and stuff
63          basicButtons_pnl.setBounds(this.getWidth() - (exit_btn.getWidth() * 3) -
    40, 10, exit_btn.getWidth() * 3 + 35,
64                  exit_btn.getHeight());
65
66          // Options panel
67          options_panel.setBounds((int) (0.60 * screenSize.getWidth()), (int) (0.34
    * screenSize.getHeight()),
68                  (int) (0.45 * screenSize.getWidth()), 700);
69
70          // Buttons in the Options Panel
71          option1_btn.setBounds(new Rectangle((int) (0.45 * screenSize.getWidth()),
    80));
72          option1_btn.setFont(buttonFont.deriveFont((float) (0.07 * getHeight())));
73
74          option2_btn.setBounds(new Rectangle((int) (0.45 * screenSize.getWidth()),
    80));
75          option2_btn.setFont(buttonFont.deriveFont((float) (0.07 * getHeight())));
76
77          option3_btn.setBounds(new Rectangle((int) (0.45 * screenSize.getWidth()),
    70));
78          option3_btn.setFont(buttonFont.deriveFont((float) (0.07 * getHeight())));
79
80          option4_btn.setBounds(new Rectangle((int) (0.45 * screenSize.getWidth()),
    70));
81          option4_btn.setFont(buttonFont.deriveFont((float) (0.07 * getHeight())));
82      }
83
84      private void reassignColors() {
85
86          if (Colors.DarkMode) {
87              backgroundPanel.setBackground("/run/media/krishnaraj/Programs/Java/How
     Much/src/main/resources/images/choose topic dark.png");
88          } else {
89              backgroundPanel.setBackground("/run/media/krishnaraj/Programs/Java/How
     Much/src/main/resources/images/choose topic.png");
90          }
91          Colors.reassignColors();
92          basicButtons_pnl.setBackground(Colors.bgColor);
93          exit_btn.setBackground(Colors.bgColor);
94          resize_btn.setBackground(Colors.bgColor);
```

```
95          backToMenu_btn.setBackground(Colors.primaryColor);
96          backToMenu_btn.setForeground(Colors.bgColor);
97          minimize_btn.setBackground(Colors.bgColor);
98          option1_btn.setBackground(Colors.bgColor);
99          option1_btn.setForeground(Colors.primaryColor);
100         option2_btn.setBackground(Colors.bgColor);
101         option2_btn.setForeground(Colors.primaryColor);
102         option4_btn.setBackground(Colors.bgColor);
103         option4_btn.setForeground(Colors.primaryColor);
104         option3_btn.setBackground(Colors.bgColor);
105         option3_btn.setForeground(Colors.primaryColor);
106     }
107
108     private void createPanels() {
109         options_panel = new JPanel();
110         BoxLayout bl = new BoxLayout(options_panel, BoxLayout.Y_AXIS);
111         options_panel.setLayout(bl);
112         options_panel.add(option1_btn);
113         options_panel.add(Box.createRigidArea(new Dimension(0, 25)));
114         options_panel.add(option2_btn);
115         options_panel.add(Box.createRigidArea(new Dimension(0, 25)));
116         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();
```

```
            grantAccess = true;
            Main.changeFrame(6);
        });

        option2_btn = new JButton();
        option2_btn.setText(Topics[1] + " ");
        option2_btn.setAlignmentY(Box.CENTER_ALIGNMENT);
        option2_btn.setAlignmentX(Box.LEFT_ALIGNMENT);
        option2_btn.setFocusPainted(false);
        option2_btn.setContentAreaFilled(false);
        option2_btn.setOpaque(true);
        option2_btn.setBorder(null);
        option2_btn.addChangeListener(evt -> {
            if (option2_btn.getModel().isPressed()) {
                option2_btn.setForeground(Colors.accentColor);
            } else if (option2_btn.getModel().isRollover()) {
                option2_btn.setForeground(Colors.accentColor);
            } else {
                option2_btn.setForeground(Colors.primaryColor);
            }
        });
        option2_btn.addActionListener(e -> {
            grantAccess = true;
            currentTopic = Topics[1];
            this.setVisible(false);
            this.dispose();
            grantAccess = true;
            Main.changeFrame(6);
        });

        option3_btn = new JButton();
        option3_btn.setText(Topics[2] + " ");
        option3_btn.setAlignmentY(Box.CENTER_ALIGNMENT);
        option3_btn.setAlignmentX(Box.LEFT_ALIGNMENT);
        option3_btn.setFocusPainted(false);
        option3_btn.setContentAreaFilled(false);
        option3_btn.setOpaque(true);
        option3_btn.setBorder(null);
        option3_btn.addChangeListener(evt -> {
            if (option3_btn.getModel().isPressed()) {
                option3_btn.setForeground(Colors.accentColor);
            } else if (option3_btn.getModel().isRollover()) {
                option3_btn.setForeground(Colors.accentColor);
            } else {
                option3_btn.setForeground(Colors.primaryColor);
            }
        });
        option3_btn.addActionListener(e -> {
            grantAccess = true;
            currentTopic = Topics[2];
            this.setVisible(false);
            this.dispose();
            grantAccess = true;
            Main.changeFrame(6);
        });

        option4_btn = new JButton();
        option4_btn.setText(Topics[3] + " ");
        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)
    (0.80 * screenSize.getHeight()),
58                  (int) (0.20 * screenSize.getWidth()), (int) (0.07 * screenSize.
    getHeight()));
59          backtoTopic_btn.setFont(buttonFont.deriveFont((float) (0.06 * getHeight())
    ));
60
61          // The Entire basic button panel for closing minimizing and stuff
62          basicButtons_pnl.setBounds(this.getWidth() - (exit_btn.getWidth() * 3) -
    40, 10, exit_btn.getWidth() * 3 + 35,
63                  exit_btn.getHeight());
64
65          // Score Label
66          score_lbl.setBounds((int) (0.76 * screenSize.getWidth()), (int) (0.80 *
    screenSize.getHeight()),
67                  (int) (0.31 * screenSize.getWidth()), (int) (0.13 * screenSize.
    getHeight()));
68          score_lbl.setFont(buttonFont.deriveFont((float) (0.14 * getHeight())));
69      }
70
71      private void reassignColors() {
72          Colors.reassignColors();
73          if (GameFrame.gameWon) {
74              if (Colors.DarkMode) {
75                  backgroundPanel.setBackground("/run/media/krishnaraj/Programs/Java
    /How Much/src/main/resources/images/game won over dark.png");
76              } else {
77                  backgroundPanel.setBackground("/run/media/krishnaraj/Programs/Java
    /How Much/src/main/resources/images/game won over.png");
78              }
79          } else {
80              if (Colors.DarkMode) {
81                  backgroundPanel.setBackground("/run/media/krishnaraj/Programs/Java
```

```
/How Much/src/main/resources/images/game over dark.png");
            } else {
                backgroundPanel.setBackground("/run/media/krishnaraj/Programs/Java
/How Much/src/main/resources/images/game over.png");
            }
        }
        backtoTopic_btn.setBackground(Colors.primaryColor);
        backtoTopic_btn.setForeground(Colors.bgColor);

        score_lbl.setBackground(Colors.bgColor);
        score_lbl.setForeground(Colors.accentColor);

        basicButtons_pnl.setBackground(Colors.bgColor);
        exit_btn.setBackground(Colors.bgColor);
        resize_btn.setBackground(Colors.bgColor);
        minimize_btn.setBackground(Colors.bgColor);
    }

    private void createLabels() {
        score_lbl = new JLabel();
        score_lbl.setText(String.valueOf(DataBaseManager.currentScore));
        score_lbl.setAlignmentY(Box.CENTER_ALIGNMENT);
        score_lbl.setAlignmentX(Box.CENTER_ALIGNMENT);
        score_lbl.setOpaque(true);
        score_lbl.setBorder(null);
    }

    private void createButtons() {
        backtoTopic_btn = new JButton();
        backtoTopic_btn.setText("Try Again");
        backtoTopic_btn.setAlignmentY(Box.CENTER_ALIGNMENT);
        backtoTopic_btn.setAlignmentX(Box.LEFT_ALIGNMENT);
        backtoTopic_btn.setFocusPainted(false);
        backtoTopic_btn.setContentAreaFilled(false);
        backtoTopic_btn.setOpaque(true);
        backtoTopic_btn.setBorder(null);
        backtoTopic_btn.addChangeListener(evt -> {
            if (backtoTopic_btn.getModel().isPressed()) {
                backtoTopic_btn.setForeground(Colors.bgColor);
            } else if (backtoTopic_btn.getModel().isRollover()) {
                backtoTopic_btn.setForeground(Colors.accentColor);
            } else {
                backtoTopic_btn.setForeground(Colors.bgColor);
            }
        });
        backtoTopic_btn.addActionListener(e -> {
            this.setVisible(false);
            this.dispose();
            grantAccess = true;
            Main.changeFrame(2);
        });

        // Removing Change and Action Listeners.
        GameFrame.removeAllChangeAndActionListenersFromBasicButtons();

        exit_btn.addChangeListener(evt -> {
            if (exit_btn.getModel().isPressed()) {
                exit_btn.setForeground(Colors.primaryColor);
                Main.changeFrame(0);
```

```
            } else if (exit_btn.getModel().isRollover()) {
                exit_btn.setForeground(Colors.secondaryColor);
            } else {
                exit_btn.setForeground(Colors.primaryColor);
            }
        });
        resize_btn.addActionListener(e -> {
            if (!Main.maximized) {
                this.setExtendedState(MAXIMIZED_BOTH);
                resize_btn.setIcon(new ImageIcon(resizeDown_image));
            } else {
                this.setExtendedState(JFrame.NORMAL);
                this.setLocationRelativeTo(null);
                Dimension dimension = Toolkit.getDefaultToolkit().getScreenSize();
                int x = (int) ((dimension.getWidth() - Main.WIDTH) / 2);
                int y = (int) ((dimension.getHeight() - Main.HEIGHT) / 2);
                this.setBounds(x, y, Main.WIDTH, Main.HEIGHT);
                resize_btn.setIcon(new ImageIcon(resizeUp_image));
            }
            Main.maximized = !Main.maximized;
        });

        minimize_btn.addChangeListener(evt -> {
            if (minimize_btn.getModel().isPressed()) {
                this.setState(JFrame.ICONIFIED);
                minimize_btn.setForeground(Colors.primaryColor);
            } else if (minimize_btn.getModel().isRollover()) {
                minimize_btn.setForeground(Colors.secondaryColor);
            } else {
                minimize_btn.setForeground(Colors.primaryColor);
            }
        });
    }
}
```

Listing 8: Main Java file

```
/*
 * Class that does everything that we wanna do with mongodb. Things like inserting
    , deleting, creating and fetching data.
 */

package org.howmuch;

import com.mongodb.MongoClient;
import com.mongodb.client.*;
import org.bson.Document;

public class MongoManager {
    static MongoDatabase database;
    public static String MONGO_DATABASE_NAME = "HowMuch";
    public static int MONGO_PORT_NO = 27017;
    public static String MONGO_HOST = "localhost";

    public static String[] fetchDataFromMongo(String currentTopic, int randomIndex
    ) {
        MongoCollection<org.bson.Document> collection = database.getCollection(
    currentTopic);
        FindIterable<Document> iterDoc = collection.find();
```

```
        int i = 0;
        for (Document document : iterDoc) {
            System.out.println(document);
            if (i == randomIndex) {
                return new String[] { (String) document.get("Name"), document.getString("Price"),
                        document.getString("Image") };
            }
            i++;
        }
        return new String[] { "Sadly Not Found", "Sadly Not Found", "Sadly Not Found" };

    }

    public static boolean establishConnectionWithMongo() {
        // Creating a MongoDB client
        try {
            MongoClient mongoClient = new MongoClient(MONGO_HOST, MONGO_PORT_NO);
            // Connecting to the database
            database = mongoClient.getDatabase(MONGO_DATABASE_NAME);
            System.out.println("Connected Successfully to mongoDb");
            return true;

        } catch (Exception e) {
            System.out.println("Couldnt establish connection due to some reason");
            System.out.println(e.getMessage());
            return false;
        }
    }

    public static void addDataToMongo(String Topic, String[] data) {
        try {
            MongoClient mongoClient = new MongoClient(MONGO_HOST, MONGO_PORT_NO);
            // Connecting to the database
            database = mongoClient.getDatabase(MONGO_DATABASE_NAME);
            Topic = Topic.substring(0, 1).toUpperCase() + Topic.substring(1);
            MongoCollection<Document> collection = database.getCollection(Topic);
            Document dataDocToAdd = new Document();
            dataDocToAdd.append("Name", data[0]);
            dataDocToAdd.append("Price", data[1]);
            dataDocToAdd.append("Image", data[2]);
            collection.insertOne(dataDocToAdd);
            System.out.println(data[0]);
            System.out.println("\n\nAdded record to mongo---------\n\n");
        } catch (Exception e) {
            System.out.println("Couldnt add data");
        }
    }

    public static void clearMongoDb() {
        MongoCollection<Document> collection = database.getCollection(Main.Topics[0]);
        collection.drop();
        collection = database.getCollection(Main.Topics[1]);
        collection.drop();
        collection = database.getCollection(Main.Topics[2]);
        collection.drop();
        collection = database.getCollection(Main.Topics[3]);
```

```
76          collection.drop();
77      }
78  }
```

Listing 9: Main Java file

```
1  /*
2   * An Important class, As it controls a lot of the important variables, and all
       interactions with the Local CSV File databases.
3   */
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                 if (subfile.isDirectory()) {
50                     for (File f : Objects.requireNonNull(subfile.listFiles())) {
51                         f.delete();
52                     }
53                 }
54             }
55 
56             // Also clear the csv files.
57             data_deleter = new File(LOCAL_CSV_FOLDER);
58             listFilesForFolder(data_deleter);
59             for (File subfile : Objects.requireNonNull(data_deleter.listFiles()))
    {
60                 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 excepition 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             List<String[]> csvBody = reader.readAll();
101             USER_INDEX = csvBody.size();
102         } catch (IOException e) {
103             throw new RuntimeException(e);
104         }
105
106         // append the new user to the login file.
107         try (FileWriter userDataFileWriter = new FileWriter(userDatafile, true)) {
108
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)) {
112
113                 String[] data = { currentUsername, currentPassword, String.valueOf(currentScore) };
114                 writer.writeNext(data);
115                 System.out.println("added new user");
116             }
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.DEFAULT_SEPARATOR,
131                     CSVWriter.NO_QUOTE_CHARACTER, CSVWriter.DEFAULT_ESCAPE_CHARACTER, CSVWriter.DEFAULT_LINE_END)) {
132                 // System.out.println(Arrays.toString(data));
133                 writer.writeNext(data);
134             }
135
136         } catch (IOException e) {
137             System.out.println("Cant open user data file. ");
138         }
139     }
140
141     public static boolean doesUsernameExist(String username) {
142         File inputFile = new File(USERDATA_FILEPATH);
143         try (CSVReader reader = new CSVReader(new FileReader(inputFile), ',')) {
144             List<String[]> csvBody = reader.readAll();
145             for (String[] s : csvBody) {
146                 if (s[0].equals(username)) {
```

```
147                     System.out.println("User Already Exists");
148                     return true;
149                 }
150             }
151         } catch (IOException e) {
152             System.out.println("couldnt create csvreader in username exists
    checker method. ");
153         }
154         return false;
155     }
156 
157     public static boolean doesPasswordMatch(String username, String password) {
158         File inputFile = new File(USERDATA_FILEPATH);
159         try (CSVReader reader = new CSVReader(new FileReader(inputFile), ',')) {
160             List<String[]> csvBody = reader.readAll();
161             for (int i = 0; i < csvBody.size(); i++) {
162                 String[] s = csvBody.get(i);
163                 if (s[0].equals(username)) {
164                     System.out.println("User Found");
165                     if (s[1].equals(password)) {
166                         System.out.println("Password Matches");
167                         USER_INDEX = i;
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 excepition 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
298     public static int findLength(String Topic) {
299         File inputFile;
300         if (Main.isLocalDatabaseUpToDate) {
301             inputFile = new File(LOCAL_CSV_FOLDER + '/' + Topic.toLowerCase() + ".
    csv");
302         } else {
303             inputFile = new File(LOCAL_BACKUP_CSV_FOLDER + '/' + Topic.toLowerCase
    () + ".csv");
304         }
305         try (CSVReader reader = new CSVReader(new FileReader(inputFile), ',')) {
306             List<String[]> csvBody = reader.readAll();
307             return csvBody.size();
```

```
308         } catch (IOException e) {
309             throw new RuntimeException(e);
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 -> {
```

```
            if (exit_btn.getModel().isPressed()) {
                exit_btn.setForeground(Colors.primaryColor);
                Main.changeFrame(0);
            } else if (exit_btn.getModel().isRollover()) {
                exit_btn.setForeground(Colors.secondaryColor);
            } else {
                exit_btn.setForeground(Colors.primaryColor);
            }
        });
        resize_btn.addActionListener(e -> {
            if (!Main.maximized) {
                this.setExtendedState(MAXIMIZED_BOTH);
                resize_btn.setIcon(new ImageIcon(resizeDown_image));
            } else {
                this.setExtendedState(JFrame.NORMAL);
                this.setLocationRelativeTo(null);
                Dimension dimension = Toolkit.getDefaultToolkit().getScreenSize();
                int x = (int) ((dimension.getWidth() - Main.WIDTH) / 2);
                int y = (int) ((dimension.getHeight() - Main.HEIGHT) / 2);
                this.setBounds(x, y, Main.WIDTH, Main.HEIGHT);
                resize_btn.setIcon(new ImageIcon(resizeUp_image));
            }
            Main.maximized = !Main.maximized;
        });

        minimize_btn.addChangeListener(evt -> {
            if (minimize_btn.getModel().isPressed()) {
                this.setState(JFrame.ICONIFIED);
                minimize_btn.setForeground(Colors.primaryColor);
            } else if (minimize_btn.getModel().isRollover()) {
                minimize_btn.setForeground(Colors.secondaryColor);
            } else {
                minimize_btn.setForeground(Colors.primaryColor);
            }
        });
    }
}
```

Listing 11: Main Java file

```
package org.howmuch;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ComponentAdapter;
import java.awt.event.ComponentEvent;

import static org.howmuch.Main.*;

public class HelpFrame extends JFrame {
    BackgroundPanel backgroundPanel;
    JButton backToMenu_btn;

    HelpFrame() {
        backgroundPanel = new BackgroundPanel();

        this.setTitle("How Much? ");
        if (maximized) {
            this.setExtendedState(MAXIMIZED_BOTH);
```

```
        } else {
            this.setPreferredSize(new Dimension(Main.WIDTH, Main.HEIGHT));
        }
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setResizable(true);
        this.setUndecorated(true);
        this.setMinimumSize(new Dimension(1280, 720));

        createFonts();
        createBasicButtonPanel();
        createButtons();
        reassignColors();
        reassignBounds();

        this.addComponentListener(new ComponentAdapter() {
            @Override
            public void componentResized(ComponentEvent e) {
                reassignBounds();
                repaint();
            }
        });

        this.add(backToMenu_btn);
        this.add(basicButtons_pnl);
        this.add(backgroundPanel);
        this.pack();
        this.setLocationRelativeTo(null);
        this.setVisible(true);
    }

    private void reassignBounds() {
        Dimension screenSize = this.getSize();

        // The back to menu mode label
        backToMenu_btn.setBounds((int) (0.015 * screenSize.getWidth()), (int)
    (0.80 * screenSize.getHeight()),
                (int) (0.20 * screenSize.getWidth()), (int) (0.07 * screenSize.
    getHeight()));
        backToMenu_btn.setFont(buttonFont.deriveFont((float) (0.05 * getHeight()))
    );

        // The Entire basic button panel for closing minimizing and stuff
        basicButtons_pnl.setBounds(this.getWidth() - (exit_btn.getWidth() * 3) -
    40, 10, exit_btn.getWidth() * 3 + 35,
                exit_btn.getHeight());

    }

    private void reassignColors() {
        Colors.reassignColors();
        if (Colors.DarkMode) {
            backgroundPanel.setBackground("/run/media/krishnaraj/Programs/Java/How
     Much/src/main/resources/images/help and credits dark.png");
        } else {
            backgroundPanel.setBackground("/run/media/krishnaraj/Programs/Java/How
     Much/src/main/resources/images/help and credits.png");
        }
        backToMenu_btn.setBackground(Colors.primaryColor);
        backToMenu_btn.setForeground(Colors.bgColor);
```

```
73
74          basicButtons_pnl.setBackground(Colors.bgColor);
75          exit_btn.setBackground(Colors.bgColor);
76          resize_btn.setBackground(Colors.bgColor);
77          minimize_btn.setBackground(Colors.bgColor);
78      }
79
80      private void createButtons() {
81          backToMenu_btn = new JButton();
82          backToMenu_btn.setText("Back to Menu");
83          backToMenu_btn.setAlignmentY(Box.CENTER_ALIGNMENT);
84          backToMenu_btn.setAlignmentX(Box.LEFT_ALIGNMENT);
85          backToMenu_btn.setFocusPainted(false);
86          backToMenu_btn.setContentAreaFilled(false);
87          backToMenu_btn.setOpaque(true);
88          backToMenu_btn.setBorder(null);
89          backToMenu_btn.addChangeListener(evt -> {
90              if (backToMenu_btn.getModel().isPressed()) {
91                  backToMenu_btn.setForeground(Colors.bgColor);
92              } else if (backToMenu_btn.getModel().isRollover()) {
93                  backToMenu_btn.setForeground(Colors.accentColor);
94              } 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         resize_btn.addActionListener(e -> {
119             if (!Main.maximized) {
120                 this.setExtendedState(MAXIMIZED_BOTH);
121                 resize_btn.setIcon(new ImageIcon(resizeDown_image));
122             } else {
123                 this.setExtendedState(JFrame.NORMAL);
124                 this.setLocationRelativeTo(null);
125                 Dimension dimension = Toolkit.getDefaultToolkit().getScreenSize();
126                 int x = (int) ((dimension.getWidth() - Main.WIDTH) / 2);
127                 int y = (int) ((dimension.getHeight() - Main.HEIGHT) / 2);
128                 this.setBounds(x, y, Main.WIDTH, Main.HEIGHT);
129                 resize_btn.setIcon(new ImageIcon(resizeUp_image));
130             }
131             Main.maximized = !Main.maximized;
```

```
        });

        minimize_btn.addChangeListener(evt -> {
            if (minimize_btn.getModel().isPressed()) {
                this.setState(JFrame.ICONIFIED);
                minimize_btn.setForeground(Colors.primaryColor);
            } else if (minimize_btn.getModel().isRollover()) {
                minimize_btn.setForeground(Colors.secondaryColor);
            } else {
                minimize_btn.setForeground(Colors.primaryColor);
            }
        });
    }
}
```

Listing 12: Main Java file

```
/*
 * The class that Performs the role of the heart of the code. It is what downloads
    the images from aamzon, and saves them.
 * 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
 * We can then get exactly what we want.
 */

package org.howmuch;

import java.io.IOException;
import java.net.MalformedURLException;
import java.net.URL;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.*;

import com.groupdocs.conversion.Converter;
import com.groupdocs.conversion.filetypes.ImageFileType;
import com.groupdocs.conversion.options.convert.ImageConvertOptions;
import org.apache.commons.io.FileExistsException;
import org.w3c.dom.*;

import javax.xml.parsers.*;
import java.io.*;

import com.gargoylesoftware.htmlunit.*;
import com.gargoylesoftware.htmlunit.html.*;
import org.xml.sax.SAXException;

public class AmazonScrapper {
    static Converter converter;
    static ImageConvertOptions options;
    static WebClient webClient;
    static DocumentBuilder builder;
    static DocumentBuilderFactory factory;
    public static HashMap<Integer, String[]> searchQueries_map = new HashMap<>();
    public static String AMAZON_PREFIX_URL = "https://www.amazon.in/s?k=";

    AmazonScrapper() {
```

```
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(
    StandardCharsets.UTF_8));
110                             xmlParser(input,
111                                     DataBaseManager.LOCAL_IMG_FOLDER + '/' + Main.
    Topics[topic.getKey()].toLowerCase()
112                                             + '/' + topic.getValue()[topic_queries
    ] + searchResult + ".webp",
113                                     Main.Topics[topic.getKey()].toLowerCase());
114                         }
115                     } catch (IOException e) {
116                         System.out.println("An error occurred: " + e);
117                     }
118
119                 }
120             }
121
122         }
123     }
124
125     /*
126      * Function that uses Xpath to go through the xml code, parse it and then
    return
127      * the necessary strings.
128      */
129     public static void xmlParser(ByteArrayInputStream inputFile, String
    imageFilePath, String Topic)
130             throws ParserConfigurationException, IOException, SAXException {
131
132         String productName = "Sadly Not Found", productPrice = "Sadly Not Found",
    productImagePath = "Sadly Not Found";
133
134         Document doc = builder.parse(inputFile);
135
136         NodeList nListImages = doc.getElementsByTagName("img");
137         Element imageElement = (Element) nListImages.item(0);
138         String[] allImageURLs = imageElement.getAttribute("srcset").split(",");
```

```
139         String hdImageIrl = allImageURLs[allImageURLs.length - 1].split(" ")[1];
140 
141         productName = imageElement.getAttribute("alt");
142         productName = productName.replace(",", " -");
143         if (productName.contains("Sponsored Ad - ")) {
144             productName = productName.replace("Sponsored Ad - ", "");
145         }
146         System.out.println(productName);
147 
148         saveImage(hdImageIrl, imageFilePath);
149         productImagePath = imageFilePath.replace(".webp", ".png");
150 
151         NodeList nList = doc.getElementsByTagName("span");
152         for (int i = 0; i < nList.getLength(); i++) {
153             Element currElement = (Element) nList.item(i);
154             if (currElement.getAttribute("class").equals("a-price-whole")) {
155                 System.out.println("Price is: ");
156                 productPrice = currElement.getTextContent().replace(".", "");
157                 productPrice = productPrice.strip().replace(",", "");
158                 System.out.println(productPrice);
159             }
160         }
161         String[] data = new String[] { productName, productPrice, productImagePath
    };
162         if (productName.equalsIgnoreCase("Sadly Not Found") || productPrice.
    equalsIgnoreCase("Sadly Not Found")
163                 || productImagePath.equalsIgnoreCase("Sadly Not Found")) {
164             System.out.println("Not adding this data");
165         } else {
166             if (!Main.isLocalDatabaseUpToDate) {
167                 DataBaseManager.addDataToCSV(DataBaseManager.LOCAL_CSV_FOLDER + '/
    ' + Topic + ".csv", data);
168             }
169             if (Main.usingMongo) {
170                 if (!Main.isMongoUpToDate) {
171                     MongoManager.addDataToMongo(Topic, data);
172                 }
173             }
174         }
175     }
176 
177     /*
178      * Simple function save the image, but coz we cant work with webp images, we
179      * gotta convert them to png and save them right away.
180      */
181     public static void saveImage(String URLst, String filepath) {
182         if (new File(filepath).exists()) {
183             System.out.println("File Exists, gonna replace it");
184             new File(filepath).delete();
185         }
186 
187         try (InputStream in = new URL(URLst).openStream()) {
188             Files.copy(in, Paths.get(filepath));
189             converter = new Converter(filepath);
190             filepath = filepath.replace(".webp", ".png");
191             converter.convert(filepath, options);
192             Files.delete(Path.of(filepath.replace(".png", ".webp")));
193 
194         } catch (IOException e) {
```

```
195                 System.out.println("we got some issue here with this file");
196             }
197         }
198 
199 }
```

Listing 13: Main Java file