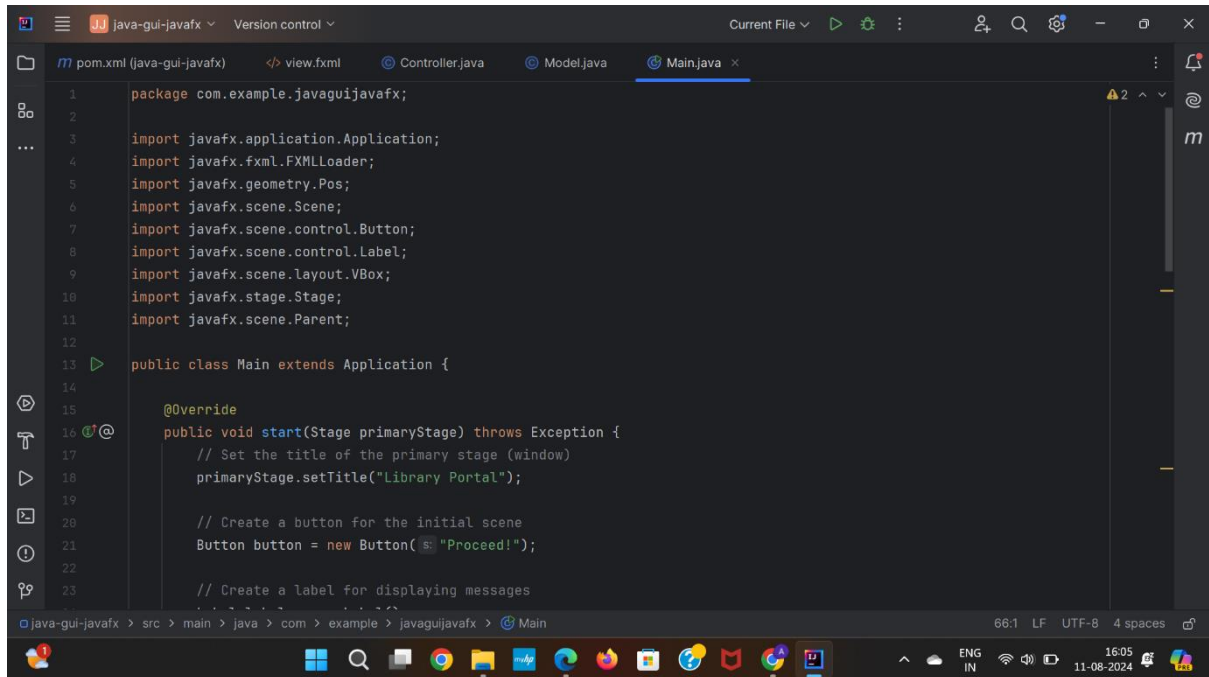
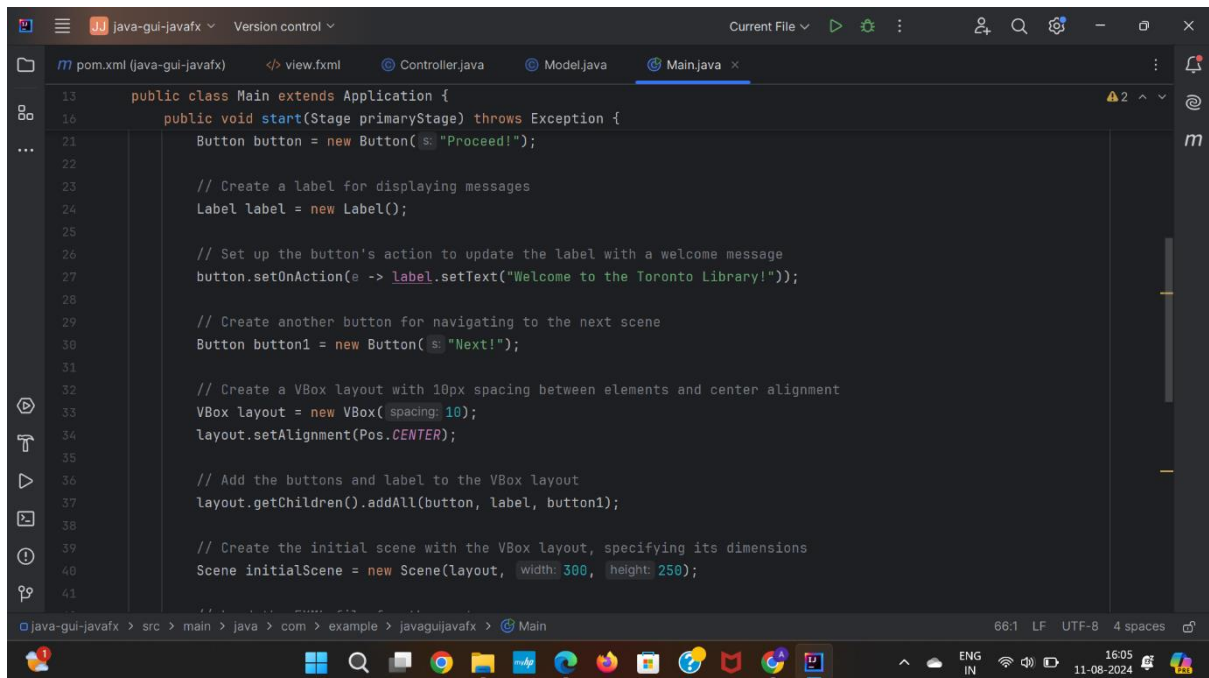


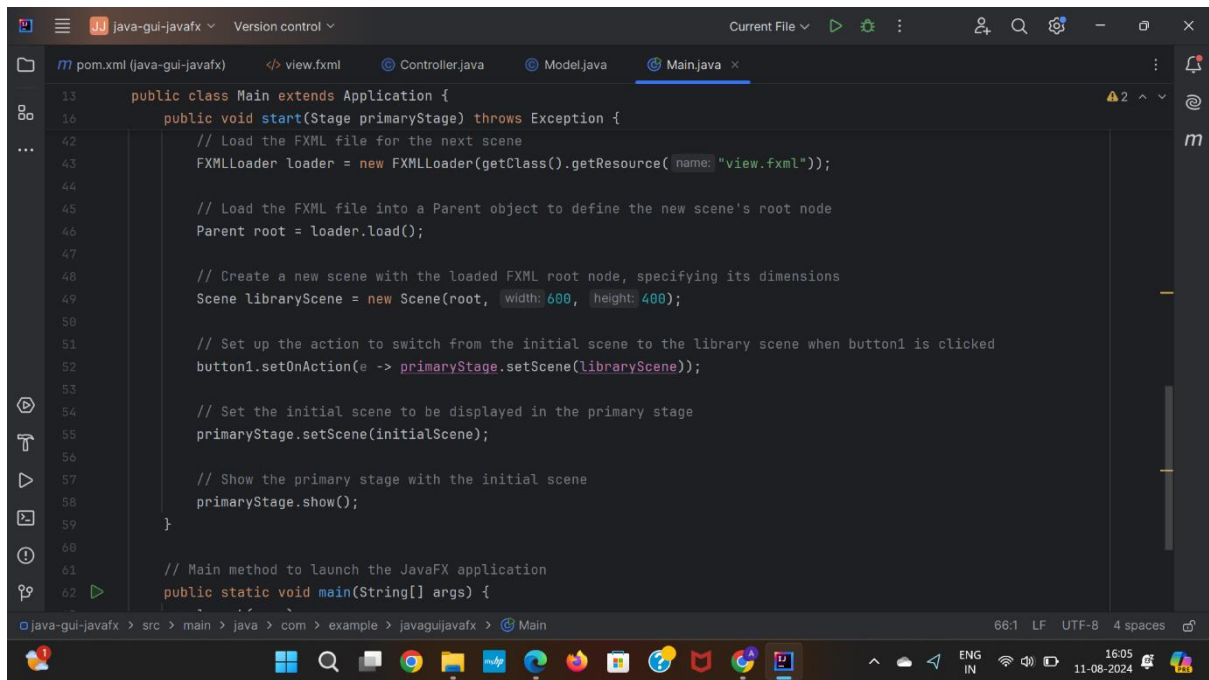
Main.java



```
1 package com.example.javaguijavafx;
2
3 import javafx.application.Application;
4 import javafx.fxml.FXMLLoader;
5 import javafx.geometry.Pos;
6 import javafx.scene.Scene;
7 import javafx.scene.control.Button;
8 import javafx.scene.control.Label;
9 import javafx.scene.layout.VBox;
10 import javafx.stage.Stage;
11 import javafx.scene.Parent;
12
13 public class Main extends Application {
14
15     @Override
16     public void start(Stage primaryStage) throws Exception {
17         // Set the title of the primary stage (window)
18         primaryStage.setTitle("Library Portal");
19
20         // Create a button for the initial scene
21         Button button = new Button("Proceed!");
22
23         // Create a label for displaying messages
```



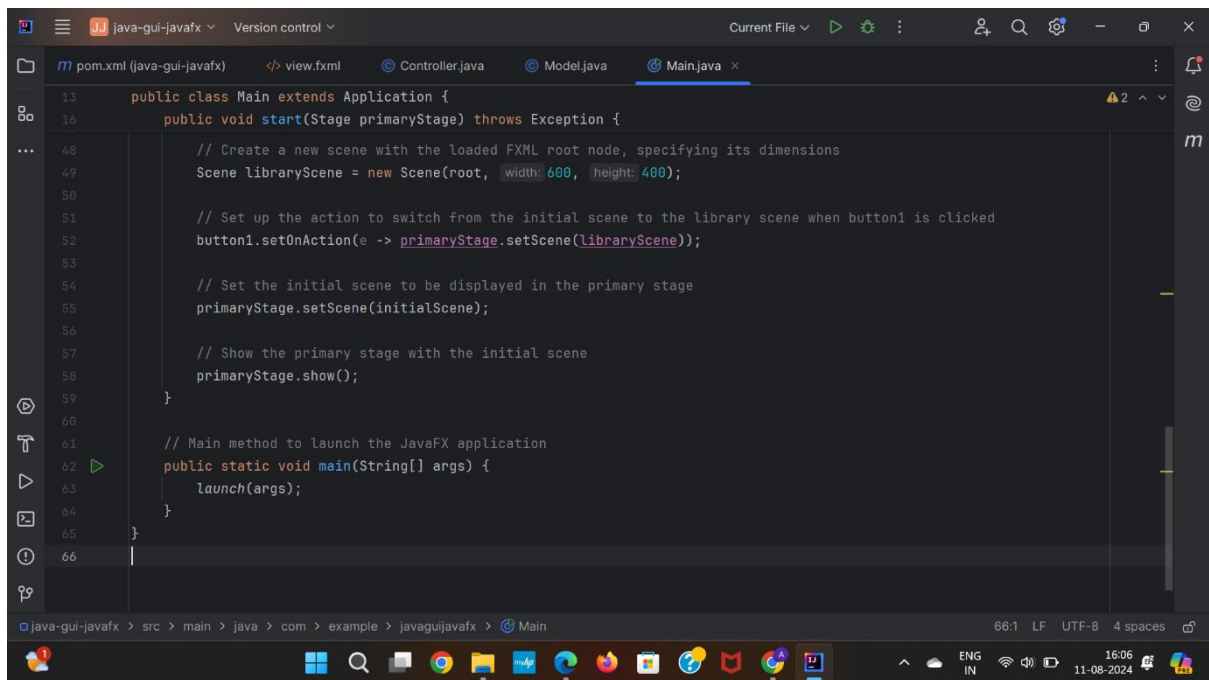
```
13 public class Main extends Application {
16     public void start(Stage primaryStage) throws Exception {
21         Button button = new Button("Proceed!");
22
23         // Create a label for displaying messages
24         Label label = new Label();
25
26         // Set up the button's action to update the label with a welcome message
27         button.setOnAction(e -> label.setText("Welcome to the Toronto Library!"));
28
29         // Create another button for navigating to the next scene
30         Button button1 = new Button("Next!");
31
32         // Create a VBox layout with 10px spacing between elements and center alignment
33         VBox layout = new VBox(spacing: 10);
34         layout.setAlignment(Pos.CENTER);
35
36         // Add the buttons and label to the VBox layout
37         layout.getChildren().addAll(button, label, button1);
38
39         // Create the initial scene with the VBox layout, specifying its dimensions
40         Scene initialScene = new Scene(layout, width: 300, height: 250);
41     }
}
```



The screenshot shows an IDE window titled 'java-gui-javafx' with a 'Version control' dropdown. The file explorer on the left shows a project structure with 'pom.xml (java-gui-javafx)', 'view.fxml', 'Controller.java', 'Model.java', and 'Main.java'. The 'Main.java' file is open, showing the following code:

```
13 public class Main extends Application {
16     public void start(Stage primaryStage) throws Exception {
42         // Load the FXML file for the next scene
43         FXMLLoader loader = new FXMLLoader(getClass().getResource("view.fxml"));
44
45         // Load the FXML file into a Parent object to define the new scene's root node
46         Parent root = loader.load();
47
48         // Create a new scene with the loaded FXML root node, specifying its dimensions
49         Scene libraryScene = new Scene(root, width: 600, height: 400);
50
51         // Set up the action to switch from the initial scene to the library scene when button1 is clicked
52         button1.setOnAction(e -> primaryStage.setScene(libraryScene));
53
54         // Set the initial scene to be displayed in the primary stage
55         primaryStage.setScene(initialScene);
56
57         // Show the primary stage with the initial scene
58         primaryStage.show();
59     }
60
61     // Main method to launch the JavaFX application
62     public static void main(String[] args) {
```

The status bar at the bottom indicates '66:1 LF UTF-8 4 spaces'.

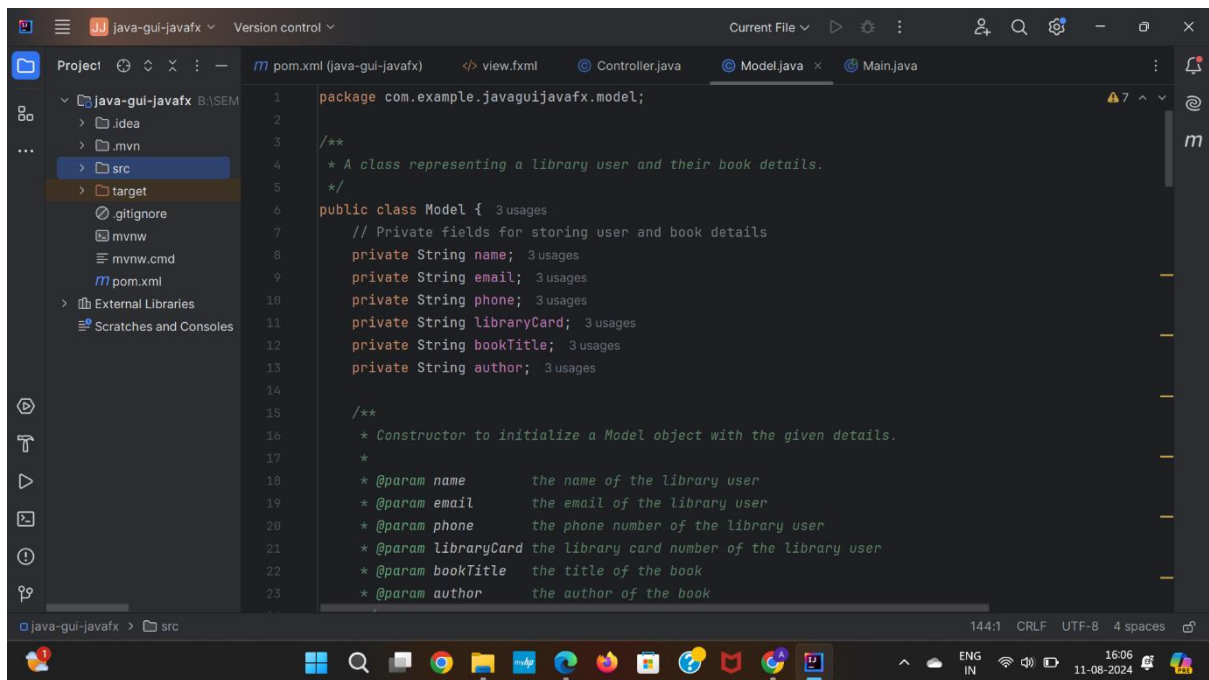


The screenshot shows the same IDE window, but the code is scrolled down to show the end of the 'main' method and the closing of the 'Main' class:

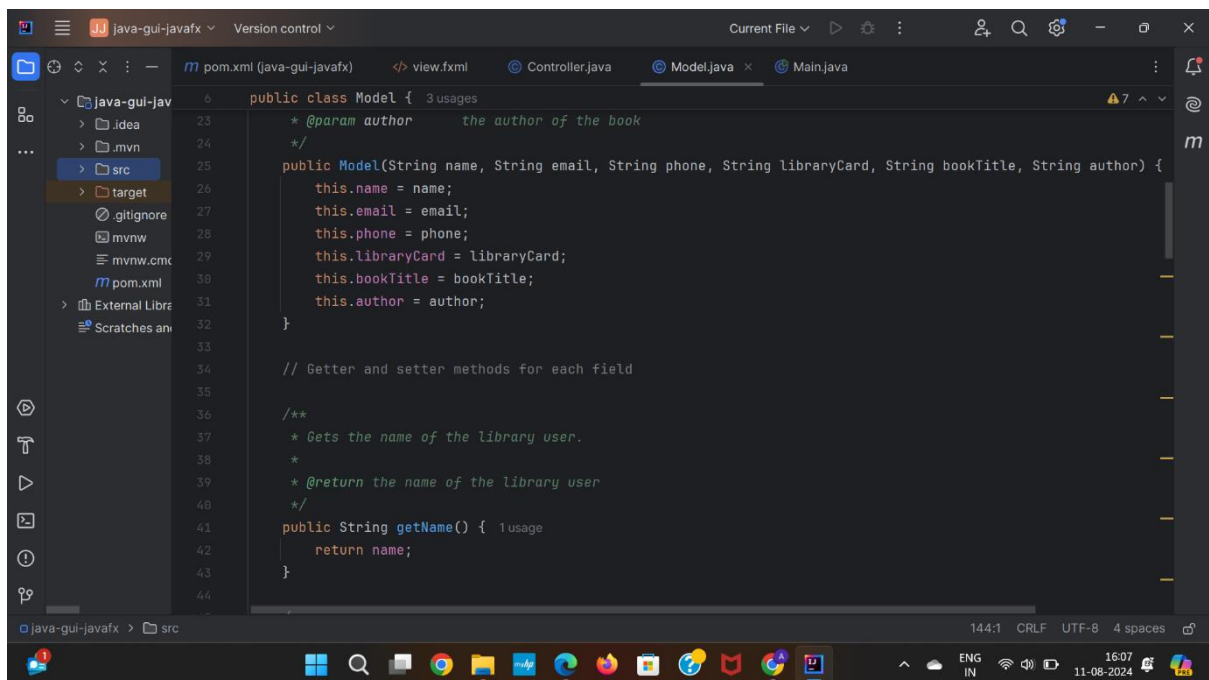
```
48         // Create a new scene with the loaded FXML root node, specifying its dimensions
49         Scene libraryScene = new Scene(root, width: 600, height: 400);
50
51         // Set up the action to switch from the initial scene to the library scene when button1 is clicked
52         button1.setOnAction(e -> primaryStage.setScene(libraryScene));
53
54         // Set the initial scene to be displayed in the primary stage
55         primaryStage.setScene(initialScene);
56
57         // Show the primary stage with the initial scene
58         primaryStage.show();
59     }
60
61     // Main method to launch the JavaFX application
62     public static void main(String[] args) {
63         launch(args);
64     }
65 }
66
```

The status bar at the bottom indicates '66:1 LF UTF-8 4 spaces'.

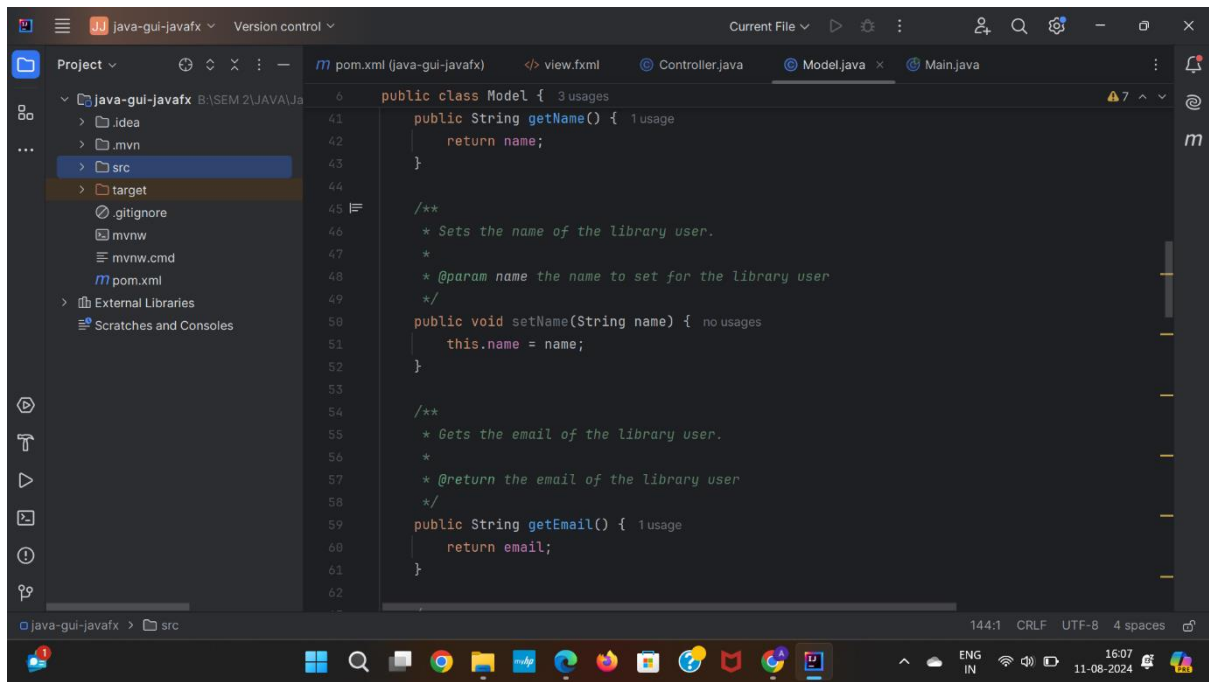
Model.java



```
1 package com.example.javaguijavafx.model;
2
3 /**
4  * A class representing a library user and their book details.
5  */
6 public class Model { 3 usages
7     // Private fields for storing user and book details
8     private String name; 3 usages
9     private String email; 3 usages
10    private String phone; 3 usages
11    private String libraryCard; 3 usages
12    private String bookTitle; 3 usages
13    private String author; 3 usages
14
15    /**
16     * Constructor to initialize a Model object with the given details.
17     *
18     * @param name the name of the library user
19     * @param email the email of the library user
20     * @param phone the phone number of the library user
21     * @param libraryCard the library card number of the library user
22     * @param bookTitle the title of the book
23     * @param author the author of the book
```



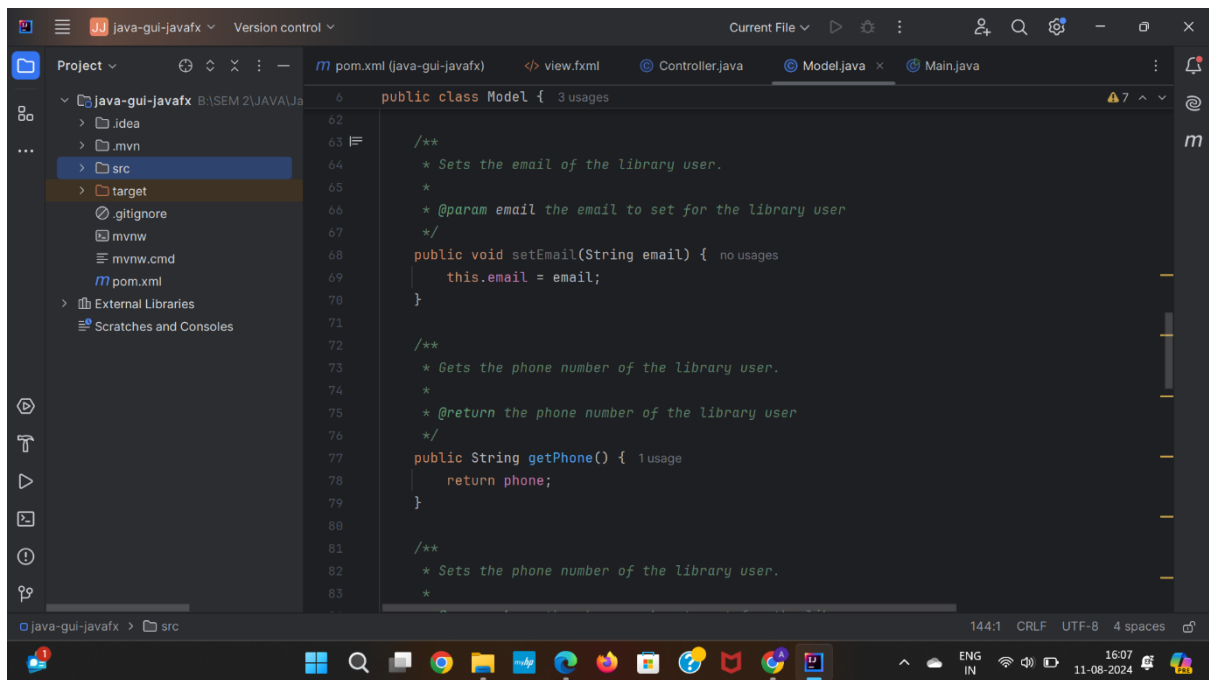
```
6 public class Model { 3 usages
7     * @param author the author of the book
8     */
9     public Model(String name, String email, String phone, String libraryCard, String bookTitle, String author) {
10         this.name = name;
11         this.email = email;
12         this.phone = phone;
13         this.libraryCard = libraryCard;
14         this.bookTitle = bookTitle;
15         this.author = author;
16     }
17
18     // Getter and setter methods for each field
19
20     /**
21     * Gets the name of the library user.
22     *
23     * @return the name of the library user
24     */
25     public String getName() { 1 usage
26         return name;
27     }
28 }
```



The screenshot shows an IDE window for a project named 'java-gui-javafx'. The 'Project' sidebar on the left shows the file structure: 'java-gui-javafx' (B:\SEM 2\JAVA\Ja) containing 'idea', '.mvn', 'src', 'target', '.gitignore', 'mvnw', 'mvnw.cmd', 'pom.xml', 'External Libraries', and 'Scratches and Consoles'. The 'src' folder is expanded. The main editor displays the 'Model.java' file with the following code:

```
6 public class Model { 3 usages
41 public String getName() { 1 usage
42     return name;
43 }
44
45 /**
46  * Sets the name of the library user.
47  *
48  * @param name the name to set for the library user
49  */
50 public void setName(String name) { no usages
51     this.name = name;
52 }
53
54 /**
55  * Gets the email of the library user.
56  *
57  * @return the email of the library user
58  */
59 public String getEmail() { 1 usage
60     return email;
61 }
62 }
```

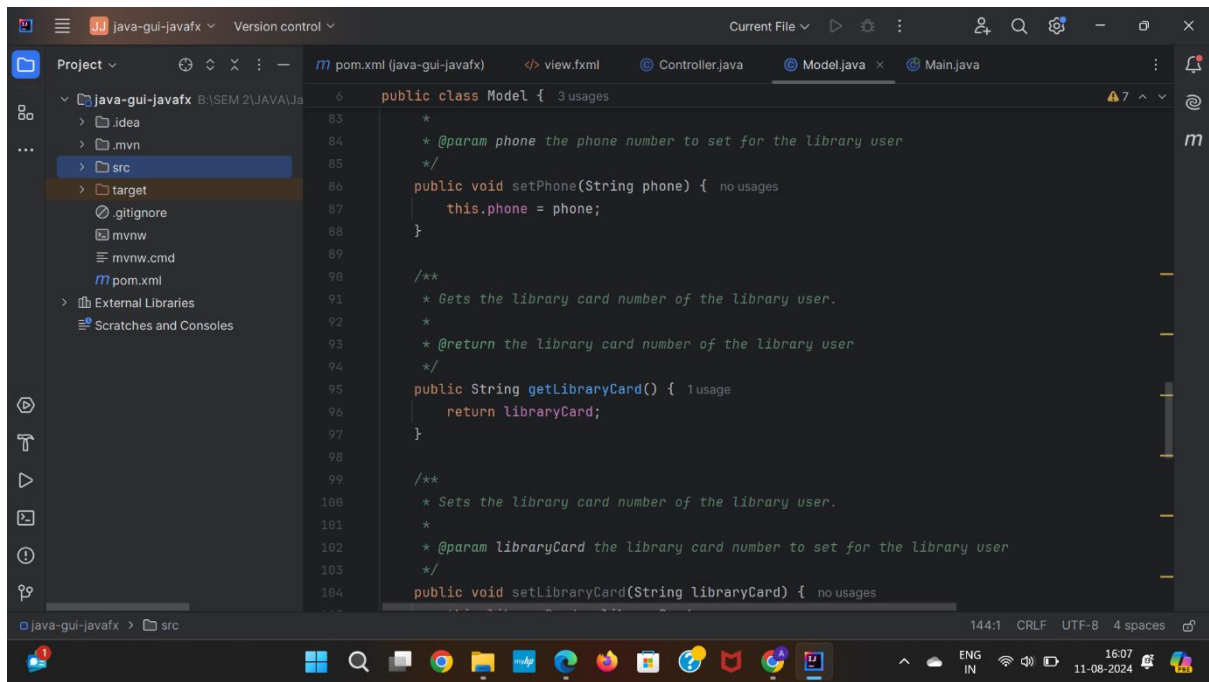
The status bar at the bottom indicates '144:1 CRLF UTF-8 4 spaces'.



The screenshot shows the same IDE window, but the 'Model.java' file is scrolled down to show different methods. The code is as follows:

```
62
63 /**
64  * Sets the email of the library user.
65  *
66  * @param email the email to set for the library user
67  */
68 public void setEmail(String email) { no usages
69     this.email = email;
70 }
71
72 /**
73  * Gets the phone number of the library user.
74  *
75  * @return the phone number of the library user
76  */
77 public String getPhone() { 1 usage
78     return phone;
79 }
80
81 /**
82  * Sets the phone number of the library user.
83  *
```

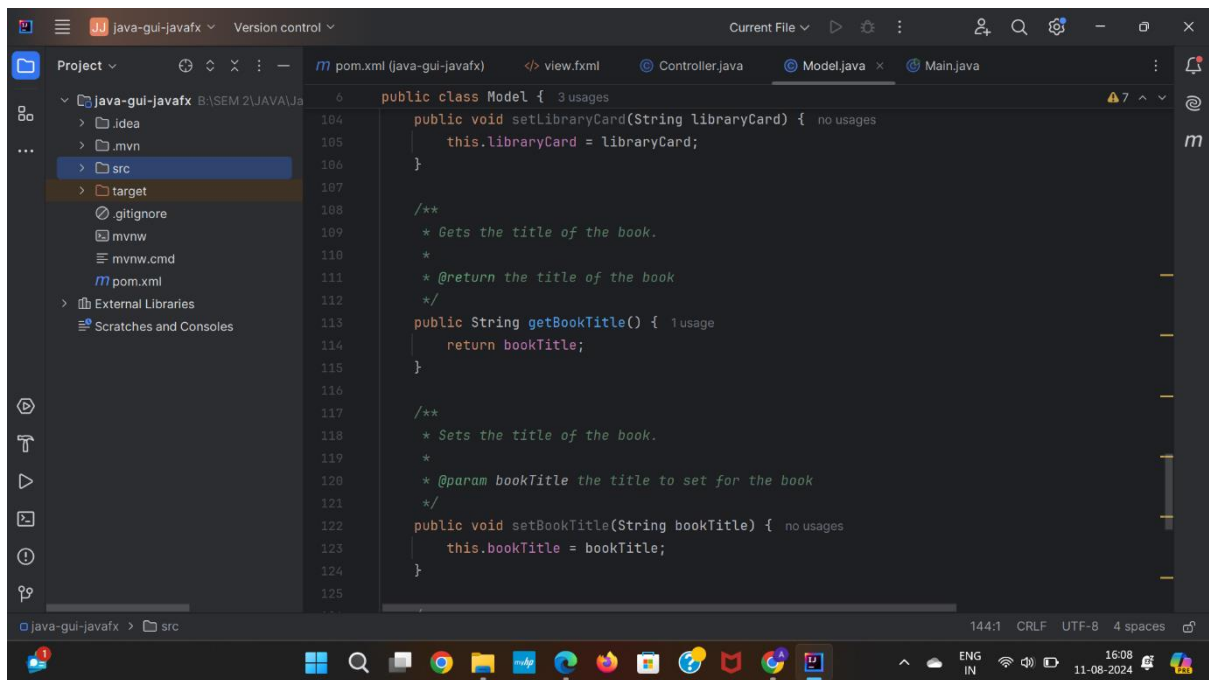
The status bar at the bottom indicates '144:1 CRLF UTF-8 4 spaces'.



The screenshot shows an IDE window for a project named 'java-gui-javafx'. The 'Project' sidebar on the left shows the file structure: 'java-gui-javafx' (B:\SEM 2\JAVA\Ja) containing 'idea', '.mvn', 'src', 'target', '.gitignore', 'mvnw', 'mvnw.cmd', 'pom.xml', 'External Libraries', and 'Scratches and Consoles'. The 'src' folder is expanded, showing 'target'. The 'pom.xml' file is selected. The main editor displays the 'Model.java' file with the following code:

```
public class Model {  
    *  
    * @param phone the phone number to set for the library user  
    */  
    public void setPhone(String phone) {  
        this.phone = phone;  
    }  
  
    /**  
    * Gets the library card number of the library user.  
    *  
    * @return the library card number of the library user  
    */  
    public String getLibraryCard() {  
        return libraryCard;  
    }  
  
    /**  
    * Sets the library card number of the library user.  
    *  
    * @param libraryCard the library card number to set for the library user  
    */  
    public void setLibraryCard(String libraryCard) {  
        this.libraryCard = libraryCard;  
    }  
}
```

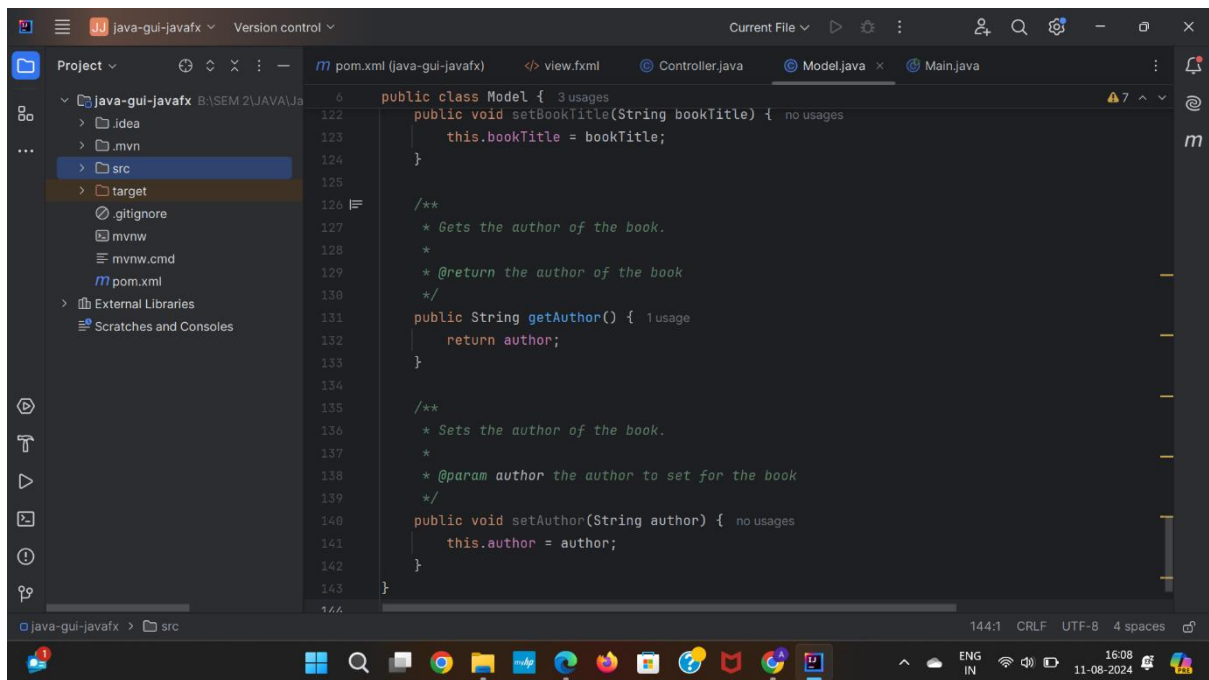
The status bar at the bottom indicates the file is 'pom.xml (java-gui-javafx)' with a line number of 144, CRLF line endings, UTF-8 encoding, and 4 spaces for indentation. The system tray shows the date and time as 16:07 on 11-08-2024.



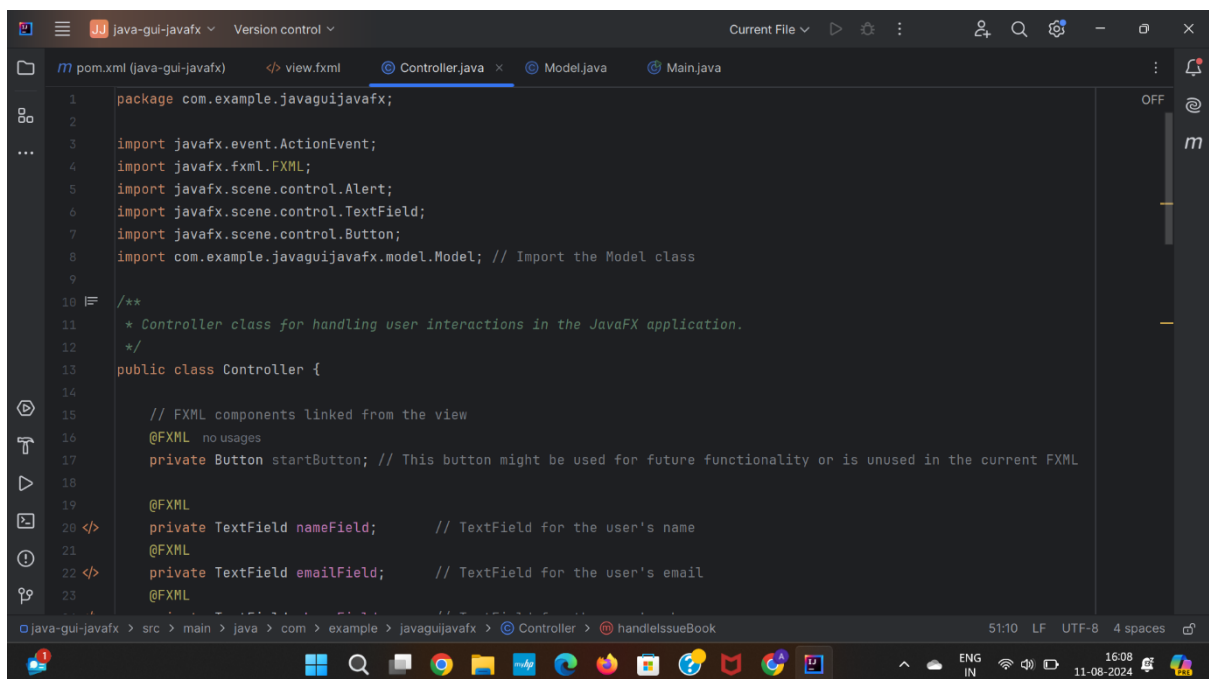
The screenshot shows the same IDE window for the 'java-gui-javafx' project. The 'pom.xml' file is still selected in the sidebar. The main editor displays the 'Model.java' file with the following code:

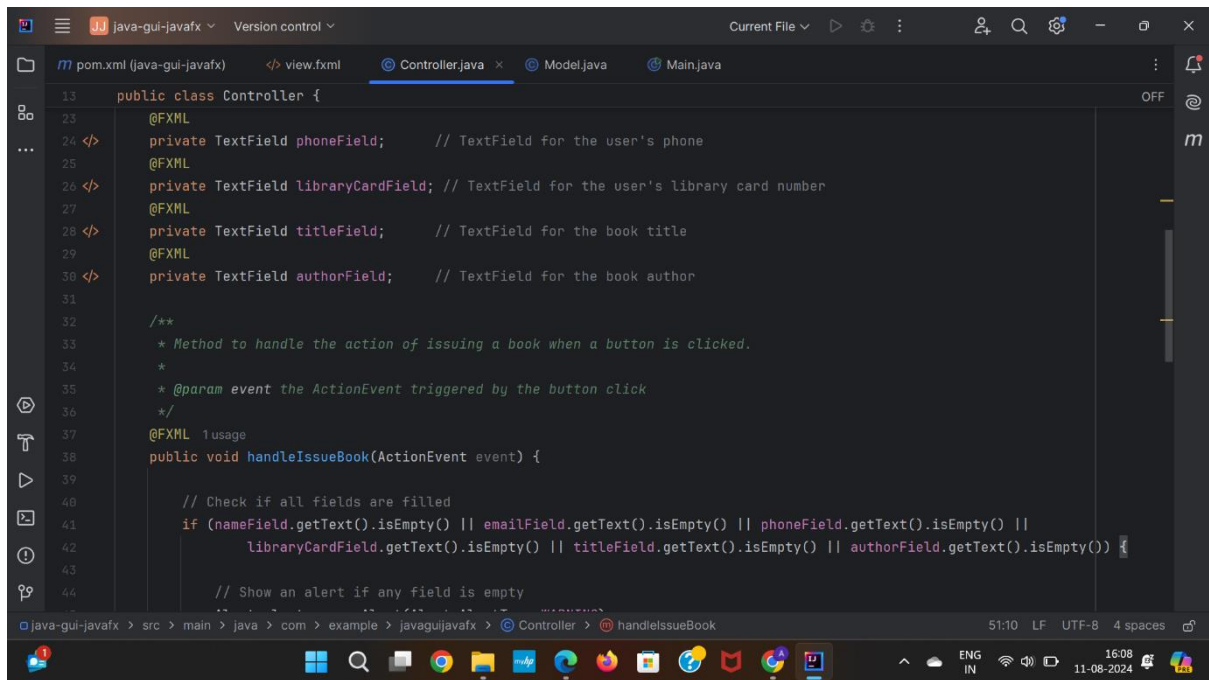
```
public class Model {  
    public void setLibraryCard(String libraryCard) {  
        this.libraryCard = libraryCard;  
    }  
  
    /**  
    * Gets the title of the book.  
    *  
    * @return the title of the book  
    */  
    public String getBookTitle() {  
        return bookTitle;  
    }  
  
    /**  
    * Sets the title of the book.  
    *  
    * @param bookTitle the title to set for the book  
    */  
    public void setBookTitle(String bookTitle) {  
        this.bookTitle = bookTitle;  
    }  
}
```

The status bar at the bottom indicates the file is 'pom.xml (java-gui-javafx)' with a line number of 144, CRLF line endings, UTF-8 encoding, and 4 spaces for indentation. The system tray shows the date and time as 16:08 on 11-08-2024.

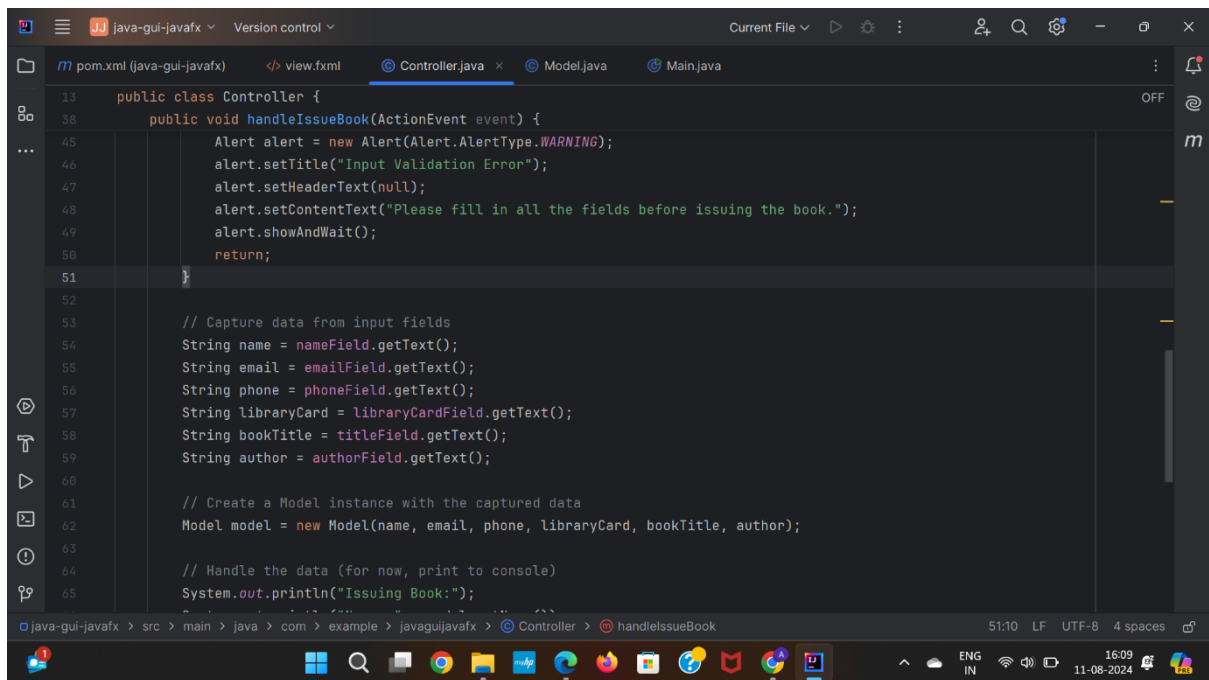


Controller.java

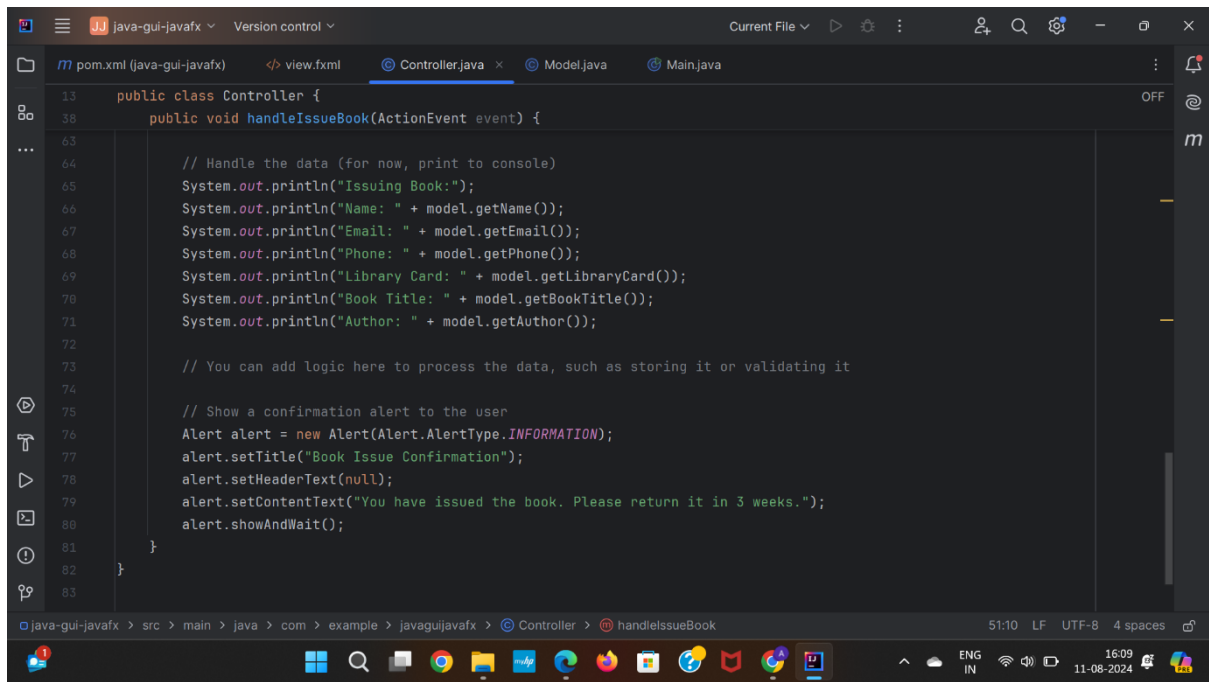




```
13 public class Controller {
23     @FXML
24     private TextField phoneField; // TextField for the user's phone
25
26     @FXML
27     private TextField libraryCardField; // TextField for the user's library card number
28
29     @FXML
30     private TextField titleField; // TextField for the book title
31
32     @FXML
33     private TextField authorField; // TextField for the book author
34
35     /**
36      * Method to handle the action of issuing a book when a button is clicked.
37      *
38      * @param event the ActionEvent triggered by the button click
39      */
40     @FXML 1 usage
41     public void handleIssueBook(ActionEvent event) {
42
43         // Check if all fields are filled
44         if (nameField.getText().isEmpty() || emailField.getText().isEmpty() || phoneField.getText().isEmpty() ||
45             libraryCardField.getText().isEmpty() || titleField.getText().isEmpty() || authorField.getText().isEmpty()) {
46
47             // Show an alert if any field is empty
48
49         }
50     }
51 }
```

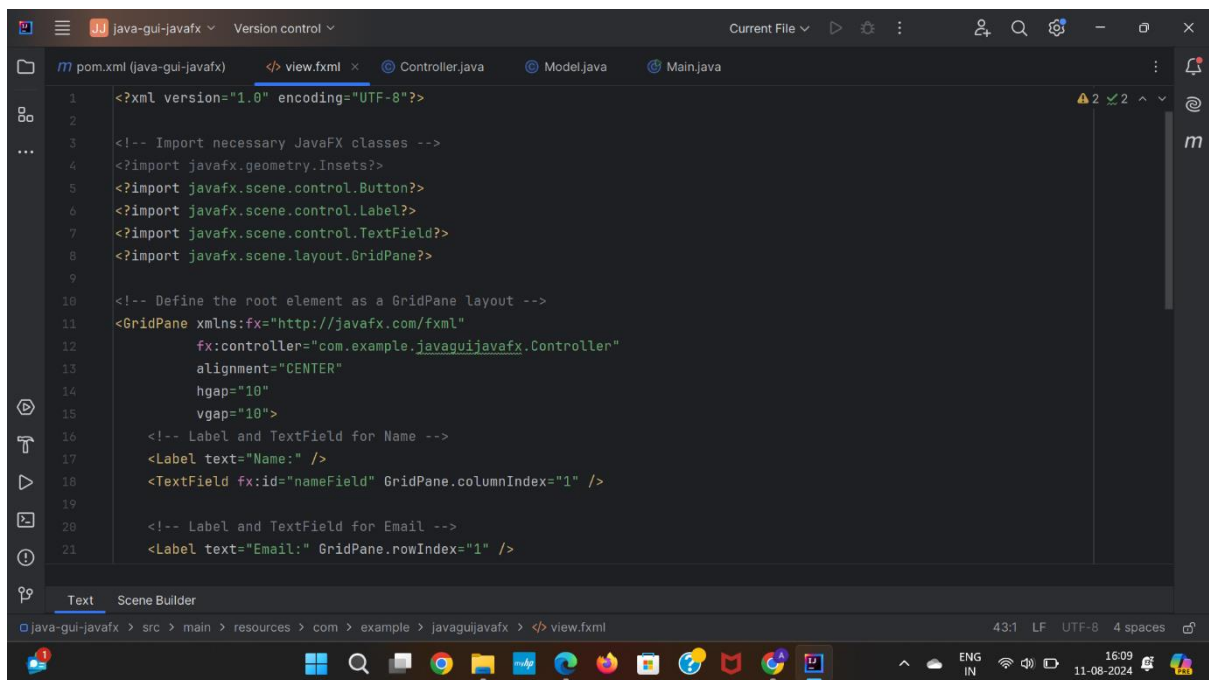


```
38 public void handleIssueBook(ActionEvent event) {
45     Alert alert = new Alert(Alert.AlertType.WARNING);
46     alert.setTitle("Input Validation Error");
47     alert.setHeaderText(null);
48     alert.setContentText("Please fill in all the fields before issuing the book.");
49     alert.showAndWait();
50     return;
51
52
53     // Capture data from input fields
54     String name = nameField.getText();
55     String email = emailField.getText();
56     String phone = phoneField.getText();
57     String libraryCard = libraryCardField.getText();
58     String bookTitle = titleField.getText();
59     String author = authorField.getText();
60
61     // Create a Model instance with the captured data
62     Model model = new Model(name, email, phone, libraryCard, bookTitle, author);
63
64     // Handle the data (for now, print to console)
65     System.out.println("Issuing Book:");
66 }
```

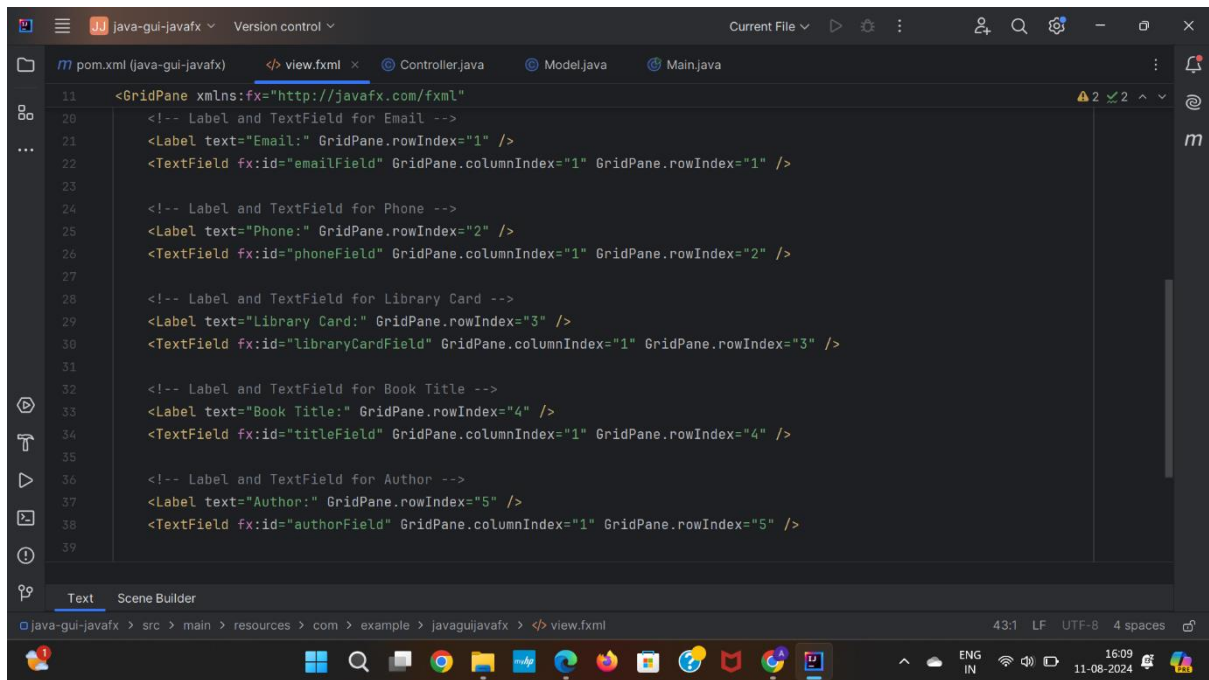


```
13 public class Controller {
38     public void handleIssueBook(ActionEvent event) {
63
64         // Handle the data (for now, print to console)
65         System.out.println("Issuing Book:");
66         System.out.println("Name: " + model.getName());
67         System.out.println("Email: " + model.getEmail());
68         System.out.println("Phone: " + model.getPhone());
69         System.out.println("Library Card: " + model.getLibraryCard());
70         System.out.println("Book Title: " + model.getBookTitle());
71         System.out.println("Author: " + model.getAuthor());
72
73         // You can add logic here to process the data, such as storing it or validating it
74
75         // Show a confirmation alert to the user
76         Alert alert = new Alert(Alert.AlertType.INFORMATION);
77         alert.setTitle("Book Issue Confirmation");
78         alert.setHeaderText(null);
79         alert.setContentText("You have issued the book. Please return it in 3 weeks.");
80         alert.showAndWait();
81     }
82 }
83
```

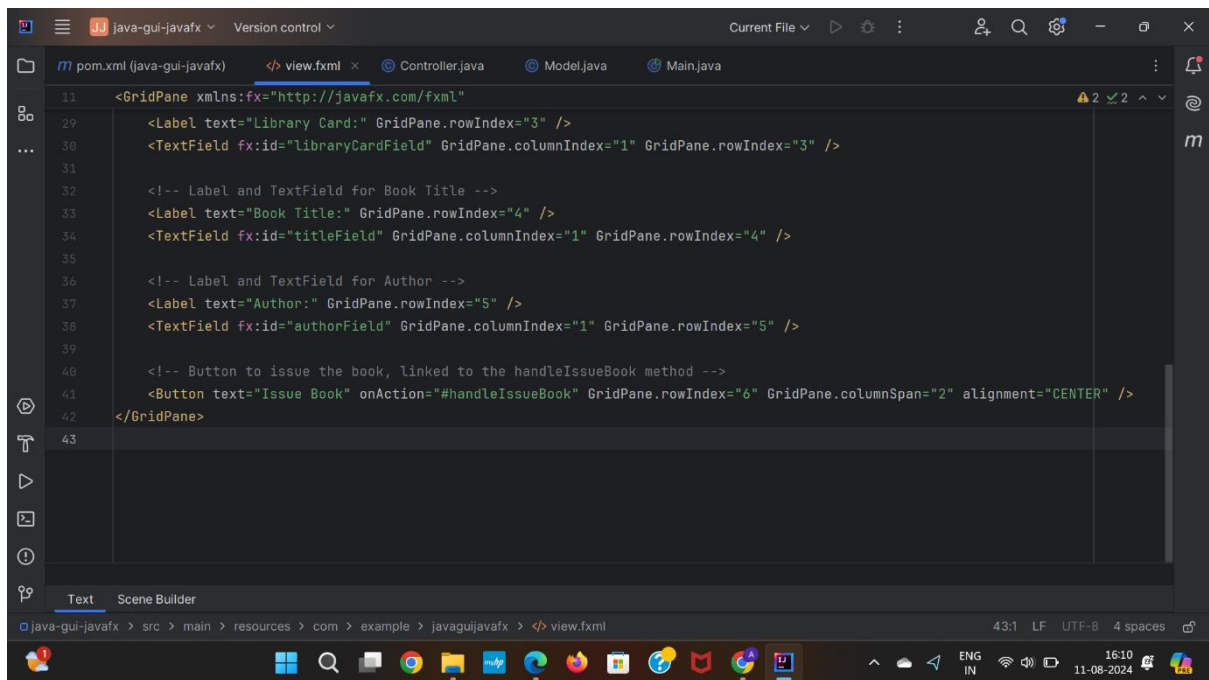
View.fxml



```
1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <!-- Import necessary JavaFX classes -->
4 <?import javafx.geometry.Insets?>
5 <?import javafx.scene.control.Button?>
6 <?import javafx.scene.control.Label?>
7 <?import javafx.scene.control.TextField?>
8 <?import javafx.scene.layout.GridPane?>
9
10 <!-- Define the root element as a GridPane layout -->
11 <GridPane xmlns:fx="http://javafx.com/fxml"
12     fx:controller="com.example.javaguijavafx.Controller"
13     alignment="CENTER"
14     hgap="10"
15     vgap="10">
16
17     <!-- Label and TextField for Name -->
18     <Label text="Name:" />
19     <TextField fx:id="nameField" GridPane.columnIndex="1" />
20
21     <!-- Label and TextField for Email -->
22     <Label text="Email:" GridPane.rowIndex="1" />
```

```
11 <GridPane xmlns:fx="http://javafx.com/fxml"
20 <!-- Label and TextField for Email -->
21 <Label text="Email:" GridPane.rowIndex="1" />
22 <TextField fx:id="emailField" GridPane.columnIndex="1" GridPane.rowIndex="1" />
23
24 <!-- Label and TextField for Phone -->
25 <Label text="Phone:" GridPane.rowIndex="2" />
26 <TextField fx:id="phoneField" GridPane.columnIndex="1" GridPane.rowIndex="2" />
27
28 <!-- Label and TextField for Library Card -->
29 <Label text="Library Card:" GridPane.rowIndex="3" />
30 <TextField fx:id="libraryCardField" GridPane.columnIndex="1" GridPane.rowIndex="3" />
31
32 <!-- Label and TextField for Book Title -->
33 <Label text="Book Title:" GridPane.rowIndex="4" />
34 <TextField fx:id="titleField" GridPane.columnIndex="1" GridPane.rowIndex="4" />
35
36 <!-- Label and TextField for Author -->
37 <Label text="Author:" GridPane.rowIndex="5" />
38 <TextField fx:id="authorField" GridPane.columnIndex="1" GridPane.rowIndex="5" />
39
```



```
40 <!-- Button to issue the book, linked to the handleIssueBook method -->
41 <Button text="Issue Book" onAction="#handleIssueBook" GridPane.rowIndex="6" GridPane.columnSpan="2" alignment="CENTER" />
42 </GridPane>
43
```

Result:

