BSc (Hons) Artificial Intelligence and Data Science

Module: CM1601 Programming Fundamentals

Coursework <02> Report

RGU Student ID : 2410212

IIT Student ID : 20233136

Student Name : Mathusha Kannathasan

Executive Summary

This report contains the well developed & executable programme for the TechExpo event to select top projects from each category. From that, they will be awarded by judges. The flow of this programme is sequentially designed for the accessible of the users. It is done using JavaFx project in IntelliJ IDEA software platform. An attractive graphical user interface is made for the event for users to connect with the platform. Furthermore, this report contains structured flowchart & well explained for each controller, necessary Junit tests, required assumptions & conclusions with wide range of references.

Table of Contents

Table of Figures

Flow Charts

1. MainMenu

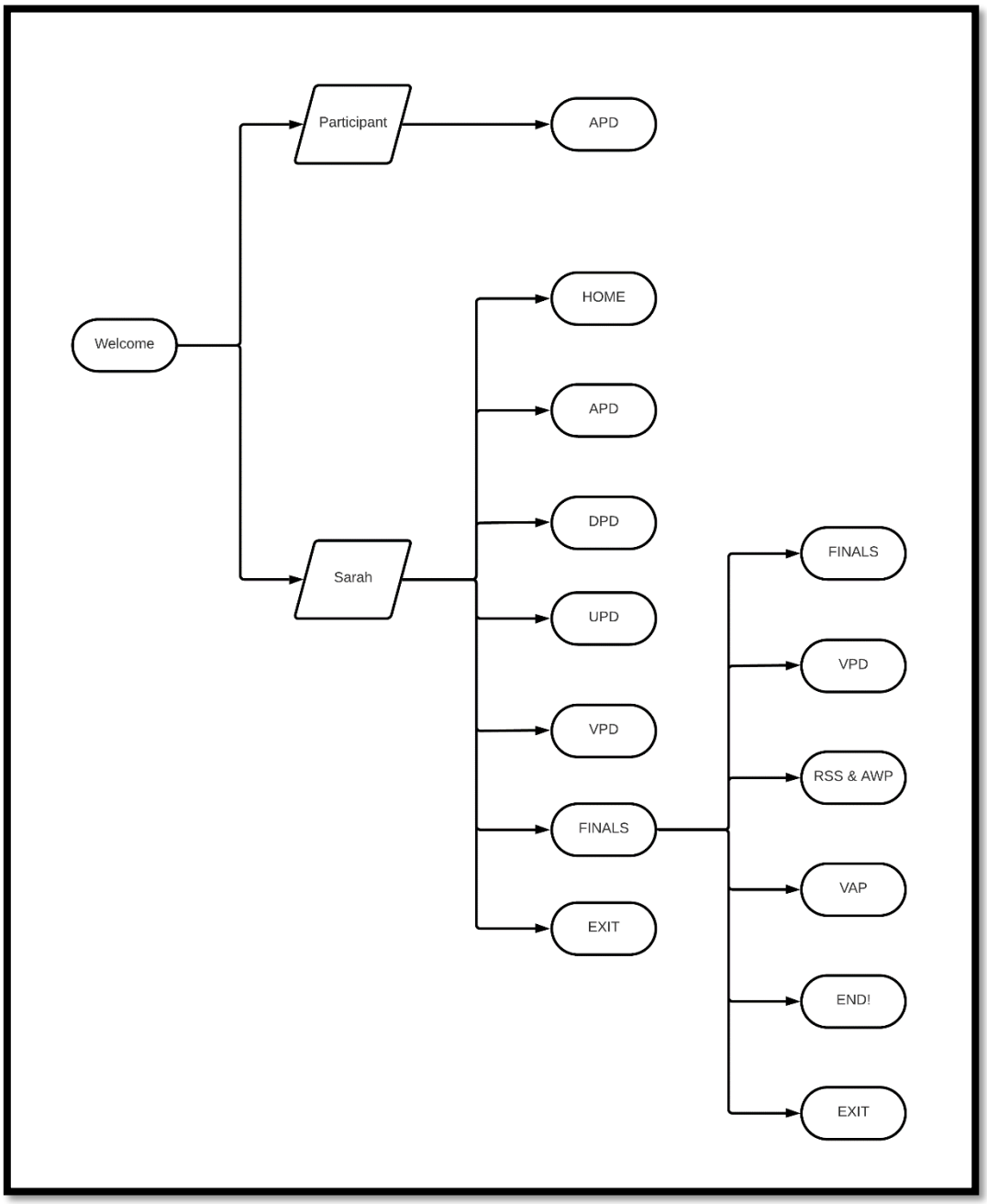


Figure 1 - FlowchartMenu

2. AddProjectDetails

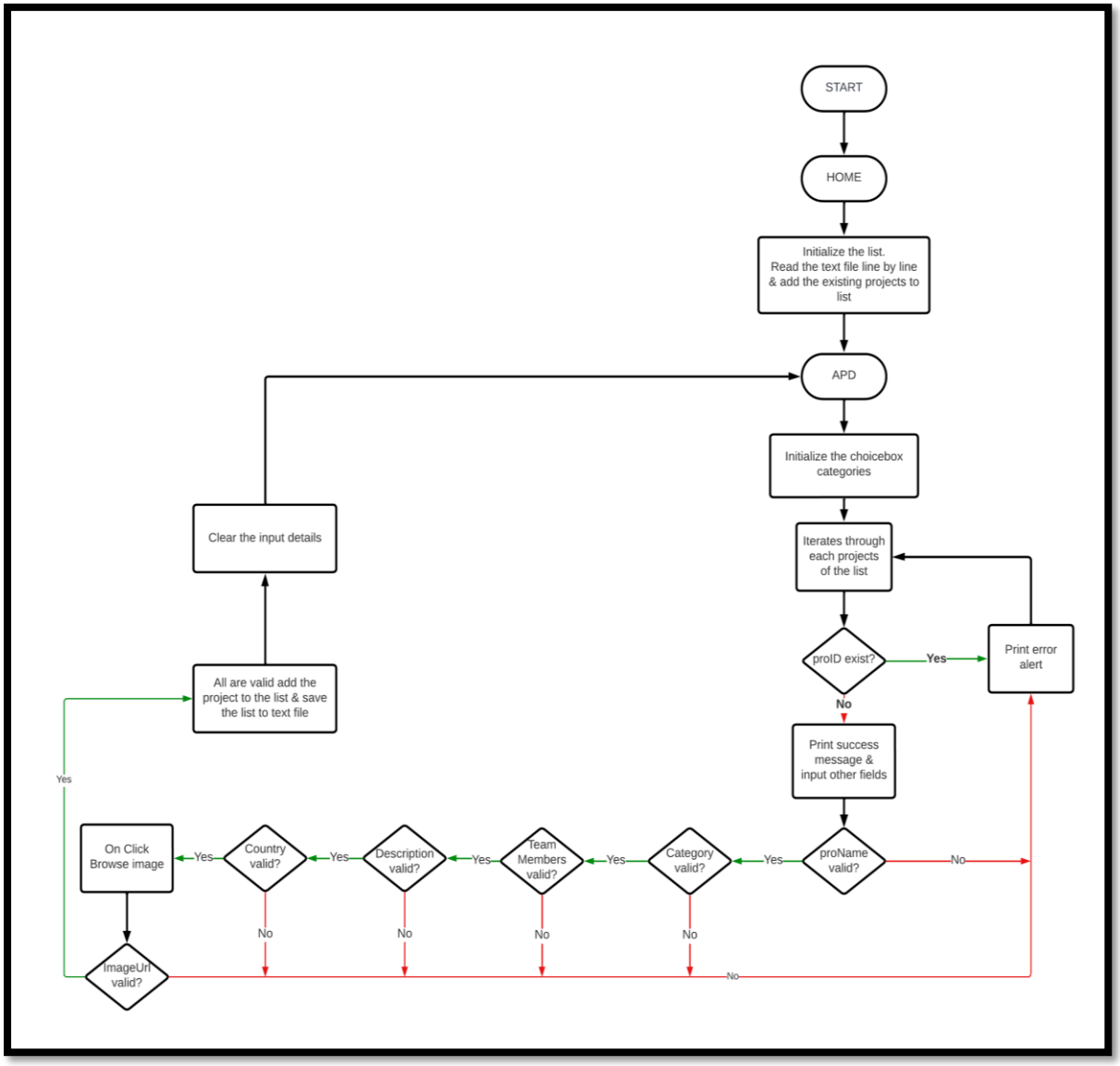


Figure 2 - FlowchartAPD

3. DeleteProjectDetails

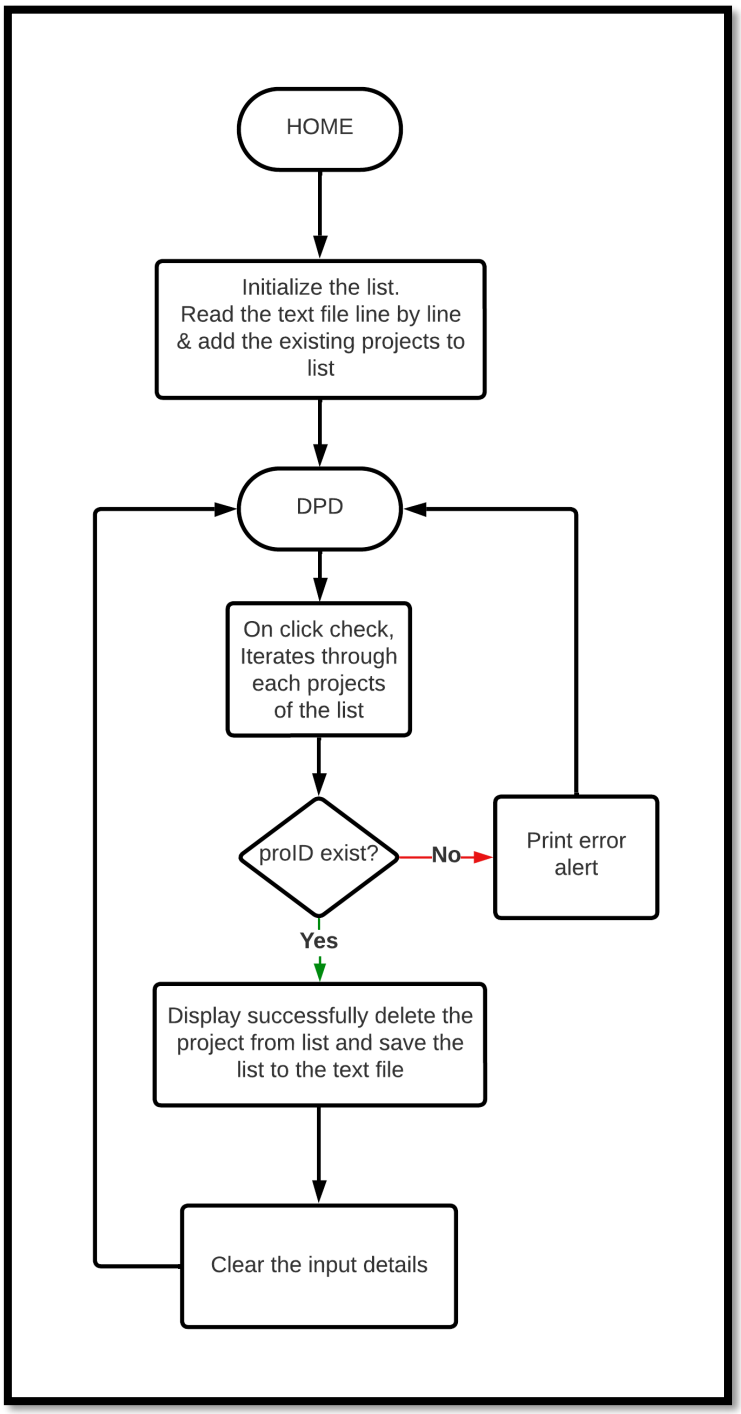


Figure 3 - FlowchartDPD

4. UpdateProjectDetails

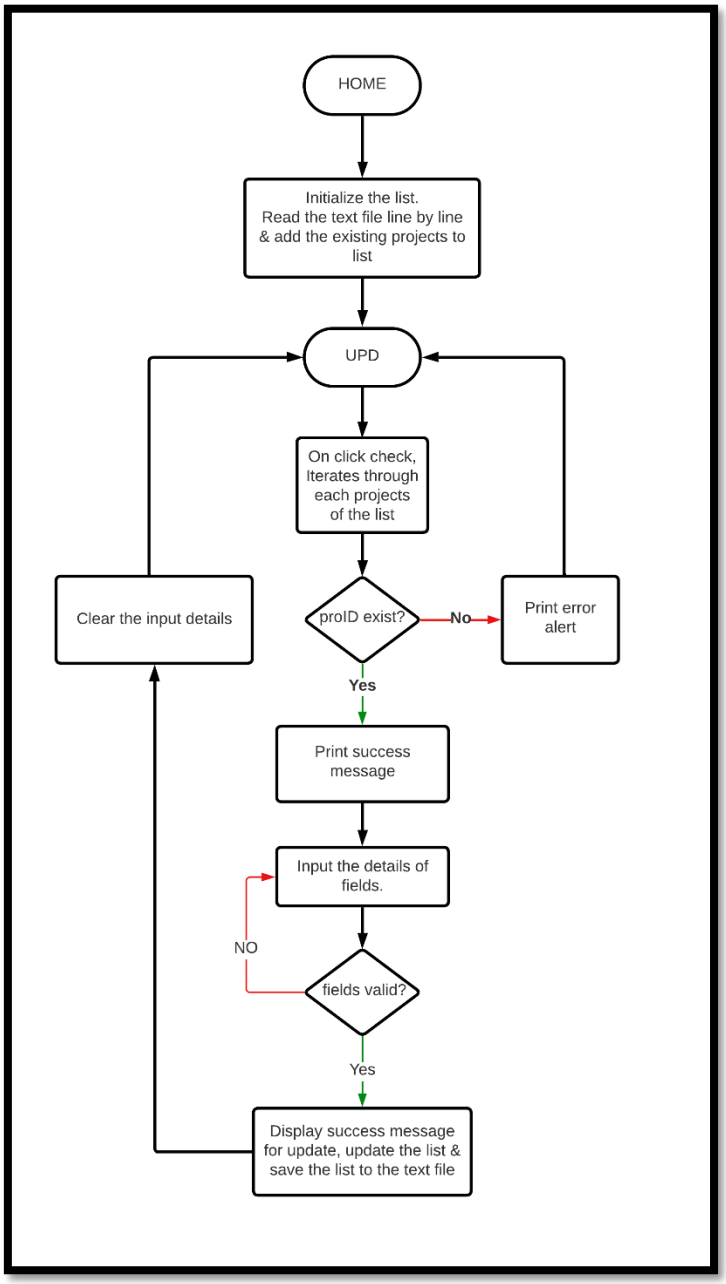


Figure 4 - FlowchartUPD

Note: These three controllers's flowcharts (APD, DPD, UPD) include SavingProjectDetails to the TextFile. Therefore did not draw a separate flowchart for that.

5. ViewProjectDetails

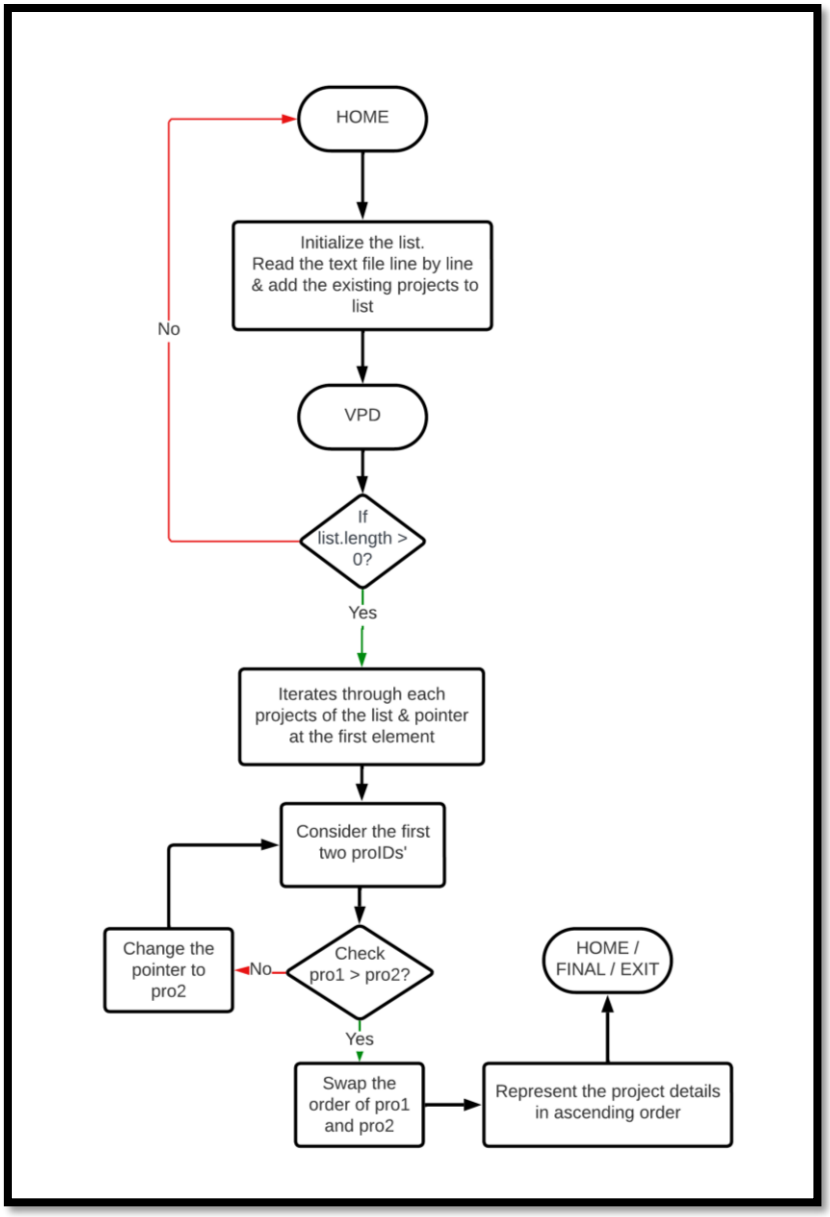


Figure 5 - FlowchartVPD

6. START (Path to FINAL)

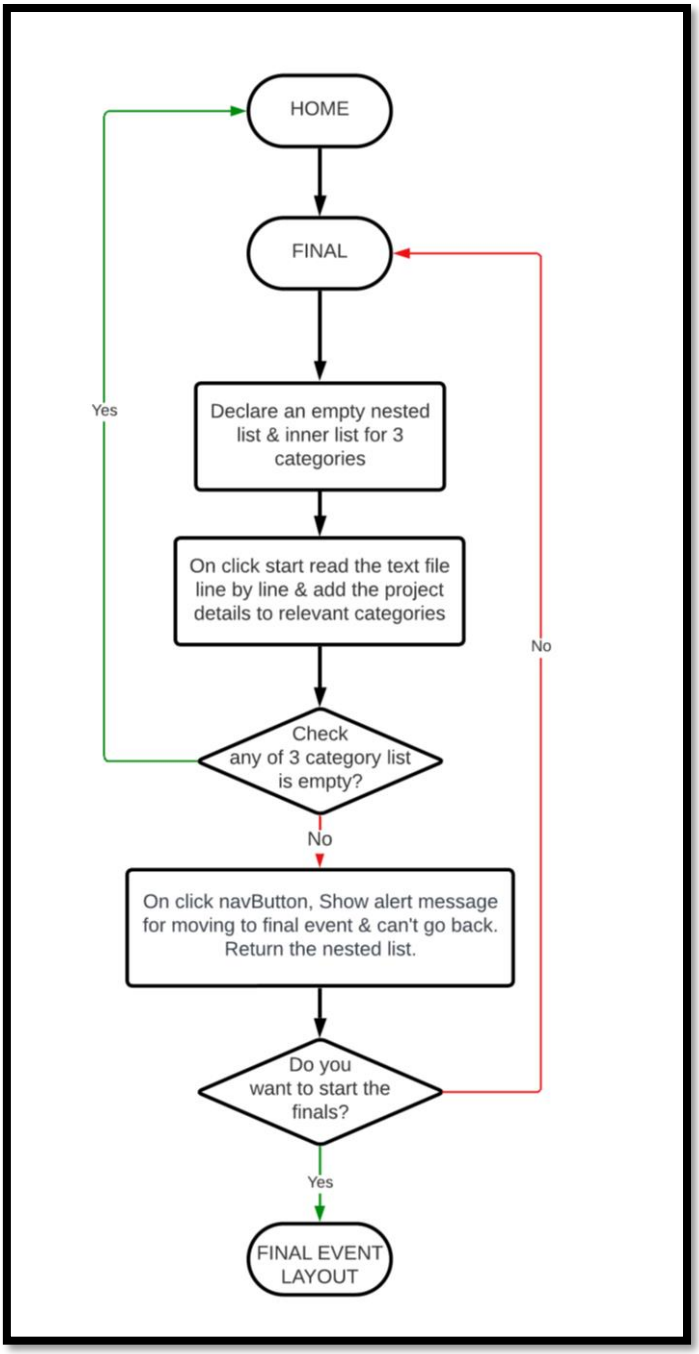


Figure 6 - FlowchartStart

7. RandomSpotlightSelection

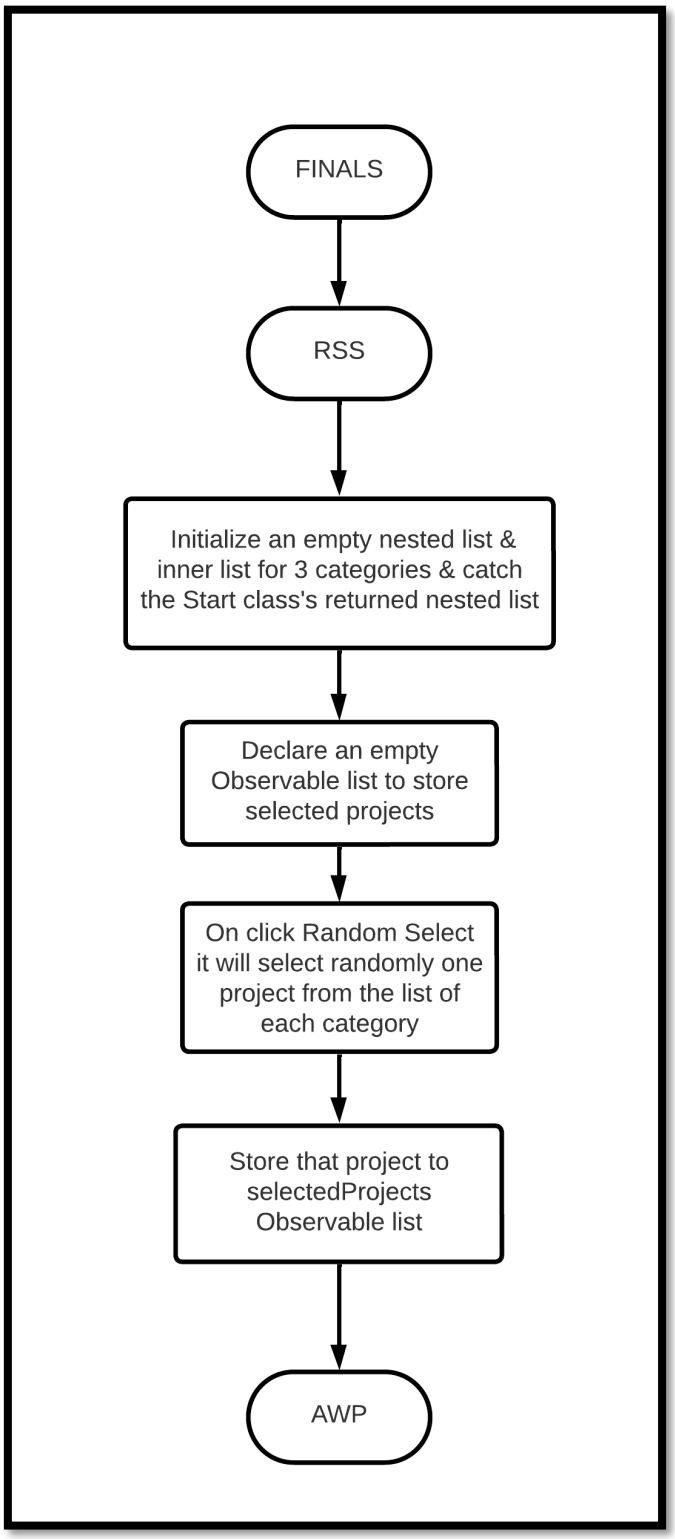


Figure 7 - FlowchartRSS

8. AwardWinningProjects

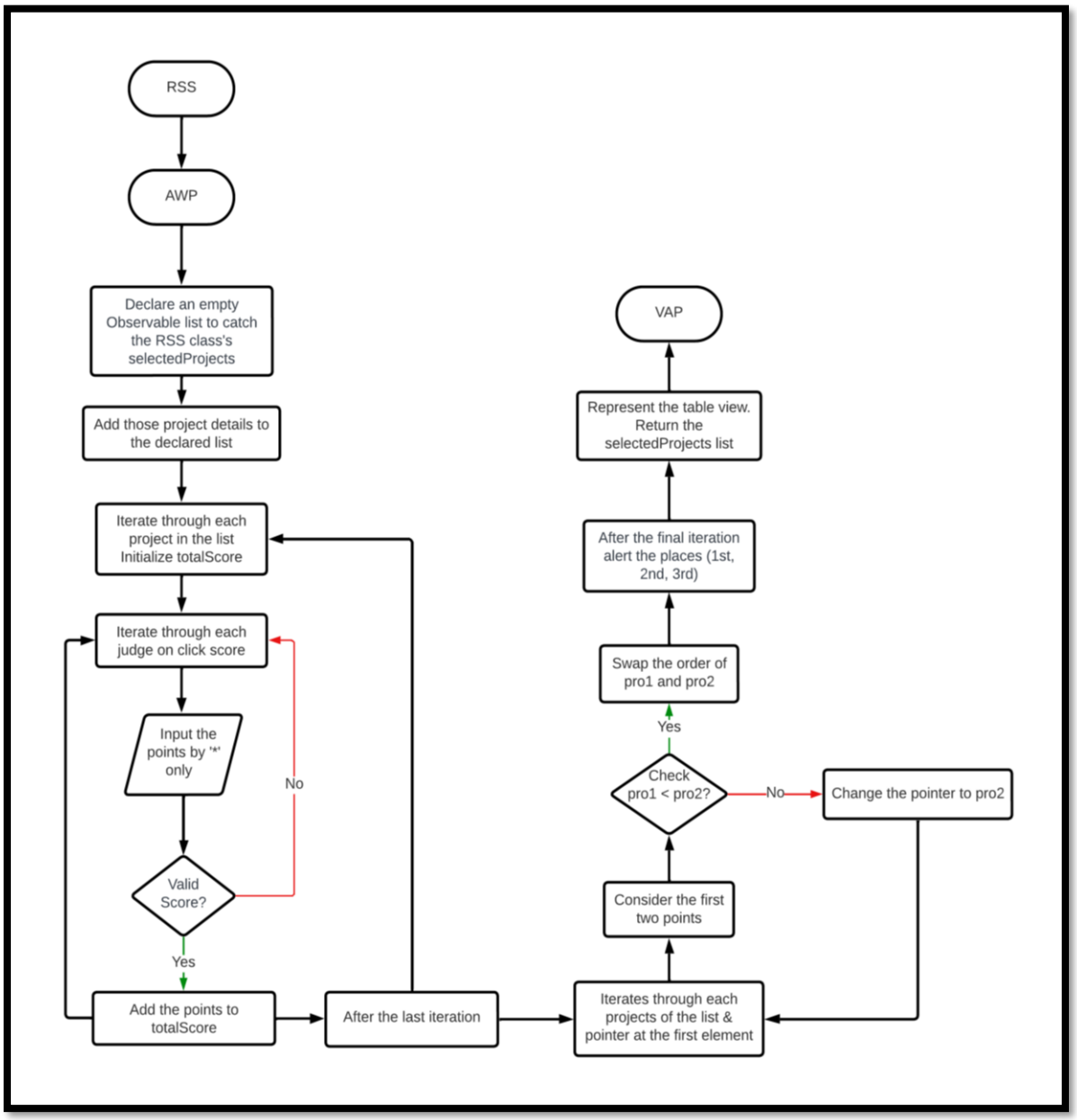


Figure 8 - FlowchartAWP

9. VisualizingAwardWinningProjects

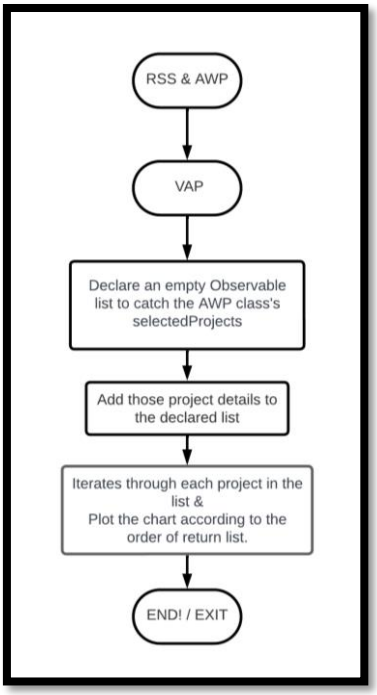


Figure 9 - FlowchartVAP

10. EXIT

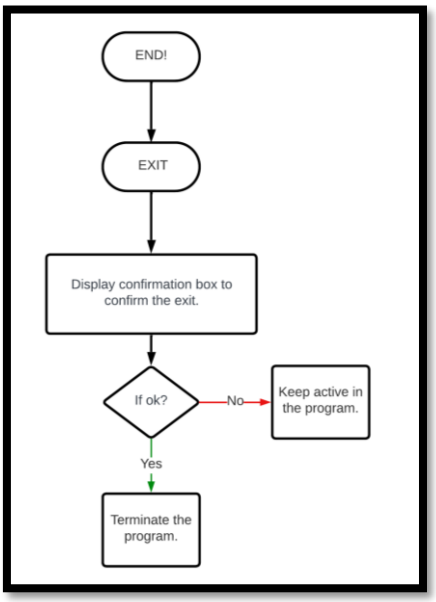


Figure 10 - FlowchartEXIT

Introduction to Functions with Code.

1. Project Class

a) Code

```
package JavaCoursework;

import javafx.scene.image.Image;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

public class Project implements Serializable {
    private int projectID;
    private String projectName;
    private String category;
    private List<String> teamMembers;
    private String description;
    private String country;
    private String teamLogoUrl; // Store URL instead of Image directly
    private Image teamLogo;
    private int judgesPoints;

    public Project(int projectID, String projectName, String category, List<String> teamMembers, String
description, String country, String teamLogoUrl) {
        this.projectID = projectID;
        this.projectName = projectName;
        this.category = category;
        this.teamMembers = teamMembers;
        this.description = description;
        this.country = country;
        this.teamLogoUrl = teamLogoUrl;
    }

    public Project(int projectID, String projectName, String category, List<String> teamMembers, String
description, String country) {
        this.projectID = projectID;
        this.projectName = projectName;
        this.category = category;
        this.teamMembers = teamMembers;
        this.description = description;
        this.country = country;
        this.judgesPoints = 0;
    }

    public int getProjectID() {
        return projectID;
    }

    public void setProjectID(int projectID) {
        this.projectID = projectID;
    }

    public String getProjectName() {
        return projectName;
    }

    public void setProjectName(String projectName) {
        this.projectName = projectName;
    }

    public String getCategory() {
        return category;
    }

    public void setCategory(String category) {
        this.category = category;
```

```java
    }

    public String getTeamMembersAsString() {
        return String.join(", ", teamMembers);
    }

    public List<String> getTeamMembers() {
        return teamMembers;
    }

    public void setTeamMembers(List<String> teamMembers) {
        this.teamMembers = teamMembers;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public String getCountry() {
        return country;
    }

    public void setCountry(String country) {
        this.country = country;
    }

    public String getTeamLogoUrl() {
        return teamLogoUrl;
    }

    public void setTeamLogoUrl(String teamLogoUrl) {
        this.teamLogoUrl = teamLogoUrl;
        this.teamLogo = loadTeamLogo();
    }

    public void setTeamLogo(Image teamLogoImage) {
        this.teamLogo = teamLogoImage;
    }

   public Image getTeamLogo() {
        return teamLogo;
    }

    public int getJudgesPoints() { return judgesPoints; }

    public void setJudgesPoints(int judgesPoints) { this.judgesPoints = judgesPoints; }

    @Override
    public String toString() {
        return "Project{" +
                "projectID=" + projectID +
                ", projectName='" + projectName + '\'' +
                ", category='" + category + '\'' +
                ", teamMembers=" + teamMembers +
                ", description='" + description + '\'' +
                ", country='" + country + '\'' +
                ", teamLogoUrl='" + teamLogoUrl + '\'' +
                ", judgesPoints=" + judgesPoints +
                '}';
    }

    public Image loadTeamLogo() {
        try {
            return new Image(teamLogoUrl, true);
        } catch (Exception e) {
            System.err.println("Failed to load team logo: " + e.getMessage());
            return null; // Return null or a default image
        }
    }

    public static List<String> formatTeamMembers(String input) {
        String[] membersArray = input.trim().split(",");
        List<String> teamMembers = new ArrayList<>();
```

```
        for (String member : membersArray) {
            teamMembers.add(capitalizeFirstLetter(member.trim()));
        }
        return teamMembers;
    }

    static String capitalizeFirstLetter(String input) {
        return input.substring(0, 1).toUpperCase() + input.substring(1).toLowerCase();
    }

    static boolean isValidCategory(String category) {
        return category.equalsIgnoreCase("AI") || category.equalsIgnoreCase("ML") ||
category.equalsIgnoreCase("RT");
    }

    static String formatDescription(String description) {
        String[] sentences = description.split("(?<=[.!?])\\s*");
        StringBuilder capitalizedDescription = new StringBuilder();
        for (String sentence : sentences) {
            capitalizedDescription.append(sentence.substring(0, 1).toUpperCase())
                    .append(sentence.substring(1).toLowerCase())
                    .append(" ");
        }
        return capitalizedDescription.toString().trim();
    }
}
```

b) Description

The ‘Project’ class is a comprehensive representation of a project's details, encapsulating essential attributes such as the project's unique ID, name, category, team members, description, country, team logo URL, and judges' points. The class offers constructors to initialize these attributes, including an optional constructor for setting the team logo URL. It provides various getter and setter methods to access and modify the project attributes, such as the project ID, name, category, team members, description, country, and team logo URL. The ‘setTeamLogoUrl’ method updates the logo URL and utilizes the ‘loadTeamLogo’ method to fetch the logo image, handling potential errors gracefully. Additional functionalities include ‘formatTeamMembers’, which formats and capitalizes team member names, ‘isValidCategory’ for validating project categories, and ‘formatDescription’, which ensures proper capitalization of sentences in the description. The ‘toString’ method provides a detailed string representation of the project, consolidating all its attributes for display and debugging purposes. Overall, the ‘Project’ class ensures efficient management and formatting of project details within the application, including robust handling of team logo URLs and proper formatting of team member names and project descriptions.

2. TechExpoApplication

a) Code

```
package JavaCoursework;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.image.Image;
import javafx.stage.Stage;
```

```java
/*** Main application class for the TechExpo application. This class extends the JavaFX Application
class and sets up the main stage. */
public class TechExpoApplication extends Application {

    @ Override
    public void start(Stage primaryStage) throws Exception {
        Parent root = FXMLLoader.load(getClass().getResource("WelcomeScreen.fxml"));
        // Create a new scene with the loaded layout and set its size
        Scene scene = new Scene(root, 980, 700);

        // Load the application icon
        Image icon = new Image("/projectIcon.png");
        primaryStage.getIcons().add(icon);
        primaryStage.setTitle("Welcome to TechExpo!!!");
        primaryStage.setScene(scene);
        primaryStage.show();
        scene.getStylesheets().add(getClass().getResource("JavaCoursework.css").toExternalForm());
    }

    public static void main(String[] args) {
        launch();
    } // Launch the JavaFX application
}
```

b) Description

The ‘TechExpoApplication’ class extends the ‘Application’ class and serves as the main entry point for the TechExpo application. It overrides the ‘start’ method to initialize the primary stage by loading the ‘WelcomeScreen.fxml’ file, which defines the main layout for the application. A new ‘Scene’ is created with this layout, set to dimensions of 980x700 pixels. The application icon is loaded from the resource path ‘/projectIcon.png’ and assigned to the stage. The window title is set to "Welcome to TechExpo!!!", and the stage is made visible by calling ‘show()’. The ‘scene’ is also styled by adding a CSS stylesheet located at ‘JavaCoursework.css’. The ‘main’ method invokes ‘launch()’, which triggers the JavaFX application lifecycle and launches the application.

3. WelcomeScreen

a) Code

```java
package JavaCoursework;

import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.Node;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Alert;
import javafx.scene.control.TextInputDialog;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.stage.Stage;
import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
import java.util.Optional;
import java.util.ResourceBundle;

public class WelcomeScreen implements Initializable {
```

```java
    private static List<Project> projectsList = new ArrayList<>();
    private final String FILENAME = "project_details.txt";
    public ImageView LoveIcon;

    @Override
    public void initialize(URL location, ResourceBundle resources) {
        projectsList = MainEventLayout.readProjectsFromFile(FILENAME);
        loadImage(LoveIcon, "THANKYOU.png");
    }

    private void loadImage(ImageView imageView, String imagePath) {
        // Use the path relative to the resources folder
        Image image = new Image(getClass().getResourceAsStream("/" + imagePath));
        if (image != null) {
            imageView.setImage(image);
        } else {
            System.err.println("Image resource not found: " + imagePath);
        }
    }

    @FXML
    public void onClickSarah(ActionEvent event) {
        // Define the correct password
        String correctPassword = "sarah123";

        // Create a TextInputDialog for password input
        TextInputDialog dialog = new TextInputDialog();
        dialog.setTitle("Password Required");
        dialog.setHeaderText("Enter Password for Sarah");
        dialog.setContentText("Password:");

        // Show the dialog and capture the user input
        Optional<String> result = dialog.showAndWait();

        // Check if the password is correct
        if (result.isPresent() && result.get().equals(correctPassword)) {
            try {
                // Load MainEventsLayout.fxml
                FXMLLoader loader = new FXMLLoader(getClass().getResource("MainEventsLayout.fxml"));
                Parent root = loader.load();

                // Get the Stage from the current scene and set new scene
                Stage stage = (Stage) ((Node) event.getSource()).getScene().getWindow();
                stage.setScene(new Scene(root, 980, 700));
               root.getStylesheets().add(getClass().getResource("JavaCoursework.css").toExternalForm());
                stage.show();
            } catch (IOException e) {
                e.printStackTrace();
            }
        } else {
            // Show an error alert if the password is incorrect
            Alert alert = new Alert(Alert.AlertType.ERROR);
            alert.setTitle("Incorrect Password");
            alert.setHeaderText("Access Denied");
            alert.setContentText("The password you entered is incorrect.");
            alert.showAndWait();
        }
    }


    @FXML
    public void onClickParticipant(ActionEvent event) {
        try {
            loadFXML("AddProjectDetails.fxml", "Hello Participant!! You can add your project details",
event);
        } catch (IOException e) {
            showAlert("Error", "Failed to load AddProjectDetails.fxml", Alert.AlertType.ERROR);
            e.printStackTrace();
        }
    }

    private void loadFXML(String fxmlFileName, String alertMessage, ActionEvent event) throws
IOException {
        // Load the FXML file
        FXMLLoader loader = new FXMLLoader(getClass().getResource(fxmlFileName));
        Parent fxml = loader.load();
```

```
        // Get the Stage from the current scene and set new scene
        Stage stage = (Stage) ((Node) event.getSource()).getScene().getWindow();
        stage.setScene(new Scene(fxml));
        fxml.getStylesheets().add(getClass().getResource("JavaCoursework.css").toExternalForm());

        // Show the alert message after loading the FXML
        Alert alert = new Alert(Alert.AlertType.INFORMATION);
        alert.setTitle("Information");
        alert.setHeaderText(null);
        alert.setContentText(alertMessage);
        alert.showAndWait();
    }

    private void showAlert(String title, String content, Alert.AlertType alertType) {
        Alert alert = new Alert(alertType);
        alert.setTitle(title);
        alert.setHeaderText(null);
        alert.setContentText(content);
        alert.showAndWait();
    }

    public static List<Project> getProjectsFromFile(String filePath) {
        return projectsList;
    }
}
```

b) Description

The ‘WelcomeScreen’ class implements the ‘Initializable’ interface for a JavaFX application. It initializes by loading a list of projects from a file and an image into an ‘ImageView’. The ‘onClickSarah’ method checks for a correct password using a ‘TextInputDialog’ before loading the ‘MainEventsLayout.fxml’ file. If the password is incorrect, it shows an error alert. The ‘onClickParticipant’ method loads the ‘AddProjectDetails.fxml’ file and shows an informational alert. Helper methods include ‘loadFXML’ for loading FXML files and showing alerts, and ‘showAlert’ for displaying alerts. The ‘scene’ is also styled by adding a CSS stylesheet located at ‘JavaCoursework.css’.The static method ‘getProjectsFromFile’ returns the list of projects.

4. MainEventLayout

a) Code

```
package JavaCoursework;

import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.Parent;
import javafx.scene.control.Alert;
import javafx.scene.control.Button;
import javafx.scene.control.ButtonBar;
import javafx.scene.control.ButtonType;
import javafx.scene.layout.StackPane;
import javafx.stage.Stage;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
import java.util.ResourceBundle;
```

```java
public class MainEventLayout implements Initializable  {

    @FXML
    public Button button;

    @FXML
    public StackPane contentArea;

    protected void loadFXML(String fxmlFileName, String alertMessage) throws IOException {
        Parent fxml = FXMLLoader.load(getClass().getResource(fxmlFileName));
        // Clear the current content
        contentArea.getChildren().clear();
        // Set the loaded FXML as the new content
        contentArea.getChildren().setAll(fxml);

        // Show the alert message after loading the FXML
        Alert alert = new Alert(Alert.AlertType.INFORMATION);
        alert.setTitle("Information");
        alert.setHeaderText(null);
        alert.setContentText(alertMessage);
        alert.showAndWait();
    }

    @Override
    public void initialize(URL location, ResourceBundle resources) {
        try {
            //projectsList = WelcomeScreen.getProjectsFromFile("project_details.txt");
            // Loading the initial scene
            loadFXML("home.fxml", "Hello Sarah!!! Welcome to TechExpo!!!");
        } catch (IOException ex) {
            // Print stack trace for debugging if the initial scene cannot be loaded
            ex.printStackTrace();
        }
    }

    @FXML
    public void Home() throws IOException {
        loadFXML("Home.fxml", "Welcome to our Home page.");
    }

    @FXML
    public void APD() throws IOException {
        loadFXML("AddProjectDetails.fxml", "This is for adding project details. " +
                "You can add your project details here.");
    }

    @FXML
    public void DPD() throws IOException {
        loadFXML("DeleteProjectDetails.fxml", "This is for deleting project details. " +
                "You can delete your project details here.");
    }

    @FXML
    public void UPD() throws IOException {
        loadFXML("UpdateProjectDetails.fxml", "This is for updating project details. " +
                "You can update your project details here.");
    }

    @FXML
    public void VPD() throws IOException {
        // Check if no projects were loaded
        loadFXML("ViewProjectDetails.fxml",
                "You can view your project details in ascending order based on proID of all projects.");
    }

    @FXML
    public void Final() throws IOException {
        loadFXML("Start.fxml", "This will navigate you to the finals of the project. " +
                "Click that button for navigation.");
    }

    @FXML
    public void Exit() {
        Alert alert = new Alert(Alert.AlertType.CONFIRMATION);
        alert.setTitle("Exit Confirmation");
        alert.setHeaderText("Are you sure you want to exit?");
```

```java
        alert.setContentText("Press OK to exit or Cancel to stay.");

        // Customizing the buttons in the alert dialog
        ButtonType okButton = new ButtonType("OK");
        ButtonType cancelButton = new ButtonType("Cancel", ButtonBar.ButtonData.CANCEL_CLOSE);
        alert.getButtonTypes().setAll(okButton, cancelButton);

        // Handling user's choice
        alert.showAndWait().ifPresent(response -> {
            if (response == okButton) {
                // Close the application
                Stage stage = (Stage) contentArea.getScene().getWindow();
                stage.close();
            }
        });
    }

    public static List<Project> readProjectsFromFile(String filePath) {
        List<Project> loadedProjects = new ArrayList<>();
        try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {
            String line;
            Project currentProject = null;
            while ((line = reader.readLine()) != null) {
                if (line.startsWith("*********** ")) {
                    continue; //skip it
                } else if (line.startsWith("---Project Details---")) {
                    // New project, create a new Project object
                    currentProject = new Project(0, "", "",
                            new ArrayList<>(), "", "", "");
                } else if (line.startsWith("Project ID: ")) {
                    if (currentProject != null) {
                        // Set the project ID of the current project
                        currentProject.setProjectID(Integer.parseInt(line.substring(11).trim()));
                    }
                } else if (line.startsWith("Project Name: ")) {
                    if (currentProject != null) {
                        currentProject.setProjectName(line.substring(13).trim());
                    }
                } else if (line.startsWith("Category: ")) {
                    if (currentProject != null) {
                        currentProject.setCategory(line.substring(9).trim());
                    }
                } else if (line.startsWith("Team Members: ")) {
                    if (currentProject != null) {
                        String[] teamMembers = line.substring(14).trim().split(",\\s*");
                        currentProject.setTeamMembers(List.of(teamMembers));
                    }
                } else if (line.startsWith("Brief Description: ")) {
                    if (currentProject != null) {
                        currentProject.setDescription(line.substring(18).trim());
                    }
                } else if (line.startsWith("Country: ")) {
                    if (currentProject != null) {
                        currentProject.setCountry(line.substring(9).trim());
                    }
                } else if (line.startsWith("Logo URL: ")) {
                    if (currentProject != null) {
                        currentProject.setTeamLogoUrl(line.substring(10).trim());
                        // Add the project to the list
                        loadedProjects.add(currentProject);
                        currentProject = null; // Reset currentProject to prepare for next project
                    }
                }
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
        return loadedProjects;
    }
}
```

b) Description

The ‘MainEventLayout’ class, which implements the ‘Initializable’ interface, manages the user interface for the TechExpo application using FXML annotations to link UI components such as ‘Button’ and ‘StackPane’. The ‘initialize’ method sets up the initial scene by loading ‘home.fxml’ and displaying a welcome message. The ‘loadFXML’ method dynamically loads specified FXML files into the ‘contentArea’ and shows an informational alert message. This method is utilized by other methods to load different views: ‘Home’ for loading ‘Home.fxml’ with a home page message, ‘APD’ for adding project details, ‘DPD’ for deleting project details, ‘UPD’ for updating project details, ‘VPD’ for viewing project details, and ‘Final’ for navigating to the finals using ‘Start.fxml’. The ‘Exit’ method prompts a confirmation alert to exit the application, closing the stage if the user confirms their choice. The class also includes a ‘readProjectsFromFile’ method to parse project details from a text file, creating ‘Project’ objects and adding them to a list, which facilitates access to project data across the application. This class ensures smooth transitions between different views and manages project data effectively.

5. HOME

a) Code

```
package JavaCoursework;

import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Alert;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.stage.Stage;

import java.io.IOException;

public class Home {
    @FXML
    public ImageView IconDPD;
    @FXML
    public ImageView IconUPD;
    @FXML
    public ImageView IconVPD;
    @FXML
    public ImageView IconNEXT;
    @FXML
    public ImageView IconEXIT;
    @FXML
    private ImageView IconAPD;

    @FXML
    public void initialize() {
        // Load images for all ImageViews
        loadImage(IconAPD, "APD.png");
        loadImage(IconDPD, "DPD.png");
        loadImage(IconUPD, "UPD.png");
        loadImage(IconVPD, "VPD.png");
        loadImage(IconNEXT, "NEXT.png");
        loadImage(IconEXIT, "EXIT.png");
    }

    private void loadImage(ImageView imageView, String imagePath) {
```

```
        // Use the path relative to the resources folder
        Image image = new Image(getClass().getResourceAsStream("/" + imagePath));
        if (image != null) {
            imageView.setImage(image);
        } else {
            System.err.println("Image resource not found: " + imagePath);
        }
    }

    public void onClickBackHome(ActionEvent event) {
        try {
            // Load AddProjectDetails.fxml
            FXMLLoader loader = new FXMLLoader(getClass().getResource("WelcomeScreen.fxml"));
            Parent root = loader.load();

            // Get the Stage from the current scene and set new scene
            Stage stage = (Stage) ((javafx.scene.control.Button)
event.getSource()).getScene().getWindow();
            stage.setScene(new Scene(root));
            root.getStylesheets().add(getClass().getResource("JavaCoursework.css").toExternalForm());
            stage.show();
        } catch (IOException e) {
            showAlert("Error", "Failed to load WelcomeScreen.fxml", Alert.AlertType.ERROR);
            e.printStackTrace();
        }
    }

    private void showAlert(String title, String content, Alert.AlertType alertType) {
        Alert alert = new Alert(alertType);
        alert.setTitle(title);
        alert.setHeaderText(null);
        alert.setContentText(content);
        alert.showAndWait();
    }
}
```

b) Description

The ‘Home’ class manages the display of various icons using ‘ImageView’ components. During initialization, the ‘initialize()’ method loads images for each ‘ImageView’ by calling the ‘loadImage()’ method. This method takes an ‘ImageView’ and an image file name as parameters, loading the image from the resources folder and setting it to the corresponding ‘ImageView’. If the image is not found, an error message is printed to the console. The ‘onClickBackHome()’ method handles the event of clicking the back button, loading the ‘WelcomeScreen.fxml’ file and setting it as the new scene. If the FXML file cannot be loaded, an error alert is displayed. The ‘scene’ is also styled by adding a CSS stylesheet located at ‘JavaCoursework.css’. This setup ensures that all icons are properly displayed when the application starts and provides a mechanism for navigating back to the welcome screen.

6. AddValidator

a) Code

```
package JavaCoursework;

import javafx.collections.ObservableList;
import javafx.scene.control.Alert;
import javafx.scene.image.Image;

import java.io.*;
```

```java
import java.util.List;

public class AddValidator {

    // Method to check if a given integer is a positive integer
    public static boolean isPositiveInteger(int value) {
        return value > 0;
    }

    public static String validateProjectID(int projectID, List<Project> projectsList) {
        if (!isPositiveInteger(projectID)) {
            return "Warning: The project ID must be a positive integer.";
        }
        if (isDuplicateProjectID(projectID, projectsList)) {
            return "Warning: The project ID already exists.";
        }
        return "Success: The project ID is available.";
    }

    static boolean isDuplicateProjectID(int projectID, List<Project> projectsList) {
        for (Project project : projectsList) {
            if (project.getProjectID() == projectID) {
                return true;
            }
        }
        return false;
    }

    public static String validateAndFormatProjectName(String projectNameText) {
        String projectName = projectNameText.trim();
        if (projectName.isEmpty()) {
            return null;
        }
        return Project.capitalizeFirstLetter(projectName);
    }

    public static String validateCategory(String category) {
        if (category.equals("AI") || category.equals("ML") || category.equals("RT")) {
            return category;
        }
        return null;
    }

    public static List<String> validateAndFormatTeamMembers(String membersText) {
        List<String> teamMembers = List.of(membersText.split(","));
        if (teamMembers.size() == 3) {
            return Project.formatTeamMembers(membersText.trim());
        }
        return null;
    }

    public static String validateAndFormatDescription(String descriptionText) {
        String description = descriptionText.trim();
        if (description.isEmpty()) {
            return null;
        }
        return Project.formatDescription(description);
    }

    public static String validateAndFormatCountry(String countryText) {
        String country = countryText.trim();
        if (country.isEmpty()) {
            return null;
        }
        return Project.capitalizeFirstLetter(country);
    }

    public static String validateAndFormatLogoUrl(String logoUrl, List<Project> projectsList) {
        if (isUrlEmpty(logoUrl) || isDuplicateLogoUrl(logoUrl, projectsList) ||
!isValidLogoUrl(logoUrl)) {
            return null;
        }
        return logoUrl;
    }

    static boolean isUrlEmpty(String url) {
        if (url.isEmpty()) {
```

```
            return true;
        }
        return false;
    }

    static boolean isDuplicateLogoUrl(String logoUrl, List<Project> projectsList) {
        for (Project project : projectsList) {
            if (project.getTeamLogoUrl().equals(logoUrl)) {
                return true;
            }
        }
        return false;
    }

    static boolean isValidLogoUrl(String logoUrl) {
        try {
            new Image(logoUrl);
            return true;
        } catch (IllegalArgumentException e) {
            return false;
        }
    }

    static void saveProjectsToFile(String filePath, List<Project> projectsList) {
        try (BufferedWriter writer = new BufferedWriter(new FileWriter(filePath))) {
            writeProjects(writer, "AI Projects", projectsList, "AI");
            writeProjects(writer, "ML Projects", projectsList, "ML");
            writeProjects(writer, "RT Projects", projectsList, "RT");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private static void writeProjects(BufferedWriter writer, String categoryHeader, List<Project>
projects, String category) throws IOException {
        writer.write("\n");
        writer.write("*********** " + categoryHeader + " ***********\n");
        writer.write("\n");
        for (Project project : projects) {
            if (project.getCategory().equals(category)) {
                writer.write("---Project Details---\n");
                writer.write("\n");
                writer.write("Project ID: " + project.getProjectID() + "\n");
                writer.write("Project Name: " + project.getProjectName() + "\n");
                writer.write("Category: " + project.getCategory() + "\n");
                writer.write("Team Members: " + project.getTeamMembersAsString() + "\n");
                writer.write("Brief Description: " + project.getDescription() + "\n");
                writer.write("Country: " + project.getCountry() + "\n");
                writer.write("Logo URL: " + project.getTeamLogoUrl() + "\n");
                writer.write("\n");
            }
        }
    }


    //For Delete Part
    // Method to validate if the project ID exists in the list
    public static boolean isProjectIDValid(List<Project> projectsList, int delProID) {
        // Check if the provided list is null
        if (projectsList == null) {
            throw new IllegalArgumentException("The list of projects cannot be null.");
        }
        // Check if the project ID is a positive integer
        if (delProID <= 0) {
            return false;
        }
        // Iterate through the list of projects to find a match
        for (Project project : projectsList) {
            // Check if the current project's ID matches the given ID
            if (project.getProjectID() == delProID) {
                return true; // Project ID found in the list
            }
        }
        // Project ID not found in the list
        return false;
    }
```

```java
    // Method to validate project list
    public static boolean validateProjectList(List<Project> projects) {
        return projects != null && !projects.isEmpty();
    }

    // Bubble sort method to sort projects by projectID in ascending order
    static void bubbleSortProjects(List<Project> projects) {
        int n = projects.size();
        for (int i = 0; i < n - 1; i++) {
            for (int j = 0; j < n - i - 1; j++) {
                Project project1 = projects.get(j);
                Project project2 = projects.get(j + 1);

                if (project1.getProjectID() > project2.getProjectID()) {
                    // Swap projects
                    projects.set(j, project2);
                    projects.set(j + 1, project1);
                }
            }
        }
    }

    // Method to validate category projects and return a boolean
    public static boolean validateCategoryProjects(List<Project> categoryProjects, String category) {
        // Check if the categoryProjects list is null
        if (categoryProjects == null) {
            return false; // Indicate that validation failed
        }

        // Check if the categoryProjects list is empty
        if (categoryProjects.isEmpty()) {
            return false; // Indicate that validation failed
        }

        // If all validations pass
        return true; // Indicate that validation passed
    }

    // Additional method to generate warning message
    public static String getValidationMessage(List<Project> categoryProjects, String category) {
        // Check if the categoryProjects list is null
        if (categoryProjects == null) {
            return "Warning: The category list for " + category + " is null.";
        }

        // Check if the categoryProjects list is empty
        if (categoryProjects.isEmpty()) {
            return "Warning: No projects found for category: " + category;
        }

        // If all validations pass
        return null;
    }


    // Method to validate project ratings
    public static boolean isValidRating(String rating) {
        return rating != null && rating.matches("[*]{1,5}");
    }

    // Method to sort projects by judgesPoints using bubble sort
    public static void bubbleSortProjects(ObservableList<Project> projects) {
        int n = projects.size();
        Project temp;
        for (int i = 0; i < n - 1; i++) {
            for (int j = 0; j < n - i - 1; j++) {
                if (projects.get(j).getJudgesPoints() < projects.get(j + 1).getJudgesPoints()) {
                    // Swap projects if current is less than next
                    temp = projects.get(j);
                    projects.set(j, projects.get(j + 1));
                    projects.set(j + 1, temp);
                }
            }
        }
    }

    // Method to validate that the selected projects are not null and the size is at least 3
```

```
    public static boolean validateSelectedProjects(ObservableList<Project> selectedProjects) {
        return selectedProjects != null && selectedProjects.size() >= 3;
    }

    // Method to validate that the selected projects are not null and the size is at least 3 for
AwardWinningProjects
    public static boolean validateAwardWinningProjects(ObservableList<Project> selectedProjects) {
        return selectedProjects != null && selectedProjects.size() >= 3;
    }
}
```

b) Description

The ‘AddValidator’ class is designed for validating and formatting project details within a JavaFX application. It includes several methods to ensure data integrity and consistency. The ‘isPositiveInteger’ method checks if a given integer is positive, while ‘validateProjectID’ ensures the project ID is positive and not a duplicate using ‘isDuplicateProjectID’. The ‘validateAndFormatProjectName’ method trims, validates, and formats the project name, capitalizing the first letter. The ‘validateCategory’ method verifies that the category is one of "AI", "ML", or "RT". ‘validateAndFormatTeamMembers’ ensures the team members string is properly formatted and contains exactly three members. The ‘validateAndFormatDescription’ and ‘validateAndFormatCountry’ methods trim, validate, and format the project description and country name, respectively. The ‘validateAndFormatLogoUrl’ method checks that the logo URL is valid, not empty, not a duplicate, and loadable as an image. Methods such as ‘isUrlEmpty’, ‘isDuplicateLogoUrl’, and ‘isValidLogoUrl’ assist in these validations. The ‘saveProjectsToFile’ method saves project details to a file, categorized by project type (AI, ML, RT), with the helper method ‘writeProjects’ handling the actual writing process. For project deletion, ‘isProjectIDValid’ checks if a project ID exists in the list, while ‘validateProjectList’ ensures the project list is not null or empty. Sorting is handled by ‘bubbleSortProjects’, which sorts projects by project ID using bubble sort. The ‘validateCategoryProjects’ and ‘getValidationMessage’ methods validate and generate warnings for category-specific project lists. Additionally, ‘isValidRating’ ensures project ratings follow a specific star pattern, and another ‘bubbleSortProjects’ method sorts projects by judges' points in descending order. The ‘validateSelectedProjects’ and ‘validateAwardWinningProjects’ methods ensure that selected project lists are valid and contain at least three projects. Lastly, the ‘showAlert’ method displays alert messages with a given type, title, and content.

7. AddProjectDetails

a) Code

```
package JavaCoursework;

import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
```

```java
import javafx.stage.FileChooser;
import javafx.stage.Stage;

import java.io.*;
import java.net.URL;
import java.util.List;
import java.util.ResourceBundle;

public class AddProjectDetails implements Initializable {

    @FXML
    private TextField proID;

    @FXML
    private TextField proName;

    @FXML
    private ChoiceBox<String> proCategory;

    @FXML
    private TextField proMembers;

    @FXML
    private TextField proCountry;

    @FXML
    private TextArea proDescription;

    @FXML
    private TextArea logoTextView;

    @FXML
    private ImageView logoImageView;

    private static final String FILENAME = "project_details.txt";
    private static List<Project> projectsList = WelcomeScreen.getProjectsFromFile(FILENAME);

    @Override
    public void initialize(URL location, ResourceBundle resources) {
        proCategory.getItems().addAll("AI", "ML", "RT");
    }

    @FXML
    private void onClickCheck() {
        try {
            int projectID = Integer.parseInt(proID.getText().trim());
            String result = AddValidator.validateProjectID(projectID, projectsList);
            if (result.contains("Error") || result.contains("Warning")) {
                showAlert(Alert.AlertType.ERROR, "Validation", result);
            } else {
                showAlert(Alert.AlertType.INFORMATION, "Validation", result);
            }
        } catch (NumberFormatException e) {
            showAlert(Alert.AlertType.ERROR, "Validation Error", "Invalid project ID. Please enter a
valid integer.");
        }
    }

    @FXML
    private void onClickAdd() {
        String result = addProjectDetails();
        if (result.contains("Error") || result.contains("Warning")) {
            showAlert(Alert.AlertType.ERROR, "Validation", result);
        } else {
            showAlert(Alert.AlertType.INFORMATION, "Validation", result);
        }
    }

    private String addProjectDetails() {
        try {
            // Validate Project ID
            int projectID = parseProjectID();
            if (projectID == -1) {
                return "Error: Invalid project ID.";
            } else if (AddValidator.isDuplicateProjectID(projectID, projectsList)) {
                return "Warning: The project ID already exists.";
            }
```

```
            // Validate Project Name
            String projectName = AddValidator.validateAndFormatProjectName(proName.getText().trim());
            if (projectName == null) {
                return "Error: Project name cannot be empty.";
            }

            // Validate Category
            String category = proCategory.getValue();
            if (category == null || category.trim().isEmpty()) {
                return "Error: Project category cannot be empty.";
            } else {
                category = AddValidator.validateCategory(category.trim());
                if (category == null) {
                    return "Error: Invalid project category.";
                }
            }

            // Validate Country
            String country = AddValidator.validateAndFormatCountry(proCountry.getText().trim());
            if (country == null) {
                return "Error: Country cannot be empty.";
            }

            // Validate Team Members
            List<String> teamMembers =
AddValidator.validateAndFormatTeamMembers(proMembers.getText().trim());
            if (teamMembers == null) {
                return "Error: Please enter exactly three team members separated by commas.";
            }

            // Validate Description
            String description =
AddValidator.validateAndFormatDescription(proDescription.getText().trim());
            if (description == null) {
                return "Error: Project description cannot be empty.";
            }

            // Validate Logo URL
            String logoUrl = AddValidator.validateAndFormatLogoUrl(logoTextView.getText().trim(),
projectsList);
            if (logoUrl == null) {
                return "Error: Invalid logo URL.";
            }

            // If all validations pass, add the project
            Project newProject = new Project(projectID, projectName, category, teamMembers, description,
country, logoUrl);
            projectsList.add(newProject);
            AddValidator.saveProjectsToFile(FILENAME, projectsList);
            clearFields();
            return "Success: Project details have been added successfully.";

        } catch (NumberFormatException e) {
            return "Error: Invalid project ID. Please enter a valid integer.";
        }
    }

    private int parseProjectID() {
        try {
            int projectID = Integer.parseInt(proID.getText().trim());
            if (projectID > 0) {
                return projectID;
            }
        } catch (NumberFormatException e) {
            return -1;
        }
        return -1;
    }

    @FXML
    private void onClickBrowse() {
        FileChooser fileChooser = new FileChooser();
        fileChooser.setTitle("Choose Logo Image File");
        fileChooser.getExtensionFilters().addAll(
                new FileChooser.ExtensionFilter("Image Files", "*.png", "*.jpg", "*.gif"),
                new FileChooser.ExtensionFilter("All Files", "*.*")
```

```
        );
        File selectedFile = fileChooser.showOpenDialog(null);
        if (selectedFile != null) {
            String imagePath = selectedFile.toURI().toString();
            if (!AddValidator.isDuplicateLogoUrl(imagePath, projectsList)) {
                logoTextView.setText(imagePath);
                logoImageView.setImage(new Image(imagePath));
            } else {
                showAlert(Alert.AlertType.ERROR, "Error", "The logo URL must be unique for each
project.");
            }
        }
    }

    static void showAlert(Alert.AlertType alertType, String title, String message) {
        Alert alert = new Alert(alertType);
        alert.setTitle(title);
        alert.setHeaderText(null);
        alert.setContentText(message);
        alert.showAndWait();
    }

    private void clearFields() {
        proID.clear();
        proName.clear();
        proCategory.setValue(null);
        proMembers.clear();
        proDescription.clear();
        proCountry.clear();
        logoTextView.clear();
        logoImageView.setImage(null);
    }

    public void onClickBack(ActionEvent event) { try {
        // Load AddProjectDetails.fxml
        FXMLLoader loader = new FXMLLoader(getClass().getResource("WelcomeScreen.fxml"));
        Parent root = loader.load();
        // Get the Stage from the current scene and set new scene
        Stage stage = (Stage) ((javafx.scene.control.Button) event.getSource()).getScene().getWindow();
        stage.setScene(new Scene(root));
        root.getStylesheets().add(getClass().getResource("JavaCoursework.css").toExternalForm());
        stage.show();
    } catch (IOException e) {
        e.printStackTrace();
    }
    }
}
```

b) Description

The ‘AddProjectDetails’ class manages the interface and logic for adding new project details. It includes several UI components such as ‘TextField’, ‘ChoiceBox’, ‘TextArea’, and ‘ImageView’, which are initialized and populated in the ‘initialize’ method. This method sets the categories for the ‘ChoiceBox’ and loads existing projects from a file into the ‘projectsList’. The ‘onClickCheck’ method validates the project ID entered by the user, ensuring it is unique and a positive integer. If the validation fails, it displays an error alert using the ‘showAlert’ method. The ‘onClickAdd’ method validates and processes user input for various project details including the project ID, name, category, team members, description, country, and logo URL. It uses helper methods from the ‘AddValidator’ class to ensure the inputs are correctly formatted and valid. If all validations pass, a new ‘Project’ object is created and added to the ‘projectsList’, which is then saved to the file. The ‘onClickBrowse’ method allows users to select an image file for the project logo, updating the ‘logoTextView’ and ‘logoImageView’ if the selected image path is valid and unique. The ‘parseProjectID’ method extracts and validates the project ID from the ‘TextField’. The

‘showAlert’ method is used throughout the class to display different types of alerts based on user interactions. The ‘clearFields’ method resets all input fields after successfully adding a project, ensuring the form is ready for new input. Additionally, the ‘onClickBack’ method handles navigation back to the welcome screen by loading the ‘WelcomeScreen.fxml’ file and setting it as the new scene. The ‘scene’ is also styled by adding a CSS stylesheet located at ‘JavaCoursework.css’. If the FXML file cannot be loaded, an error alert is displayed.

8. DeleteProjectDetails

a) Code

```
package JavaCoursework;

import javafx.fxml.FXML;
import javafx.scene.control.Alert;
import javafx.scene.control.TextField;

import java.io.IOException;
import java.util.List;

public class DeleteProjectDetails {

    @FXML
    TextField proID;

    public static String FILENAME = "project_details.txt";
    private static List<Project> projectsList = WelcomeScreen.getProjectsFromFile(FILENAME);

    @FXML
    public void deleteProject() {
        try {
            int delProID = parseProjectID();

            // Validate the project ID before attempting to delete
            if (AddValidator.isPositiveInteger(delProID)) {
                if (AddValidator.isProjectIDValid(projectsList, delProID)) {
                    boolean removed = removeProjectFromList(projectsList, delProID);
                    if (removed) {
                        updateProjectFile(projectsList);
                        showAlert(Alert.AlertType.INFORMATION, "Project Deleted",
                                "Project with ID " + delProID + " has been deleted.");
                    } else {
                        showAlert(Alert.AlertType.WARNING, "Deletion Error",
                                "Error occurred while deleting the project with ID " + delProID + ".");
                    }
                } else {
                    showAlert(Alert.AlertType.WARNING, "No Project Found",
                            "The project ID you entered was not found.");
                }
            } else {
                showAlert(Alert.AlertType.WARNING, "Invalid Project ID",
                        "The project ID must be a positive integer.");
            }

        } catch (NumberFormatException e) {
            showAlert(Alert.AlertType.WARNING, "Invalid Input",
                    "Please enter a valid integer for project ID.");
        } catch (IOException e) {
            showAlert(Alert.AlertType.ERROR, "File Error",
                    "There was an error accessing the project details file.");
        } finally {
            proID.clear();
        }
    }

    // Parse the project ID from the text field
    int parseProjectID() {
```

```
        return Integer.parseInt(proID.getText().trim());
    }

    // Remove the project with the given ID from the list
    boolean removeProjectFromList(List<Project> projectsList, int delProID) {
        for (Project project : projectsList) {
            if (project.getProjectID() == delProID) {
                projectsList.remove(project);
                return true;
            }
        }
        return false; // Indicate that the project was not found
    }

    // Update the project details file
    void updateProjectFile(List<Project> projectsList) throws IOException {
        AddValidator.saveProjectsToFile(FILENAME, projectsList);
    }

    private void showAlert(Alert.AlertType alertType, String title, String message) {
        Alert alert = new Alert(alertType);
        alert.setTitle(title);
        alert.setHeaderText(null); // No header text
        alert.setContentText(message);
        alert.showAndWait(); // Wait for the user to close the alert
    }

    @FXML
    public void onclickDelete() {
        deleteProject(); // Call deleteProject() method when the button is clicked
    }
}
```

b) Description

The ‘DeleteProjectDetails’ class manages the user interface and logic for deleting project details. . It retrieves the list of projects from the "project_details.txt" file using the ‘WelcomeScreen. getProjectsFromFile’ method. When the ‘onclickDelete’ method is called, it invokes the ‘deleteProject’ method. This method begins by parsing the project ID from the ‘proID’ text field using the ‘parseProjectID’ method. The ‘deleteProject’ method checks if the project ID is a positive integer using the ‘AddValidator.isPositiveInteger’ method and if it exists in the list using the ‘AddValidator.isProjectIDValid’ method. If the project ID is valid and exists, it attempts to remove the project from the list with the ‘removeProject FromList’ method. If the project is successfully removed, the updated list is saved back to the file using the ‘updateProjectFile’ method, and an information alert is displayed to confirm the deletion. If the project is not found or an error occurs during the deletion, a warning alert is shown. If the input is not a valid integer, a warning alert is displayed. The ‘showAlert’ method is used to display these alerts. Finally, the ‘proID’ text field is cleared after attempting to delete a project.

9. UpdateProjectDetails

a) Code

```
package JavaCoursework;

import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.*;
```

```
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.stage.FileChooser;

import java.io.File;
import java.net.URL;
import java.util.List;
import java.util.ResourceBundle;

public class UpdateProjectDetails implements Initializable {

    @FXML
    public TextArea updateLogo;
    @FXML
    public TextArea updateMembers;
    @FXML
    public TextField updateCountry;
    @FXML
    public TextField updateCategory;
    @FXML
    public TextField updateName;
    @FXML
    public Label alertUpdate;

    @FXML
    TextField proID;

    @FXML
    TextArea updateDetails;

    @FXML
    ImageView logoImageView;

    private static final String FILENAME = "project_details.txt";
    private static List<Project> projectsList = WelcomeScreen.getProjectsFromFile(FILENAME);

    @Override
    public void initialize(URL location, ResourceBundle resources) {}

    @FXML
    public void onClickCheck() {
        // Parse project ID from the text field
        int projectID = parseProjectID();
        if (projectID == -1) {clearFields();
        return;}
        // Find the project by ID in the projects list
        Project project = findProjectByID(projectID, projectsList);
        if (project == null) {
            // Show error if project not found
            showAlert(Alert.AlertType.ERROR, "Error", "Project with ID " + projectID + " is not
found.");
            clearFields();
            return;
        } else {
            showAlert(Alert.AlertType.INFORMATION, "Success", "Project found with ID " + projectID +
".");
            alertUpdate.setText("Now, you can update the fields you wish to update!!!");
            updateFieldsWithProjectDetails(project);
        }
    }

    @FXML
    public void onClickUpdate() {
        int projectID = parseProjectID();
        if (projectID == -1) return;

        Project projectToUpdate = findProjectByID(projectID, projectsList);
        if (projectToUpdate == null) {
            showAlert(Alert.AlertType.ERROR, "Error", "Project with ID " + projectID + " is not
found.");
            clearFields();
            return;
        }

        // Validate and update project details
        String projectName = AddValidator.validateAndFormatProjectName(updateName.getText());
        if (projectName == null) {
```

```
            showAlert(Alert.AlertType.ERROR, "Error", "Project name cannot be empty.");
            return;
        }
        projectToUpdate.setProjectName(projectName);

        String category = AddValidator.validateCategory(updateCategory.getText().trim().toUpperCase());
        if (category == null) {
            showAlert(Alert.AlertType.ERROR, "Error", "Invalid category! Please enter AI, ML, or RT.");
            return;
        }
        projectToUpdate.setCategory(category);

        String country = AddValidator.validateAndFormatCountry(updateCountry.getText());
        if (country == null) {
            showAlert(Alert.AlertType.ERROR, "Error", "Country cannot be empty.");
            return;
        }
        projectToUpdate.setCountry(country);

        String description = AddValidator.validateAndFormatDescription(updateDetails.getText());
        if (description == null) {
            showAlert(Alert.AlertType.ERROR, "Error", "Description cannot be empty.");
            return;
        }
        projectToUpdate.setDescription(description);

        List<String> teamMembers = AddValidator.validateAndFormatTeamMembers(updateMembers.getText());
        if (teamMembers == null) {
            showAlert(Alert.AlertType.ERROR, "Error", "Please enter exactly three team members separated
by commas.");
            return;
        }
        projectToUpdate.setTeamMembers(teamMembers);

        // Check logo URL uniqueness only if it is being updated
        if (!AddValidator.isUrlEmpty(updateLogo.getText().trim())) {
            String newLogoUrl = updateLogo.getText().trim();
            for (Project p : projectsList) {
                if (p.getProjectID() != projectToUpdate.getProjectID() &&
p.getTeamLogoUrl().equals(newLogoUrl)) {
                    showAlert(Alert.AlertType.ERROR, "Error", "The logo URL must be unique for each
project.");
                    return;
                }
            }
            projectToUpdate.setTeamLogoUrl(newLogoUrl);
            Image image = new Image(newLogoUrl);
            logoImageView.setImage(image);
        } else {
            showAlert(Alert.AlertType.ERROR, "Error", "Invalid logo URL.");
            return;
        }

        // Save the updated projects list to the file
        saveUpdatedProjectsList();
        showAlert(Alert.AlertType.INFORMATION, "Success", "Project details have been updated
successfully.");
        clearFields();
    }

    @FXML
    public void onClickBrowse() {
        // Open a file chooser to select an image file for the logo
        FileChooser fileChooser = new FileChooser();
        fileChooser.setTitle("Choose Logo Image File");
        fileChooser.getExtensionFilters().addAll(
                new FileChooser.ExtensionFilter("Image Files", "*.png", "*.jpg", "*.gif"),
                new FileChooser.ExtensionFilter("All Files", "*.*")
        );
        File selectedFile = fileChooser.showOpenDialog(null);
        if (selectedFile != null) {
            String imagePath = selectedFile.toURI().toString(); // Convert file path to URI
            // Check if the selected image path is already used by another project
            if (AddValidator.isDuplicateLogoUrl(imagePath, projectsList)) {
                showAlert(Alert.AlertType.ERROR, "Error", "The logo URL must be unique for each
project.");
                return;
```

```
            }
            updateLogo.setText(imagePath); // Set URI in text area
            Image image = new Image(imagePath); // Load image from URI
            logoImageView.setImage(image); // Display image in ImageView
        }
    }

    private int parseProjectID() {
        try {
            int id = Integer.parseInt(proID.getText().trim());
            if (!AddValidator.isPositiveInteger(id)) {
                showAlert(Alert.AlertType.ERROR, "Error", "Project ID must be a positive integer.");
                return -1;
            }
            return id;
        } catch (NumberFormatException e) {
            showAlert(Alert.AlertType.ERROR, "Error", "Invalid project ID. Please enter a valid
integer.");
            return -1;
        }
    }

    private Project findProjectByID(int projectID, List<Project> projects) {
        for (Project project : projects) {
            if (project.getProjectID() == projectID) {
                return project; // Return project if ID matches
            }
        }
        return null; // Project not found
    }

    private void updateFieldsWithProjectDetails(Project project) {
        updateName.setText(project.getProjectName());
        updateCategory.setText(project.getCategory());
        updateCountry.setText(project.getCountry());
        updateDetails.setText(project.getDescription());
        updateMembers.setText(String.join(", ", project.getTeamMembers()));
        updateLogo.setText(project.getTeamLogoUrl());
        logoImageView.setImage(new Image(project.getTeamLogoUrl()));
    }

    private void saveUpdatedProjectsList() {
        AddValidator.saveProjectsToFile(FILENAME, projectsList);
    }

    private void clearFields() {
        proID.clear();
        updateName.clear();
        updateCategory.clear();
        updateCountry.clear();
        updateDetails.clear();
        updateMembers.clear();
        updateLogo.clear();
        logoImageView.setImage(null);
        alertUpdate.setText("");
    }

    private void showAlert(Alert.AlertType alertType, String title, String message) {
        Alert alert = new Alert(alertType);
        alert.setTitle(title);
        alert.setHeaderText(null);
        alert.setContentText(message);
        alert.showAndWait();
    }
}
```

b) Description

The ‘UpdateProjectDetails’ class is responsible for updating project details. It reads project data from a file and stores it in a list. When the user enters a project ID and clicks "Check," the ‘onClickCheck’ method parses the ID using the ‘parseProjectID’ method, searches for the

project with ‘findProjectByID’, and if found, populates the input fields with the project's details using ‘updateFieldsWithProjectDetails’. If the project isn't found, an error alert is displayed and fields are cleared. When the user clicks "Update," the ‘onClickUpdate’ method updates the project details based on the input fields. It performs validations on the category, team members, and logo URL using the ‘updateProjectDetails’ method. If any validations fail, an error alert is shown. If all validations pass, the project details are updated in the list, and the updated list is saved back to the file "project_details.txt" using ‘saveUpdatedProjectsList’. The ‘onClickBrowse’ method allows the user to select a new logo image via a file chooser, ensuring the selected image URL is unique. It updates the logo's image URL in the text area and displays the image in an ‘ImageView’ if the URL is valid. Helper methods include ‘findProjectByID’ for project retrieval, ‘updateFieldsWithProjectDetails’ for populating fields with existing details, and ‘clearFields’ for clearing fields after updating. The ‘showAlert’ method is used to display messages to the user throughout these operations.

10. ViewProjectDetails

a) Code

```
package JavaCoursework;

import javafx.fxml.FXML;
import javafx.scene.control.Alert;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;

import java.util.List;

public class ViewProjectDetails {

    @FXML
    public TableView<Project> projectData;
    @FXML
    public TableColumn<Project, Integer> proID;
    @FXML
    public TableColumn<Project, String> proName;
    @FXML
    public TableColumn<Project, String> proCategory;
    @FXML
    public TableColumn<Project, String> proMembers;
    @FXML
    public TableColumn<Project, String> proCountry;
    @FXML
    public TableColumn<Project, String> proDescription;
    @FXML
    public TableColumn<Project, Image> teamLogo; // Changed to Image type

    private static final String FILENAME = "project_details.txt";
    private static List<Project> projectsList;

    @FXML
    private void initialize() {
        projectsList = WelcomeScreen.getProjectsFromFile(FILENAME);
        // Check if no projects were loaded
        if ((!AddValidator.validateProjectList(projectsList))) {
            showAlert(Alert.AlertType.WARNING, "No Projects Found",
                    "No projects were found in the file. Please add more projects.");
            return;
        }
```

```java
        // Call the method to Sort projects by projectID
        AddValidator.bubbleSortProjects(projectsList);
        initializeTableColumns();
        populateTableWithProjects();
    }

    private void initializeTableColumns() {
        // Initialize table columns
        proID.setCellValueFactory(new PropertyValueFactory<>("projectID"));
        proName.setCellValueFactory(new PropertyValueFactory<>("projectName"));
        proCategory.setCellValueFactory(new PropertyValueFactory<>("category"));
        proMembers.setCellValueFactory(new PropertyValueFactory<>("teamMembersAsString"));
        proDescription.setCellValueFactory(new PropertyValueFactory<>("description"));
        proCountry.setCellValueFactory(new PropertyValueFactory<>("country"));
        teamLogo.setCellValueFactory(new PropertyValueFactory<>("teamLogo")); // Use Image property
        // Set cell factory for teamLogo column to display images
        teamLogo.setCellFactory(column -> new ImageViewTableCell<>());
    }

    private void populateTableWithProjects() {
        // Clear any existing items
        projectData.getItems().clear();
        // Populate table with projects
        projectData.getItems().addAll(projectsList);
        // Load team logos for each project
        projectsList.forEach(project -> project.setTeamLogo(project.loadTeamLogo()));
    }


    // Custom cell factory for displaying images in a TableColumn
    static class ImageViewTableCell<S> extends javafx.scene.control.TableCell<S, Image> {
        private final ImageView imageView = new ImageView();
        @Override
        protected void updateItem(Image item, boolean empty) {
            super.updateItem(item, empty);
            if (empty || item == null) {
                setGraphic(null); // Clear cell if empty or item is null
            } else {
                imageView.setImage(item); // Set image to display in cell
                imageView.setFitWidth(50); // Set image width (adjust as needed)
                imageView.setPreserveRatio(true); // Preserve image aspect ratio
                setGraphic(imageView); // Set ImageView as cell's graphic
            }
        }
    }

    private void showAlert(Alert.AlertType alertType, String title, String message) {
        Alert alert = new Alert(alertType);
        alert.setTitle(title);
        alert.setHeaderText(null);
        alert.setContentText(message);
        alert.showAndWait();
    }
}
```

b) Description

The ‘ViewProjectDetails’ class manages displaying project details in a JavaFX ‘TableView’. Upon initialization it reads project data from "project_details.txt" and stores it in ‘projectsList’. If no projects are found, a warning alert is displayed. The projects are sorted by ID using the ‘bubbleSortProjects’ method. The ‘initializeTableColumns’ method sets up the table columns for various project attributes, including a custom cell factory (‘ImageViewTableCell’) for displaying team logo images. The ‘populateTableWithProjects’ method clears the table and adds all projects, ensuring each project's logo is displayed correctly. The ‘showAlert’ method is used to display alerts to the user. The ‘ImageViewTableCell’ inner class handles the display of images within table cells.

11. Start

a) Code

```
package JavaCoursework;

import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Node;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Alert;
import javafx.scene.control.ButtonType;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.stage.Stage;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

public class Start {

    @FXML
    public ImageView NextIcon;

    private static final String FILE_PATH = "project_details.txt";

    // List to store categorized projects after reading from the file.
    static List<List<Project>> categories;
    private Stage stage;
    private Scene scene;
    private Parent root;

    // Constructor initializes categories by reading project details from file
    public Start() {
        // Initialize categories list
        categories = readProjectDetails();
    }

    public void initialize() {
        loadImage(NextIcon, "NEXT.png");
    }

    void loadImage(ImageView imageView, String imagePath) {
        // Use the path relative to the resources folder
        Image image = new Image(getClass().getResourceAsStream("/" + imagePath));
        if (image != null) {
            imageView.setImage(image);
        } else {
            System.err.println("Image resource not found: " + imagePath);
        }
    }

    // Method to read project details from a text file and categorize them
    List<List<Project>> readProjectDetails() {
        List<List<Project>> categories = new ArrayList<>();

        try (BufferedReader reader = new BufferedReader(new FileReader(FILE_PATH))) {
            String line;
            List<Project> currentCategory = null;
            Project currentProject = null;

            while ((line = reader.readLine()) != null) {
                line = line.trim();
                if (!line.isEmpty()) {
                    if (line.startsWith("Project ID")) {
                        // Start of a new project
                        if (currentProject != null) {
                            // Add the last project to the category before starting a new one
                            if (currentCategory != null) {
```

```
                                currentCategory.add(currentProject);
                            }
                        }
                        // Start of a new project
                        currentProject = parseProject(line);
                        if (currentProject != null) {
                            String category = currentProject.getCategory();
                            int index = findCategoryIndex(categories, category);
                            if (index == -1) {
                                currentCategory = new ArrayList<>();
                                categories.add(currentCategory);
                            } else {
                                currentCategory = categories.get(index);
                            }
                        }
                    } else if (currentProject != null) {
                        // Continuation of project data
                        appendProjectData(currentProject, line);
                    }
                }
            }
            // Add the last project if the file ends without an empty line
            if (currentProject != null && currentCategory != null) {
                currentCategory.add(currentProject);
            }
        } catch (IOException e) {
            showAlert(Alert.AlertType.ERROR, "Error", "Error reading project details from file: " +
e.getMessage());
        }

        return categories;
    }

    // Parse a line from the file and create a SelectedProjects object
    private Project parseProject(String line) {
        Project project = null;
        try {
            String[] parts = line.split(":", 2);
            String key = parts[0].trim();
            String value = parts[1].trim();

            if (key.equals("Project ID")) {
                int projectID = Integer.parseInt(value);
                // Initialize new project with ID; other fields will be set in appendProjectData()
                project = new Project(projectID, "", "", new ArrayList<>(), "", "", "");
            }
        } catch (NumberFormatException | ArrayIndexOutOfBoundsException e) {
            showAlert(Alert.AlertType.ERROR, "Error", "Failed to parse project details: " +
e.getMessage());
        }
        return project;
    }

    // Append data from a line to an existing SelectedProjects object
    private void appendProjectData(Project project, String line) {
        String[] parts = line.split(":", 2);
        if (parts.length == 2) {
            String key = parts[0].trim();
            String value = parts[1].trim();

            switch (key) {
                case "Project Name":
                    project.setProjectName(value);
                    break;
                case "Category":
                    project.setCategory(value);
                    break;
                case "Team Members":
                    String[] teamMembersArray = value.split(", ");
                    List<String> teamMembersList = new ArrayList<>();
                    for (String member : teamMembersArray) {
                        teamMembersList.add(member);
                    }
                    project.setTeamMembers(teamMembersList);
                    break;
                case "Country":
                    project.setCountry(value);
```

```
                    break;
                case "Brief Description":
                    project.setDescription(value);
                    break;
                case "Logo URL":
                    project.setTeamLogoUrl(value);
                    break;
                default:
                    break;
            }
        }
    }
    // Find the index of a category in the categories list
    private int findCategoryIndex(List<List<Project>> categories, String category) {
        for (int i = 0; i < categories.size(); i++) {
            List<Project> categoryList = categories.get(i);
            if (!categoryList.isEmpty() && categoryList.get(0).getCategory().equals(category)) {
                return i;
            }
        }
        return -1;
    }
    // Method to get the list of categorized projects
    public static List<List<Project>> getCategories() {
        return categories;
    }

    // Handle the start event to check categories and show a confirmation dialog
    @FXML
    public void onClickStart(ActionEvent event) {
        // Check if ML, AI, RT categories are empty
        if (!isCategoryNotEmpty("ML") || !isCategoryNotEmpty("AI") || !isCategoryNotEmpty("RT")) {
            showAlert(Alert.AlertType.WARNING, "Empty Categories", "Some categories are empty. " +
                    "Please make sure all these categories have at least one project.");
        } else {
            showConfirmationDialog(event);
        }
    }

    // Check if a specific category is not empty
    static boolean isCategoryNotEmpty(String categoryName) {
        for (List<Project> category : categories) {
            if (!category.isEmpty() && category.get(0).getCategory().equals(categoryName)) {
                return true;
            }
        }
        return false;
    }

    // Show a confirmation dialog before proceeding to the final event layout
    private void showConfirmationDialog(ActionEvent event) {
        // Alert that the user cannot go back once the event starts
        Alert confirmationAlert = new Alert(Alert.AlertType.CONFIRMATION);
        confirmationAlert.setTitle("Confirmation");
        confirmationAlert.setHeaderText("You cannot go back once the event starts.");
        confirmationAlert.setContentText("Are you sure you want to proceed?");
        confirmationAlert.showAndWait().ifPresent(response -> {
            if (response == ButtonType.OK) {
                try {
                    // Load FinalEventLayout.fxml
                    FXMLLoader loader = new FXMLLoader(getClass().getResource("FinalEventLayout.fxml"));
                    root = loader.load();
                    stage = (Stage) ((Node) event.getSource()).getScene().getWindow();
                    scene = new Scene(root, 1100,630);
               root.getStylesheets().add(getClass().getResource("JavaCoursework.css").toExternalForm());
                    // Get the current stage
                    stage.setScene(scene);
                    stage.show();
                } catch (IOException e) {
                    showAlert(Alert.AlertType.ERROR, "Error", "Failed to load FinalEventLayout.fxml: " +
e.getMessage());
                }
            }
        });
    }
    // Method to display an alert dialog with specified type, title, and message
    private void showAlert(Alert.AlertType alertType, String title, String message) {
```

```
        Alert alert = new Alert(alertType);
        alert.setTitle(title);
        alert.setHeaderText(null);
        alert.setContentText(message);
        alert.showAndWait();
    }
}
```

b) Description

The ‘Start’ class initializes and categorizes project details for the TechExpo application. It reads project data from ‘project_details.txt’, categorizing projects into ML, AI, and RT using methods like ‘readProjectDetails()’, ‘parseProject()’, and ‘appendProjectData()’. The ‘findCategoryIndex()’ method locates the category index within the categories list. The ‘initialize()’ method sets up the ‘NextIcon’ image, while ‘loadImage()’ handles image loading. The ‘onClickStart(ActionEvent event)’ method validates that essential categories (ML, AI, RT) are not empty and shows an alert if any are empty; otherwise, it displays a confirmation dialog. Upon confirmation, it transitions to the ‘FinalEventLayout.fxml’ using ‘showConfirmationDialog()’. The ‘scene’ is also styled by adding a CSS stylesheet located at ‘JavaCoursework.css’. Utility methods include ‘isCategoryNotEmpty()’ for category checks and ‘showAlert()’ for displaying alerts.

12. FinalEventLayout

a) Code

```
package JavaCoursework;

import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.control.Alert;
import javafx.scene.control.ButtonBar;
import javafx.scene.control.ButtonType;
import javafx.scene.layout.StackPane;
import javafx.stage.Stage;

import java.io.IOException;

public class FinalEventLayout {

    @FXML
    private StackPane contentArea;

    public void initialize() {
        try {
            // Load the initial scene
            loadFXML("FinalEventStart.fxml", "Welcome to the Finals of the TechExpo!!!");
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }

    private void loadFXML(String fxmlFileName, String alertMessage) throws IOException {
        Parent fxml = FXMLLoader.load(getClass().getResource(fxmlFileName));
        contentArea.getChildren().clear();
        contentArea.getChildren().add(fxml);
        // Show the alert message
        showAlert(Alert.AlertType.INFORMATION, "Information", alertMessage);
    }
```

```
    @FXML
    public void FINALS() throws IOException {
        loadFXML("FinalEventStart.fxml", "Are you ready to start the finals of the TechExpo???");
    }

    public void VPD() throws IOException {
        loadFXML("ViewProjectDetails.fxml", "This is for viewing project details. " +
                "You can view your project details in ascending order based on proID of all projects.");
    }

    @FXML
    public void RSS() throws IOException {
        RandomSpotlightSelect.onClickRandomSelection();
        if (!AddValidator.validateSelectedProjects(RandomSpotlightSelect.getSelectedProjects())) {
            showAlert(Alert.AlertType.WARNING, "Data Incomplete", "Please ensure at least 3 projects are
available.");
            return;
        }
        loadFXML("AwardWinningProjects.fxml", "For each category, one project has been randomly
selected. " +
                "Click on the top button and judge can give the points for the selected project using
'*' to select the winners.");
    }

    @FXML
    public void VAP() throws IOException {
        if (!AddValidator.validateAwardWinningProjects(AwardWinningProjects.getSelectedProjects())) {
            showAlert(Alert.AlertType.WARNING, "Data Incomplete", "Please ensure at least 3 projects are
available.");
            return;
        }
        loadFXML("VisualizingAwardWinningProjects.fxml", "This is for visualizing award-winning
projects. " +
                "Here, the awarded projects will be visualized in a bar chart for graphical view.");
    }

    @FXML
    public void END() throws IOException {
        if (AwardWinningProjects.getSelectedProjects() == null ||
AwardWinningProjects.getSelectedProjects().size() < 3) {
            showAlert(Alert.AlertType.WARNING, "Incomplete Event!", "Please complete the events.");
            return;
        }
        loadFXML("ThankYou.fxml", "Thank you for participating in TechExpo!!!");
    }

    @FXML
    public void Exit() {
        Alert alert = new Alert(Alert.AlertType.CONFIRMATION);
        alert.setTitle("Confirmation Dialog");
        alert.setHeaderText("Are you sure you want to exit?");
        alert.setContentText("Press OK to exit or Cancel to stay.");

        // Customizing the buttons in the alert dialog
        ButtonType okButton = new ButtonType("OK");
        ButtonType cancelButton = new ButtonType("Cancel", ButtonBar.ButtonData.CANCEL_CLOSE);
        alert.getButtonTypes().setAll(okButton, cancelButton);

        // Handling user's choice
        alert.showAndWait().ifPresent(response -> {
            if (response == okButton) {
                // Close the application
                Stage stage = (Stage) contentArea.getScene().getWindow();
                stage.close();
            }
        });
    }

    private void showAlert(Alert.AlertType alertType, String title, String message) {
        Alert alert = new Alert(alertType);
        alert.setTitle(title);
        alert.setHeaderText(null);
        alert.setContentText(message);
        alert.showAndWait();
    }
}
```

b) Description

The ‘FinalEventLayout’ class oversees the TechExpo finals interface. It starts by loading ‘FinalEventStart.fxml’ into a ‘StackPane’ called ‘contentArea’. The class includes several event handlers: ‘FINALS()’ loads the finals layout, ‘VPD()’ opens the project details view, ‘RSS()’ initiates random project selection and, if validated, loads the award-winning projects layout, ‘VAP()’ ensures the validity of award-winning projects before displaying them in a bar chart, and ‘END()’ transitions to a thank-you screen if at least three projects are selected. The ‘Exit()’ method prompts for confirmation before closing the application. Error handling and user notifications are managed through the ‘showAlert()’ method.

13. FinalEventStart

a) Code

```
package JavaCoursework;

import javafx.fxml.FXML;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;

public class FinalEventStart {
    @FXML
    public ImageView IconRSS;
    @FXML
    public ImageView IconAWP;
    @FXML
    public ImageView IconVAP;
    @FXML
    public ImageView IconVPD;
    @FXML
    public ImageView IconEND;
    @FXML
    public ImageView IconEXIT;

    @FXML
    public void initialize() {
        // Load images for all ImageViews
        loadImage(IconVPD, "VPD.png");
        loadImage(IconRSS, "RSS.png");
        loadImage(IconAWP, "AWP.png");
        loadImage(IconVAP, "VAP.png");
        loadImage(IconEND, "THANKYOU.png");
        loadImage(IconEXIT, "EXIT.png");
    }

    private void loadImage(ImageView imageView, String imagePath) {
        // Use the path relative to the resources folder
        Image image = new Image(getClass().getResourceAsStream("/" + imagePath));
        if (image != null) {
            imageView.setImage(image);
        } else {
            System.err.println("Image resource not found: " + imagePath);
        }
    }
}
```

b) Description

The ‘FinalEventStart’ class in the JavaFX application handles the display of icons for different sections using ‘ImageView’ components. The ‘initialize()’ method is responsible for loading images into these ‘ImageView’ components by invoking the ‘loadImage()’ method. This method loads images from the resources folder and sets them to the respective ‘ImageView’. If an image is not found, an error message is printed to the console, ensuring that any missing resources are reported.

14. RandomSpotlightSelect

a) Code

```
package JavaCoursework;

import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.scene.control.Alert;

import java.util.ArrayList;
import java.util.List;
import java.util.Random;

public class RandomSpotlightSelect {

    static List<List<Project>> categories; // List to hold categories of projects
    // ObservableList to hold selected projects
    private static ObservableList<Project> selectedProjects = FXCollections.observableArrayList();

    public static ObservableList<Project> getSelectedProjects() {
        return selectedProjects;
    }

    public static void onClickRandomSelection() {
        categories = Start.getCategories();
        selectedProjects.clear(); // Clear the list before adding new projects
        // Select random project for each category
        selectRandomProjectForCategory("AI", selectedProjects);
        selectRandomProjectForCategory("ML", selectedProjects);
        selectRandomProjectForCategory("RT", selectedProjects);
    }

    // Method to select a random project for a given category and add it to selectedProjects list
    private static void selectRandomProjectForCategory(String category, ObservableList<Project>
selectedProjects) {
        List<Project> categoryProjects = getCategoryProjects(category);
        // Get projects for the specified category (Temporary List to store)
        // Check if categoryProjects is not empty and select a random project
        if  (AddValidator.validateCategoryProjects(categoryProjects, category)) {
            int randomIndex = generateRandomIndex(categoryProjects.size());
            Project selectedProject = categoryProjects.get(randomIndex);
            selectedProjects.add(selectedProject); // Add the selected project to selectedProjects list
        }else {
            // Show warning message based on the validation result
            String message = AddValidator.getValidationMessage(categoryProjects, category);
            if (message != null) {
                showAlert(Alert.AlertType.WARNING, "Warning", message);
            }
        }
    }

    // Method to fetch projects belonging to a specific category
    static List<Project> getCategoryProjects(String category) {
        List<Project> categoryProjects = new ArrayList<>();
        for (List<Project> categoryList : categories) {
            for (Project project : categoryList) {
```

```
                if (project.getCategory().equals(category)) {
                    categoryProjects.add(project);
                }
            }
        }
        return categoryProjects;
    }

    // Method to generate a random index within a given range
    static int generateRandomIndex(int upperBound) {
        Random random = new Random();
        return random.nextInt(upperBound);
    }

    // Method to display an alert with specified type, title, and message
    private static void showAlert(Alert.AlertType alertType, String title, String message) {
        Alert alert = new Alert(alertType);
        alert.setTitle(title);
        alert.setHeaderText(null);
        alert.setContentText(message);
        alert.showAndWait();
    }
}
```

b) Description

The ‘RandomSpotlightSelect’ class is designed to dynamically select and display random projects from the AI, ML, and RT categories in a JavaFX application. It manages the selection process through the ‘onClickRandomSelection()’ method, which clears the ‘selectedProjects’ list and selects one random project from each category using ‘selectRandomProjectForCategory()’. This method filters projects by category with ‘getCategoryProjects()’, selects a random project using ‘generateRandomIndex()’, and adds it to ‘selectedProjects’. If no projects are found for a category, a warning alert is displayed via ‘showAlert()’. The ‘getCategoryProjects()’ method iterates through the ‘categories’ list to gather projects matching the specified category. Alerts are shown using ‘showAlert()’, providing user feedback.

15. AwardWinningProjects

a) Code

```
package JavaCoursework;

import javafx.beans.property.SimpleStringProperty;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.scene.control.Alert;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.TextInputDialog;
import javafx.scene.control.cell.PropertyValueFactory;

public class AwardWinningProjects {

    @FXML
    TableView<Project> projectData;

    @FXML
    TableColumn<Project, Integer> proID;

    @FXML
```

```
    TableColumn<Project, String> proName;

    @FXML
    TableColumn<Project, String> proCategory;

    @FXML
    TableColumn<Project, String> proMembers;

    @FXML
    TableColumn<Project, String> proCountry;

    @FXML
    TableColumn<Project, String> proDescription;

    @FXML
    TableColumn<Project, Integer> judgesPoints;

    @FXML
    TableColumn<Project, String> place; // TableColumn for places

    private static ObservableList<Project> selectedProjects = FXCollections.observableArrayList();

    public AwardWinningProjects() {
        // Clear the list before adding new projects
        selectedProjects.clear();
        // Fetch the randomly selected projects from RandomSpotlightSelection
        //used as intermediate between selectedProjects and RSS
        ObservableList<Project> selected = RandomSpotlightSelect.getSelectedProjects();

        // Convert SelectedProjects to Projects and add to selectedProjects list
        for (Project project : selected) {
            Project newProject = new Project(
                    project.getProjectID(),
                    project.getProjectName(),
                    project.getCategory(),
                    project.getTeamMembers(),
                    project.getDescription(),
                    project.getCountry()
            );
            selectedProjects.add(newProject);
        }
    }

    public static ObservableList<Project> getSelectedProjects() {
        return selectedProjects;
    }

    @FXML
    public void initialize() {
        setupTableColumns();
        projectData.setItems(selectedProjects);
    }

    private void setupTableColumns() {
        projectData.setVisible(false);
        proID.setCellValueFactory(new PropertyValueFactory<>("projectID"));
        proName.setCellValueFactory(new PropertyValueFactory<>("projectName"));
        proCategory.setCellValueFactory(new PropertyValueFactory<>("category"));
        proMembers.setCellValueFactory(new PropertyValueFactory<>("teamMembersAsString"));
        proCountry.setCellValueFactory(new PropertyValueFactory<>("country"));
        proDescription.setCellValueFactory(new PropertyValueFactory<>("description"));
        judgesPoints.setCellValueFactory(new PropertyValueFactory<>("judgesPoints"));
        place.setCellValueFactory(cellData -> new
SimpleStringProperty(getPlaceString(selectedProjects.indexOf(cellData.getValue()))));
    }

    String getPlaceString(int index) {
        return switch (index) {
            case 0 -> "1st";
            case 1 -> "2nd";
            case 2 -> "3rd";
            default -> "";
        };
    }

    @FXML
    public void onClickJudgeScoring() {
```

```
        scoreProjects();
        determineWinners(); //Call the function to determine winners
        projectData.setVisible(true); // Show TableView after scoring is complete
    }

    void scoreProjects() {
        for (Project project : selectedProjects) {
            int totalScore = getJudgeScore(project);
            project.setJudgesPoints(totalScore);
        }
        // Update TableView
        projectData.setItems(selectedProjects);
        projectData.refresh(); // Refresh TableView to reflect updated values
    }

    private int getJudgeScore(Project project) {
        int totalScore = 0;
        for (int i = 1; i <= 4; i++) {
            boolean validInput = false;
            while (!validInput) {
                TextInputDialog dialog = createScoreDialog(i, project);
                dialog.showAndWait();
                String rating = dialog.getEditor().getText();
                if (AddValidator.isValidRating(rating)) {
                    totalScore += rating.length();
                    validInput = true;
                } else {
                    showAlert(Alert.AlertType.WARNING, "Invalid Input", "Please enter between 1 and 5
'*' characters.");
                }
            }
        }
        return totalScore;
    }

    private TextInputDialog createScoreDialog(int judgeNumber, Project project) {
        TextInputDialog dialog = new TextInputDialog();
        dialog.setTitle("Judge Scoring");
        dialog.setHeaderText("Judge " + judgeNumber + ": Scoring for " + project.getCategory() + "
Project with ProID - " + project.getProjectID()
                + "  & ProName - " + project.getProjectName());
        dialog.setContentText("Enter rating using stars only (1-5):");
        return dialog;
    }

    void determineWinners() {
        AddValidator.bubbleSortProjects(selectedProjects);
        String resultMessage = getResultMessage();
        showAlert(Alert.AlertType.INFORMATION, "Winners", resultMessage);
    }

    private String getResultMessage() {
        StringBuilder resultMessage = new StringBuilder("Overall Ranking:\n");
        if (!selectedProjects.isEmpty()) {
            resultMessage.append("1st Place: ").append(selectedProjects.get(0).getCategory()).append("
with ").
                    append(selectedProjects.get(0).getJudgesPoints()).append(" points\n");
        }
        if (selectedProjects.size() > 1) {
            resultMessage.append("2nd Place: ").append(selectedProjects.get(1).getCategory()).append("
with ").
                    append(selectedProjects.get(1).getJudgesPoints()).append(" points\n");
        }
        if (selectedProjects.size() > 2) {
            resultMessage.append("3rd Place: ").append(selectedProjects.get(2).getCategory()).append("
with ").
                    append(selectedProjects.get(2).getJudgesPoints()).append(" points\n");
        }
        return resultMessage.toString();
    }

    // Utility method to show alert with specified type, title, and message
    private void showAlert(Alert.AlertType alertType, String title, String message) {
        Alert alert = new Alert(alertType);
        alert.setTitle(title);
        alert.setHeaderText(null);
        alert.setContentText(message);
```

```
        alert.showAndWait();
    }
}
```

b) Description

The ‘AwardWinningProjects’ class manages and displays top projects for the TechExpo awards in JavaFX. It initializes a ‘TableView’ with columns for project details such as ID, name, category, team members, country, description, and judges' points, with the ‘initialize()’ method setting up these columns and initially hiding the table. The ‘onClickJudgeScoring()’ method prompts judges to input ratings via ‘TextInputDialog’, validates these ratings, computes total scores using ‘getJudgeScore()’, and updates the ‘TableView’. ‘determineWinners()’ sorts the projects by their scores and displays the top projects using ‘getResultMessage()’. The ‘createScoreDialog()’ method generates dialogs for rating input, while ‘showAlert()’ provides feedback on invalid input or displays results.

16. VisualizingAwardWinningProjects

a) Code

```
package JavaCoursework;

import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.chart.BarChart;
import javafx.scene.chart.XYChart;
import javafx.collections.ObservableList;

import java.net.URL;
import java.util.ResourceBundle;

public class VisualizingAwardWinningProjects implements Initializable {

    @FXML
    private BarChart<Number, String> pointsBarChart;

    private ObservableList<Project> topProjects; // Use ObservableList directly

    @Override
    public void initialize(URL location, ResourceBundle resources) {
        // Fetch selected top projects directly into ObservableList
        topProjects = fetchTopProjects();
        plotData(); // Plot data on the bar chart
    }

    private ObservableList<Project> fetchTopProjects() {
        // Fetch the award-winning projects from AwardWinningProjects
        return AwardWinningProjects.getSelectedProjects();
    }

    private void plotData() {
        XYChart.Series<Number, String> series = new XYChart.Series<>();
        series.setName("Points"); // Set series name for the bar chart
        // Clear any existing data in the chart
        pointsBarChart.getData().clear();
        // Use the provided order from selectedProjects without sorting
        for (int i = 0; i < Math.min(3, topProjects.size()); i++) {
            Project project = topProjects.get(i); // Get each top project
            String label = project.getProjectName() + " (" + project.getCountry() + ")"; // Generate
label for the bar
            series.getData().add(new XYChart.Data<>(project.getJudgesPoints(),label)); // Add data to
series
```

```
        }
        pointsBarChart.getData().add(series); // Add series to the bar chart
    }
}
```

b) Description

The ‘VisualizingAwardWinningProjects’ class uses JavaFX to display award-winning projects in a ‘BarChart’. It initializes by fetching top projects from ‘AwardWinningProjects’ using the ‘getSelectedProjects()’ method, storing them in an ‘ObservableList’. The ‘initialize()’ method calls ‘plotData()’ to populate the chart. ‘fetchTopProjects()’ retrieves the latest project data for visualization, while ‘plotData()’ clears existing chart data, creates a new ‘XYChart.Series’ with the name "Points", and adds each project's judges' points and label to the series. The labels include the project name and country. Finally, the series is added to the ‘BarChart’, providing a visual representation of the top projects' performance.

Junit Tests

a) Code

```
package JavaCoursework;

import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import org.junit.jupiter.api.*;

import java.io.*;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import static org.junit.jupiter.api.Assertions.*;

public class AddValidatorTest {

    @Test
    public void testValidateProjectID() {
        List<Project> projectsList = new ArrayList<>();
        projectsList.add(new Project(1, "Test Project", "AI", List.of("Member1", "Member2", "Member3"),
"Description", "Country", "http://example.com/logo.png"));

        assertEquals("Warning: The project ID must be a positive integer.",
AddValidator.validateProjectID(0, projectsList));
        assertEquals("Warning: The project ID already exists.", AddValidator.validateProjectID(1,
projectsList));
        assertEquals("Success: The project ID is available.", AddValidator.validateProjectID(2,
projectsList));
    }

    @Test
    public void testIsDuplicateProjectID() {
        List<Project> projectsList = new ArrayList<>();
        projectsList.add(new Project(1, "Test Project", "AI", List.of("Member1", "Member2", "Member3"),
"Description", "Country", "http://example.com/logo.png"));

        assertTrue(AddValidator.isDuplicateProjectID(1, projectsList));
        assertFalse(AddValidator.isDuplicateProjectID(2, projectsList));
    }

    @Test
```

```
    public void testValidateAndFormatProjectName() {
        assertNull(AddValidator.validateAndFormatProjectName(""));
        assertEquals("Test project", AddValidator.validateAndFormatProjectName("test project"));
    }

    @Test
    public void testValidateCategory() {
        assertNull(AddValidator.validateCategory("InvalidCategory"));
        assertEquals("AI", AddValidator.validateCategory("AI"));
        assertEquals("ML", AddValidator.validateCategory("ML"));
        assertEquals("RT", AddValidator.validateCategory("RT"));
    }

    @Test
    public void testValidateAndFormatTeamMembers() {
        assertNull(AddValidator.validateAndFormatTeamMembers("Member1,Member2"));
        assertEquals(List.of("Member1", "Member2", "Member3"),
AddValidator.validateAndFormatTeamMembers("Member1,Member2,Member3"));
    }

    @Test
    public void testValidateAndFormatDescription() {
        assertNull(AddValidator.validateAndFormatDescription(""));
        assertEquals("This is a project description.", AddValidator.validateAndFormatDescription("this
is a project description."));
    }

    @Test
    public void testValidateAndFormatCountry() {
        assertNull(AddValidator.validateAndFormatCountry(""));
        assertEquals("Country", AddValidator.validateAndFormatCountry("country"));
    }

    @Test
    public void testIsDuplicateLogoUrl() {
        List<Project> projectsList = new ArrayList<>();
        projectsList.add(new Project(1, "Test Project", "AI", List.of("Member1", "Member2", "Member3"),
"Description", "Country", "http://example.com/logo.png"));

        assertTrue(AddValidator.isDuplicateLogoUrl("http://example.com/logo.png", projectsList));
        assertFalse(AddValidator.isDuplicateLogoUrl("http://example.com/newlogo.png", projectsList));
    }


    @Test
    public void testSaveProjectsToFile() {
        String filePath = "testProjects.txt";
        List<Project> projectsList = new ArrayList<>();
        projectsList.add(new Project(1, "AI Project", "AI", List.of("Member1", "Member2", "Member3"),
"AI Description", "CountryA", "http://example.com/ai_logo.png"));
        projectsList.add(new Project(2, "ML Project", "ML", List.of("Member4", "Member5", "Member6"),
"ML Description", "CountryB", "http://example.com/ml_logo.png"));
        projectsList.add(new Project(3, "RT Project", "RT", List.of("Member7", "Member8", "Member9"),
"RT Description", "CountryC", "http://example.com/rt_logo.png"));

        AddValidator.saveProjectsToFile(filePath, projectsList);

        try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {
            String line;
            boolean foundAI = false, foundML = false, foundRT = false;
            while ((line = reader.readLine()) != null) {
                if (line.contains("AI Projects")) {
                    foundAI = true;
                }
                if (line.contains("ML Projects")) {
                    foundML = true;
                }
                if (line.contains("RT Projects")) {
                    foundRT = true;
                }
            }
            assertTrue(foundAI, "AI Projects header not found in file");
            assertTrue(foundML, "ML Projects header not found in file");
            assertTrue(foundRT, "RT Projects header not found in file");
        } catch (IOException e) {
            e.printStackTrace();
            fail("IOException occurred while reading the file");
```

```java
        }
    }

    @Test
    public void testSaveProjectsToTextFile() throws IOException {
        // Setup
        String testFilePath = "test_projects.txt";
        List<Project> projects = new ArrayList<>();

        // Create sample projects
        projects.add(new Project(1, "AI Project 1", "AI", List.of("Member A", "Member B", "Member C"),
"Description 1", "Country A", null));
        projects.add(new Project(2, "ML Project 1", "ML", List.of("Member A", "Member B", "Member C"),
"Description 2", "Country B", null));
        projects.add(new Project(3, "RT Project 1", "RT", List.of("Member A", "Member B", "Member C"),
"Description 3", "Country C", null));

        // Save projects to file
        AddValidator.saveProjectsToFile(testFilePath, projects);

        // Read the file and verify content
        StringBuilder fileContent = new StringBuilder();
        try (BufferedReader reader = new BufferedReader(new FileReader(testFilePath))) {
            String line;
            while ((line = reader.readLine()) != null) {
                fileContent.append(line).append("\n");
            }
        }

        String expectedContent =
                "\n*********** AI Projects ***********\n\n" +
                        "---Project Details---\n\n" +
                        "Project ID: 1\n" +
                        "Project Name: AI Project 1\n" +
                        "Category: AI\n" +
                        "Team Members: Member A, Member B, Member C\n" +
                        "Brief Description: Description 1\n" +
                        "Country: Country A\n" +
                        "Logo URL: null\n\n" +
                        "\n*********** ML Projects ***********\n\n" +
                        "---Project Details---\n\n" +
                        "Project ID: 2\n" +
                        "Project Name: ML Project 1\n" +
                        "Category: ML\n" +
                        "Team Members: Member A, Member B, Member C\n" +
                        "Brief Description: Description 2\n" +
                        "Country: Country B\n" +
                        "Logo URL: null\n\n" +
                        "\n*********** RT Projects ***********\n\n" +
                        "---Project Details---\n\n" +
                        "Project ID: 3\n" +
                        "Project Name: RT Project 1\n" +
                        "Category: RT\n" +
                        "Team Members: Member A, Member B, Member C\n" +
                        "Brief Description: Description 3\n" +
                        "Country: Country C\n" +
                        "Logo URL: null\n\n";

        // Print differences for debugging
        String actualContent = fileContent.toString();
        if (!expectedContent.equals(actualContent)) {
            System.out.println("Expected Content:");
            System.out.println(expectedContent);
            System.out.println("Actual Content:");
            System.out.println(actualContent);
        }

        // Assertion
        assertEquals(expectedContent, actualContent, "The file content should match the expected
content.");

        // Clean up
        new java.io.File(testFilePath).delete();
    }


    @Test
```

```
    public void testIsProjectIDValid_withValidID() {
        List<Project> projects = Arrays.asList(
                new Project(1, "Project1", "AI", Arrays.asList("Member1", "Member2", "Member3"),
"Description1", "Country1", "url1"),
                new Project(2, "Project2", "ML", Arrays.asList("Member1", "Member2", "Member3"),
"Description2", "Country2", "url2")
        );
        assertTrue(AddValidator.isProjectIDValid(projects, 1));
        assertTrue(AddValidator.isProjectIDValid(projects, 2));
    }

    @Test
    public void testIsProjectIDValid_withInvalidID() {
        List<Project> projects = Arrays.asList(
                new Project(1, "Project1", "AI", Arrays.asList("Member1", "Member2", "Member3"),
"Description1", "Country1", "url1")
        );
        assertFalse(AddValidator.isProjectIDValid(projects, 2)); // ID not in the list
        assertFalse(AddValidator.isProjectIDValid(projects, -1)); // Negative ID
    }

    @Test
    public void testIsProjectIDValid_withNullList() {
        assertThrows(IllegalArgumentException.class, () -> {
            AddValidator.isProjectIDValid(null, 1);
        });
    }

    @Test
    public void testIsValidProjectID() {
        assertTrue(AddValidator.isPositiveInteger(1));   // Valid positive integer
        assertTrue(AddValidator.isPositiveInteger(100)); // Valid positive integer
        assertFalse(AddValidator.isPositiveInteger(0));  // Zero is not positive
        assertFalse(AddValidator.isPositiveInteger(-1)); // Negative number is not positive
    }

    @Test
    public void testValidateProjectList WithValidProjects() {
        // Create a list with some projects
        List<Project> projects = new ArrayList<>();
        projects.add(new Project(1, "Project 1", "Category 1", List.of("Ma", "Ka", "Vi"), "Description
1", "Country 1", "Url.png"));

        // Validate the list
        boolean result = AddValidator.validateProjectList(projects);

        // Assert that the result is true
        assertTrue(result, "The project list should be considered valid.");
    }

    @Test
    public void testValidateProjectList_WithEmptyList() {
        // Create an empty list
        List<Project> projects = new ArrayList<>();

        // Validate the list
        boolean result = AddValidator.validateProjectList(projects);

        // Assert that the result is false
        assertFalse(result, "The project list should be considered invalid.");
    }

    @Test
    public void testValidateProjectList_WithNullList() {
        // Create a null list
        List<Project> projects = null;

        // Validate the list
        boolean result = AddValidator.validateProjectList(projects);

        // Assert that the result is false
        assertFalse(result, "The project list should be considered invalid.");
    }

    // Method to test bubbleSortProjects
    @Test
    public void testBubbleSortProjects() {
```

```java
        // Create a list of projects with unsorted projectIDs
        List<Project> projects = new ArrayList<>();
        projects.add(new Project(5, "Project5", "CategoryA", Arrays.asList("Member1", "Member2"),
"Description5", "Country5", "URL5"));
        projects.add(new Project(2, "Project2", "CategoryB", Arrays.asList("Member3", "Member4"),
"Description2", "Country2", "URL2"));
        projects.add(new Project(9, "Project9", "CategoryC", Arrays.asList("Member5", "Member6"),
"Description9", "Country9", "URL9"));
        projects.add(new Project(1, "Project1", "CategoryD", Arrays.asList("Member7", "Member8"),
"Description1", "Country1", "URL1"));
        projects.add(new Project(6, "Project6", "CategoryE", Arrays.asList("Member9", "Member10"),
"Description6", "Country6", "URL6"));

        // Expected sorted list by projectID
        List<Project> expectedProjects = new ArrayList<>();
        expectedProjects.add(new Project(1, "Project1", "CategoryD", Arrays.asList("Member7",
"Member8"), "Description1", "Country1", "URL1"));
        expectedProjects.add(new Project(2, "Project2", "CategoryB", Arrays.asList("Member3",
"Member4"), "Description2", "Country2", "URL2"));
        expectedProjects.add(new Project(5, "Project5", "CategoryA", Arrays.asList("Member1",
"Member2"), "Description5", "Country5", "URL5"));
        expectedProjects.add(new Project(6, "Project6", "CategoryE", Arrays.asList("Member9",
"Member10"), "Description6", "Country6", "URL6"));
        expectedProjects.add(new Project(9, "Project9", "CategoryC", Arrays.asList("Member5",
"Member6"), "Description9", "Country9", "URL9"));

        // Sort the projects using bubbleSortProjects
        AddValidator.bubbleSortProjects(projects);

        // Assert that the sorted list matches the expected list based on projectID
        for (int i = 0; i < projects.size(); i++) {
            assertEquals(expectedProjects.get(i).getProjectID(), projects.get(i).getProjectID());
        }
    }

    @Test
    public void testValidateCategoryProjectsWithNullCategoryProjects() {
        List<Project> nullProjects = null;
        String category = "AI";
        boolean result = AddValidator.validateCategoryProjects(nullProjects, category);
        assertFalse(result, "Expected validation to fail for null categoryProjects.");
    }

    @Test
    public void testValidateCategoryProjectsWithEmptyCategoryProjects() {
        List<Project> emptyProjects = new ArrayList<>();
        String category = "AI";
        boolean result = AddValidator.validateCategoryProjects(emptyProjects, category);
        assertFalse(result, "Expected validation to fail for empty categoryProjects.");
    }

    @Test
    public void testValidateCategoryProjectsWithValidCategoryProjects() {
        List<Project> validProjects = new ArrayList<>();
        validProjects.add(new Project(1, "Project 1", "AI", new ArrayList<>(), "Description 1", "Country
1", "URL1"));
        String category = "AI";
        boolean result = AddValidator.validateCategoryProjects(validProjects, category);
        assertTrue(result, "Expected validation to pass for valid categoryProjects.");
    }

    @Test
    public void testGetValidationMessageWithNullCategoryProjects() {
        List<Project> nullProjects = null;
        String category = "AI";
        String result = AddValidator.getValidationMessage(nullProjects, category);
        assertEquals("Warning: The category list for AI is null.", result);
    }

    @Test
    public void testGetValidationMessageWithEmptyCategoryProjects() {
        List<Project> emptyProjects = new ArrayList<>();
        String category = "AI";
        String result = AddValidator.getValidationMessage(emptyProjects, category);
        assertEquals("Warning: No projects found for category: AI", result);
    }
```

```
    @Test
    public void testGetValidationMessageWithValidCategoryProjects() {
        List<Project> validProjects = new ArrayList<>();
        validProjects.add(new Project(1, "Project 1", "AI", new ArrayList<>(), "Description 1", "Country
1", "URL1"));
        String category = "AI";
        String result = AddValidator.getValidationMessage(validProjects, category);
        assertNull(result, "Expected no validation message for valid categoryProjects.");
    }

    @Test
    void testIsValidRating() {
        assertTrue(AddValidator.isValidRating("**"));
        assertTrue(AddValidator.isValidRating("*****"));
        assertFalse(AddValidator.isValidRating(""));
        assertFalse(AddValidator.isValidRating("******"));
        assertFalse(AddValidator.isValidRating("abc"));
    }

    @Test
    void testBubbleSortPointsProjects() {
        // Create a list of Project without specifying judgesPoints
        ObservableList<Project> projects = FXCollections.observableArrayList(
                new Project(1, "Project A", "ML", List.of("TeamA"), "Description A", "Country A"),
                new Project(2, "Project B", "AI", List.of("TeamB"), "Description B", "Country B"),
                new Project(3, "Project C", "RT", List.of("TeamC"), "Description C", "Country C")
        );

        // Set judgesPoints for each project
        projects.get(0).setJudgesPoints(15);
        projects.get(1).setJudgesPoints(10);
        projects.get(2).setJudgesPoints(20);

        // Perform the bubble sort
        AddValidator.bubbleSortProjects(projects);

        // Validate that the projects are sorted by judgesPoints in descending order
        assertEquals(20, projects.get(0).getJudgesPoints());
        assertEquals(15, projects.get(1).getJudgesPoints());
        assertEquals(10, projects.get(2).getJudgesPoints());
    }

    @Test
    void testGetPlaceString() {
        // Create an instance of the class containing getPlaceString
        AwardWinningProjects awardWinningProjects = new AwardWinningProjects();

        // Test for 1st place
        assertEquals("1st", awardWinningProjects.getPlaceString(0), "Expected 1st for index 0");

        // Test for 2nd place
        assertEquals("2nd", awardWinningProjects.getPlaceString(1), "Expected 2nd for index 1");

        // Test for 3rd place
        assertEquals("3rd", awardWinningProjects.getPlaceString(2), "Expected 3rd for index 2");

        // Test for index greater than 2
        assertEquals("", awardWinningProjects.getPlaceString(3), "Expected empty string for index 3");

        // Test for negative index
        assertEquals("", awardWinningProjects.getPlaceString(-1), "Expected empty string for negative
index");
    }

    @Test
    void testValidateSelectedProjects() {
        // Test with null
        assertFalse(AddValidator.validateSelectedProjects(null));

        // Test with empty list
        ObservableList<Project> emptyList = FXCollections.observableArrayList();
        assertFalse(AddValidator.validateSelectedProjects(emptyList));

        // Test with list containing fewer than 3 projects
        ObservableList<Project> lessThanThree = FXCollections.observableArrayList(
                new Project(1, "Project A", "ML", List.of("TeamA"), "Description A", "Country A", null),
                new Project(2, "Project B", "AI", List.of("TeamB"), "Description B", "Country B", null)
```

```java
        );
        assertFalse(AddValidator.validateSelectedProjects(lessThanThree));

        // Test with list containing exactly 3 projects
        ObservableList<Project> exactlyThree = FXCollections.observableArrayList(
                new Project(1, "Project A", "ML", List.of("TeamA"), "Description A", "Country A", null),
                new Project(2, "Project B", "AI", List.of("TeamB"), "Description B", "Country B", null),
                new Project(3, "Project C", "RT", List.of("TeamC"), "Description C", "Country C", null)
        );
        assertTrue(AddValidator.validateSelectedProjects(exactlyThree));

        // Test with list containing more than 3 projects
        ObservableList<Project> moreThanThree = FXCollections.observableArrayList(
                new Project(1, "Project A", "ML", List.of("TeamA"), "Description A", "Country A", null),
                new Project(2, "Project B", "AI", List.of("TeamB"), "Description B", "Country B", null),
                new Project(3, "Project C", "RT", List.of("TeamC"), "Description C", "Country C", null),
                new Project(4, "Project D", "ML", List.of("TeamD"), "Description D", "Country D", null)
        );
        assertTrue(AddValidator.validateSelectedProjects(moreThanThree));
    }

    @Test
    void testValidateAwardWinningProjects() {
        // Test with null
        assertFalse(AddValidator.validateAwardWinningProjects(null));

        // Test with empty list
        ObservableList<Project> emptyList = FXCollections.observableArrayList();
        assertFalse(AddValidator.validateAwardWinningProjects(emptyList));

        // Test with list containing fewer than 3 projects
        ObservableList<Project> lessThanThree = FXCollections.observableArrayList(
                new Project(1, "Project A", "ML", List.of("TeamA"), "Description A", "Country A"),
                new Project(2, "Project B", "AI", List.of("TeamB"), "Description B", "Country B")
        );
        assertFalse(AddValidator.validateAwardWinningProjects(lessThanThree));

        // Test with list containing exactly 3 projects
        ObservableList<Project> exactlyThree = FXCollections.observableArrayList(
                new Project(1, "Project A", "ML", List.of("TeamA"), "Description A", "Country A"),
                new Project(2, "Project B", "AI", List.of("TeamB"), "Description B", "Country B"),
                new Project(3, "Project C", "RT", List.of("TeamC"), "Description C", "Country C")
        );
        assertTrue(AddValidator.validateAwardWinningProjects(exactlyThree));

        // Test with list containing more than 3 projects
        ObservableList<Project> moreThanThree = FXCollections.observableArrayList(
                new Project(1, "Project A", "ML", List.of("TeamA"), "Description A", "Country A"),
                new Project(2, "Project B", "AI", List.of("TeamB"), "Description B", "Country B"),
                new Project(3, "Project C", "RT", List.of("TeamC"), "Description C", "Country C"),
                new Project(4, "Project D", "ML", List.of("TeamD"), "Description D", "Country D")
        );
        assertTrue(AddValidator.validateAwardWinningProjects(moreThanThree));
    }
}
```

b) Description

The ‘AddValidatorTest’ class contains JUnit test cases to verify the functionality of various methods in the ‘AddValidator’ class. Tests include validating project IDs, checking for duplicates, and ensuring correct formatting of project names, categories, team members, descriptions, and countries. It also tests for duplicate logo URLs, saving projects to files, and sorting projects. The test methods ensure that validation and formatting methods work as expected, handle edge cases like null or empty inputs, and verify the sorting logic. Additionally, tests cover category-specific project validation, rating validation, and sorting by judges' points. The class also includes tests for correctly generating place strings and validating selected projects.

c) Test Output

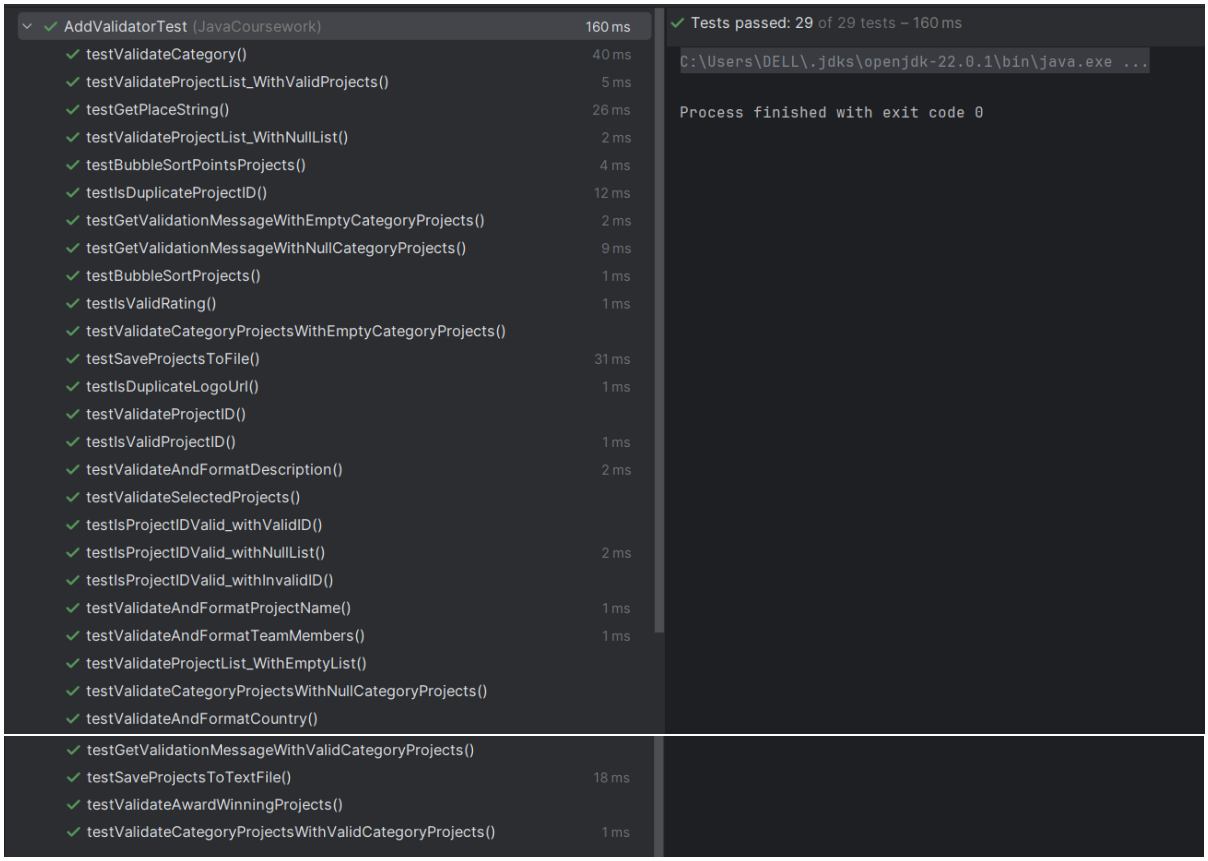
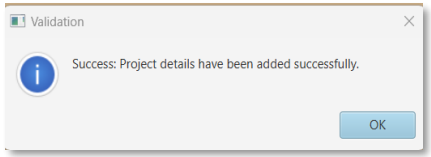
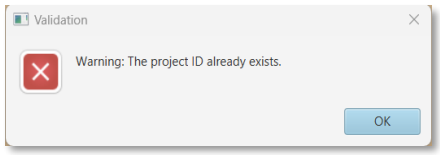
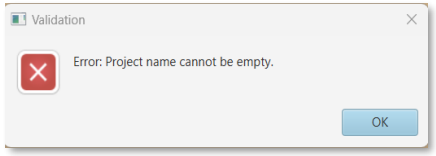
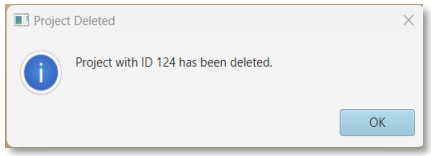
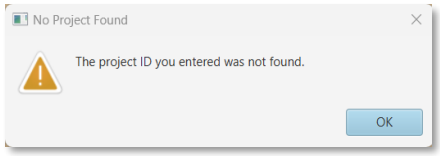
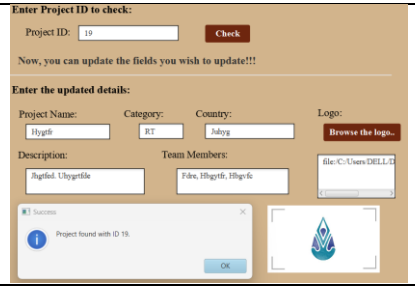
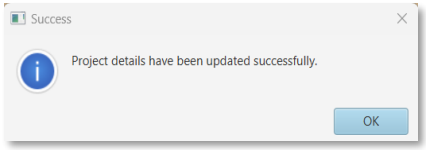
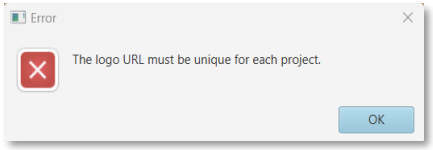
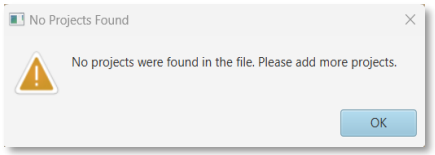
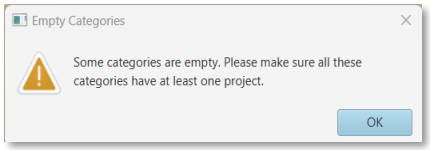


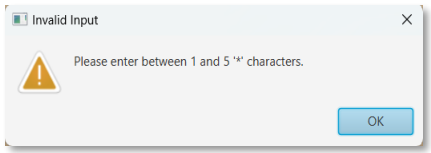
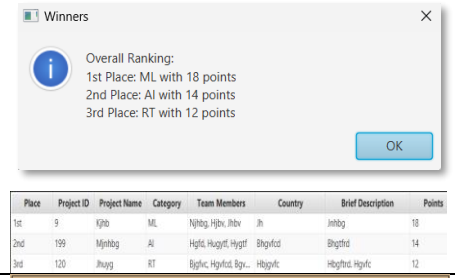
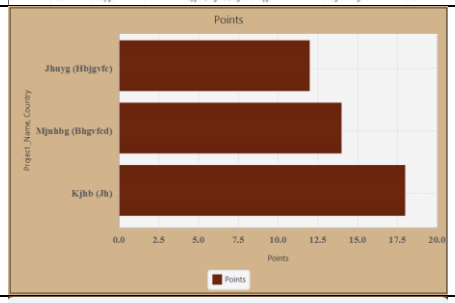
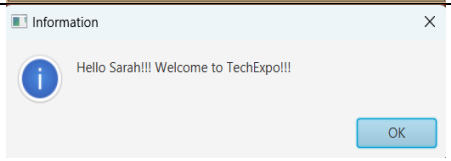
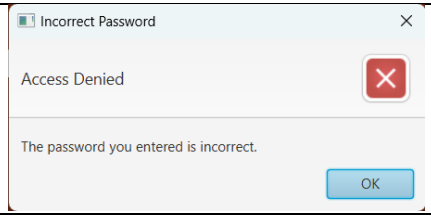
Figure 11: Junit Test Output

Test plans & Test cases

ID	Description	Pre-Condition	Test Steps	Test Inputs	Expected Results	Actual Result	Status
T1	Validate the projectID	Application is running, and the AddProjectDetail form is loaded.	Enter a valid positive integer into the proID field. Click Check button.	proID: 123	Alert displays "Validation" with a success message if the ID is valid and not a duplicate.		Success
T2	Verify handling of invalid projectID	Application is running, and proID contains invalid input	Enter invalid input in the proID field. Click the 'Check'button	proID: 'abc' / 0 / -1	An alert should display: "Invalid Input" with a message indicating invalid integer.		Success
T3	Add a new project wirh valid details	Application is running, and the AddProjectDetail form is loaded.	Fill in all required fields.	proID: 124 proName: 'Mack' proCategory: 'AI' proMembers: 'Alice', 'Bob', 'Charlie' proCountry: 'USA' proDescription: 'This is a new project.' logoTextView: 'APD.png'	Alert displays "Validation" with a success message, and the project is added to the list & saved to the text file.		Success
T4	Handle duplicate projectID	Application is running. A project with ID '125' already exists.	Enter exist ID into 'proID' field & click the "Check" button	proID: 125	Alert displays "Validation" with a warning message about the duplicate ID.		Success
T5	Validate & display error for empty fields	Application is running, and the AddProjectDetail form is loaded.	Leave required fields empty & click the "Submit" button	Empty fields	Alert displays "Validation" with error messages for each empty field.		Success

T6	Browse & load a logo image	Application is running, and the AddProjectDetail form is loaded.	Click the “Browse” button	Path to a valid image file.	Image file is displayed in logoImageView, and the file path is set in logoTextView.		Success
T7	Verify saving projects to a file	Application is running, and projectsList is populated with several Project objects.	Click the button for saving and check the content of the generated file.	filePath: "project_details.txt" projectsList: List containing projects	The file "project_details.txt" should contain sections for "AI", "ML", and "RT" & projects saved to it’s category.	The projects are saved into it’s relevant categories.	Success
T8	Verify successful deletion of a project	Application is running, list exist with projects and ‘proID’ contains a valid project ID to delete	Enter a valid ‘proID’in field. Click on button. Check the file to ensure the ID is removed.	‘proID’: A valid existing projectID	The project with the specified ID should be removed from file. An alert should display: "Project Deleted" with the project ID.	 Project has been deleted in the file.	Success
T9	Verify that a non-existent project ID is handled correctly	Application is running, projectsList exist with projects and ‘proID’ contains a non-existent projectID	Enter a non-existent project ID in the proID field. Click the delete button.	proID: A non-existent project ID	An alert should display: "No Project Found" with the project ID. The file project_details.txt should remain unchanged.	 Text file has been unchanged.	Success
T10	Verify successful field population	Application is running, projectsList exist with projects and ‘proID’ contains a valid project ID to delete	Enter a valid projectID in the proID. Click ‘Check’ button.	‘proID’: A valid existing projectID	An alert should display: "Success" with the project ID. Fields should be populated with the project details .		Success

T11	Verify successful project update.	Application is running, projectsList exists with projects, and proID contains a valid project ID.	Modify the project details Click the "Update" button & Check the file to ensure the project details are updated.	Fields: Only modify the fields need to Update.	An alert should display: "Success" with a message indicating the project details have been updated. Text file should reflect the updated project details.	 Text file has been updated with modified details.	Success
T12	Verify uniqueness of logo URL	Application is running, projectsList exists with projects, and proID contains a valid projectID.	Modify the logo URL to one that is already used. Click the "Update" button.	'Browse' button: Choose a existing image.	An alert should display: "Error" indicating that the logo URL must be unique. The project details in text file should remain unchanged.	 Text file has been unchanged.	Success
T13	Verify sorting of projects by 'projectID'	Application is running, and projectsList contains projects.	Observe the projects' sorting order in the table.	Projects with unsorted projectIDs in the list.	Projects should be sorted by projectID in ascending order in the table view.	The projects has been displayed in the table view with the sorted order of projectID	Success
T14	Verify handling of empty projectList	Application is running, and projectsList exists but is empty.	Observe the alert displayed for no projects found.	Empty projectsList	A warning alert should display: "No Projects Found" with an appropriate message.		Success
T15	Verify random project selection for each category.	categories list is populated with projects for each category.	Click the RSS button to call the random selection fuction.	Populated categories list with projects for categories "AI", "ML", and "RT".	The selectedProjects list should contain one random project for each category. Navigate to AWP.	A project has been selected for each category. It allows to rate the selected points.	Success
T16	Verify project selection when a category has no projects	categories list contains some categories with no projects.	Click the Start button to call navigation.	categories list with categories, some of which have no projects.	Alerts should be displayed for categories with no projects available.		Success

T17	Verify alert display for invalid input during scoring	Rating entered is invalid (not between 1-5 stars).	Enter an invalid rating in the scoring dialog.	Invalid ratings (0 stars, 6 stars, except stars as input).	An alert should be displayed with a warning message for invalid input.		Success
T18	Verify the ranking results sorted according to points.	Input the points for each selected projects. Observe the alert message and ranking results.	Enter the valid ratings for each project.	Valid ratings (1 star – 5 stars as input)	Projects should be sorted by judgesPoints and displayed in ranking order in the table view. An alert should be shown with the overall ranking of projects (1st, 2nd, 3rd).		Success
T19	Verify the bar chart displays the correct data for Project	topProjects is populated with specific projects.	Check if the bar chart reflects the data correctly.	topProjects containing known value for required details.	The bar chart should show bars with heights corresponding to the getJudgesPoints() values.		Success
T20	Confirm the presence of Sarah	None	Click the button for sarah. Enter the password to visit the interface belongs to Sarah.	Password: sarah123	The welcome message will display as “Hello Sarah, Welcome to TechExpo”		Success
T21	Validate the password	None	Enter the invalid password	Password: 12345Sarah	The alert message will display as “The password you entered is incorrect.”		

Summary

The JavaFX application is robust, maintainable, and adheres to core OOP principles. It features comprehensive error handling with try-catch blocks, rigorous input validation to prevent invalid data entry, and systematic unit and integration testing to ensure stable and resilient performance across various scenarios. Maintainability is achieved through a modular codebase, promoting separation of concerns and encapsulation. Clear naming conventions, inline comments, and Git-based version control enhance code readability and facilitate collaborative development.

Core OOP principles are exemplified with each class encapsulating specific functionalities. Encapsulation is enforced with private fields and methods, ensuring data integrity and controlled access. Abstraction is used to hide implementation details from higher-level modules, such as in 'MainEventLayout', which abstracts navigation functionalities. Polymorphism, through method overriding and interface implementations, provides flexibility and extensibility, allowing different classes to offer varying behaviors while adhering to a common interface.

The codebase follows SOLID principles: classes have single responsibilities (SRP), dependency injection (DIP) promotes loose coupling, and the Open/Closed Principle (OCP) ensures classes are open for extension but closed for modification. Polymorphism supports the Liskov Substitution Principle (LSP), enabling different behaviors through interface implementations. This design approach enhances software quality, making the application easier to understand, maintain, and extend.

Conclusion

The JavaFX application exemplifies a robust and maintainable design rooted in object-oriented programming principles and best practices. Each class is meticulously crafted to encapsulate specific functionalities, promoting modularity and ensuring clarity in code structure. Error handling and input validation mechanisms enhance robustness, guaranteeing a stable user experience across various interactions. Adherence to SOLID principles facilitates extensibility and scalability, laying a strong foundation for future enhancements.

Assumptions

Assume that the projectID and logoUrl must be unique for each project. Further UPD cannot be done for projectID as it is based on projectID. Assume that the event will only 3 categories (AI, ML, RT). The user can update the fields according to their wish. There no any restriction while updating, they can leave some fields without updating also. The finals will start, If only each category contains atleast one project. The APD, DPD, UPD cannot be done after starting of the event. But VPD can be access even after starting the finals (RSS, AWP, VAP). If only RSS happens, then only AWP & VAP will access else it will not allow to access.

References

Edureka. (2018, Oct 4). *Java OOPs Concepts*. Retrieved from YouTube: https://youtu.be/7GwptabrYyk?si=Otq71aeKlz6_XiBM

GeeksforGeeks, Sanchhaya Education Private Limited. (2024, June 27). *Java Tutorial*. Retrieved from GeeksforGeeks: https://www.geeksforgeeks.org/java/?ref=home-articlecards

Javatpoint. (2024). *JavaFx Tutorial*. Retrieved from Javatpoint: https://www.javatpoint.com/javafx-tutorial

Simplilearn. (2021, Mar 16). *Java OOPs Concepts*. Retrieved from YouTube: https://www.youtube.com/watch?v=6T_HgnjoYwM

Tutorials Point India Private Limited. (2024). *JavaFX Tutorial*. Retrieved from tutorialspoint: https://www.tutorialspoint.com/javafx/index.htm

W3schools. (2024). *Java Tutorial*. Retrieved from W3schools: https://www.w3schools.com/java/default.asp

Appendices

Figure 12: WelcomeScreen & SarahHomePage

Figure 13: APD & DPD

Figure 14: UPD & VPD

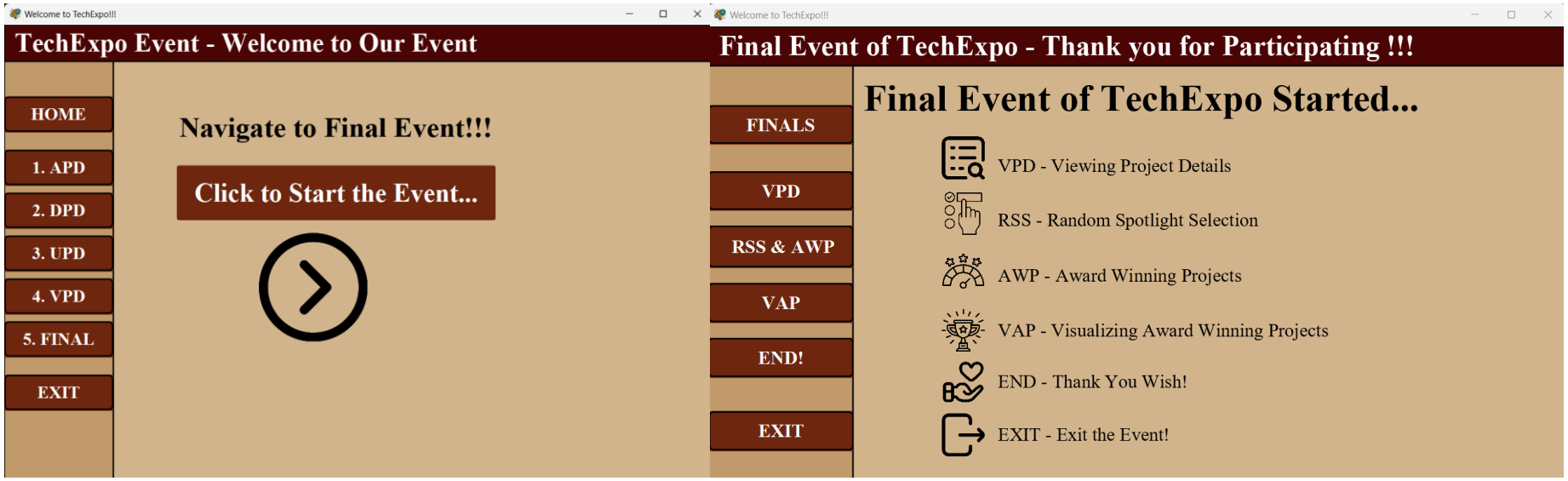


Figure 15: EventNav & EventHomePage

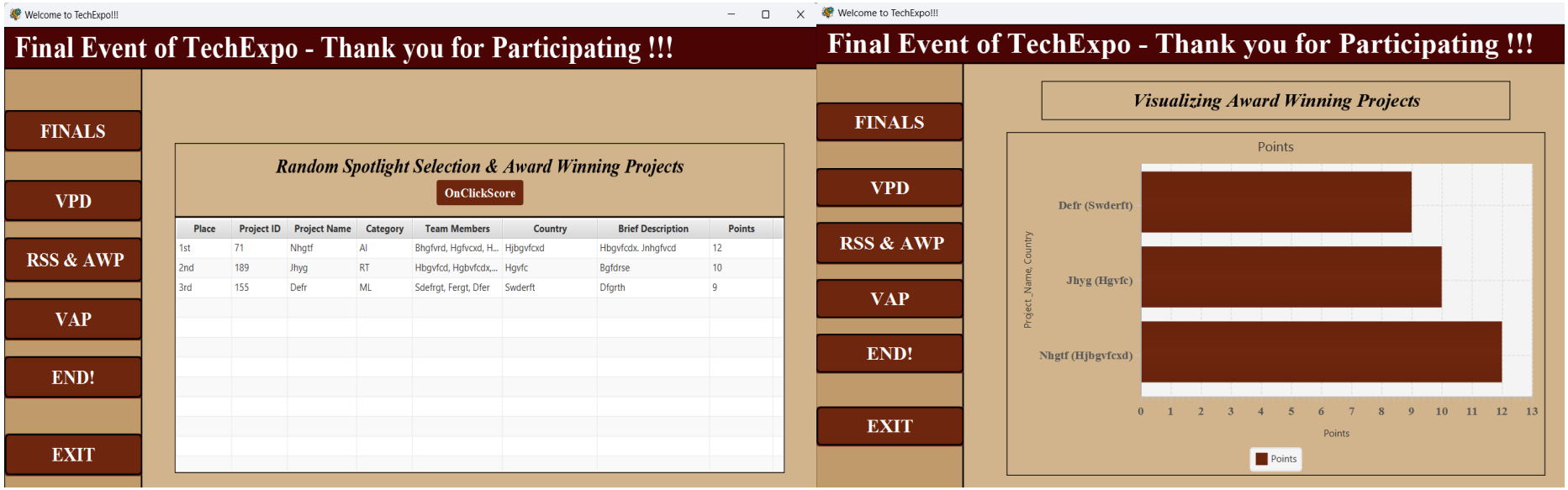


Figure 16: RSS & AWP & VAP