**PROJECTNAME**

**“Admission portal”**

**CLASS MAIN**

This JavaFX program creates a graphical user interface (GUI) for an **Admission Portal**. Here’s a breakdown of the key components and functionality:

**1. Class Declaration**

* public class Main extends Application:
  + The class Main extends the Application class, which is the entry point for JavaFX applications. The start method must be overridden to define the GUI.

**2. Window Setup**

* primaryStage.setTitle("Admission Portal");: Sets the title of the window (stage) to "Admission Portal".
* primaryStage.setMaximized(true);: Maximizes the window to full screen.

**3. Background Image**

* An Image object is created by loading an image from a resource (/welcome.jpg).
* ImageView backgroundImageView: This is used to display the image. The image is resized to fit the full width and height of the stage, while maintaining its aspect ratio.

**4. Text Elements**

* welcomeText: A large welcoming message "WELCOME TO COMSATS LAHORE." is displayed in a bold font, with black color and a white stroke.
* quoteText: A smaller quote message "The future belongs to those who believe in the beauty of their dreams." is displayed in blue with a white stroke.
* The texts are created using the Text class, and their font style, size, color, and stroke are set.

**5. Buttons**

* Two buttons are created:
  + **Proceed Further Button**: When clicked, the current stage is closed, and a new stage for the login page (LoginPage) is opened.
  + **Admin Login Button**: This button opens a TextInputDialog for the user to enter an admin password. If the correct password ("admin123") is entered, it opens a new stage for viewing all students (ViewAllStudents). If the password is incorrect, an error message is displayed in an alert box.

**6. Button Styling**

* A styleButton method is used to style the buttons. The button's background is orange (#FFA500), and the text is black. Additionally, it has a hover effect where the background changes to red when the mouse hovers over the button.
* The buttons also have a border and padding to make them visually appealing.

**7. Button Layout**

* HBox buttonBox = new HBox(20, adminButton, nextButton);: Both buttons are placed in a horizontal box (HBox), with a spacing of 20 pixels between them.
* buttonBox.setAlignment(Pos.CENTER);: Centers the buttons within the box.
* buttonBox.setStyle("-fx-background-color: #333333;");: Sets the background color of the box to charcoal gray.

**8. Main Layout**

* A BorderPane layout is used to organize the components:
  + root.setTop(new StackPane(welcomeText));: The welcome text is placed at the top.
  + root.setCenter(new StackPane(quoteText));: The quote is placed in the center.
  + root.setBottom(buttonBox);: The button box is placed at the bottom.
* StackPane stackPane = new StackPane(backgroundImageView, root);: The background image and the main content (BorderPane) are layered on top of each other using a StackPane.

**9. Scene and Stage Setup**

* A Scene is created with the stackPane as its root. The scene is then set on the primary stage (primaryStage.setScene(scene);), and the stage is displayed (primaryStage.show();).

**10. Helper Methods**

* **styleButton(Button button)**: This method defines the visual style of the buttons and handles the hover effects (changes the button color when the mouse hovers over it).
* **showAlert(String title, String message)**: This method is used to show an error alert with a custom title and message.

**11. Main Method**

* launch(args);: This method launches the JavaFX application, which will invoke the start method to display the UI.

**Summary:**

* The program creates a full-screen admission portal window with a background image.
* It displays a welcome message and a motivational quote, along with two buttons:
  + One for proceeding to the login page.
  + Another for admin login, which checks for a correct password before allowing access to the admin page.
* The buttons are styled with colors and hover effects, and the layout is organized using JavaFX layout managers like BorderPane and HBox.

**Class BS**

**Package and Imports**

The code is part of the package com.example.oopproj2 and utilizes JavaFX libraries to create a graphical user interface (GUI). Necessary imports include components for layout, controls, and the Application class for launching the JavaFX application.

**Class Overview**

The Bs class extends the JavaFX Application class and implements a program selection interface. It displays various educational program options, with buttons for navigation. The start method initializes and configures the primary stage, serving as the application's main window.

**Primary GUI Setup**

The start method organizes the GUI into distinct sections using a BorderPane layout. These sections include:

1. **Main Container**  
   A BorderPane is used as the root layout for the application. The background is styled with a charcoal color, creating a clean and professional look.
2. **Title Section**  
   A Label positioned at the top of the layout serves as the title, with bold, white text styled for emphasis. The title is centrally aligned with additional padding for aesthetic spacing.
3. **Program Buttons (Center Layout)**  
   Program selection buttons are arranged in two rows using vertical (VBox) and horizontal (HBox) layouts. Each button represents a specific program, such as "Computer Science" or "Electrical Engineering." Buttons are uniformly styled and programmed to navigate to an informational page when clicked.
4. **Navigation Buttons (Bottom Layout)**  
   A "Back" button is placed at the bottom to allow users to return to the previous stage. This button features hover effects for enhanced user interaction and visual feedback.

**Helper Methods**

1. **Button Creation**  
   A dedicated method standardizes button creation, ensuring consistency in styling, size, and functionality. This promotes code reusability and simplifies maintenance.
2. **Page Navigation**  
   A navigation method opens new windows corresponding to the selected program while closing the current stage. This ensures a seamless transition between different parts of the application.

**Features Summary**

* **Dynamic Styling**  
  Buttons incorporate hover effects, changing their appearance to provide interactive feedback to the user.
* **Modular Layout**  
  The use of intuitive layouts (BorderPane, VBox, HBox) organizes components effectively for a clear interface.
* **Reusability**  
  The application employs a helper method to create buttons, ensuring uniformity across the interface.
* **User Navigation**  
  The program allows smooth transitions between windows, enhancing the overall user experience.

**Potential Improvements**

1. **CSS Adjustments**  
   Fix minor issues like correcting the CSS property name from -fx-background-colour to -fx-background-color for compatibility.
2. **Exception Handling**  
   Add error handling for potential failures in the Selection and SchoolingInfoPage stages to improve robustness.
3. **Responsive Design**  
   Optimize the GUI for varying screen resolutions by dynamically adjusting button sizes and layout spacing.

**CLASS DATABASE**

**Purpose**

The database class provides core functionality for database connectivity and query execution in the application. It focuses on connecting to a MySQL database, executing secure queries for login validation, and handling insert or update operations.

**Class Overview**

The database class consists of three static methods:

1. **Database Connection**  
   The connectDB method establishes a connection with the MySQL database using JDBC.
2. **Login Query Execution**  
   The executePreparedQuery method validates user login credentials by running a parameterized query to check the email and password against the database.
3. **Write Operations**  
   The executeWriteQuery method runs queries for updating or inserting data into the database.

**Key Components**

1. **Database Connection (connectDB)**  
   This method sets up a connection to the MySQL database comsats\_admission using the DriverManager. It requires the database URL, username, and password. If the connection fails, it catches and prints the exception stack trace.
2. **Secure Query for Login (executePreparedQuery)**  
   This method ensures secure handling of login credentials. It performs the following steps:
   * Establishes a database connection.
   * Uses a parameterized SQL query to prevent SQL injection attacks.
   * Binds the user-provided email and password to the query parameters.
   * Executes the query and returns the result set, which can be used to verify if the user exists.
3. **Write Query Execution (executeWriteQuery)**  
   This method handles SQL queries that modify database records. It accepts a query string, establishes a connection, and executes the query using a Statement. Any errors during execution are handled by throwing a runtime exception.

**Features**

1. **Centralized Database Connection**  
   All methods rely on a single connection mechanism, ensuring consistent and reusable database access.
2. **Secure Authentication**  
   The executePreparedQuery method uses prepared statements to avoid SQL injection, a common security vulnerability.
3. **Simplified Write Operations**  
   The executeWriteQuery method allows seamless execution of SQL commands like INSERT or UPDATE, making it versatile for handling database modification

**Interactive Functionality**

* **Login Validation**  
  The executePreparedQuery method takes user-provided email and password inputs, checks them against the database, and returns a result set indicating whether the credentials are valid.
* **Data Insertion/Update**  
  The executeWriteQuery method executes SQL commands, enabling features like adding new users or updating existing records..

**Conclusion**

The database class is a vital utility for managing database operations within the application. Its modular design simplifies database interactions while adhering to basic security principles. However, improvements in resource management, logging, and configuration can make the implementation more robust and scalable.

**CLASS INTERMEDIATE INFO PAGE**

### ****Purpose****

The IntermediateInfoPage class represents a JavaFX-based GUI application for collecting and validating intermediate-level academic information from users. It incorporates form elements, input validation, and basic navigation.

### ****Class Overview****

This class extends the Application class provided by JavaFX and overrides the start method to define the user interface. It contains various UI components such as labels, text fields, combo boxes, and buttons, arranged in a clean and organized layout

### ****GUI Design****

### **Root Layout**

* **Container Type**: BorderPane
  + The main layout organizes the content into distinct sections (e.g., title, form, and buttons).
  + A dark background color (#333333) is applied to the entire window for a modern appearance.

### ****Title****

* Positioned at the top using a VBox layout.
* Styled with a light gray font color and bold size (Arial, 24).

### ****Form Section****

* **Layout**: GridPane
  + A grid-based layout ensures a neat alignment of labels and input fields.
  + Padding, horizontal gaps, and vertical gaps are used for spacing.
* **Form Fields**:
  + **Year of Study**: A text field for entering the academic year.
  + **Field of Study**: A combo box with predefined options (Pre-Medical, Pre-Engineering, ICS).
  + **Gender**: A combo box with Male and Female options.
  + **Marks**: A text field for entering total marks obtained out of 1200.
  + **Grade**: A text field for specifying the grade (editable).
  + **Exam Board Name**: A text field for the name of the examination board.
  + **College Name**: A text field for the name of the college.

### ****Buttons Section****

* **Layout**: HBox with center alignment.
* **Buttons**:
  1. **Back**: Navigates to the previous page or closes the current stage.
  2. **Submit**:
     + Validates user inputs to ensure required fields are filled.
     + Opens the main page upon successful validation.

### ****Styling****

* **Labels**: Styled using a helper method with consistent font (Arial, 16) and light gray color.
* **Text Fields**: Styled for uniform width.
* **Combo Boxes**: Styled with predefined options and a uniform width.
* **Buttons**:
  + Styled with a golden background color (#f39c12) and rounded corners.
  + Mouse hover effects change the button color to red for interactive feedback.

### ****Functionality****

1. **Input Validation**
   * Checks if mandatory fields are filled (e.g., Year of Study, Marks, Exam Board Name, College Name).
   * Displays an error alert if validation fails.
2. **Navigation**
   * The **Back** button closes the current stage.
   * The **Submit** button validates inputs and, on success, navigates to the main page.
3. **Dynamic Alerts**
   * Alerts notify the user about missing fields or errors.

### ****Helper Methods****

* createStyledLabel(String text): Creates styled labels with a consistent appearance.
* createStyledTextField(): Creates styled text fields with a defined width.
* createStyledComboBox(String... items): Creates combo boxes with predefined options.
* createStyledButton(String text, EventHandler<ActionEvent> action): Creates buttons with consistent styling and interactive hover effects.
* showAlert(String title, String message): Displays error alerts to the user.

### ****Features****

1. **User-Friendly Interface**:
   * Clean layout and logical grouping of related fields.
   * Hover effects on buttons for better usability.
2. **Validation**:
   * Prevents submission of incomplete forms, ensuring data integrity.
3. **Dynamic Navigation**:
   * Seamless page transitions using JavaFX's Stage.
4. **Reusability**:
   * Modular methods for creating UI components enable consistency and reusability.

### ****Potential Improvements****

1. **Enhanced Validation**:
   * Add specific validation for numeric fields (e.g., Marks).
   * Ensure Year of Study is a valid year.
2. **Custom Themes**:
   * Use an external CSS file for easier maintenance of styling.
3. **Error Handling**:
   * Handle exceptions (e.g., invalid inputs) with detailed error messages.
4. **Internationalization**:
   * Add support for multiple languages for broader usability.

### ****Conclusion****

The IntermediateInfoPage class provides a visually appealing and functional interface for collecting user information. It effectively uses JavaFX layouts and components to create a robust and maintainable application.

**CLASS LOGIN PAGE**

The LoginPage class is a JavaFX application that provides a graphical user interface for user login functionality. It includes fields for email and password, buttons for login and signup, and database authentication to validate user credentials.

**Main Container**

The main layout is a BorderPane, which serves as the root container. It is styled with a dark background color (#333333) for a modern look. The center of the layout contains a VBox for organizing the login form.

**Login Form**

The login form consists of:

* **Title:** A bold label that reads "Login to Your Account" with a large font size (28) and white text.
* **Email Field:** A TextField styled with a dark gray background, white text, and placeholder text in light gray.
* **Password Field:** A PasswordField styled similarly to the email field.
* **Buttons:** Two buttons for "Login" and "Signup," styled with orange backgrounds and interactive hover effects that change the color to red.

**Button Functionality**

* **Login Button:**
  + Validates that both the email and password fields are filled.
  + Calls a method from the database class to query the database using the provided credentials.
  + Displays a success message if credentials are valid and navigates to the next page.
  + Displays an error message if credentials are invalid.
* **Signup Button:**
  + Hides the current login page.
  + Opens a new signup page where users can create an account.

**Styling**

Helper methods ensure consistent styling:

* **Text Fields:** The styleTextField method applies a uniform design, including a dark background, white text, and padding for all text fields.
* **Buttons:** The styleButton method applies hover effects and changes the button's background color, providing visual feedback.

**Alerts**

The showAlert method displays pop-up messages to inform the user about errors or successful actions. It accepts a title, message, and type of alert (e.g., error, information, warning).

**Database Integration**

The login functionality integrates with the database class. It uses a prepared statement to securely check user credentials in the database.

**Application Launch**

The main method launches the JavaFX application. The start method initializes the UI components and displays the login form for user interaction.

Users can log in with their credentials or navigate to the signup page to create a new account.

**CLASS SCHOOLING INFO**

This JavaFX program creates a **Schooling Information Page** where users can input various details such as personal information, guardian details, and schooling information. Below is a detailed breakdown of the components and functionality:

**1. Class Declaration**

* public class SchoolingInfoPage extends Application:
  + This class extends Application, making it the entry point for the JavaFX application. The start method is overridden to define the user interface (UI).

**2. Window Setup**

* primaryStage.setTitle("School Info Page");: Sets the title of the window (stage) to "School Info Page".
* primaryStage.setMaximized(true);: Maximizes the window to full screen.

**3. Root Container**

* BorderPane root = new BorderPane();: A BorderPane is used to manage the layout of the UI components.
* root.setBackground(...): The background of the root pane is set to a dark gray color (#333333).

**4. Title Section**

* Label titleLabel = new Label("Enter Schooling Information");: A label displaying the page title.
* HBox titleBox: A horizontal box is used to center the title label. Padding is applied around the title.

**5. Form Layout (GridPane)**

* GridPane formGrid = new GridPane();: A GridPane is used to arrange the form elements in a grid. This layout includes labels and input fields for various pieces of information.
  + **Hgap & Vgap**: Horizontal and vertical gaps between the grid cells.
  + **Padding**: Padding around the grid to provide spacing.
  + **Alignment**: Aligns the form content to the top center.
* **Form Elements**:
  + **TextField Elements**: Input fields for applicantNameField, contactField, emailField, guardianNameField, guardianContactField, guardianEmailField, schoolNameField, and cnicField are created using the helper method createLabeledField.
  + **ComboBox for Date of Birth**: Three combo boxes are used for selecting the day, month, and year of birth:
    - dayComboBox: A combo box for selecting the day (1 to 31).
    - monthComboBox: A combo box for selecting the month (January to December).
    - yearComboBox: A combo box for selecting the year (1980 to 2026).
  + **Gender Dropdown**: A combo box for selecting gender (Male, Female, Other), created using the createLabeledDropdown method.

**6. Helper Methods**

* **createLabeledField**: This method creates a labeled text field. It takes a GridPane, a label text, column and row positions, label font, and field font as parameters.
* **createLabeledDropdown**: This method creates a labeled dropdown (combo box). It accepts a GridPane, label text, column and row positions, font, and options (gender options in this case).
* **createStyledLabel**: This method creates a Label with a specific font and white text color.
* **createStyledButton**: This method creates a button with a given text and background color. It also defines hover effects where the background color changes when the mouse hovers over the button.

**7. Buttons**

* Two buttons are created:
  + **Back Button**: When clicked, it closes the current window (page).
  + **Next Button**: When clicked, it:
    1. Constructs a SQL query to insert the user's data into the database (though the database.executeWriteQuery(query) method is used, it is assumed that the database object is defined elsewhere).
    2. Opens a new IntermediateInfoPage and closes the current stage.

**8. Event Handlers**

* **Back Button**: Closes the current page (stage).
* **Next Button**: Executes the SQL query to store the data and transitions to the next page (opens the IntermediateInfoPage).

**9. Final Layout**

* VBox mainLayout = new VBox(10, titleBox, formGrid, buttonBox);: A vertical box (VBox) arranges the title, form grid, and button box in a vertical stack.
* **Centering**: The entire layout is centered using mainLayout.setAlignment(Pos.CENTER);.

**10. Scene and Stage Setup**

* Scene scene = new Scene(root);: A Scene is created with the root (which contains all the UI elements) as its root node.
* primaryStage.setScene(scene);: Sets the scene to the primary stage.
* primaryStage.show();: Displays the stage.

**11. Main Method**

* launch(args);: Launches the JavaFX application, which will invoke the start method to display the UI.

**Summary:**

This program creates a user interface for inputting schooling-related information:

* Personal details (name, contact, email, etc.).
* Date of birth and gender selection.
* Guardian's contact information and schooling details.
* The form data can be submitted by clicking the "Next" button, which triggers a database insertion (assuming a database connection is established).
* The UI is responsive, styled with custom fonts, colors, and layouts, and uses JavaFX layout components such as BorderPane, GridPane, HBox, and VBox.

**CLASS SELECTION**

This JavaFX program creates a **Program Selection Page** with two buttons for selecting a program (in this case, "BS Program") or going back to the login page. Here’s a detailed breakdown of the code:

**1. Class Declaration**

* public class Selection extends Application:
  + This class extends the Application class, meaning it is the entry point for the JavaFX application. The start method is overridden to define the layout and behavior of the UI components.

**2. Window Setup**

* primaryStage.setTitle("Program Selection");: Sets the title of the window (stage) to "Program Selection".
* VBox layout = new VBox(30);: Creates a vertical layout (VBox) where the components are arranged vertically with a 30px gap between them.
* layout.setAlignment(Pos.CENTER);: Aligns all elements inside the layout to the center.
* layout.setStyle("-fx-background-color: #333333;");: Sets the background color of the layout to a charcoal gray (#333333).

**3. Title Section**

* Label titleLabel = new Label("Select Your Program");: A label is created with the text "Select Your Program".
* titleLabel.setStyle("-fx-font-size: 32px; -fx-font-weight: bold; -fx-text-fill: white;");: The label is styled with a large font size (32px), bold weight, and white text color.

**4. Buttons (Program Selection and Back)**

* **BS Program Button (bsButton)**:
  + A button is created for selecting the "BS Program" with the label "BS Program".
  + Styling is applied to the button:
    - Background color: gray (-fx-background-color: gray).
    - Text color: black (-fx-text-fill: black).
    - Font size: 18px (-fx-font-size: 18px).
    - Padding: 10px top and bottom, 20px left and right (-fx-padding: 10px 20px).
  + **Mouse Hover Effects**: The background color changes to white when the mouse hovers over the button, and returns to gray when the mouse exits.
  + bsButton.setOnAction(...): The button’s action is to close the current stage (window) and open the BS program page by calling new Bs().start(new Stage());.
* **Back Button (backButton)**:
  + A button is created for navigating back to the login page with the label "Back".
  + Styling is similar to the BS Program button but with an orange background (-fx-background-color: orange).
  + **Mouse Hover Effects**: The background color changes to red when the mouse hovers over the button, and returns to orange when the mouse exits.
  + backButton.setOnAction(...): The button’s action is to close the current stage and open the login page by calling new LoginPage().start(new Stage());.

**5. Adding Content to Layout**

* layout.getChildren().addAll(titleLabel, bsButton, backButton);: Adds the title label, BS program button, and back button to the layout.

**6. Scene Setup**

* Scene scene = new Scene(layout, 1600, 1000);: Creates a new scene with the layout, setting the initial window size to 1600x1000 pixels.
* primaryStage.setScene(scene);: Sets the created scene to the primary stage.
* primaryStage.setMaximized(true);: Maximizes the window so that it fills the screen.
* primaryStage.show();: Displays the stage (window) with the scene.

**7. Main Method**

* launch(args);: Launches the JavaFX application, triggering the start method to display the UI.

**Summary:**

* The program creates a **Program Selection Page** with two buttons:
  + **BS Program**: Opens the BS program page when clicked.
  + **Back**: Takes the user back to the login page.
* The layout uses a vertical box (VBox) with a centered alignment, and the window is maximized with a charcoal gray background.
* Styling is applied to buttons and labels, including mouse hover effects to change button colors. The application is designed to provide a simple user interface for navigating between pages in a program selection context.

**CLASS SIGNUP PAGE**

The SignupPage class is a JavaFX application for creating a user sign-up page. It allows users to register by filling out a form with their name, email, password, and date of birth. The user can also navigate back to the login page. Below is a breakdown of the code and its functionality:

**1. Class and Constructor:**

* SignupPage extends Application: The class extends Application, making it the entry point for the JavaFX application.
* The constructor public SignupPage(Stage loginStage) accepts a Stage object representing the login page, which is used for navigation back to the login page once the user signs up.

**2. UI Layout:**

* **Main Container** (BorderPane):
  + The root layout is a BorderPane, which is a container that can hold UI components in the top, bottom, left, right, and center.
  + Background color is set to charcoal (#333333).
* **GridPane**:
  + The form layout is created using GridPane, which arranges the input fields in a grid-like structure. The form is centered, and gaps between the rows and columns are defined.

**3. Form Fields:**

* **Title**: A label Signup for an Account is displayed at the top of the form.
* **Name**: A text field is provided for the user to enter their name.
* **Email**: A text field for the user to input their email.
* **Password**: A password field for the user to input a password (password will be masked).
* **Date of Birth**: A date of birth field consists of three combo boxes for day, month, and year. The items for each combo box are populated with valid values (days 1-31, months 1-12, and years 2000-2026).

**4. Buttons:**

* **Sign Up Button**: This button validates the form fields when clicked. If any field is empty, an error message is shown. Otherwise, the form data is inserted into a database (students table), and a success message is shown. After successful registration, the sign-up page closes, and the login page is shown.
* **Back Button**: When clicked, this button closes the sign-up page and returns to the login page.

**5. Styling:**

* styleTextField(TextField): This method applies consistent styling to text fields, including a font size, background color, text color, and padding.
* styleComboBox(ComboBox): Similar styling is applied to the combo boxes used for date of birth.
* styleButton(Button): Buttons are styled with background color, text color, font size, and padding. Mouse hover effects are added to change the background color when the user hovers over a button.

**6. Alert Handling:**

* showAlert(Alert.AlertType, String, String): This method shows alert dialogs to notify the user of success or error messages. It is used for showing form validation errors or database operation results.

**7. Scene and Stage Setup:**

* A Scene with the root layout (BorderPane) is created and set on the primary stage.
* The stage is maximized to fill the screen.
* The primaryStage.show() method is called to display the stage.

**8. Main Method:**

* The main method launches the JavaFX application by calling launch(args).

**Key Points:**

* **Form Validation**: Ensures all fields are filled before submitting the form.
* **Database Operation**: Attempts to insert the user data into a database table (students), handling any exceptions and showing appropriate alerts.
* **Navigation**: Allows the user to return to the login page after signing up.

**Suggestions for Improvement:**

* **Database Connection**: The database operation (database.executeWriteQuery(...)) assumes that a database object is available but is not defined in the provided code. You may need to include a database connection setup for this to work.
* **Error Handling**: You can enhance error handling by checking the validity of email format and password strength.

This is a clean and user-friendly implementation for a sign-up page that provides essential features for user registration, form validation, and smooth navigation.

**CLASS VIEW ALL STUDENT**

The ViewAllStudents class is a JavaFX application designed to display a list of students from a database in a table format. It fetches data from a MySQL database and presents it in a TableView. Here's an explanation of the code structure and key functionalities:

**1. Class Structure:**

* ViewAllStudents extends Application: The class extends Application, making it the entry point for the JavaFX application.
* **Database Connection**: The class connects to a MySQL database to fetch student data and populate the table.
* **Inner Student Class**: This class represents the Student object that is displayed in the table.

**2. UI Components:**

* **Root Layout**: A BorderPane is used as the root layout for the application. It is styled with padding and a black background.
* **TableView**: The TableView<Student> component is used to display the list of students. The table is styled with an orange background and centered text. The table is set to automatically resize columns.

**3. Table Columns:**

The TableView consists of five columns:

* Student Name
* Student Number
* Student Email
* Student CNIC
* Student Degree

Each column is created using the createColumn() method, which sets the title and the property of the Student object to display the respective data. The columns are centered and have a white text color with a black background.

**4. Fetching and Displaying Data:**

* The fetchAndDisplayData() method retrieves student data from the database using a SELECT query and populates the table with the result. Each student's information is fetched from the admission table and added to the TableView.
* If the database query fails or encounters an issue, an error message is displayed using the showAlert() method.

**5. Database Class:**

* The Database class is a helper class with a static method executeReadQuery() that is used to execute the SQL read query and return a ResultSet.

**6. Error Handling:**

* If there is an error fetching the data from the database, an error message is displayed using a showAlert() method. This method creates an alert dialog with the error message.

**7. Scene and Stage Setup:**

* A Scene with the root layout (BorderPane) is created and set on the primary stage.
* The stage is displayed using primaryStage.show().

**8. Main Method:**

* The main() method launches the JavaFX application by calling launch(args).

**9. Student Class:**

* The Student class has five properties (studentName, studentNumber, studentEmail, studentCnic, studentDegree), each with a corresponding getter method.
* Instances of the Student class are added to the TableView to display student information.

**10. Database Connection:**

* The application connects to a MySQL database using DriverManager.getConnection(). The connection string specifies the database (comsats\_admission), username (root), and password (umair@2004).

**Suggestions for Improvement:**

* **Database Connection Handling**: The database connection in the Database class could be improved by closing the Connection, Statement, and ResultSet objects properly to prevent resource leaks. A finally block or try-with-resources could be used to ensure resources are closed after use.
* **Error Messages**: The error handling in the fetchAndDisplayData() method only prints the error to the console. You could expand the error messages in the alert dialog to give more detailed feedback to the user.
* **Security**: Avoid hardcoding database credentials (like username and password) directly in the code. You can use environment variables or configuration files to store sensitive information securely.

**Key Functionalities:**

* Display a list of students fetched from the database in a table format.
* Allow resizing and customization of table columns.
* Handle database errors and display alerts when necessary.
* Provide a clear and user-friendly interface with a clean design using JavaFX.