




Application Program Development

Segment : Interfaces and Implementations

Presenter Name

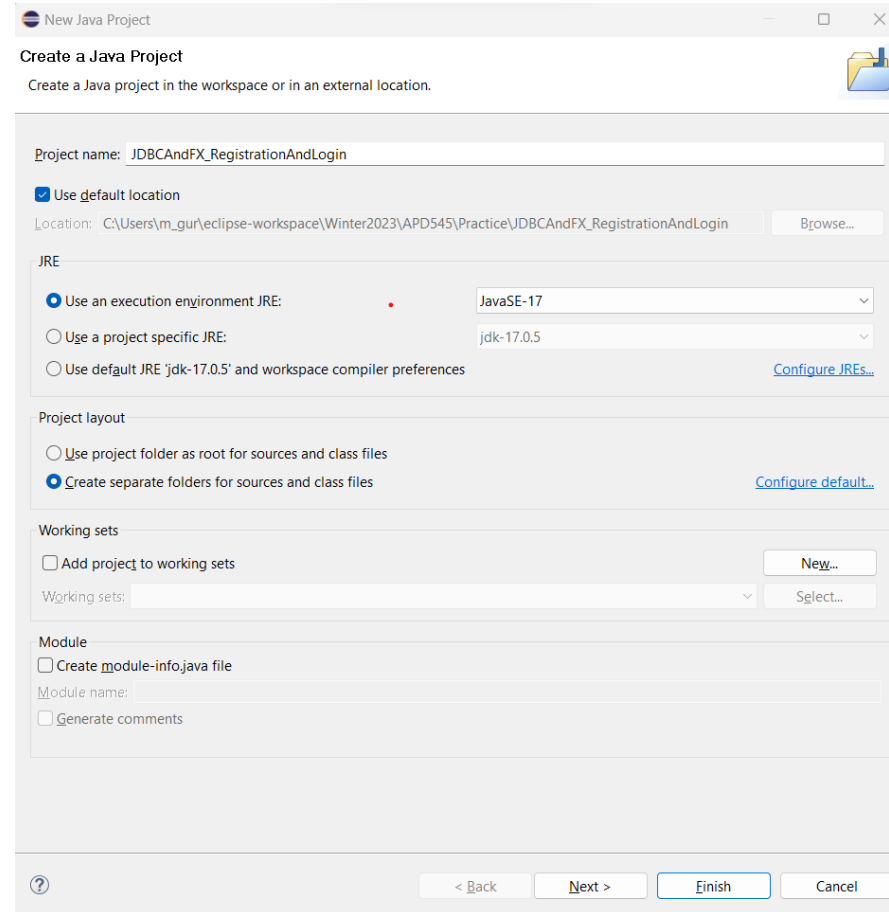


Outcomes

- JDBC with FX example using MySQL Connector.
 - Creating layouts of JavaFX
 - Adding UI controls
 - Validating controls
 - Connecting with MySQL via JDBC API
- 

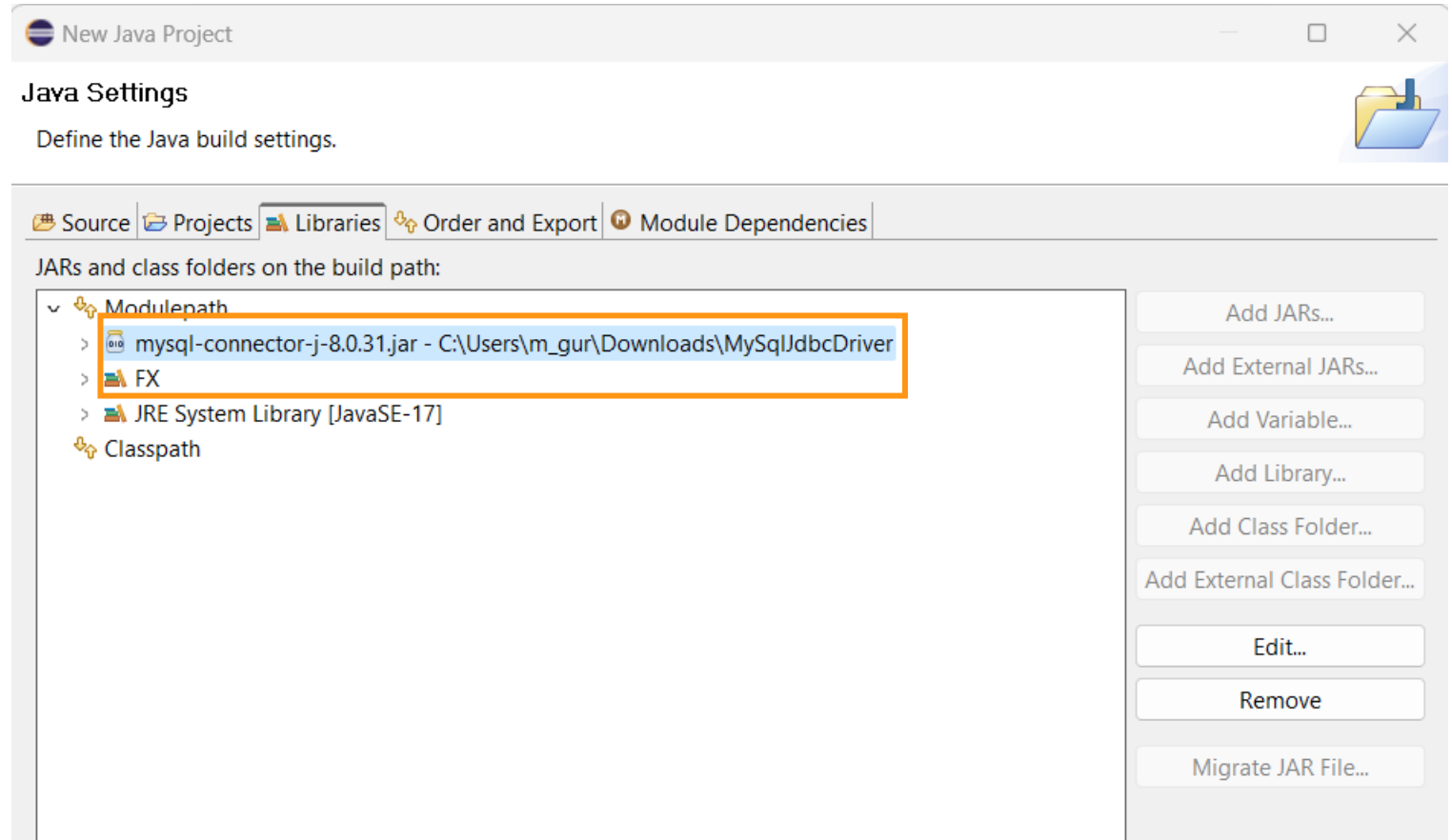
Step – 1 : Creating Project

- Create a JavaFX project using eclipse

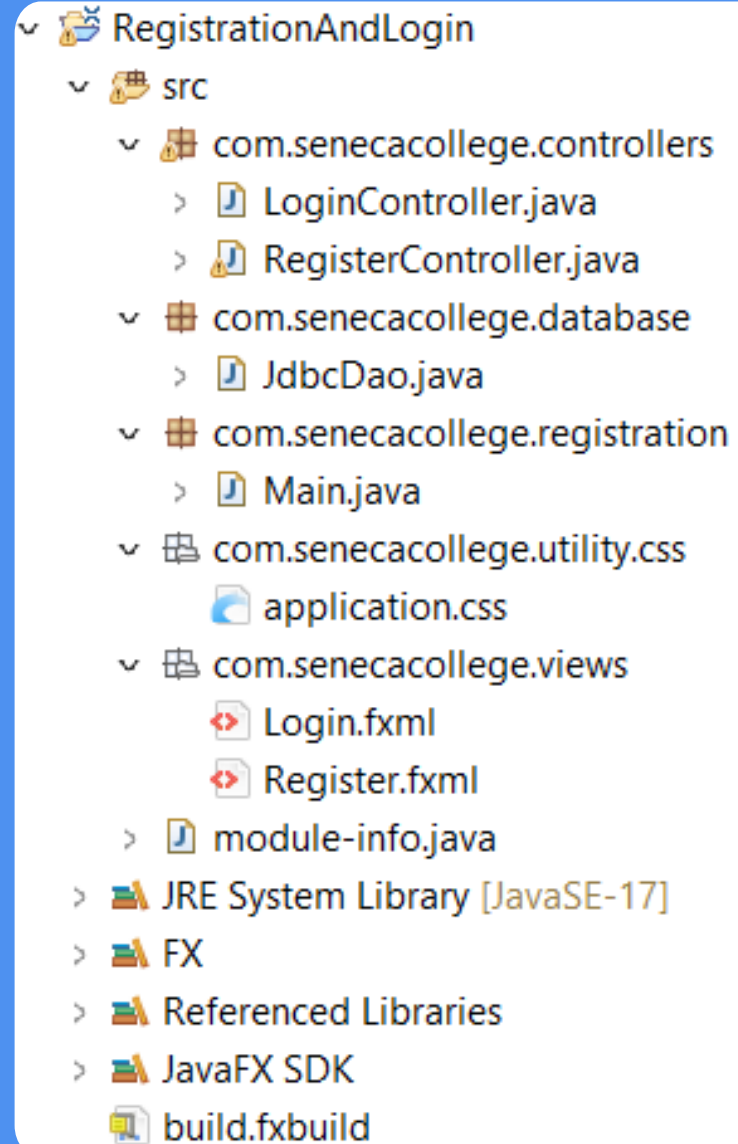


Step – 2 : Adding Required Libraries

- Add JavaFx libraries required for the project.
- Download the [MySQL JDBC Connector](#) and add it to the project as well.



Project Structure – Once completed



Step – 3 : Database Setup

- Make sure you have MySQL installed on your machine.
- First create the database using the following MySQL statement

```
create database registrationloginFX;
```

- Next create a table in the created database like

```
CREATE TABLE registration (  
    id BIGINT PRIMARY KEY AUTO_INCREMENT,  
    full_name varchar(250) NOT NULL,  
    email_id varchar(250) NOT NULL,  
    password varchar(250)  
);
```

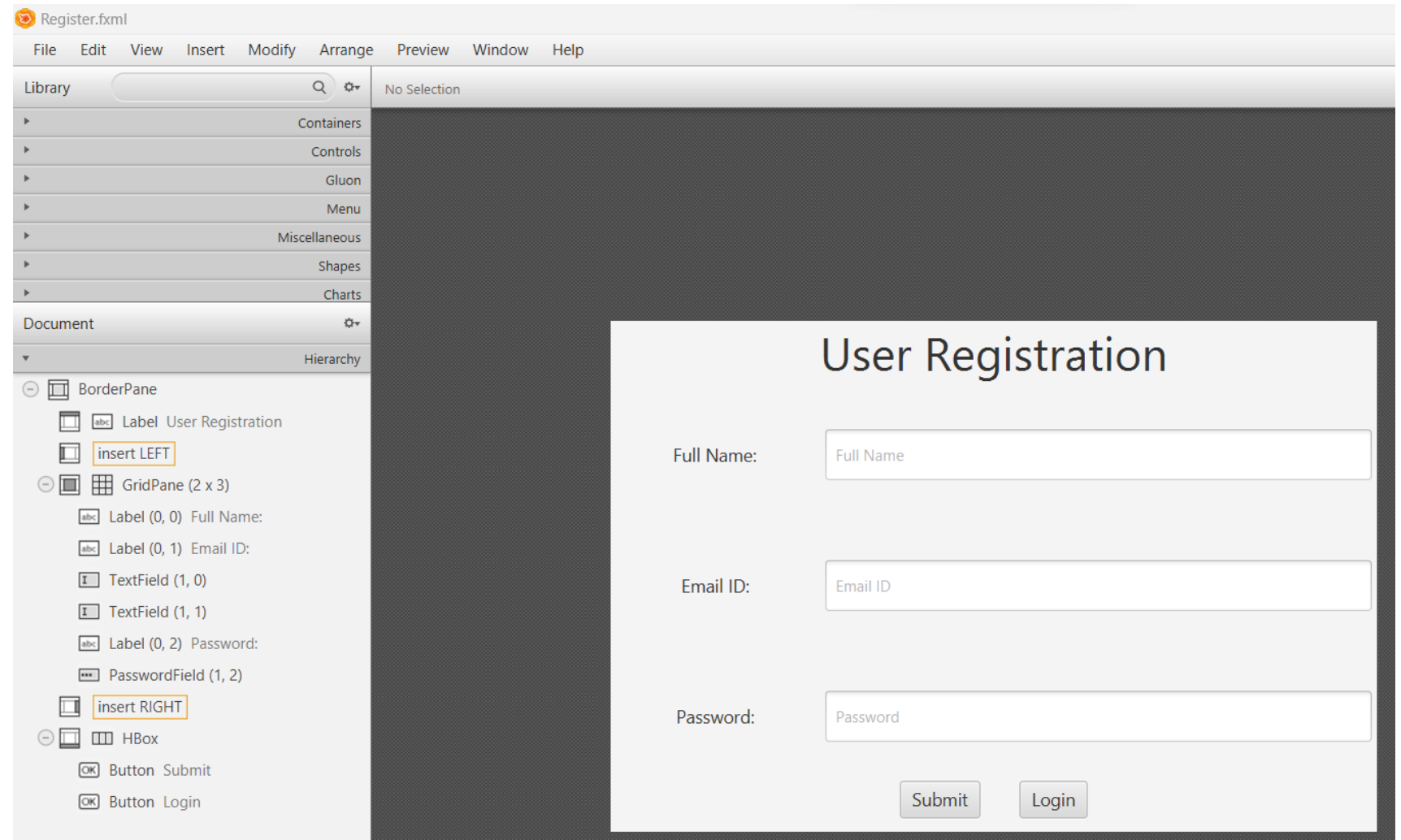
Step – 5 : Main.java class

- As before, for creating a JavaFX application, we'll need to extend our Main class from `javafx.application.Application` and override its `start()` method.

```
public class Main extends Application {
    @Override
    public void start(Stage primaryStage) {
        try {
            BorderPane root =
                (BorderPane) FXMLLoader.load(getClass()
                    .getResource("/com/senecacollege/views/Register.fxml"));
            primaryStage.setTitle("User Registration");
            Scene scene = new Scene(root, 800, 500);
            primaryStage.setScene(scene);
            primaryStage.show();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Step – 6 : Creating Registration.fxml

- Open Registration.fxml in the views folder in the scene builder and create the layout.



Step – 7 : Creating Registration Controller to Handle Data

- Open the RegistratoinController.java from the controller's folder.

@FXML

```
void submitButton(ActionEvent event) throws SQLException {
    Window owner = submitButton.getScene().getWindow();
    if (fullNameText.getText().isEmpty()) {
        showAlert(Alert.AlertType.ERROR, owner, "Form Error!", "Please enter your name");
        return;
    }
    if (emailidText.getText().isEmpty()) {
        showAlert(Alert.AlertType.ERROR, owner, "Form Error!", "Please enter your email id");
        return;
    }
    if (passwordField.getText().isEmpty()) {
        showAlert(Alert.AlertType.ERROR, owner, "Form Error!", "Please enter a password");
        return;
    }
    String fullName = fullNameText.getText();
    String emailId = emailidText.getText();
    String password = passwordField.getText();
    JdbcDao jdbcDao = new JdbcDao();
    jdbcDao.insertRecord(fullName, emailId, password);
    showAlert(Alert.AlertType.CONFIRMATION, owner, "Registration Successful!",
        "Welcome " + fullNameText.getText());
}
```

Step – 7 : Creating Registration Controller to Handle Data

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

- JdbcDao class is being used for validating the fields and to store and to retrieve or to manipulate data from the database.

Step – 8 : Creating JdbcDao for database operations

- Here are steps to connect to the MySQL database:
 - Establishing a connection
 - Create a statement
 - Execute the query
 - Using try-with-resources Statements to Automatically Close JDBC Resources
- From JDBC 4.0, we don't need to include 'Class.forName()' in our code, to load JDBC driver. When the method 'getConnection' is called, the 'DriverManager' will automatically load the suitable driver among the JDBC drivers that were loaded at initialization and those loaded explicitly using the same class loader as the current application.

```
Connection conn = DriverManager.getConnection(UrlDatabase, Username, Password);
```

```
public class JdbcDao {
    // Replace below database url, username and password with your actual database credentials
    private static final String DATABASE_URL = "jdbc:mysql://localhost:3306/registrationFX?useSSL=false";
    private static final String DATABASE_USERNAME = "root";
    private static final String DATABASE_PASSWORD = "root";
    private static final String INSERT_QUERY = "INSERT INTO registration (full_name, email_id, password) VALUES (?, ?, ?)";
    public void insertRecord(String fullName, String emailId, String password) throws SQLException
    {
        // Step 1: Establishing a Connection and
        // try-with-resource statement will auto close the connection.
        try (Connection connection = DriverManager.getConnection(DATABASE_URL, DATABASE_USERNAME,
                                                                DATABASE_PASSWORD);

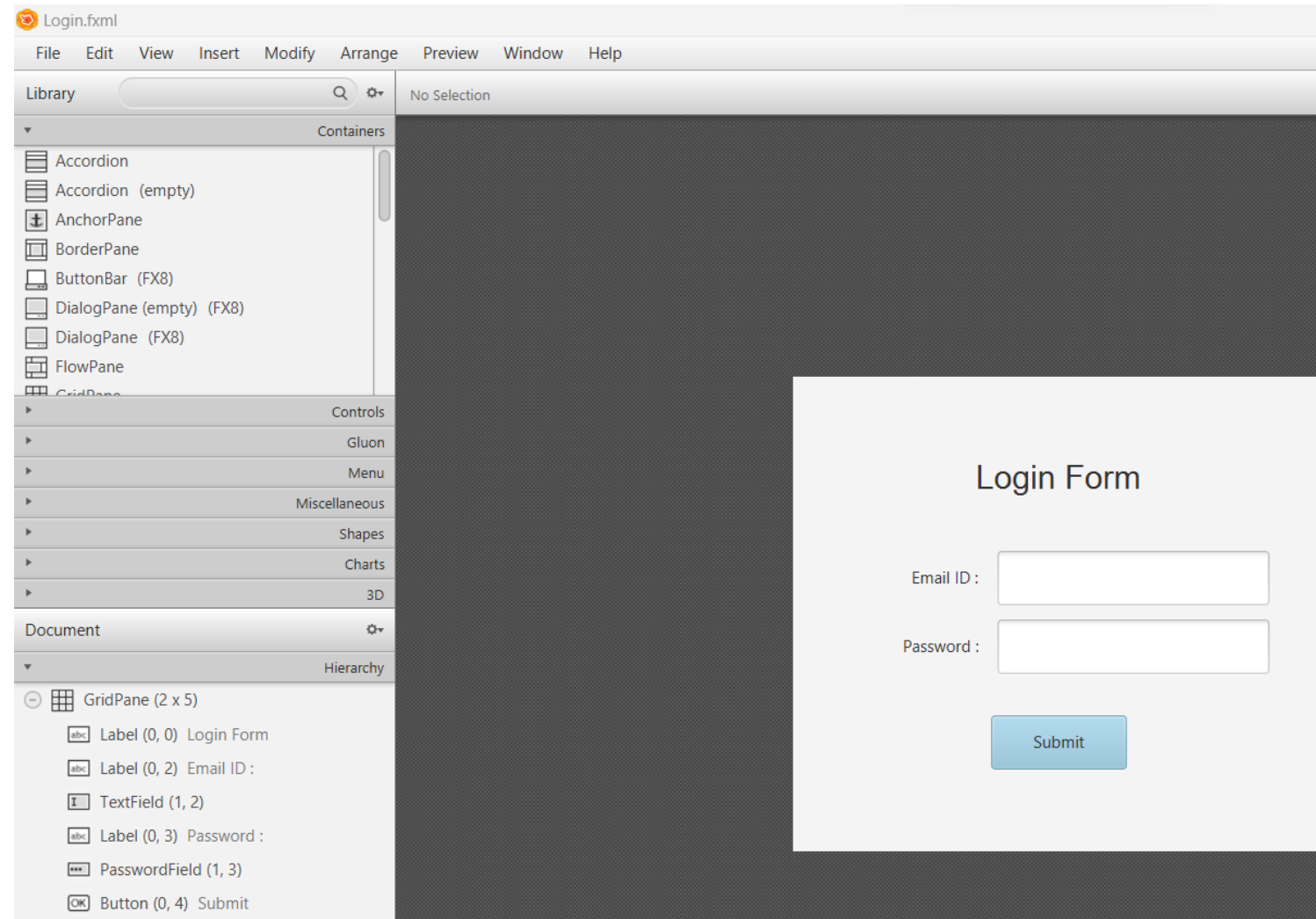
            // Step 2: Create a statement using connection object
            PreparedStatement preparedStatement = connection.prepareStatement(INSERT_QUERY))
        {
            preparedStatement.setString(1, fullName);
            preparedStatement.setString(2, emailId);
            preparedStatement.setString(3, password);

            // Step 3: Execute the query or update query
            preparedStatement.executeUpdate();
        } catch (SQLException e) {
            // print SQL exception information printSQLException(e);
        }
    }
}
```

```
public static void printSQLException(SQLException ex)
{
    for (Throwable e: ex)
    {
        if (e instanceof SQLException)
        {
            e.printStackTrace(System.err); System.err.println("SQLState: "
+ ((SQLException) e).getSQLState()); System.err.println("Error Code: "
+ ((SQLException) e).getErrorCode()); System.err.println("Message: "
+ e.getMessage()); Throwable t = ex.getCause();
            while (t != null)
            {
                System.out.println("Cause: " + t);
                t = t.getCause();
            }
        }
    }
}
```

Creating Login.fxml

- Create Login.fxml in the views folder and then open it in the scene builder and create the layout.



Updating RegisterController.java

- Update RegisterController class with the new code to add the functionality of Login button

@FXML

```
void loginClickEvent(ActionEvent event) {  
    try {  
        FXMLLoader loader = new FXMLLoader(getClass()  
            .getResource("/com/senecacollege/views/Login.fxml"));  
        Parent loginRoot = loader.load();  
        Stage st = new Stage();  
        st.setTitle("User Login");  
        Scene scene = new Scene(loginRoot, 800, 500);  
        st.setScene(scene);  
        st.show();  
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
}
```

Creating LoginController.java

@FXML

```
void login(ActionEvent event) throws SQLException {
    Window owner = submitButton.getScene().getWindow();
    if (emailIdField.getText().isEmpty()) {
        showAlert(Alert.AlertType.ERROR, owner, "Form Error!", "Please enter your email id");
        return;
    }
    if (passwordField.getText().isEmpty()) {
        showAlert(Alert.AlertType.ERROR, owner, "Form Error!", "Please enter a password");
        return;
    }
    String emailId = emailIdField.getText();
    String password = passwordField.getText();
    JdbcDao jdbcDao = new JdbcDao();
    boolean flag = jdbcDao.validate(emailId, password);
    if (!flag) {
        infoBox("Please enter correct Email and Password", null, "Failed");
    } else {
        infoBox("Login Successful!", null, "Failed");
    }
}
```


Creating LoginController.java

```
public static void infoBox(String infoMessage, String headerText, String title) {
    Alert alert = new Alert(AlertType.CONFIRMATION);
    alert.setContentText(infoMessage);
    alert.setTitle(title);
    alert.setHeaderText(headerText);
    alert.showAndWait();
}

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

Updating JdbcDao.java

```
private static final String SELECT_QUERY = "SELECT * FROM registration WHERE email_id = ? and password = ?";

public boolean validate(String emailId, String password) throws SQLException {
    // Step 1: Establishing a Connection and
    // try-with-resource statement will auto close the connection.
    try (Connection connection = DriverManager
        .getConnection(DATABASE_URL, DATABASE_USERNAME, DATABASE_PASSWORD);
        // Step 2: Create a statement using connection object
        PreparedStatement preparedStatement = connection.prepareStatement(SELECT_QUERY))
    {
        preparedStatement.setString(1, emailId);
        preparedStatement.setString(2, password);
        ResultSet resultSet = preparedStatement.executeQuery();
        if (resultSet.next()) {
            return true;
        }
        catch (SQLException e) {
            // print SQL exception information
            printSQLException(e);
        }
        return false;
    }
}
```