JAVA SWING BASED – INTERVIEWSKILLS BOT SQL CONNECTIVITY USING JDBC

A Report
Submitted in partial fulfillment of the Requirements
for the COURSE

DATABASE MANAGEMENT SYSTEMS

By

B ABHINAY
<1602-21-737-002>
Under the guidance of Ms B. Leelavathy

Department of Information Technology
Vasavi College of Engineering
(Autonomous) (Affiliated to Osmania University) Ibrahimbagh, Hyderabad-31
2022-2023

BONAFIDE CERTIFICATE

This is to certify that this project report titled ***'INTERVIEW SKILL BOT'***is a project work of B Abhinay bearing roll no. 1602-21-737-002 who carried out this project under my supervision in the IV semester for the academic year 2022- 2023.

Signature
External Examiner

Signature
Internal Examiner

ABSTRACT

The E-learning project aims to provide a web-based platform for students to access course materials, take mock quizzes, track their progress, and to access question bank. The system includes a user table to manage user information. The system also includes a Progress table to track the user's progress in the course. It also includes the list of students who solved questions from question bank. This project provides a comprehensive solution for online learning, which is especially relevant in the current context where remote learning is becoming more popular. It allows users to learn at their own pace and provides tools to track their progress and improve their learning outcomes.

Requirement Analysis

List of Tables:

- users
- Profession
- suggestion
- suggestion_bot

List of Attributes with their Domain Types:

1.Users Table:

ATTRIBUTE	DOMAIN	CONSTARINT
User_id	NUMBER	Primary_Key
User_name	VARCHAR	Not Null
Mobile_no	NUMBER	
email_id	VARCHAR	Not Null

2.Profession Table:

ATTRIBUTE	DOMAIN	CONSTRAINT
profession_id	NUMBER	Primary_Key
profession_name	VARCHAR	Not Null

3.Suggestion:

ATTRIBUTE	DOMAIN	CONSTARINT
Profession_id	NUMBER	Primary_Key
Profession_name	VARCHAR	Not Null
User_id	NUMBER	Not Null
Suggestion	Varchar	

4.Profession:

ATTRIBUTE	DOMAIN	CONSTARINT
User_id	NUMBER	Foreign_Key
prof_id	NUMBER	Foreign_Key
Suggestion	VARCHAR	Not Null

AIM AND PRIORITY OF THE PROJECT

The aim of the Interview Skills Bot project is to provide users with an interactive and helpful tool to enhance their interview preparation and performance. The bot is designed to simulate a real interview experience and assist users in developing their interview skills, boosting their confidence, and increasing their chances of success.

ARCHITECTURE AND TECHNOLOGY

Software used:

Java, Oracle 11g Database, Java SE version 14, Run SQL.

Java SWING:

Java SWING is a GUI widget toolkit for Java. It is part of Oracle's Java Foundation Classes (JFC) - an API for providing a graphical user interface (GUI) for Java programs.

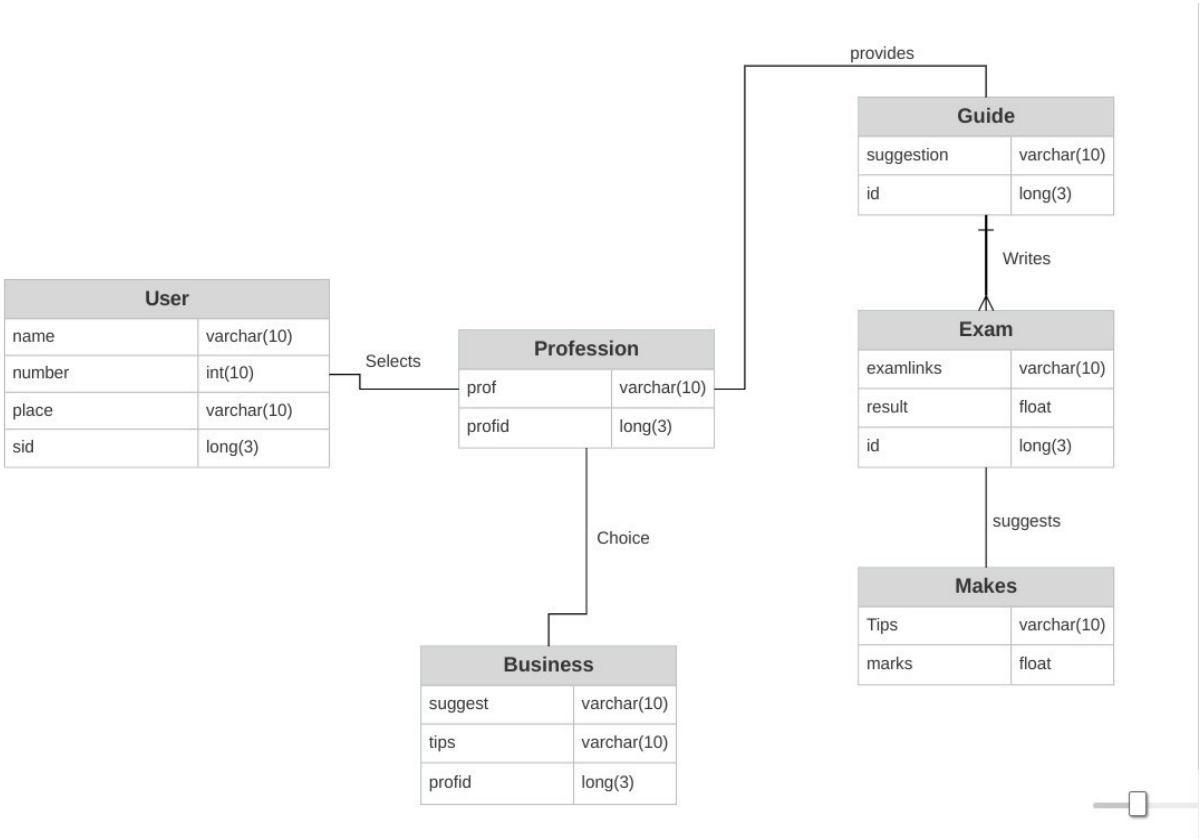
Swing was developed to provide a more sophisticated set of GUI components than the earlier AWT. Swing provides a look and feel that emulates the look and feel of several platforms, and also supports a pluggable look and feel that allows applications to have a look and feel unrelated to the underlying platform. It has more powerful and flexible components than AWT. In addition to familiar components such as buttons, check boxes and labels, Swing provides several advanced components such as tabbed panel, scroll panes, trees, tables, and lists.

SQL:

Structure Query Language(SQL) is a database query language used for storing and managing data in **Relational** DBMS. SQL was the first commercial language introduced for E.F Codd's Relational model of database. Today almost all RDBMS (MySql, Oracle, Infomix, Sybase, MS Access) use **SQL** as the standard database query language. SQL is used to perform all types of data operations in RDBMS.

DESIGN

Entity Relationship Diagram



1.Creating table for students with constraints :

QUERY: create table users(
2 user_id number(4),
3 user_name varchar(20),
4 email varchar(5),
5 mobile number(10),
6 primary key(user_id));

```
Connected.
SQL> create table students(
  2  student_id number(4),
  3  student_name varchar(20),
  4  class varchar(5),
  5  primary key(student_id));

Table created.

SQL> desc students;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 STUDENT_ID                                NOT NULL NUMBER(4)
 STUDENT_NAME                                       VARCHAR2(20)
```

2. Creating notes table:

QUERY: create table profession(
2 profession_id number(2),
3 profession_name varchar(20) not null,
4 primary key(profession_id));

3.Creating suggestion :

QUERY: create table suggestion(
2 uesr_id number(4) primary key,
3 profession_id number(4),suggestion varchar(255));

```
SQL> create table mock_quiz(
  2  quiz_id number(4) primary key,
  3  duration varchar(10));

Table created.

SQL> desc mock_quiz
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 QUIZ_ID                                   NOT NULL NUMBER(4)
 DURATION                                           VARCHAR2(10)
```

4.Creating Questions table:

QUERY: create table questions(
2 question_no number(4) primary key,
3 question varchar2(50) not null);

```
SQL> create table questions(
  2  question_no number(4) primary key,
  3  question varchar2(50) not null);

Table created.

SQL> desc questions;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 QUESTION_NO                               NOT NULL NUMBER(4)
 QUESTION                                  NOT NULL VARCHAR2(50)
```

DML COMMANDS:

INSERT into User VALUES("sia",9551710377,"uk",12);
INSERT into User VALUES("chris",9552710377,"canada",
13);INSERT into User VALUES("stones",9553710377,"dubai",
14); INSERT into User VALUES("virat",9554710377,"india",15);

INSERT into Profession VALUES("software",13);
INSERT into Profession VALUES("government",14);
INSERT into Profession VALUES("medicine",15);

INSERT into Guide VALUES("Be confident while answeringMakes",
13); INSERT into Guide VALUES("Be patriotic",
14);INSERT into Guide VALUES("treat life as important factor while answering",
15);INSERT into Exam VALUES("it.com",90,13);
INSERT into Exam VALUES("gov.in",100,14);
INSERT into Exam VALUES("med.ac.in",70,15);
INSERT into Makes VALUES("keep doing well",90

IMPLEMENTATION

JAVA-SQL Connectivity using JDBC:

Java Database Connectivity (JDBC) is an application programming interface (API) for the programming language Java, which defines how a client may access a database. It is a Java-based data access technology used for Java database connectivity. It is part of the Java Standard Edition platform, from Oracle Corporation. It provides methods to query and update data in a database and is oriented towards relational databases.

The connection to the database can be performed using Java programming (JDBC API) as:

```
}import javafx.application.Application;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.*;
import javafx.stage.Stage;

import java.sql.*;

public class DatabaseOperationsInterface extends Application {

    private static final String JDBC_URL = "jdbc:oracle:thin:@localhost:1521:xe";
    private static final String USERNAME = "abhi";
    private static final String PASSWORD = "002";

    private Connection connection;
    private ObservableList<User> usersData;

    private TableView<User> usersTable;
    private TextField nameField;
```

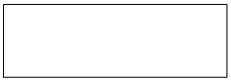
```
private TextField emailField;
private TextField mobileField;

private PreparedStatement insertStatement;
private PreparedStatement deleteStatement;
private PreparedStatement updateStatement;

public static void main(String[] args) {
   launch(args);
}

@Override
public void start(Stage primaryStage) {
   primaryStage.setTitle("Database Operations");

   // Initialize database connection
   connectToDatabase();

   // Create UI components
   Label nameLabel = new Label("Name:");
   Label emailLabel = new Label("Email:");
   Label mobileLabel = new Label("Mobile:");

   nameField = new TextField();
   emailField = new TextField();
   mobileField = new TextField();

   Button addButton = new Button("Add");
   Button deleteButton = new Button("Delete");
   Button updateButton = new Button("Update");

   usersTable = new TableView<>();
   usersData = FXCollections.observableArrayList();
   usersTable.setItems(usersData);

   TableColumn<User, String> nameColumn = new TableColumn<>("Name");
   nameColumn.setCellValueFactory(cellData -> cellData.getValue().nameProperty());
   TableColumn<User, String> emailColumn = new TableColumn<>("Email");
   emailColumn.setCellValueFactory(cellData -> cellData.getValue().emailProperty());
   TableColumn<User, String> mobileColumn = new TableColumn<>("Mobile");
   mobileColumn.setCellValueFactory(cellData -> cellData.getValue().mobileProperty());

   usersTable.getColumns().addAll(nameColumn, emailColumn, mobileColumn);
```

```
        addButton.setOnAction(e -> addUser());
        deleteButton.setOnAction(e -> deleteUser());
        updateButton.setOnAction(e -> updateUser());

        // Layout the UI components
        GridPane gridPane = new GridPane();
        gridPane.setHgap(10);
        gridPane.setVgap(10);
        gridPane.setPadding(new Insets(10));

        gridPane.add(nameLabel, 0, 0);
        gridPane.add(nameField, 1, 0);
        gridPane.add(emailLabel, 0, 1);
        gridPane.add(emailField, 1, 1);
        gridPane.add(mobileLabel, 0, 2);
        gridPane.add(mobileField, 1, 2);

        HBox buttonsBox = new HBox(10, addButton, deleteButton, updateButton);
        VBox vbox = new VBox(10, gridPane, buttonsBox, usersTable);

        Scene scene = new Scene(vbox);
        primaryStage.setScene(scene);
        primaryStage.show();

        // Prepare SQL statements
        prepareStatements();

        // Load existing users from the database
        loadUsers();
    }

    private void connectToDatabase() {
        try {
            connection = DriverManager.getConnection(JDBC_URL, USERNAME, PASSWORD);
            System.out.println("Connected to the database!");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    private void prepareStatements() {
        try {
```

```
            insertStatement = connection.prepareStatement(
                "INSERT INTO users (user_id, name, email, mobile) VALUES
(user_id_seq.NEXTVAL, ?, ?, ?)"
            );

            deleteStatement = connection.prepareStatement(
                "DELETE FROM users WHERE user_id = ?"
            );

            updateStatement = connection.prepareStatement(
                "UPDATE users SET name = ?, email = ?, mobile = ? WHERE user_id = ?"
            );
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    private void loadUsers() {
        try {
            Statement statement = connection.createStatement();
            ResultSet resultSet = statement.executeQuery("SELECT * FROM users");

            while (resultSet.next()) {
                int userId = resultSet.getInt("user_id");
                String name = resultSet.getString("name");
                String email = resultSet.getString("email");
                String mobile = resultSet.getString("mobile");

                User user = new User(userId, name, email, mobile);
                usersData.add(user);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    private void addUser() {
        String name = nameField.getText();
        String email = emailField.getText();
        String mobile = mobileField.getText();

        try {
            insertStatement.setString(1, name);
```

```
            insertStatement.setString(2, email);
            insertStatement.setString(3, mobile);
            insertStatement.executeUpdate();

            ResultSet generatedKeys = insertStatement.getGeneratedKeys();
            if (generatedKeys.next()) {
                int userId = generatedKeys.getInt(1);
                User user = new User(userId, name, email, mobile);
                usersData.add(user);
            }

            clearFields();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    private void deleteUser() {
        User selectedUser = usersTable.getSelectionModel().getSelectedItem();
        if (selectedUser != null) {
            int userId = selectedUser.getUserId();

            try {
                deleteStatement.setInt(1, userId);
                deleteStatement.executeUpdate();

                usersData.remove(selectedUser);
                clearFields();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }

    private void updateUser() {
        User selectedUser = usersTable.getSelectionModel().getSelectedItem();
        if (selectedUser != null) {
            int userId = selectedUser.getUserId();
            String name = nameField.getText();
            String email = emailField.getText();
            String mobile = mobileField.getText();

            try {
```

```
                updateStatement.setString(1, name);
                updateStatement.setString(2, email);
                updateStatement.setString(3, mobile);
                updateStatement.setInt(4, userId);
                updateStatement.executeUpdate();

                selectedUser.setName(name);
                selectedUser.setEmail(email);
                selectedUser.setMobile(mobile);
                usersTable.refresh();

                clearFields();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }

    private void clearFields() {
        nameField.clear();
        emailField.clear();
        mobileField.clear();
    }
}
```



Front-end Programs (User Interfaces) Home Page:

1.Login and Signup Page:

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

public class LoginForm extends JFrame {
   private JTextField usernameField;
   private JPasswordField passwordField;
   private JButton loginButton;

   public LoginForm() {
      setTitle("Login Form");
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      setSize(300, 200);
      setLocationRelativeTo(null);

      // Create components
      JLabel usernameLabel = new JLabel("Username:");
      JLabel passwordLabel = new JLabel("Password:");
      usernameField = new JTextField(20);
      passwordField = new JPasswordField(20);
      loginButton = new JButton("Login");
```

```
        // Create panel and layout
        JPanel panel = new JPanel(new GridLayout(3, 2));
        panel.add(usernameLabel);
        panel.add(usernameField);
        panel.add(passwordLabel);
        panel.add(passwordField);
        panel.add(new JLabel());
        panel.add(loginButton);

        // Add panel to the frame
        add(panel);

        // Add action listener to the login button
        loginButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                String username = usernameField.getText();
                char[] passwordChars = passwordField.getPassword();
                String password = new String(passwordChars);

                // Perform login validation here
                if (username.equals("admin") && password.equals("password")) {
                    JOptionPane.showMessageDialog(null, "Login successful");
                } else {
                    JOptionPane.showMessageDialog(null, "Invalid username or password");
                }
            }
        });
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                LoginForm loginForm = new LoginForm();
                loginForm.setVisible(true);
            }
        });
    }
}
```

2. Login Changes

```
import javafx.beans.property.SimpleStringProperty;
import javafx.beans.property.StringProperty;

public class User {
    private final int userId;
    private String name;
```

```
    private String email;
    private String mobile;

    public User(int userId, String name, String email, String mobile) {
        this.userId = userId;
        this.name = name;
        this.email = email;
        this.mobile = mobile;
    }

    public int getUserId() {
        return userId;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getMobile() {
        return mobile;
    }

    public void setMobile(String mobile) {
        this.mobile = mobile;
    }

    public StringProperty nameProperty() {
        return new SimpleStringProperty(name);
    }

    public StringProperty emailProperty() {
        return new SimpleStringProperty(email);
    }

    public StringProperty mobileProperty() {
        return new SimpleStringProperty(mobile);
    }
```

```
}
```

3. Interview Skill Bot page

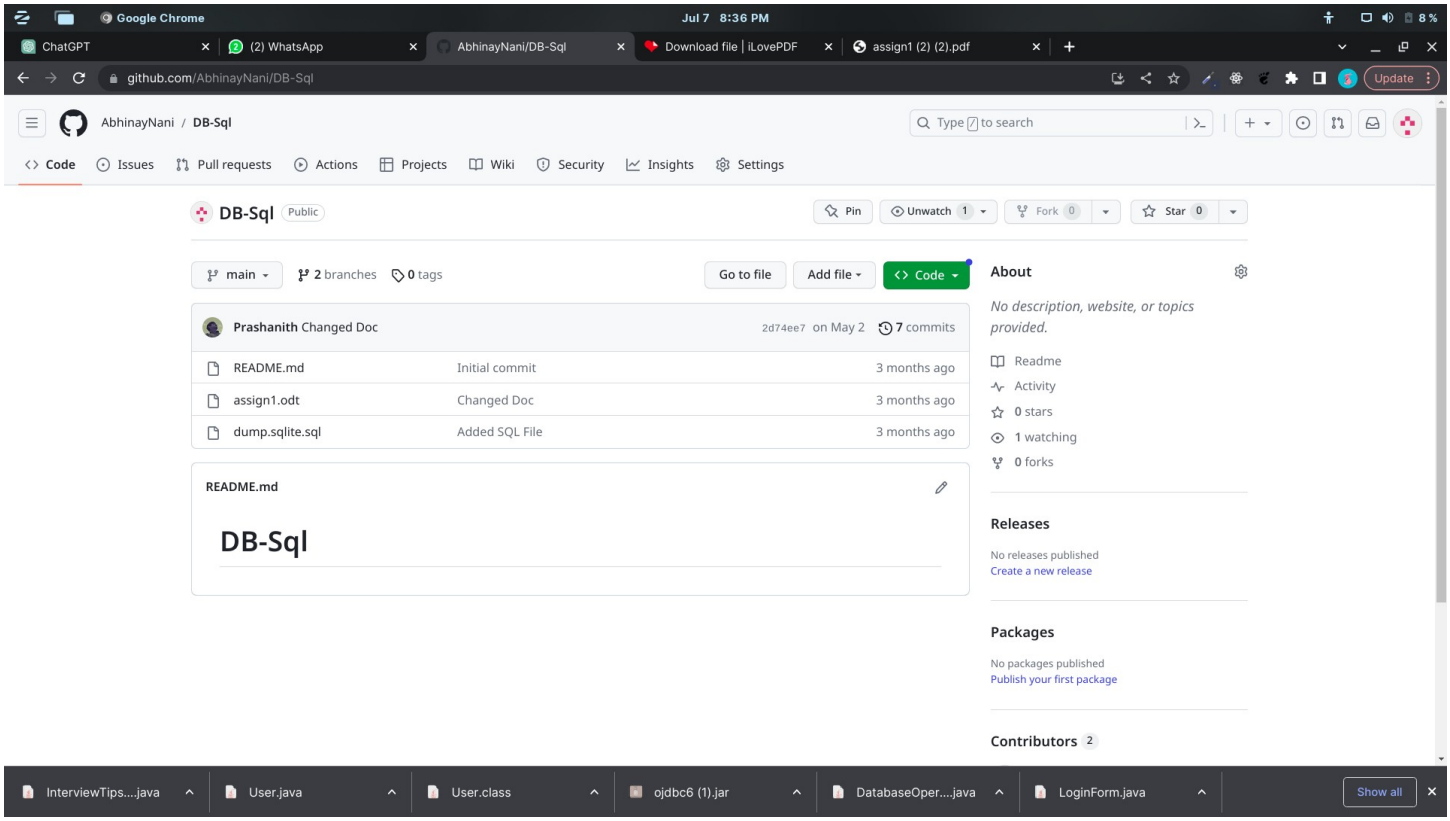
```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.HashMap;
import java.util.Map;

public class InterviewTipsChatbotUI extends JFrame {
    private JTextArea chatArea;
    private JTextField inputField;
    private JButton sendButton;
    private JComboBox<String> fieldComboBox;

    private Map<String, String[]> fieldTips;

    public InterviewTipsChatbotUI() {
        setTitle("Interview Tips Chatbot");
        setSize(400, 300);
        setBackground(Color.BLUE);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Create field tips data
        fieldTips = new HashMap<>();
        fieldTips.put("Software Development", new String[]{"Tip 1", "Tip 2", "Tip 3"});
        fieldTips.put("Data Science", new String[]{"Tip 1", "Tip 2", "Tip 3"});
        fieldTips.put("Marketing", new String[]{"Tip 1", "Tip 2", "Tip 3"});

        // Create components
        chatArea = new JTextArea();
        chatArea.setEditable(false);
        JScrollPane chatScrollPane = new JScrollPane(chatArea);

        inputField = new JTextField(20);

        sendButton = new JButton("Send");
        sendButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                sendMessage();
            }
        });
```

```
        fieldComboBox = new JComboBox<>(fieldTips.keySet().toArray(new String[0]));

        // Create a panel to hold the input field, send button, and field selection
        JPanel inputPanel = new JPanel();
        inputPanel.setLayout(new BorderLayout());
        inputPanel.add(inputField, BorderLayout.CENTER);
        inputPanel.add(sendButton, BorderLayout.EAST);
        inputPanel.add(fieldComboBox, BorderLayout.NORTH);

        // Add components to the frame
        Container container = getContentPane();
        container.add(chatScrollPane, BorderLayout.CENTER);
        container.add(inputPanel, BorderLayout.SOUTH);
    }

    private void sendMessage() {
        String message = inputField.getText();
        if (!message.isEmpty()) {
            chatArea.append("You: " + message + "\n");
            inputField.setText("");

            String selectedField = (String) fieldComboBox.getSelectedItem();
            String[] tips = fieldTips.get(selectedField);
            if (tips != null && tips.length > 0) {
                int randomIndex = (int) (Math.random() * tips.length);
                String tip = tips[randomIndex];
                chatArea.append("Chatbot: Here's a tip for " + selectedField + " interviews: \n" + tip + "\n");
            } else {
                chatArea.append("Chatbot: I'm sorry, I don't have any tips for that field.\n");
            }
        }
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {
                new InterviewTipsChatbotUI().setVisible(true);
            }
        });
    }
}
```

GitHub Links and Folder Structure

LINK: https://github.com/AbhinayNani/DB-Sql

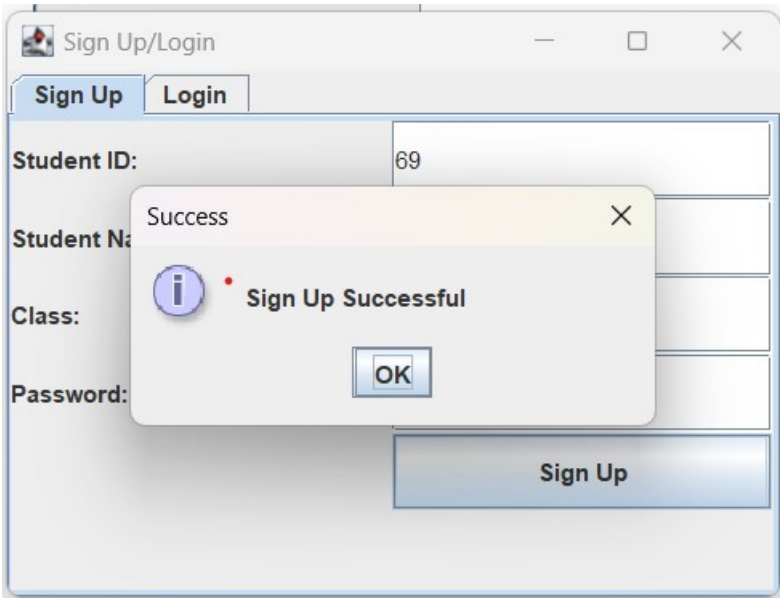
FOLDER STRUCTURE:

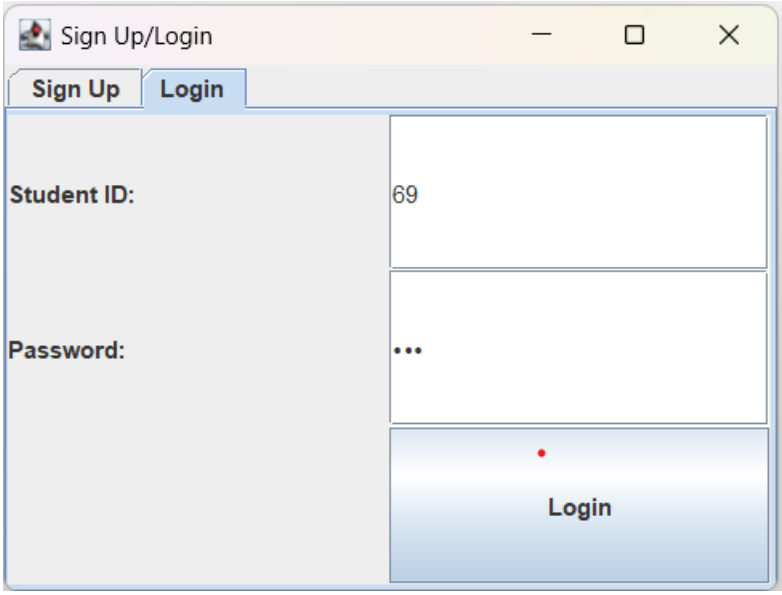

Google Chrome
Jul 7 8:36 PM
ChatGPT
(2) WhatsApp
AbhinayNani/DB-Sql
Download file | iLovePDF
assign1 (2) (2).pdf
github.com/AbhinayNani/DB-Sql
Update
AbhinayNani / DB-Sql
Type / to search
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
DB-Sql
Public
Pin
Unwatch 1
Fork 0
Star 0
main
2 branches
0 tags
Go to file
Add file
Code
Prashanith Changed Doc
2d74ee7 on May 2
7 commits
README.md
Initial commit
3 months ago
assign1.odt
Changed Doc
3 months ago
dump.sqlite.sql
Added SQL File
3 months ago
README.md
DB-Sql
About
No description, website, or topics provided.
Readme
Activity
0 stars
1 watching
0 forks
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package
Contributors 2
InterviewTips....java
User.java
User.class
ojdbc6 (1).jar
DatabaseOper....java
LoginForm.java
Show all

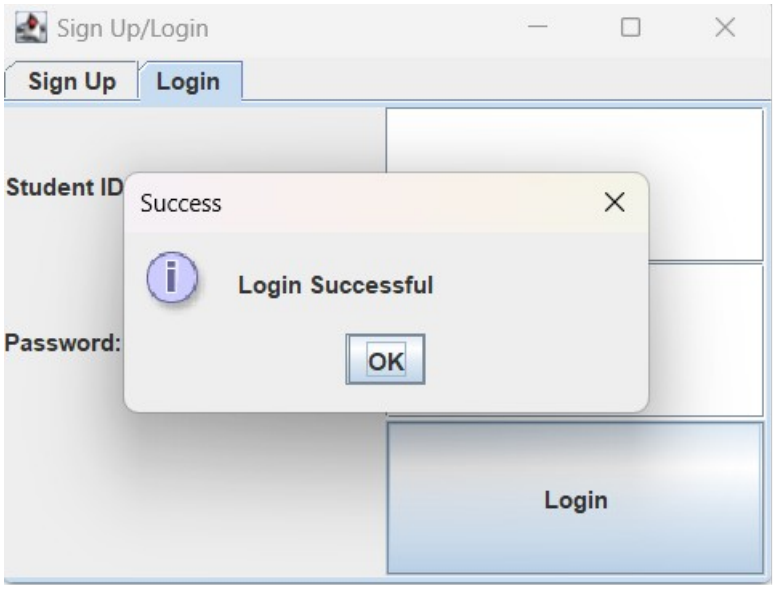
TESTING

LOGIN PAGE

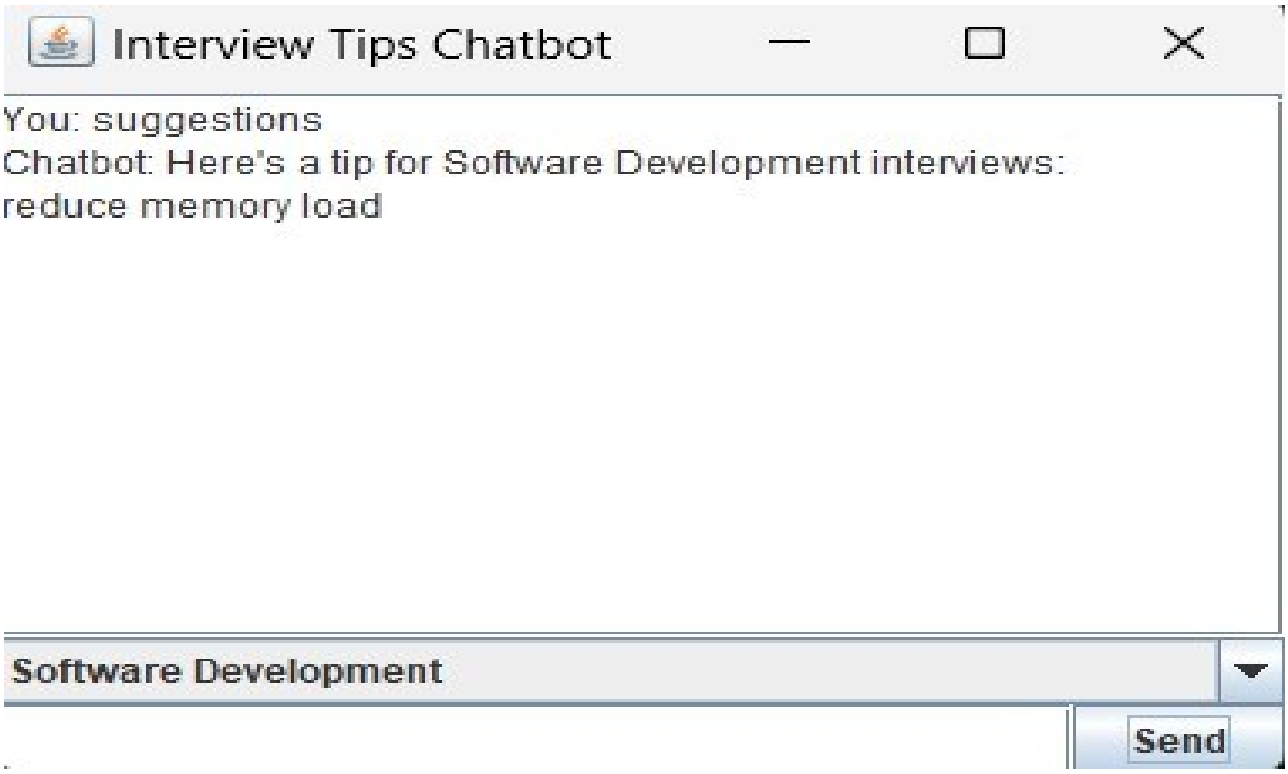
Signup:






Sign Up/Login
Sign Up
Login
Student ID
Password:
Success
Login Successful
OK
Login

Home:


Interview Tips Chatbot
You: suggestions
Chatbot: Here's a tip for Software Development interviews:
reduce memory load
Software Development
Send

RESULTS

I have completed ***'INTERVIEW SKILLS BOT '*** project successfully.

DISCUSSION AND FUTURE WORK

This project contains quizzes so that based on that user will be given efficient suggestions for they future endeavors. As a future work I thought of adding Quiz page where User can write exam and get perfect suggestions.

REFERENCES

- https://docs.oracle.com/javase/7/docs/api/
- https://www.javatpoint.com/java-swing
- https://stackoverflow.com/