CAREER RECOMMENDATION SYSTEM

A MINI-PROJECT REPORT

Submitted by

ROSHAN BRIGHT	**240701439**
SAKTHI BALA SUNDARAM	**240701460**

in partial fulfillment of the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

An Autonomous Institute

CHENNAI

NOVEMBER 2025

BONAFIDE CERTIFICATE

Certified that this project **“CAREEER RECOMMENDATION SYSTEM”** is the bonafide work of **“ROSHAN BRIGTH R,SAKTHI BALA SUNDARAM M”** who carried out the project work under my supervision.

SIGNATURE

S.VINOTHIINI

ASSISTANT PROFESSOR

Dept. of Computer Science and Engg,
Rajalakshmi Engineering College
Chennai

This mini project report is submitted for the viva voce examination to be held on

____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ABSTRACT

Choosing the right career path is often a challenging and confusing task for students. The Career Path Recommender System addresses this problem by providing personalized career suggestions based on a student's interests, skills, and quiz responses. Students interact with a Java-based frontend, where they complete a short quiz assessing their aptitude, preferences, and strengths. Each response is analyzed and mapped to a database of careers, which includes required skills, educational paths, and industry relevance. The system then generates a tailored list of career options that best match the student's profile.

ACKNOWLEDGEMENT

We express our sincere thanks to our beloved and honorable chairman **MR. S. MEGANATHAN** and the chairperson **DR. M.THANGAM MEGANATHAN** for their timely support and encouragement.

We are greatly indebted to our respected and honorable principal **Dr. S.N. MURUGESAN** for his able support and guidance.

No words of gratitude will suffice for the unquestioning support extended to us by our Head Of The Department **Dr. E.M. MALATHY** and our Deputy Head Of The Department **Dr. J. MANORANJINI** for being ever supporting force during our project work

We also extend our sincere and hearty thanks to our internal guide **S.VINOTHINI ,** for her valuable guidance and motivation during the completion of this project.

Our sincere thanks to our family members, friends and other staff members of computer science engineering.

1. ROSHAN BRIGHT R

2. SATHI BALA SUNDARAM M

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

The Career Recommendation System is a desktop-based Java application designed to guide users toward suitable career paths by matching their skills, qualifications, and interests with predefined career profiles. Built using Java (JDK 17+), MySQL, and Swing/JavaFX for the UI, it combines OOP principles, JDBC connectivity, and a custom recommendation algorithm to deliver personalized, data-driven suggestions

1.2 SCOPE OF THE WORK

This project serves as a proof-of-concept for AI-driven career guidance, expandable with machine learning, web deployment, or mobile integration in the future. It empowers users with actionable insights and supports educational institutions in career counseling program

1.3 PROBLEM STATEMENT

In today's competitive job market, many individuals—especially students and early-career professionals—struggle to identify careers that align with their strengths and passions. Traditional career counseling is often inaccessible or subjective. This system addresses the gap by providing an automated, objective, and scalable solution.

1.4 AIM AND OBJECTIVES OF THE PROJECT

The aim of this project is to develop a desktop-based Career Recommendation System using Java, MySQL, and Swing/JavaFX that matches user profiles (skills, qualifications, interests) with career requirements via a match-score algorithm to provide personalized, data-driven career suggestions. The objectives include setting up the development environment, designing a relational database, building an OOP-layered architecture, creating an interactive UI, integrating JDBC for data persistence, implementing the recommendation logic, thorough testing, optional enhancements, comprehensive documentation, and final delivery of a fully functional application.

CHAPTER 2

SYSTEM SPECIFICATIONS

2.1 HARDWARE SPECIFICATIONS

Processor	:	Intel i7
Memory Size	:	8GB (Minimum)
HDD	:	1 TB (Minimum)

2.2 SOFTWARE SPECIFICATIONS

Operating System	:	WINDOWS 10
Front – End	:	Python
Back - End	:	MySql
Language	:	python,SQL

CHAPTER 3

MODULE DESCRIPTION

A desktop-based Java application that guides users toward suitable careers by matching their skills, qualifications, and interests against a database of career profiles. Uses JDBC for MySQL integration and Swing/JavaFX for UI.

1. **User Management Module**

- Handles registration & profile storage.
- Fields: Name, Age, Qualification, Skills (comma-separated), Interests.
- DAO: UserDAO → CRUD operations on users table.

2. **Career Data Module**

- Stores & manages career entries with required skills.
- Table: careers (career_id, name, required_skills, description).
- DAO: CareerDAO → Insert/fetch/update careers.

3. **Database Integration Module**

- DBConnection utility class (singleton pattern).
- Uses MySQL Connector/J (JDBC).
- Schema: career_recommendation DB with 3 tables + sample data.

4. **User Interface Module** (Swing/JavaFX)

- **Main Frame:** Menu with Register | Recommend | Exit.
- **Registration Form:** Input fields + "Save" button.
- **Recommendation Panel:** User dropdown → "Recommend" → JTable display (Career | Score | Match %).

5. **Admin Panel Module** (Optional)

- Add/Edit/Delete careers.
- View all users & past recommendations.

CHAPTER 4

MAINFRAME CODING

```
package ui;


import javax.swing.*;

import java.awt.*;

import java.util.List;

import dao.CareerDAO;

import model.Career;


public class MainFrame extends JFrame {


  public MainFrame() {

    try {

      // Apply FlatLaf theme

      Theme.applyTheme(this);

      System.out.println("MainFrame: Applied FlatLaf theme");


      // Frame setup

      setTitle("Career Recommendation System");
```

```
        setSize(1650, 1080);

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        setLocationRelativeTo(null);


        /* ---------- BACKGROUND PANEL AS CONTENT PANE ----
------ */

        String bgPath =
"C:/Projects/CareerRecommendationSystem/src/resources/background.j
pg";

        BackgroundPanel bg = new BackgroundPanel(bgPath);

        bg.setLayout(new BorderLayout());

        setContentPane(bg); // ← CRITICAL: Only this


        /* ---------- TITLE (TOP) ---------- */

        JLabel title = new JLabel("", JLabel.CENTER);

        title.setFont(new Font("Segoe UI", Font.BOLD, 48));

        title.setForeground(Color.white);

        title.setBorder(BorderFactory.createEmptyBorder(40, 0, 40, 0));

        bg.add(title, BorderLayout.NORTH);


        /* ---------- CENTER BUTTONS ---------- */

        JPanel centerPanel = new JPanel(new GridBagLayout());
        centerPanel.setOpaque(false);


        GridBagConstraints gbc = new GridBagConstraints();

        gbc.insets = new Insets(15, 15, 15, 15);

        gbc.gridx = 0;

        gbc.gridy = GridBagConstraints.RELATIVE;

        gbc.anchor = GridBagConstraints.CENTER;


        JButton registerBtn = new JButton("Register New User");

        JButton recommendBtn = new JButton("Get
Recommendations");


        Theme.styleButton(registerBtn);

        Theme.styleButton(recommendBtn);


        centerPanel.add(registerBtn, gbc);

        centerPanel.add(Box.createVerticalStrut(20), gbc);

        centerPanel.add(recommendBtn, gbc);


        bg.add(centerPanel, BorderLayout.CENTER);
        /* ---------- EXIT BUTTON — BOTTOM LEFT ---------- */

        JPanel exitPanel = new JPanel(new
FlowLayout(FlowLayout.LEFT, 30, 25));

        exitPanel.setOpaque(false);



        JButton exitBtn = new JButton("Exit");

        Theme.styleButton(exitBtn);

        exitBtn.addActionListener(_ -> System.exit(0));



        exitPanel.add(exitBtn);

        bg.add(exitPanel, BorderLayout.SOUTH);



        /* ---------- BUTTON ACTIONS ---------- */

        registerBtn.addActionListener(_ -> new RegistrationFrame());

        recommendBtn.addActionListener(_ -> new
RecommendFrame());



        /* ---------- TEST DATABASE ---------- */

        testDBConnection();



        /* ---------- SHOW FRAME ---------- */

        setVisible(true);
```

```
            System.out.println("MainFrame: Set visible");


        } catch (Exception e) {

            System.err.println("Error initializing MainFrame: " +
e.getMessage());

            e.printStackTrace();

        }

    }


    private void testDBConnection() {

        try {

            List<Career> careers = new CareerDAO().getAllCareers();

            System.out.println("DB Connected! Found " + careers.size() + "
careers.");

        } catch (Exception ex) {

            JOptionPane.showMessageDialog(

                    this,

                    "Database Connection Failed!\n" + ex.getMessage(),

                    "Error",

                    JOptionPane.ERROR_MESSAGE);

        }

    }
    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            try {
                new MainFrame();
            } catch (Exception e) {
                System.err.println("Error starting application: " +
e.getMessage());
                e.printStackTrace();
            }
        });
    }
}
```

RECOMMEND FRAME CODING

```
 package ui;
import dao.RecommendationDAO;
import dao.UserDAO;
import logic.RecommendationEngine;
import model.Recommendation;
import javax.swing.*;

import javax.swing.table.*;

import java.awt.*;



import java.util.List;

import java.util.concurrent.*;



public class RecommendFrame extends JFrame {



   private final JComboBox<String> userCombo = new
JComboBox<>();

   private  JTable resultTable;

   private  DefaultTableModel tableModel;

   private final CardLayout cardLayout = new CardLayout();

   private final JPanel cardPanel = new JPanel(cardLayout);

   private final ExecutorService executor =
Executors.newSingleThreadExecutor();



   public RecommendFrame() {

      setTitle("Career Recommendations");

      setSize(900, 650);

      setLocationRelativeTo(null);
```

```
        setDefaultCloseOperation(DISPOSE_ON_CLOSE);



        /* ---------- BACKGROUND ---------- */

        String bgPath =
"C:/Projects/CareerRecommendationSystem/src/resources/rec_back.jpg"
;  // or reg_back.jpg

        BackgroundPanel bg = new BackgroundPanel(bgPath);

        bg.setLayout(new BorderLayout());

        setContentPane(bg);  // ← MUST BE HERE

        bg.add(cardPanel, BorderLayout.CENTER);

        /* ---------- CARD PANEL (selector / loading / result) ---------- */

        buildSelectorPanel();

        buildLoadingPanel();

        buildResultPanel();




        setVisible(true);

    }



    /* -------------------------------------------------- */

    private void buildSelectorPanel() {

        JPanel selector = new JPanel(new BorderLayout(20, 20));
        selector.setOpaque(false);

        selector.setBorder(BorderFactory.createEmptyBorder(30, 40, 30, 
40));



        // Header

        JLabel title = new JLabel("Select a user to get recommendations", 
SwingConstants.CENTER);

        title.setFont(new Font("Segoe UI", Font.BOLD, 24));

        title.setForeground(Color.BLACK);

        selector.add(title, BorderLayout.NORTH);



        // Combo + Button

        JPanel ctrl = new JPanel();

        ctrl.setOpaque(false);

        ctrl.add(new JLabel("<html><font color=white>Select 
User:</font></html>"));

        ctrl.add(userCombo);

        JButton go = new JButton("Generate");

        go.setFont(new Font("Segoe UI", Font.BOLD, 14));

        go.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSO
R));

        ctrl.add(go);

        selector.add(ctrl, BorderLayout.CENTER);
        // Load users
        loadUsers();

        go.addActionListener(_ -> generateRecommendations());

        cardPanel.add(selector, "<span style=\"color: 
black;\">SELECT</span>");
    }

    private void buildLoadingPanel() {
        JPanel loading = new JPanel(new GridBagLayout());
        loading.setOpaque(false);

        JProgressBar spinner = new JProgressBar();
        spinner.setIndeterminate(true);
        spinner.setPreferredSize(new Dimension(80, 80));
        spinner.setForeground(new Color(0, 180, 255));

        JLabel txt = new JLabel("Generating recommendations…", 
SwingConstants.CENTER);
        txt.setFont(new Font("Segoe UI", Font.PLAIN, 18));
        txt.setForeground(Color.WHITE);


        JPanel box = new JPanel();
        box.setLayout(new BoxLayout(box, BoxLayout.Y_AXIS));
        box.setOpaque(false);
        box.add(Box.createVerticalStrut(20));
        box.add(spinner);
        box.add(Box.createVerticalStrut(15));
        box.add(txt);
        box.setAlignmentX(Component.CENTER_ALIGNMENT);


        loading.add(box);
        cardPanel.add(loading, "LOADING");
    }


    private void buildResultPanel() {
        // Table model (non-editable)
        String[] cols = {"#", "Career", "Match", "Score"};
        tableModel = new DefaultTableModel(cols, 0) {
            @Override public boolean isCellEditable(int r, int c) { return
false; }
```

```
        };

        resultTable = new JTable(tableModel);

        resultTable.setRowHeight(48);

        resultTable.setFont(new Font("Segoe UI", Font.PLAIN, 15));

        resultTable.setShowGrid(false);

        resultTable.setIntercellSpacing(new Dimension(0, 0));

        resultTable.getTableHeader().setFont(new Font("Segoe UI",
Font.BOLD, 14));

        resultTable.getTableHeader().setBackground(new Color(0, 120,
215));

        resultTable.getTableHeader().setForeground(Color.WHITE);

        resultTable.setSelectionMode(ListSelectionModel.SINGLE_SELE
CTION);



        // Zebra striping & hover

        resultTable.setDefaultRenderer(Object.class, new
ZebraRenderer());



        // Progress-bar column

        TableColumn scoreCol =
resultTable.getColumnModel().getColumn(2);

        scoreCol.setCellRenderer(new ProgressBarRenderer(0, 100));
        // Wrap in scroll pane

        JScrollPane scroll = new JScrollPane(resultTable);

        scroll.setOpaque(false);

        scroll.getViewport().setOpaque(false);

        scroll.setBorder(BorderFactory.createEmptyBorder());


        // Glass card

        JPanel card = new JPanel(new BorderLayout());

        card.setOpaque(false);

        card.setBorder(BorderFactory.createEmptyBorder(30, 40, 30, 40));

        card.add(scroll, BorderLayout.CENTER);


        // Back button

        JButton back = new JButton("← Back");

        back.setFont(new Font("Segoe UI", Font.BOLD, 13));

        back.addActionListener(_ -> cardLayout.show(cardPanel,
"SELECT"));

        JPanel south = new JPanel(new FlowLayout(FlowLayout.LEFT));

        south.setOpaque(false);

        south.add(back);

        card.add(south, BorderLayout.SOUTH);
```

```
    cardPanel.add(card, "RESULT");
}

/* -------------------------------------------------- */
private void loadUsers() {
    var users = new UserDAO().getAllUsers();
    userCombo.removeAllItems();
    if (users.isEmpty()) {
        userCombo.addItem("No users – register first");
        return;
    }
    for (var u : users) {
        userCombo.addItem(u.getName() + " (ID: " + u.getId() + ")");
    }
}

private void generateRecommendations() {
    if (userCombo.getSelectedItem() == null ||
        userCombo.getSelectedItem().toString().contains("No users")) {
            JOptionPane.showMessageDialog(this, "Please register a user
first.", "No User", JOptionPane.WARNING_MESSAGE);
            return;
        }

        cardLayout.show(cardPanel, "LOADING");

        executor.submit(() -> {
            try {
                int userId = Integer.parseInt(
                        userCombo.getSelectedItem().toString().split("ID:
")[1].replace(")", ""));

                List<Recommendation> recs = new
RecommendationEngine().generate(userId, 10); // top 10

                SwingUtilities.invokeLater(() -> {
                    tableModel.setRowCount(0);
                    int rank = 1;
                    for (Recommendation r : recs) {
                        tableModel.addRow(new Object[]{
                                rank++,
                                r.getCareerName(),
                                (int) r.getMatchScore(),   // for progress bar
                                String.format("%.1f%%", r.getMatchScore())
                        });
                    }
                    if (recs.isEmpty()) {
                        tableModel.addRow(new Object[]{"—", "No matching
careers", 0, "—"});
                    }
                    new RecommendationDAO().saveRecommendations(recs);
                    cardLayout.show(cardPanel, "RESULT");
                });
            } catch (Exception ex) {
                SwingUtilities.invokeLater(() -> {
                    JOptionPane.showMessageDialog(RecommendFrame.this,
                        "Error: " + ex.getMessage(), "Generation Failed",
                        JOptionPane.ERROR_MESSAGE);
                    cardLayout.show(cardPanel, "SELECT");
                });
            }
        });
```

```
    }


    /* -------------------------------------------------- */
    @Override
    public void dispose() {
        executor.shutdownNow();
        super.dispose();
    }
}


/* ===================== RENDERERS
===================== */


class ZebraRenderer extends DefaultTableCellRenderer {
    private final Color even = new Color(255, 255, 255, 230);
    private final Color odd  = new Color(245, 245, 255, 230);
    private final Color hover = new Color(0, 120, 215, 30);


    @Override
    public Component getTableCellRendererComponent(JTable table,
Object value,
                                                   boolean isSelected, boolean hasFocus,
                                        int row, int column) {
        super.getTableCellRendererComponent(table, value, isSelected,
hasFocus, row, column);
        if (isSelected) {
            setBackground(table.getSelectionBackground());
            setForeground(table.getSelectionForeground());
        } else {
            setBackground(row % 2 == 0 ? even : odd);
            setForeground(Color.BLACK);
        }
        setHorizontalAlignment(column == 0 ? SwingConstants.CENTER
: SwingConstants.LEFT);
        setBorder(BorderFactory.createEmptyBorder(0, 12, 0, 12));
        return this;
    }

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        // hover effect
        JTable table = (JTable)
SwingUtilities.getAncestorOfClass(JTable.class, this);
```

```
        if (table != null) {
            int row = table.rowAtPoint(g.getClipBounds().getLocation());
            if (row == table.getSelectedRow()) {
                g.setColor(hover);
                g.fillRect(0, 0, getWidth(), getHeight());
            }
        }
    }
}


class ProgressBarRenderer extends JProgressBar implements
TableCellRenderer {
    public ProgressBarRenderer(int min, int max) {
        super(min, max);
        setStringPainted(true);
        setForeground(new Color(0, 180, 255));
    }


    @Override
    public Component getTableCellRendererComponent(JTable table,
Object value,
                                                   boolean isSelected, boolean hasFocus,
                                    int row, int column) {
        int v = (value instanceof Number) ? ((Number) value).intValue() :
0;
        setValue(v);
        return this;
    }
}
```

REGISTRATION FRAME CODING

```
package ui;

import dao.UserDAO;
import model.User;

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

public class RegistrationFrame extends JFrame {

    private final JTextField nameFld = new JTextField(20);
    private final JTextField ageFld  = new JTextField(5);
    private final JTextField qualFld = new JTextField(20);

    private final JTextField skillFld = new JTextField(30);

    private final JTextField interestFld = new JTextField(30);


    public RegistrationFrame() {

        setTitle("Register New User");

        setSize(600, 500);

        setLocationRelativeTo(null);

        setDefaultCloseOperation(DISPOSE_ON_CLOSE);


        /* ---------- BACKGROUND ---------- */

        String bgPath =
"C:/Projects/CareerRecommendationSystem/src/resources/reg_
back.jpg";

        BackgroundPanel bg = new BackgroundPanel(bgPath);

        bg.setLayout(new BorderLayout());

        setContentPane(bg);



        /* ---------- OVERLAY ---------- */
        JPanel overlay = new JPanel(new GridBagLayout());

        overlay.setBackground(new Color(255, 255, 255, 180));

        overlay.setBorder(BorderFactory.createEmptyBorder(30,
40, 30, 40));



        GridBagConstraints c = new GridBagConstraints();

        c.insets = new Insets(8, 8, 8, 8);

        c.anchor = GridBagConstraints.WEST;


        // ----- form rows -----

        addRow(overlay, c, 0, "Name:", nameFld);

        addRow(overlay, c, 1, "Age:", ageFld);

        addRow(overlay, c, 2, "Qualification:", qualFld);

        addRow(overlay, c, 3, "Skills (comma sep.):", skillFld);

        addRow(overlay, c, 4, "Interests (comma sep.):",
interestFld);


        // ----- buttons -----

        JPanel btnPanel = new JPanel();
        btnPanel.setOpaque(false);

        JButton saveBtn = new JButton("Save User");

        JButton cancelBtn = new JButton("Cancel");

        btnPanel.add(saveBtn);

        btnPanel.add(cancelBtn);


        c.gridx = 0; c.gridy = 5; c.gridwidth = 2; c.anchor =
GridBagConstraints.CENTER;

        overlay.add(btnPanel, c);


        bg.add(overlay, BorderLayout.CENTER);




        /* ---------- ACTIONS ---------- */

        saveBtn.addActionListener(_ -> saveUser());

        cancelBtn.addActionListener(_ -> dispose());


        setVisible(true);

    }
```

```
   private void addRow(JPanel panel, GridBagConstraints c, int
row,
                  String label, JComponent field) {
      c.gridx = 0; c.gridy = row; c.gridwidth = 1;
      panel.add(new JLabel(label), c);
      c.gridx = 1; c.gridwidth =
GridBagConstraints.REMAINDER;
      panel.add(field, c);
   }

   private void saveUser() {
      try {
         User u = new User();
         u.setName(nameFld.getText().trim());
         u.setAge(Integer.parseInt(ageFld.getText().trim()));
         u.setQualification(qualFld.getText().trim());
         u.setSkills(skillFld.getText());
         u.setInterests(interestFld.getText());

         new UserDAO().saveUser(u);
```

```
            JOptionPane.showMessageDialog(this, "User saved
successfully!");

            dispose();

        } catch (Exception ex) {

            JOptionPane.showMessageDialog(this,

                    "Error: " + ex.getMessage(), "Save Failed",

                    JOptionPane.ERROR_MESSAGE);

        }

    }

}
```

SQL QUERY

```
CREATE DATABASE IF NOT EXISTS
career_recommendation;

USE career_recommendation;



DROP TABLE IF EXISTS recommendations;

DROP TABLE IF EXISTS users;

DROP TABLE IF EXISTS careers;



CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    age INT,
    qualification VARCHAR(100),
    skills TEXT,
    interests TEXT
);

CREATE TABLE careers (
    id INT AUTO_INCREMENT PRIMARY KEY,
    career_name VARCHAR(100) NOT NULL,
    required_skills TEXT NOT NULL
);

CREATE TABLE recommendations (
    id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT,
    career_id INT,
    match_score DECIMAL(5,2),
   FOREIGN KEY (user_id) REFERENCES users(id) ON
DELETE CASCADE,

   FOREIGN KEY (career_id) REFERENCES careers(id) ON
DELETE CASCADE

);



INSERT INTO careers (career_name, required_skills)
VALUES

('Software Engineer', 'Java,SQL,OOP,Algorithms'),

('Data Analyst', 'Python,SQL,Excel,Statistics'),

('UX Designer', 'Figma,Prototyping,User Research,Creativity'),

('DevOps Engineer', 'Docker,Kubernetes,Linux,CI/CD'),

('Digital Marketer', 'SEO,Google Analytics,Content
Writing,Social Media');



SELECT * FROM careers;

SELECT * FROM recommendations;

select * from users;
```

CHAPTER 5

SCREEN SHOTS

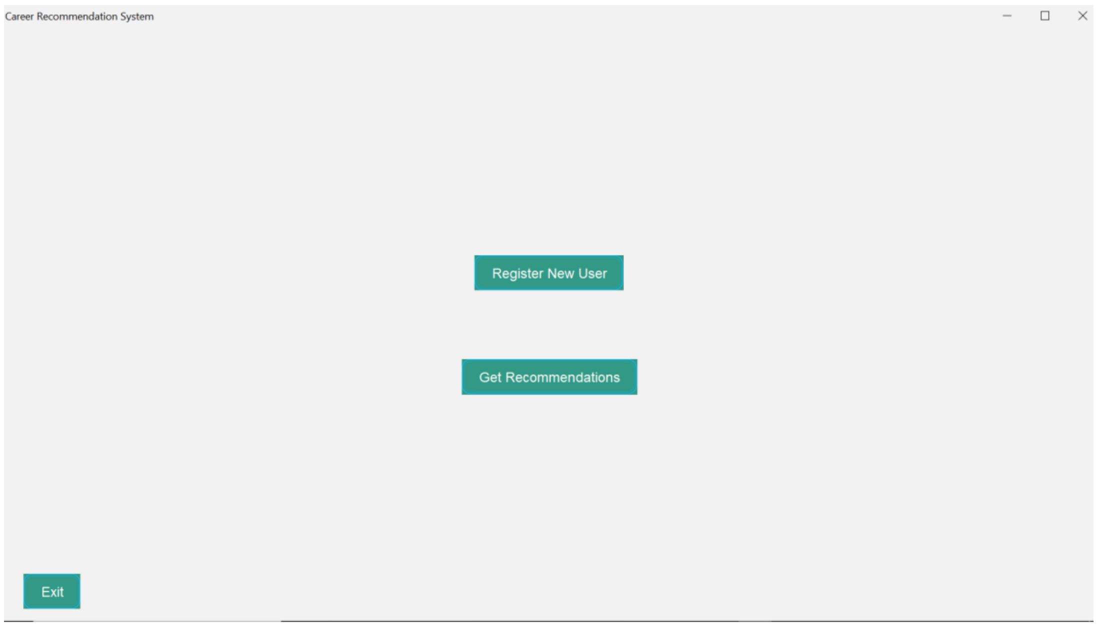
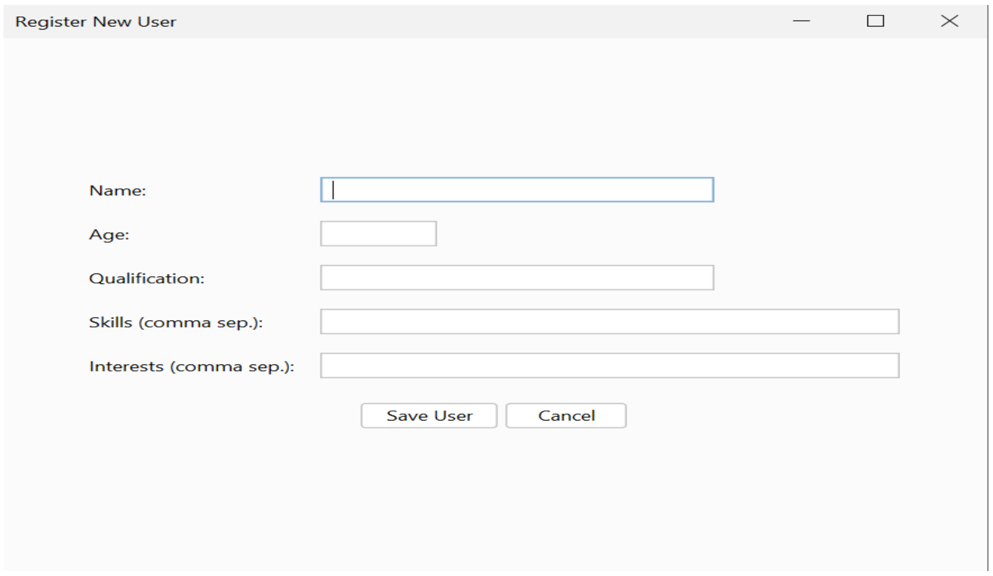


Fig 5.1 Front page



Name:

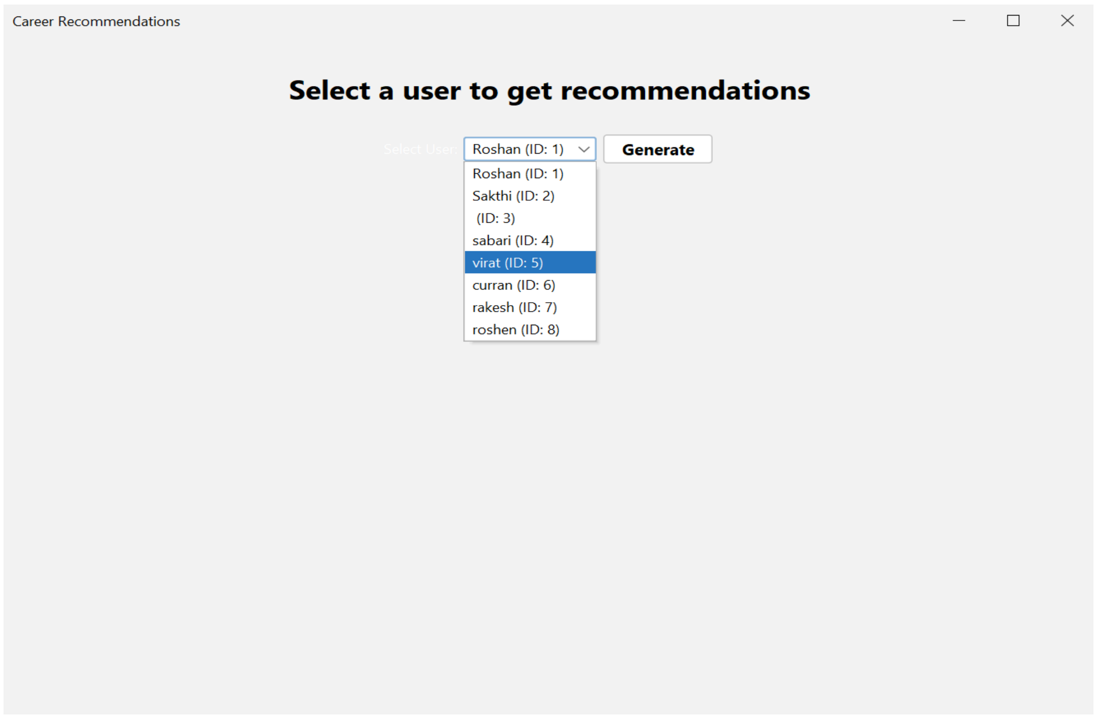
Age:

Qualification:

Skills (comma sep.):

Interests (comma sep.):

Fig 5.2 User details

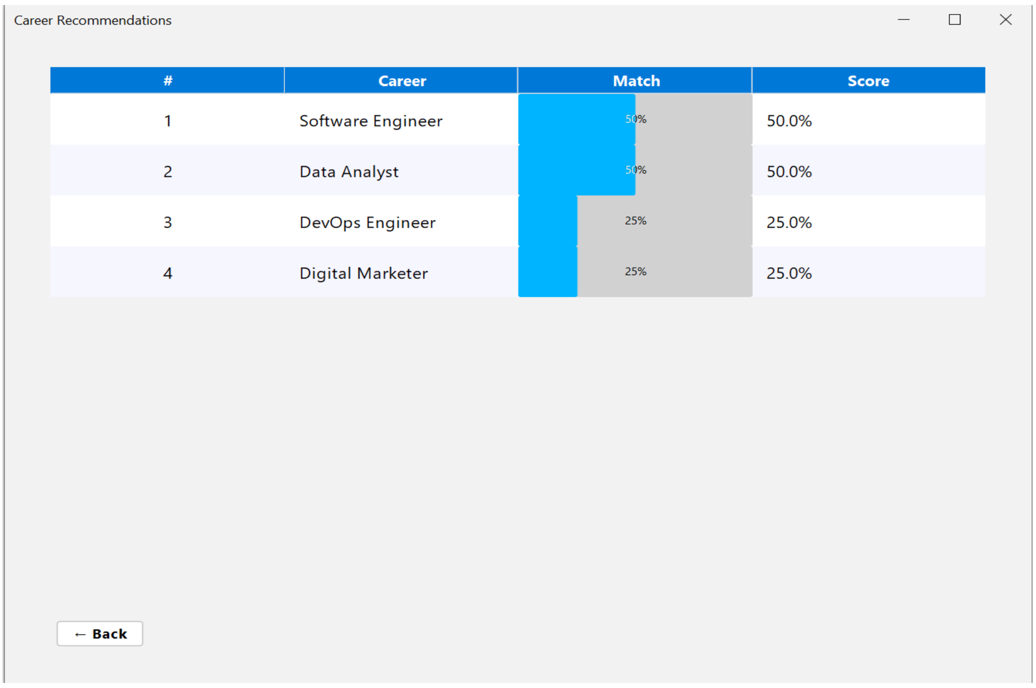

Career Recommendations

Select a user to get recommendations

Select User: Roshan (ID: 1) | Generate

- Roshan (ID: 1)
- Sakthi (ID: 2)
- (ID: 3)
- sabari (ID: 4)
- virat (ID: 5)
- curran (ID: 6)
- rakesh (ID: 7)
- roshen (ID: 8)

Fig 5.3 user selection page


Career Recommendations

#	Career	Match	Score
1	Software Engineer	50%	50.0%
2	Data Analyst	50%	50.0%
3	DevOps Engineer	25%	25.0%
4	Digital Marketer	25%	25.0%

← Back

Fig 5.4 Recommeding page

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT

The *Career Recommendation System* successfully provides personalized career suggestions to users based on their interests, skills, and academic background. By leveraging Java for the frontend and MySQL for the backend, the system offers a reliable, secure, and interactive platform for students and professionals to explore suitable career paths. This project demonstrates the effective use of database management, logical decision-making, and user-friendly design to simplify the complex process of career guidance. Overall, it bridges the gap between user aspirations and potential career opportunities through technology-driven recommendations.

REFERENCES

1. https://www.w3schools.com/sql/
2. https://www.wikipedia.org/
3. https://www.learnpython.org/
4. https://www.codecademy.com/learn/learn-python