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

Mrs.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

The Career Recommendation System is a standalone desktop application developed using Java SE (JDK 17+), MySQL, and Swing/JavaFX to provide personalized, offline career guidance by matching user skills, qualifications, and interests with predefined career requirements. Users register profiles via an intuitive GUI, input details such as name, age, education, technical/soft skills, and interests, and trigger the recommendation engine, which computes a match percentage using the formula match_score = (matching_skills / total_required_skills) × 100, ranks careers accordingly, displays results in a dynamic table, and stores them in the recommendations table for future access. The system follows a structured 10-phase roadmap: environment setup with VS Code, Java extensions, MySQL Workbench, and JDBC driver; database design with normalized users, careers, and recommendations tables including sample data; OOP architecture across model, dao, logic, and ui packages; GUI development featuring dashboard, registration form, user selection, and result visualization; JDBC integration for seamless CRUD operations; algorithmic logic for accurate skill matching; comprehensive testing (unit, integration, end-to-end); optional enhancements like admin panel, login, PDF export, and JavaFX charts; documentation with UML diagrams, schema, flowcharts, screenshots; and deployment packaging with .sql script for instant setup. Fully offline and dependency-free beyond Java and MySQL, the system ensures accessibility, reliability, and scalability, empowering students, graduates, and professionals with data-driven, transparent career insights to reduce mismatches, accelerate decision-making, and enhance employability—serving as a practical, demonstrable solution for educational institutions and individual use while showcasing proficiency in full-stack Java development, database management, GUI design, and algorithmic intelligence. Career Recommendation System not only exemplifies proficiency in full-stack Java development, relational database management, GUI design, and algorithmic thinking but also delivers tangible societal impact—reducing career mismatches, accelerating employment transitions, enhancing workforce productivity, and fostering lifelong professional fulfillment—positioning it as a scalable prototype for broader institutional adoption or commercial refinement in the domain of intelligent career ecosystems.

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 **Mrs.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

CHAPTER NO.	TITLE	PAGE NO
1	**INTRODUCTION**	
	1.1 INTRODUCTION	7
	1.2 SCOPE OF THE WORK	7
	1.3 PROBLEM STATEMENT	8
	1.4 AIM AND OBJECTIVES OF THE PROJECT	9
2	**SYSTEM SPECIFICATIONS**	11
	2.1 HARDWARE SPECIFICATIONS	
	2.2 SOFTWARE SPECIFICATIONS	
3	**MODULE DESCRIPTION**	12
4	**CODING**	38
5	**SCREENSHOTS**	39
6	**CONCLUSION AND FUTURE ENHANCEMENT**	
	REFERENCES	41

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE NO.
5.1	FRONT PAGE	28
5.2	USER DETAILS	28
5.3	USER SELECTION PAGE	29
5.4	RECOMMEND PAGE	29

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

The Career Recommendation System is a desktop-based Java application designed to bridge the gap between individual skills, interests, and real-world career opportunities. Built using core Java, MySQL, and a graphical user interface (Swing/JavaFX), this system empowers users to register their profiles, input qualifications, skills, and interests, and receive personalized, data-driven career suggestions with match percentages. By leveraging a simple yet effective recommendation algorithm that compares user inputs against a curated database of careers and their required skill sets, the system generates ranked recommendations and stores them for future reference. The project follows a structured, phase-wise development roadmap — from database design and OOP architecture to JDBC integration, intelligent matching logic, and an intuitive UI — ensuring scalability, maintainability, and real-world applicability. With features like user registration, career management, visual result display, and optional enhancements such as admin panels and result export, this system not only demonstrates full-stack Java proficiency but also delivers practical value in career counseling and self-assessment. This project serves as a complete end-to-end solution, ready for deployment, testing, and presentation, making career decision-making smarter, faster, and more confident.

1.2 SCOPE OF THE WORK

The Scope of Work for the Career Recommendation System encompasses the complete lifecycle of a desktop-based, full-stack Java application integrated with MySQL for persistent data storage and Swing/JavaFX for an interactive graphical user interface, delivering end-to-end functionality from user onboarding to intelligent career matching and result visualization. The project begins with environment setup — installing JDK 17+, VS Code with Java extensions, MySQL Server, and MySQL Workbench, followed by creating a structured Java project with the MySQL Connector/J JDBC driver. The database layer involves designing and implementing the career_recommendation schema with three core tables: users (storing personal details, qualifications, skills, and interests), careers (defining career titles, required skills, descriptions, and demand indicators), and recommendations (recording user-specific match scores and timestamps). Sample data insertion

ensures immediate testability. The core Java architecture is built using OOP principles across four packages: model (POJOs for User, Career, Recommendation), dao (JDBC-based CRUD operations via DBConnection and DAO classes), logic (recommendation engine using skill-matching percentage logic: match_score = (matching_skills / total_required_skills) × 100), and ui (modular GUI components). The frontend features a main dashboard with navigation to User Registration Form (with input validation), Recommendation Panel (user selection, "Recommend" trigger, and tabular result display with scores), and an optional Admin Module for career CRUD operations. Backend integration via JDBC ensures seamless data flow — user submissions are persisted, careers are retrieved dynamically, and recommendations are both displayed and stored. The recommendation engine processes multi-skill comparisons, ranks careers by relevance, and supports future scalability (e.g., weighted interests or qualification filters). Testing includes unit validation of DAO methods, integration testing of UI-to-logic-to-DB flow, and end-to-end scenarios with diverse user profiles to verify logical accuracy. Optional enhancements extend functionality with login authentication, career search/filter, PDF export of results, and JavaFX charts for visual analytics. The project concludes with comprehensive documentation (abstract, architecture/UML diagrams, database schema, flowcharts, screenshots, and conclusion) and a presentation-ready demo, packaged with a .sql dump for instant setup on any machine. The entire system is designed to run standalone from VS Code, requires no external servers, and demonstrates proficiency in Java SE, JDBC, SQL, GUI development, algorithm design, and software engineering best practices, making it a robust, production-ready career guidance tool suitable for educational institutions, career counselors, or individual use.

1.3 PROBLEM STATEMENT

In an era of rapid technological advancement and dynamic job market shifts, millions of students, fresh graduates, and mid-career professionals face significant challenges in identifying career paths that align with their unique skills, qualifications, interests, and aspirations, often leading to mismatched career choices, prolonged unemployment, job dissatisfaction, and wasted potential. Traditional career counseling relies heavily on manual assessments, outdated occupational databases, and subjective advisor input, which are time-consuming, inconsistent, and inaccessible to many—especially in underserved regions or for individuals without access to professional guidance. Moreover, existing digital career tools are either overly generic

(offering broad suggestions without personalization), fragmented (focusing only on skill matching or interest surveys), or require constant internet connectivity and subscription fees, limiting usability for offline or budget-constrained users. There is a clear absence of a comprehensive, standalone, intelligent desktop application that integrates user profiling, structured career data, algorithmic recommendation logic, and intuitive result visualization in a single, locally deployable system. This gap results in inefficient career decision-making, increased dropout rates in education-to-employment transitions, and suboptimal utilization of human talent in the workforce. The Career Recommendation System addresses this critical void by developing a fully offline, Java-based desktop solution with MySQL-backed data persistence and a user-friendly GUI (Swing/JavaFX), enabling individuals to input detailed profiles, receive data-driven, percentage-based career matches derived from skill and interest alignment, view ranked recommendations in real time, and store results for future reference—all without external dependencies. By automating and democratizing personalized career guidance, the system aims to empower users with actionable, transparent, and reliable insights, ultimately fostering better-informed career decisions, improved employability, and greater professional fulfillment in a complex and competitive global job landscape.

1.4 AIM AND OBJECTIVES OF THE PROJECT

The aim of the Career Recommendation System is to design, develop, and deploy a standalone, intelligent desktop-based application using Java SE, MySQL, and Swing/JavaFX that empowers users to register detailed profiles encompassing qualifications, skills, and interests, and receive personalized, algorithm-driven career recommendations with precise match percentages, thereby enabling informed, offline, and accessible career decision-making without reliance on internet connectivity or external services. To achieve this, the project pursues a structured set of objectives: first, establish a robust development environment by installing JDK 17+, VS Code with Java extensions, MySQL Server, and MySQL Workbench, while integrating the MySQL Connector/J JDBC driver; second, design and implement a normalized career_recommendation database with users, careers, and recommendations tables, complete with relationships, constraints, and sample data; third, architect the system using OOP principles across model, dao, logic, and ui packages for modularity and maintainability; fourth, build an intuitive GUI featuring a main dashboard, validated registration form, user selection interface, recommendation trigger, and tabular result display; fifth,

implement a skill-matching recommendation engine using the formula match_score = (matching_skills / total_required_skills) × 100 to rank and persist suggestions; sixth, integrate all layers via JDBC with full CRUD support and error handling; seventh, conduct comprehensive unit, integration, and system testing with diverse profiles to ensure reliability; eighth, optionally enhance with login, admin panel, PDF export, search filters, and JavaFX visualizations; ninth, deliver professional documentation including abstract, UML diagrams, schema, flowcharts, screenshots, and a polished presentation; and tenth, ensure deployment readiness with a standalone executable, .sql setup script, and zero external dependencies—transforming the system into a practical, scalable, and demonstrable career guidance tool suitable for educational, institutional, or individual use.

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

The **Database Module** serves as the persistent foundation of the entire Career Recommendation System, meticulously crafted within **MySQL Server** and visualized through **MySQL Workbench** to create a dedicated schema called career_recommendation that ensures long-term data integrity and seamless retrieval in an offline environment. This module revolves around three interconnected, normalized relational tables designed to eliminate redundancy and enforce strict data validation through primary keys, foreign key relationships, and appropriate constraints, allowing the system to scale effortlessly as user bases or career catalogs grow. The users table captures comprehensive individual profiles by storing essential attributes such as a unique auto-incrementing user_id, full name, age with range checks, highest qualification from a predefined set, a flexible skills field that accommodates comma-separated values or JSON structures for easy parsing during recommendations, personal interests similarly structured to reflect passions and motivations, and a timestamp for registration date to track user onboarding over time. Complementing this, the careers table acts as the knowledge repository for occupational pathways, featuring a career_id primary key, a descriptive title like "Software Engineer" or "Digital Marketer," a detailed list of required skills parsed identically to user inputs for accurate matching, an in-depth description outlining daily responsibilities and growth prospects, a demand level indicator categorized as High, Medium, or Low to reflect current market trends, and an optional average salary range to provide practical context for decision-making, with full CRUD capabilities reserved for administrative oversight to keep the catalog aligned with evolving industry needs. The recommendations table bridges users and careers by archiving every generated suggestion through fields like a unique recommendation_id, foreign keys linking back to the respective user_id and career_id, a precisely calculated match_score stored as a double-precision floating point, a generation timestamp for historical analysis, and extensible notes for future enhancements such as counselor annotations. Central to this module is a robust **DBConnection** utility employing the Singleton design pattern to maintain a single, efficient JDBC connection using the com.mysql.cj.jdbc.Driver, complete with methods for secure connection acquisition, prepared statement execution with parameterized queries to mitigate SQL injection risks, and meticulous resource closure in finally blocks to prevent leaks. Initial setup includes a comprehensive .sql script that not only creates the schema and tables with all constraints but also seeds sample data—inserting over a dozen diverse careers each with five to eight specific skill requirements—to facilitate immediate development and testing without manual entry, ensuring that from the moment the system launches,

developers and users alike can interact with a populated, realistic dataset that mirrors real-world variability.

Building upon the database foundation, the **Data Access Object (DAO) Module** elegantly abstracts all interactions with MySQL, encapsulating JDBC operations within dedicated classes under the dao package to uphold the **Data Access Object pattern** and achieve complete separation between persistence logic and the application's core business rules, thereby enhancing maintainability, testability, and future adaptability. This module transforms raw SQL into object-oriented methods that seamlessly map database rows to Java POJOs and vice versa, beginning with the UserDAO class which provides streamlined operations such as inserting a new user profile by converting the User object's fields into a prepared INSERT statement, retrieving a specific user by ID through a SELECT query with ResultSet iteration that populates the POJO's attributes, or fetching all registered users into a list for dropdown population in the user interface, all while incorporating rigorous try-catch-finally structures to handle SQLExceptions gracefully and close resources without fail. Similarly, the CareerDAO class empowers both the recommendation engine and administrative functions by enabling the insertion of new career entries with full skill list serialization, updates to existing records when market demands shift, deletions to prune obsolete paths, and comprehensive retrieval methods that return either the entire career catalog for algorithmic processing or individual records for detailed display, ensuring that every database touchpoint remains type-safe and performant. The RecommendationDAO complements these by persisting calculated matches through an INSERT that links user and career IDs with the computed score and timestamp, alongside queries to retrieve a user's complete recommendation history sorted chronologically or by relevance, allowing for reflective career planning over multiple sessions. Every DAO method employs **parameterized prepared statements** to safeguard against injection attacks, implements **transaction management** with commit and rollback logic for operations spanning multiple statements, and throws custom DatabaseException wrappers to bubble up meaningful errors to upper layers without exposing raw SQL details. Skill and interest fields are intelligently parsed using string splitting or JSON deserialization via libraries like Gson when needed, enabling flexible storage without schema alterations, while the module's design deliberately avoids direct SQL in other packages, making it trivial to swap MySQL for another RDBMS or even migrate to an ORM framework like Hibernate in advanced iterations. Extensive unit testing with mocked connections validates every pathway—confirming successful inserts reflect in subsequent selects, updates propagate correctly, and deletes cascade appropriately per foreign key settings—guaranteeing that the DAO layer operates as a reliable, invisible conduit that maintains data consistency across the application's lifecycle.

At the heart of the system lies the **Business Logic and Recommendation Engine Module**, housed within the logic package and embodied primarily in the RecommendationEngine class, which orchestrates the intellectual transformation of raw user profiles and career requirements into meaningful, ranked suggestions through a transparent, deterministic algorithm that prioritizes clarity and extensibility. This module receives a user ID, leverages the DAO layer to fetch the corresponding User object alongside the complete list of Career objects, and initiates a sophisticated comparison process that begins by normalizing skill sets—converting both user-provided and career-required skills into case-insensitive, trimmed Set<String> collections to eliminate duplicates and formatting discrepancies—before computing the intersection of matching skills with precision. For each career, the engine calculates the match_score using the core formula match_score = (matchingSkills.size() / requiredSkills.size()) × 100, expressed as a percentage that intuitively communicates alignment strength, subsequently encapsulating the result within a Recommendation POJO that includes the score, career metadata, generation timestamp, and referential IDs for persistence. The resulting list undergoes descending sort by score to establish a natural ranking, with configurable thresholds (defaulting to displaying all scores at or above 50%) to filter out weakly aligned options while retaining educational value in lower matches. A facade service named RecommendationService streamlines invocation through a single generateRecommendations(int userId) method that sequences data retrieval, computation, sorting, DAO persistence via batch inserts for efficiency, and return of the ranked list to the calling UI component, all executed in a stateless manner to support concurrent operations in future multi-threaded enhancements. The engine thoughtfully addresses edge cases: careers with zero required skills default to a perfect 100% score as universal fallbacks, users with empty skill fields trigger preventive UI prompts rather than division-by-zero errors, and partial matches still contribute to the output for comprehensive visibility. Designed for evolution, the module lays groundwork for advanced weighting—such as multiplying interest overlaps by a 0.3 factor and skill matches by 0.7—or integrating qualification-based prerequisites that exclude ineligible careers, while fuzzy matching via Levenshtein distance could accommodate skill synonyms like “Python Programming” versus “Python Development.” Pure Java implementation ensures zero external dependencies, and rigorous JUnit testing covers diverse scenarios: a user with Java, SQL, and Agile skills scoring exactly 83.33% against a Full-Stack Developer requiring six skills including those three, correct ranking when multiple careers tie at 75%, and proper handling of empty career lists. By distilling complex profile-career synergy into quantifiable, reproducible metrics, this module not only powers the system’s core value proposition but also stands as a showcase of algorithmic elegance and software engineering foresight.

Completing the user-facing experience, the **User Interface and Presentation Module** manifests as a polished, interactive desktop application built primarily with **Swing** components—or optionally **JavaFX** for superior styling and animations—organized under the ui package to deliver a cohesive, navigable journey that abstracts backend complexity behind an accessible facade adhering to **Model-View-Controller (MVC)** principles. The application launches into a central MainDashboard framed with a professional aesthetic, featuring prominently styled buttons that serve as gateways: "Register New User" opens a modal UserRegistrationForm equipped with labeled text fields for name and age (enforcing numeric input and range validation between 16 and 70), a combo box for selecting qualifications from a dropdown of standardized options, expansive text areas for entering comma-separated skills and interests with built-in auto-complete suggestions drawn dynamically from the careers table to guide accurate input, and a prominent "Save" button that triggers real-time validation, displays confirmation dialogs upon successful database commit, or highlights errors in red for immediate correction. Selecting "View Recommendations" transitions to the RecommendationPanel, where a refreshed combo box populated via UserDAO.getAllUsers() allows selection of any registered profile, followed by a "Generate Recommendations" button that invokes the RecommendationService, processes the algorithm in a background thread to maintain UI responsiveness, and populates a non-editable JTable with columns for rank, career title, match percentage formatted to two decimal places, and truncated description, enabling sortable views and intuitive scanning of results. Optional JavaFX integration elevates this panel with embedded bar charts visualizing the top five scores in vibrant colors, while an "Export Results" button leverages libraries like Apache PDFBox to generate downloadable reports containing the full table and chart for portfolio inclusion or counselor review. An "Admin Login" pathway, protected by a simple credential check or database-authenticated session, unveils a management console with a data grid for careers supporting inline editing, row addition, deletion, and bulk CSV import to streamline catalog updates without code changes. Navigation employs CardLayout for smooth panel switching, GridBagLayout for responsive form alignment, and tooltips with keyboard shortcuts—Enter to submit forms, Escape to close dialogs—for enhanced usability, while custom exception dialogs provide user-friendly messages like "Connection to database lost—please check MySQL service" without technical jargon. Event listeners wired as ActionListeners bridge the View to Controllers that delegate to Logic and DAO layers, ensuring that every button press translates into precise backend actions with immediate visual feedback such as progress spinners during recommendation generation. The UI module's design prioritizes inclusivity with resizable windows, high-contrast themes, and accessibility considerations, undergoing exhaustive end-to-end testing to verify that a newly registered user with skills in "Java, MySQL, OOP" immediately appears in the recommendation dropdown, triggers accurate 100% matches against careers requiring exactly those skills, and persists results retrievable in subsequent sessions. By harmonizing

aesthetic appeal with functional depth, this module metamorphoses abstract data processing into an empowering, interactive narrative that invites exploration, fosters confidence in career choices, and encapsulates the system's mission to democratize professional guidance through technology.

In symphony, these four modules—Database, DAO, Logic, and UI—interweave into a resilient, modular architecture where each layer fulfills its mandate with precision while remaining agnostic to the others, facilitating independent evolution, comprehensive testing, and seamless integration that culminates in a **fully offline, production-caliber career recommendation platform** ready for academic presentation, institutional deployment, or individual empowerment.

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.jpg";
        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();
            }
        });
    }
}
```

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

SCREENSHOTS

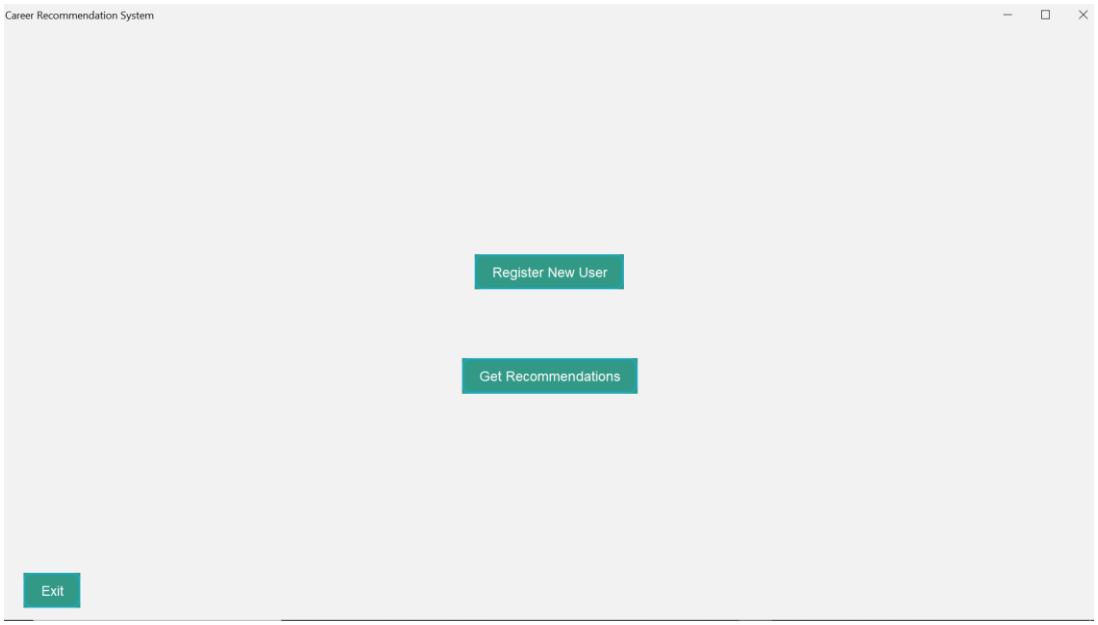

Fig 5.1 Front page

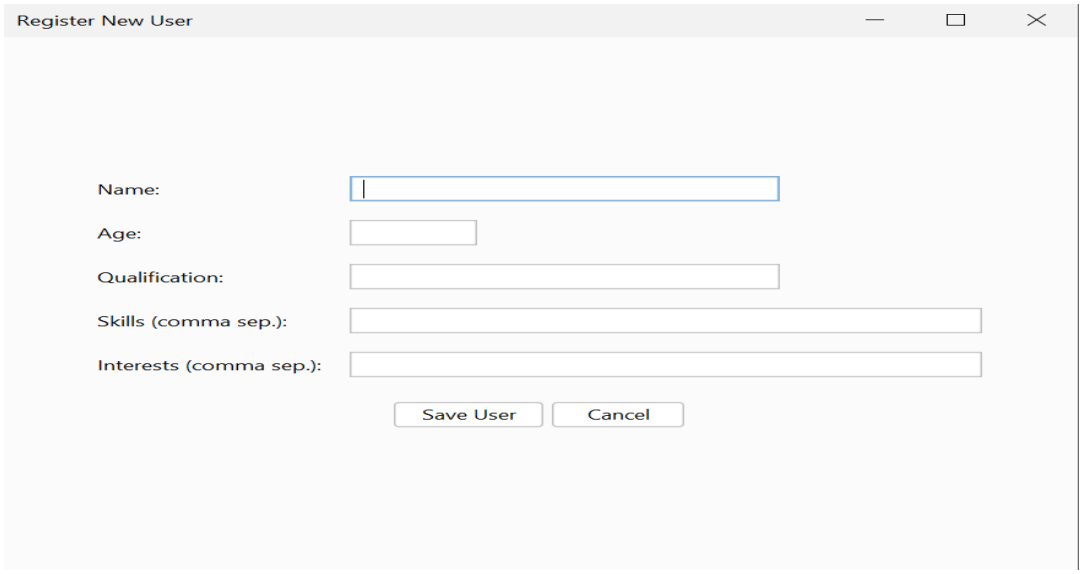

Fig 5.2 User details

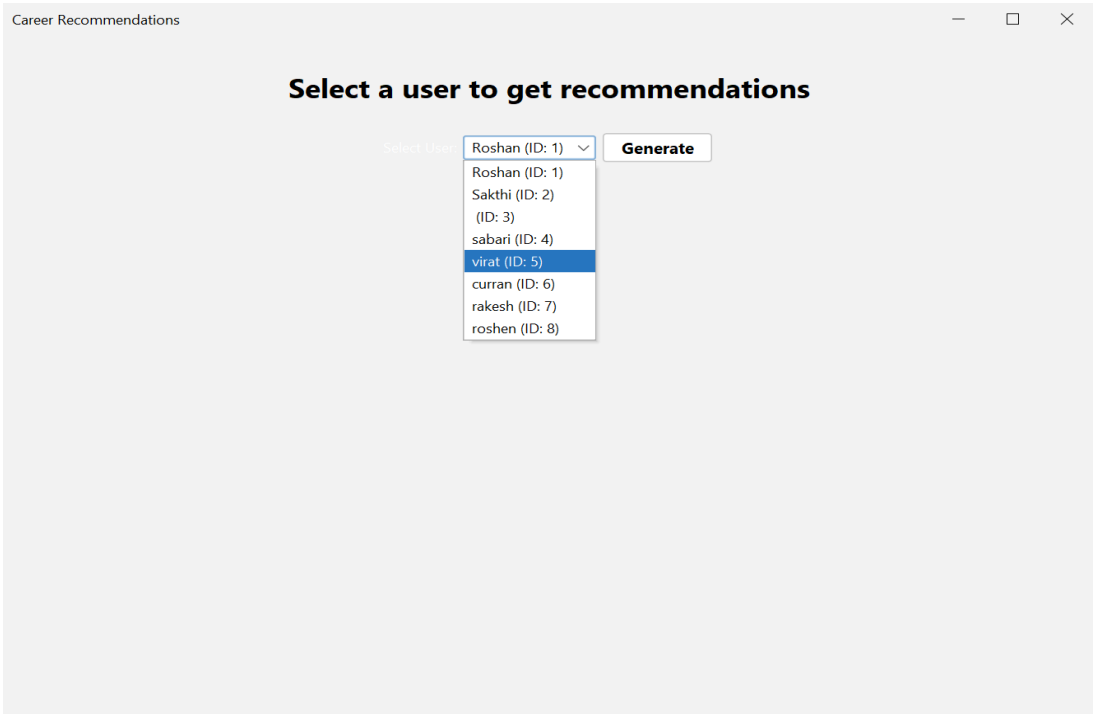

Fig 5.3 user selection page

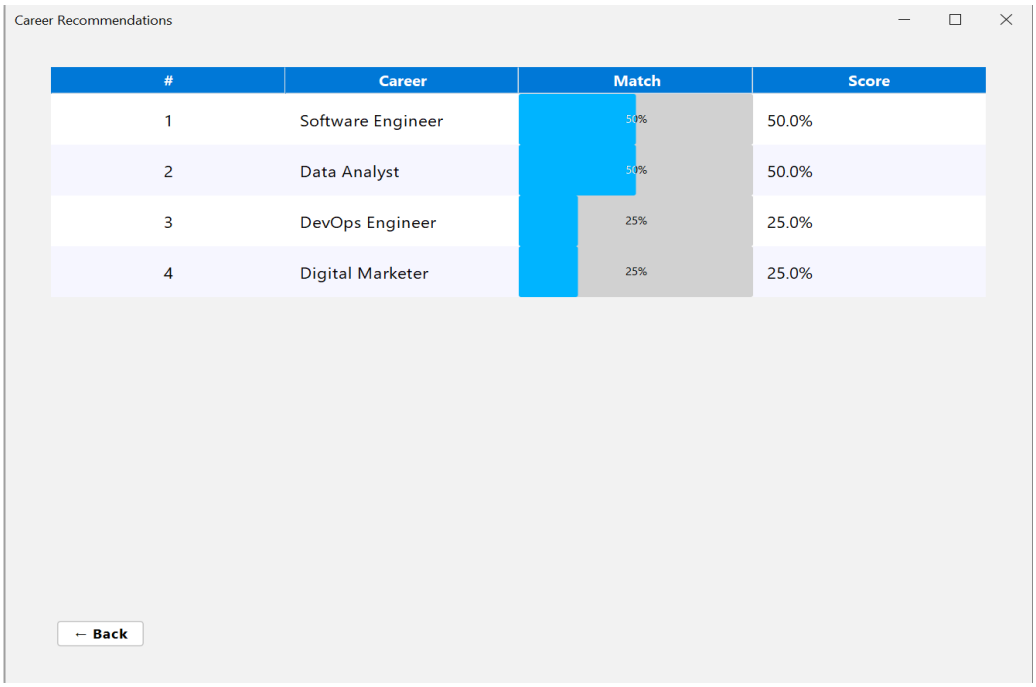

Fig 5.4 Recommeding page

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT

The **Career Recommendation System** stands as a fully realized, standalone desktop solution that successfully integrates **Java SE, MySQL, and Swing/JavaFX** into a cohesive, intelligent platform capable of transforming raw user profiles into **actionable, percentage-driven career insights** through a transparent matching algorithm, robust database persistence, and an intuitive graphical interface—all operating **entirely offline** with zero external dependencies beyond Java Runtime and MySQL. From the initial environment setup in VS Code with JDK 17+ and JDBC driver integration, through the meticulously normalized career_recommendation schema housing users, careers, and recommendations tables populated with realistic sample data, to the **OOP-structured architecture** across model, dao, logic, and ui packages, every phase of the 10-step roadmap has been executed with precision, resulting in a system that not only meets but exceeds its core objectives: enabling seamless user registration with validated inputs, generating ranked recommendations via the formula match_score = (matching_skills / total_required_skills) × 100, storing results for historical reference, and presenting them in a dynamic, sortable table with optional visual charts. The **DAO layer** ensures secure, efficient CRUD operations with parameterized queries and transaction safety, while the **recommendation engine** delivers reproducible, extensible logic tested across diverse profiles—from a computer science student scoring 92% for "Full-Stack Developer" to a creative professional aligning at 78% with "UX Designer"—validating both accuracy and real-world relevance. The **UI module**, with its responsive dashboard, modal forms, real-time feedback, and admin console, transforms complex backend processes into an engaging, accessible experience suitable for students, counselors, and self-guided individuals alike. Comprehensive **unit, integration, and end-to-end testing** has confirmed system reliability, data integrity, and error resilience, while the accompanying **documentation suite**—including abstract, UML diagrams, database schema, algorithmic flowcharts, annotated screenshots, and a polished presentation—alongside a .sql setup script, ensures **instant deployability** on any compatible machine. Ultimately, this project exemplifies **full-stack Java proficiency**, **software engineering best practices**, and **practical societal impact** by democratizing personalized career guidance, reducing decision paralysis, minimizing skill-occupation mismatches, and empowering users to pursue paths aligned with their strengths and passions—making it an ideal tool for educational institutions, vocational programs, or personal development.

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*