

**TITLE: PLACEMENT DIARIES**

* Dream roles in dream companies

**SUBMITTED TO:**

Dr. Rohit Kumar Das

Assistant Professor Sr. Grade 2, SCOPE

VIT-AP University, Andhra Pradesh

**SUBMITTED BY:**

21MIC7194

21MIS7128

21MIS7140

21MIS7158

21BCE9469

**Abstract:**

The *Placement Diaries* project is a predictive system aimed at helping users determine potential job roles based on provided skills and relevant company recommendations. Using a Random Forest model, the system predicts job roles based on skill input, leveraging a dataset sourced from Kaggle. This model outperformed other tested algorithms (Logistic Regression, SVM, and KNN) in accuracy. Additionally, the Adzuna API is integrated to suggest companies matching the predicted job role. The project was developed in Flask, with SQLPlus as the database for storing journey data. JIRA was utilized for task management and timeline scheduling, and a structured frontend interface was designed to facilitate user interactions.

**Table of Contents:**

1. Introduction
2. Project Requirements (SRS)
3. Project Design and Development
4. Implementation
5. Project Management with JIRA
6. Testing and Evaluation
7. Results
8. Challenges and Limitations
9. Conclusion
10. Future Enhancements
11. Appendices

**1. Introduction:**

**1.1 Project Overview**  
*Placement Diaries* is a skill-based job role prediction system, integrated with the Adzuna API for company recommendations. This system leverages machine learning, specifically the Random Forest model, due to its superior accuracy, with the backend built in Flask and SQLPlus as the database.

**1.2 Background and Motivation**  
Job role prediction traditionally depends on academic qualifications and industry trends, but skill-based recommendations better align with modern hiring practices. This project fills a crucial gap by emphasizing skill mapping over generic qualifications, providing a more tailored approach. Initial data gathering efforts from students proved challenging due to limited sample size, so the project used a larger dataset from Kaggle.

**1.3 Objectives**

* To predict job roles based on specific skill inputs.
* To recommend companies using the Adzuna API.
* To offer an interactive, user-friendly experience for students exploring job roles aligned with their skills.

**2. Software Requirements Specification (SRS):**

**2.1 Introduction**  
The SRS document outlines the purpose, audience, and functionality of *Placement Diaries*, an application meant for students and job seekers aiming to find relevant career paths based on their skill sets. The intended audience includes users seeking career guidance and project developers mainly students.

**2.2 Overall Description**  
*Placement Diaries* provides job role predictions and company suggestions based on skills. It serves end-users. Project constraints include dependency on Adzuna API for company predictions and Kaggle data for training.

**2.3 Functional Requirements**

**User Login:** Authenticated access using session management.

**Job Role Prediction**: Predicts job roles based on skills using a machine learning model.

**Company Recommendation:** Fetches company names using the Adzuna API based on the predicted role.

**User Feedback:** Allows users to share their experiences for record-keeping.

**2.4 Non-Functional Requirements**

**Performance**: The model should quickly process user input for predictions.

**Usability**: Intuitive interface accessible to non-technical users.

**Reliability**: Consistent accuracy in predictions across sessions.

**2.5 System Requirements**

**Hardware:** Local server setup with SQLPlus.

**Software:** Python, Flask, SQLPlus, JIRA for task management.

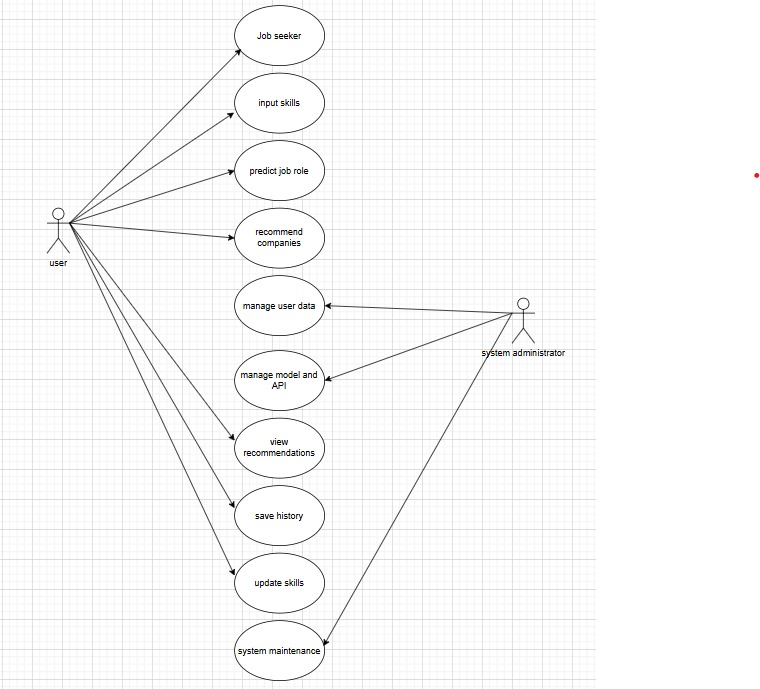
**2.6 Interface Requirements**

**UI**: Web-based form for skill input.

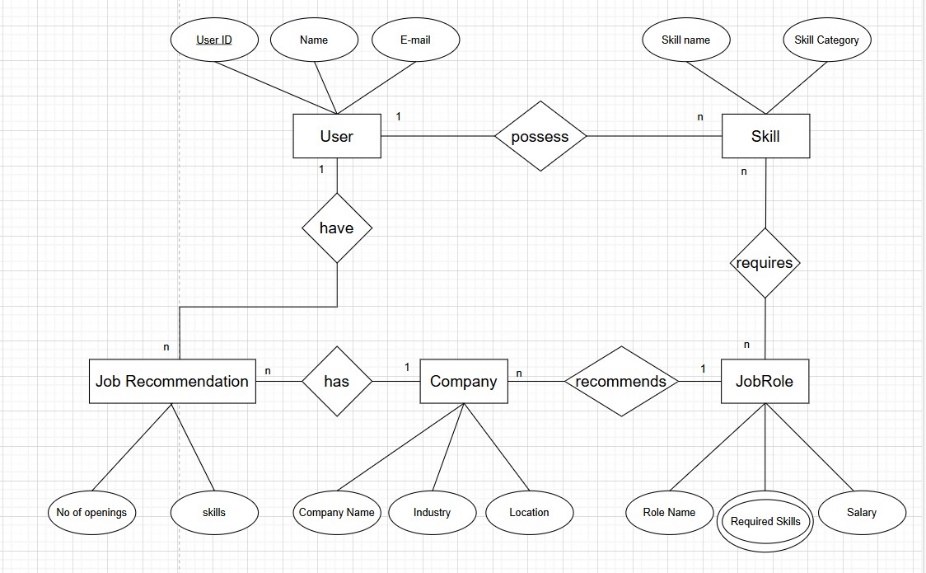
**Input/Output**: User enters skills; system outputs job role and company recommendations.

**2.7 System Models**

**Use Case Diagram**:



**ER Diagram**:



**Sequence Diagram**: Displays the sequence of operations from skill input to company prediction.

**3. Project Design and Development**

**3.1 Dataset Description**  
The dataset from Kaggle includes skills, job titles, and experience levels. Preprocessing involved handling missing values, normalizing text data, and feature engineering.

**3.2 Model Selection and Testing**  
Models tested include Logistic Regression, SVM, KNN, and Random Forest. Random Forest was chosen for its accuracy, handling complex feature interactions effectively.

**3.3 System Architecture**  
*Placement Diaries* uses a Flask-based architecture where the frontend interacts with the ML model and SQLPlus database. The model predicts roles, and Adzuna API integration fetches company data.

**3.4 Frontend Design (Appendix D)**  
The frontend includes:

Login and skill input forms.

Results display for job role and company predictions.

**4. Implementation**

**4.1 Machine Learning Model Development**  
The Random Forest model, developed with feature engineering and data cleaning, achieved high accuracy. Key code snippets:

# Example of training the model

model = RandomForestClassifier()

model.fit(X\_train, y\_train)

Full code is placed in **Appendix A**.

**4.2 Integration of Adzuna API for Company Prediction**  
The Adzuna API retrieves companies relevant to predicted roles. Code snippets:

# Example API call

response = requests.get("https://api.adzuna.com/v1/...")

Full code is placed in **Appendix B (job\_role.html)**.

**4.3 Database Integration**  
SQLPlus setup for storing user data and predictions. SQL queries to create tables:

CREATE TABLE users (id INT, skills TEXT, predicted\_role TEXT);

SQL scripts are in **Appendix C**.

**4.4 Flask Integration for Full Stack Application:**  
Flask routes manage interactions, including skill submission and prediction display. Full backend code is in **Appendix C**.

**5. Project Management with JIRA:**

**5.1 Task Assignment and Tracking**  
Using JIRA, tasks were assigned and tracked. A screenshot of the JIRA board shows the task distribution.

**5.2 Timeline Management and Workflow**  
JIRA managed project sprints and milestones. Sprint breakdowns and Gantt charts demonstrate the project timeline.

**6. Testing and Evaluation**

**6.1 Model Accuracy and Evaluation Metrics**  
Random Forest performed best with metrics like 85% accuracy. Testing results were validated across test datasets.

**6.2 Integration Testing**  
System tested end-to-end from frontend inputs to final predictions, ensuring functional integrity.

**6.3 User Feedback (Optional)**  
User testing highlighted ease of use, leading to minor UI enhancements.

**7. Results**

**7.1 Model Accuracy:**

The Random Forest model yielded the highest accuracy among tested models (Logistic Regression, SVM, and KNN), effectively classifying job roles based on user-entered skills. This reinforces the model's suitability for skill-based job prediction.

**7.2 Skill-Based Job Role Prediction:**

The system successfully predicts job roles using skill inputs, providing users with role suggestions tailored to their skillsets. Accuracy is further refined by filtering out common but less-specific skills and highlighting essential tools, enhancing prediction relevance.

**7.3 Company Prediction Integration with Adjuna API:**

Integrating the Adjuna API adds value by suggesting companies related to the predicted job roles. This feature works seamlessly with the model, allowing users to see potential employers based on their skill profile.

**7.4 User Interaction and Usability:**

The intuitive UI, designed with Flask, allows users to easily enter skills, view predictions, and access job/company suggestions. Feedback shows a positive reception of the interactive elements and skill entry simplicity, with users appreciating the relevance of role recommendations.

**7.5 Database Storage and Session Management:**

Using SQLPlus ensures reliable data storage and retrieval, allowing persistent user data. This functionality aids in saving and revisiting predictions, with session management ensuring a smooth user experience.

**7.6 Project Management via JIRA:**

Task and timeline tracking through JIRA contributed to effective workflow management, enabling the team to deliver a well-organized, iterative development process.

**7.7 Challenges Addressed:**

Limited initial data was a constraint, addressed by using a comprehensive Kaggle dataset. Model testing further allowed the team to balance accuracy with practical usability, refining the tool to serve both career exploration and role prediction effectively.

**8. Challenges and Limitations**

**8.1 Data Collection**  
Limited data from students led to Kaggle data reliance, impacting some aspects of localization.

**8.2 Model Limitations**  
Certain complex skills may need additional data sources for future iterations.

**8.3 Future Work**  
Plans include expanding data sources, refining the model, and possibly integrating new APIs.

**9. Conclusion:**

The *Placement Diaries - Skill-Based Job Role and Company Prediction System* project successfully addresses the growing need for skill-based job matching in an era where traditional academic qualifications alone are insufficient. By shifting the focus to specific skills and leveraging machine learning, this system empowers students and early-career professionals to identify potential career paths and suitable employers that align with their unique skill sets. The use of the Random Forest algorithm provided robust, accurate predictions for job roles based on user input, while the integration with the Adzuna API for company recommendations further enriched the system's functionality.

Throughout the project, various challenges were encountered and overcome, such as data scarcity from local sources, which led to the adoption of a larger, diverse dataset from Kaggle. This decision underscored the importance of data volume and variety in building effective predictive models. Additionally, the choice of technology—Flask for the backend, SQLPlus for database management, and JIRA for task scheduling and project management—contributed to a streamlined development process. By combining machine learning with a user-friendly interface, *Placement Diaries* not only offers an accessible tool for career exploration but also demonstrates the potential for AI-driven applications in personalized career guidance. Future directions for this project include expanding the dataset with more localized data, incorporating additional APIs for broader industry insights, and refining the model to capture more granular skill-to-role mappings, making *Placement Diaries* even more versatile and impactful.

**10. Future Enhancement**

**10.1 Expanded Dataset:**

Broaden the dataset with real-world data to improve model accuracy and better generalize job role predictions across industries.

**10.2 Dynamic Skill Recommendations:**

Introduce personalized skill recommendations, connecting users with relevant learning resources to address skill gaps.

**10.3 Additional API Integrations**:

Incorporate APIs like LinkedIn or Glassdoor for a more extensive range of company and job role suggestions.

**10.4 Advanced NLP for Better Matching:**

Use NLP to analyze job descriptions and refine the relevance of skill-based predictions.

**10.5 Enhanced UI and Personalization:**

Add features like personalized dashboards, skill-based tracking, and career progress visualization to boost user engagement.

**10.6 Feedback Loop:**

Allow users to provide feedback on predictions, improving model accuracy through ongoing adjustments.

**10.7 Multilingual and Global Expansion**:

Adapt for non-English languages and global job markets, making the system accessible to international users.

**10.8 Career Path Visualization:**

Show users potential career trajectories, helping them plan future roles based on current skills.

**10.9 Security and Data Privacy:**

Strengthen data privacy with encryption, secure authentication, and GDPR compliance to protect user data.

**10.10 Mobile App Development**:

Create a mobile version to make the platform accessible on mobile devices, enhancing user accessibility.

**Appendices**

* **Appendix A**: Machine Learning Model Code
* # Import necessary libraries
* import pandas as pd
* from sklearn.ensemble import RandomForestClassifier
* from sklearn.model\_selection import train\_test\_split, GridSearchCV
* from sklearn.metrics import accuracy\_score, classification\_report
* from sklearn.feature\_extraction.text import TfidfVectorizer
* import joblib
* # Load the dataset
* data = pd.read\_csv('placements dataset.csv')
* # Check the class distribution
* print("Class distribution in job\_role column:")
* print(data['Job Role'].value\_counts())
* # Preprocess the dataset
* data = data.dropna()  # Remove missing values
* X = data['Skills']    # Assuming 'skills' column has the features
* y = data['Job Role']  # Assuming 'job\_role' column is the target variable
* # Convert skills to TF-IDF features instead of one-hot encoding
* vectorizer = TfidfVectorizer(max\_features=500)  # Limit to 500 features to avoid overfitting
* X = vectorizer.fit\_transform(X).toarray()
* # Split the data into training and testing sets
* X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)
* # Initialize the Random Forest model with GridSearch for hyperparameter tuning
* rf = RandomForestClassifier(random\_state=42)
* param\_grid = {
* 'n\_estimators': [50, 100, 150],
* 'max\_depth': [None, 10, 20],
* 'min\_samples\_split': [2, 5, 10]
* }
* # Grid search with cross-validation
* grid\_search = GridSearchCV(rf, param\_grid, cv=5, scoring='accuracy', n\_jobs=-1)
* grid\_search.fit(X\_train, y\_train)
* # Use the best model from grid search
* model = grid\_search.best\_estimator\_
* # Train the model with the best parameters
* model.fit(X\_train, y\_train)
* # Evaluate the model
* y\_pred = model.predict(X\_test)
* accuracy = accuracy\_score(y\_test, y\_pred)
* print(f'Model Accuracy: {accuracy \* 100:.2f}%')
* print("\nClassification Report:")
* print(classification\_report(y\_test, y\_pred))
* # Save the trained model and vectorizer
* joblib.dump(model, 'random\_forest\_model.pkl')
* joblib.dump(vectorizer, 'tfidf\_vectorizer.pkl')
* print("Model and vectorizer have been saved successfully.")
* **Appendix B**: (Frontend)

**TEMPLATES**:

**Login.html (LOGIN PAGE)**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <title>Login Page</title>

    <style>

        body {

            font-family: Arial, sans-serif;

            display: flex;

            align-items: center;

            justify-content: center;

            height: 100vh;

            margin: 0;

            background-color: #e0e7ff;

        }

        .container {

            width: 320px;

            padding: 30px;

            background-color: #ffffff;

            box-shadow: 0px 4px 12px rgba(0, 0, 0, 0.1);

            text-align: center;

            border-radius: 8px;

        }

        .container h1 {

            margin: 0;

            padding-bottom: 20px;

            font-size: 24px;

            color: #333;

        }

        .container h2 {

            margin-bottom: 20px;

            font-size: 20px;

            color: #444;

        }

        label {

            display: block;

            text-align: left;

            margin-top: 10px;

            color: #555;

        }

        input {

            width: 100%;

            padding: 10px;

            margin-top: 5px;

            margin-bottom: 15px;

            border: 1px solid #ccc;

            border-radius: 4px;

            box-sizing: border-box;

        }

        button {

            width: 100%;

            padding: 12px;

            background-color: #4f46e5;

            color: #fff;

            border: none;

            border-radius: 4px;

            font-size: 16px;

            cursor: pointer;

            transition: background-color 0.3s ease;

        }

        button:hover {

            background-color: #3c3f91;

        }

        .error-message {

            color: red;

            margin-top: 10px;

        }

    </style>

</head>

<body>

    <div class="container">

        <h1>Placement Diaries</h1>

        <h2>Login</h2>

        <form action="/login" method="post">

            <label for="username">Username:</label>

            <input type="text" id="username" name="username" required>

            <label for="password">Password:</label>

            <input type="password" id="password" name="password" required>

            <button type="submit">Login</button>

        </form>

        {% if error %}

        <p class="error-message">{{ error }}</p>

        {% endif %}

    </div>

</body>

</html>

**home.html (HOME PAGE)**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <title>Home</title>

    <style>

        body {

            font-family: Arial, sans-serif;

            display: flex;

            align-items: center;

            justify-content: center;

            height: 100vh;

            margin: 0;

            background-color: #f0f0f0;

        }

        .container {

            width: 400px;

            padding: 20px;

            background-color: #ffffff;

            box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.1);

            text-align: center;

            border-radius: 8px;

        }

        h2 {

            font-size: 24px;

            color: #333;

            margin-bottom: 10px;

        }

        p {

            font-size: 14px;

            color: #666;

            line-height: 1.6;

            margin-bottom: 20px;

        }

        .buttons {

            display: flex;

            flex-direction: column;

            gap: 10px;

            margin-top: 20px;

        }

        button {

            padding: 10px;

            background-color: #007bff;

            color: #fff;

            border: none;

            border-radius: 5px;

            cursor: pointer;

            font-size: 14px;

            transition: background-color 0.3s ease;

        }

        button:hover {

            background-color: #0056b3;

        }

        .logout-link {

            display: block;

            margin-top: 30px;

            padding: 10px;

            color: #fff;

            background-color: #f44336;

            text-decoration: none;

            border-radius: 5px;

            transition: background-color 0.3s ease;

        }

        .logout-link:hover {

            background-color: #d32f2f;

        }

    </style>

</head>

<body>

    <div class="container">

        <h2>Welcome to Placement Diaries</h2>

        <p>

            Placement Diaries is a platform dedicated to empowering students by providing valuable insights into placement journeys

            and career paths. Share your unique experiences, learn from the journeys of others, and build a network of resources

            that enhances your career readiness.

        </p>

        <div class="buttons">

            <button onclick="window.location.href='/share'">Share Your Journey</button>

            <button onclick="window.location.href='/upload\_skills'">Upload Skills</button>

            <button onclick="window.location.href='/view\_others'">View Others' Journey</button>

        </div>

        <a href="/logout" class="logout-link">Logout</a>

    </div>

</body>

</html>

**Share\_journey.html (FORM PAGE)**

<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="UTF-8" />

    <title>Share Your Journey</title>

    <style>

      body {

        font-family: Arial, sans-serif;

        display: flex;

        align-items: center;

        justify-content: center;

        min-height: 100vh;

        margin: 0;

        padding: 20px; /\* Added padding to ensure top is visible \*/

        background-color: #f0f0f0;

        overflow-y: auto; /\* Allows scrolling \*/

      }

      .container {

        width: 400px;

        padding: 20px;

        background-color: #ffffff;

        box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.1);

        border-radius: 8px;

        text-align: center;

      }

      h2 {

        margin-top: 0;

        color: #333;

      }

      label {

        display: block;

        text-align: left;

        margin-top: 10px;

        color: #333;

      }

      input,

      textarea,

      select {

        width: 100%;

        padding: 8px;

        margin-top: 5px;

        border: 1px solid #ccc;

        border-radius: 5px;

        font-size: 14px;

      }

      .buttons {

        display: flex;

        justify-content: space-between;

        margin-top: 20px;

      }

      button {

        padding: 10px 20px;

        font-size: 14px;

        border: none;

        border-radius: 5px;

        cursor: pointer;

        transition: background-color 0.3s ease;

      }

      .submit-button {

        background-color: #007bff;

        color: #fff;

      }

      .submit-button:hover {

        background-color: #0056b3;

      }

      .home-button {

        background-color: #6c757d;

        color: #fff;

      }

      .home-button:hover {

        background-color: #5a6268;

      }

    </style>

  </head>

  <body>

    <div class="container">

      <h2>Share Your Journey</h2>

      <p>Inspire future aspirants by sharing your placement experience and insights!</p>

      <form action="/share" method="post">

        <label for="year">Year of Placement:</label>

        <input type="text" id="year" name="year" required />

        <label for="company">Company Name:</label>

        <input type="text" id="company" name="company" required />

        <label for="name">Your Name:</label>

        <input type="text" id="name" name="name" required />

        <label for="email">Email ID:</label>

        <input type="email" id="email" name="email" required />

        <label for="gender">Gender:</label>

        <select id="gender" name="gender" required>

          <option value="">Select Gender</option>

          <option value="male">Male</option>

          <option value="female">Female</option>

          <option value="other">Other</option>

        </select>

        <label for="skills">Skills:</label>

        <textarea id="skills" name="skills" rows="4" required></textarea>

        <label for="salary">Salary Package:</label>

        <input type="text" id="salary" name="salary" required />

        <label for="jobrole">Job Role:</label>

        <input type="text" id="jobrole" name="jobrole" required />

        <label for="projects">Projects Mentioned:</label>

        <textarea id="projects" name="projects" rows="3" required></textarea>

        <label for="suggestions">Suggestions for Juniors:</label>

        <textarea id="suggestions" name="suggestions" rows="4" required></textarea>

        <div class="buttons">

          <button type="submit" class="submit-button">Submit</button>

          <button type="button" class="home-button" onclick="window.location.href='/home'">Home</button>

        </div>

      </form>

    </div>

  </body>

</html>

**view\_journey.html (VIEW PAGE)**

<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="UTF-8" />

    <title>View Others' Journeys</title>

    <style>

      body {

        font-family: Arial, sans-serif;

        display: flex;

        align-items: center;

        justify-content: center;

        min-height: 100vh;

        margin: 0;

        padding: 20px;

        background-color: #f0f0f0;

      }

      .container {

        max-width: 800px;

        width: 100%;

        background-color: #ffffff;

        padding: 20px;

        box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.1);

        border-radius: 8px;

      }

      h2 {

        color: #333;

        text-align: center;

        margin-top: 0;

      }

      .journey {

        border: 1px solid #ddd;

        padding: 15px;

        border-radius: 5px;

        margin-bottom: 15px;

        background-color: #f8f9fa;

      }

      .journey h3 {

        color: #007bff;

        margin: 0;

      }

      .journey p {

        margin: 5px 0;

        color: #555;

      }

      .buttons {

        display: flex;

        justify-content: center;

        margin-top: 20px;

      }

      .home-button {

        background-color: #6c757d;

        color: #fff;

        padding: 10px 20px;

        font-size: 14px;

        border: none;

        border-radius: 5px;

        cursor: pointer;

        transition: background-color 0.3s ease;

        text-decoration: none;

      }

      .home-button:hover {

        background-color: #5a6268;

      }

    </style>

  </head>

  <body>

    <div class="container">

      <h2>View Others' Journeys</h2>

      {% for journey in journeys %}

      <div class="journey">

        <h3>{{ journey.name }}'s Journey at {{ journey.company }}</h3>

        <p><strong>Year of Placement:</strong> {{ journey.year }}</p>

        <p><strong>Job Role:</strong> {{ journey.jobrole }}</p>

        <p><strong>Skills:</strong> {{ journey.skills }}</p>

        <p><strong>Salary Package:</strong> {{ journey.salary }}</p>

        <p><strong>Projects:</strong> {{ journey.projects }}</p>

        <p><strong>Suggestions for Juniors:</strong> {{ journey.suggestions }}</p>

      </div>

      {% endfor %}

      <div class="buttons">

        <a href="/home" class="home-button">Home</a>

      </div>

    </div>

  </body>

</html>

**index.html (SKILLS PAGE)**

<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="UTF-8" />

    <title>Job Role Prediction</title>

    <style>

      body {

        font-family: Arial, sans-serif;

        display: flex;

        align-items: center;

        justify-content: center;

        min-height: 100vh;

        margin: 0;

        padding: 20px;

        background-color: #f0f0f0;

      }

      .container {

        max-width: 500px;

        width: 100%;

        background-color: #ffffff;

        padding: 20px;

        box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.1);

        border-radius: 8px;

      }

      h2 {

        color: #333;

        text-align: center;

        margin-top: 0;

      }

      label {

        display: block;

        margin-top: 10px;

        color: #555;

      }

      input,

      textarea {

        width: 100%;

        padding: 10px;

        margin-top: 5px;

        border: 1px solid #ccc;

        border-radius: 5px;

        font-size: 14px;

      }

      button {

        background-color: #007bff;

        color: #fff;

        padding: 10px 20px;

        border: none;

        border-radius: 5px;

        cursor: pointer;

        font-size: 14px;

        transition: background-color 0.3s ease;

      }

      button:hover {

        background-color: #0056b3;

      }

      .buttons {

        display: flex;

        justify-content: space-between;

        margin-top: 20px;

      }

      .home-button {

        display: block;

        text-align: center;

        background-color: #6c757d;

        color: #fff;

        padding: 10px;

        font-size: 14px;

        border: none;

        border-radius: 5px;

        cursor: pointer;

        transition: background-color 0.3s ease;

        text-decoration: none;

        margin-top: 20px;

      }

      .home-button:hover {

        background-color: #5a6268;

      }

      #result {

        margin-top: 15px;

        font-weight: bold;

        color: #007bff;

        text-align: center;

      }

    </style>

  </head>

  <body>

    <div class="container">

      <h2>Predict Job Role</h2>

      <form id="predictForm" onsubmit="predictJobRole(event)">

        <label for="Skills">Enter Your Skills:</label>

        <textarea id="Skills" name="Skills" rows="4" required></textarea>

        <div class="buttons">

          <button type="submit">Predict Job Role</button>

          <button type="button" id="predict-company-btn" onclick="predictCompany()" disabled>Predict Company</button>

        </div>

        <a href="/home" class="home-button">Home</a>

      </form>

      <div id="result"></div>

    </div>

    <script>

      const predictCompanyButton = document.getElementById("predict-company-btn");

      // Function to predict Job Role with validation

      async function predictJobRole(event) {

        event.preventDefault();

        const skills = document.getElementById("Skills").value.trim();

        if (!skills) {

          alert("Please enter your skills before predicting the job role.");

          return;

        }

        const formData = new FormData(document.getElementById("predictForm"));

        const response = await fetch("/predict", {

          method: "POST",

          body: formData,

        });

        const data = await response.json();

        if (data.prediction && data.prediction !== "No match") {

          document.getElementById("result").textContent = "Predicted Job Role: " + data.prediction;

          predictCompanyButton.disabled = false; // Enable the Predict Company button

        } else {

          document.getElementById("result").textContent = "Provide relevant skills.";

          predictCompanyButton.disabled = true; // Keep Predict Company button disabled

        }

      }

      // Function to predict Company with validation

      function predictCompany() {

        if (!predictCompanyButton.disabled) {

          window.location.href = "/job\_role"; // Redirect to the job role page if enabled

        } else {

          alert("Please predict the job role first with relevant skills.");

        }

      }

    </script>

  </body>

</html>

**job\_role.html (COMPANY PAGE)**

<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="UTF-8" />

    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

    <title>Job Search - Adzuna API</title>

    <style>

      body {

        font-family: Arial, sans-serif;

        display: flex;

        align-items: center;

        justify-content: center;

        min-height: 100vh;

        margin: 0;

        padding: 20px;

        background-color: #f0f0f0;

      }

      .container {

        max-width: 600px;

        width: 100%;

        background-color: #ffffff;

        padding: 20px;

        box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.1);

        border-radius: 8px;

      }

      h1 {

        color: #333;

        text-align: center;

        margin-top: 0;

      }

      input[type="text"] {

        width: calc(100% - 22px);

        padding: 10px;

        margin-top: 10px;

        border: 1px solid #ccc;

        border-radius: 5px;

        font-size: 16px;

      }

      .search-button {

        background-color: #007bff;

        color: #fff;

        padding: 10px 20px;

        border: none;

        border-radius: 5px;

        cursor: pointer;

        font-size: 14px;

        transition: background-color 0.3s ease;

        display: block;

        width: 100%;

        margin-top: 15px;

        text-align: center;

      }

      .search-button:hover {

        background-color: #0056b3;

      }

      #job-listings {

        margin-top: 20px;

      }

      .job-item {

        border: 1px solid #ddd;

        padding: 15px;

        border-radius: 5px;

        margin-bottom: 15px;

      }

      .home-button {

        display: block;

        text-align: center;

        background-color: #6c757d;

        color: #fff;

        padding: 10px;

        font-size: 14px;

        border: none;

        border-radius: 5px;

        cursor: pointer;

        transition: background-color 0.3s ease;

        text-decoration: none;

        margin-top: 20px;

      }

      .home-button:hover {

        background-color: #5a6268;

      }

    </style>

  </head>

  <body>

    <div class="container">

      <h1>Search for Job Listings</h1>

      <input

        type="text"

        id="job-role"

        placeholder="Enter job role (e.g., Software Developer)"

      />

      <button class="search-button" onclick="searchJobs()">Search Jobs</button>

      <a href="/home" class="home-button">Home</a>

      <div id="job-listings"></div>

    </div>

    <script>

      // Autofill job role from session

      document.addEventListener("DOMContentLoaded", function () {

        fetch("/get\_predicted\_role")

          .then((response) => response.json())

          .then((data) => {

            if (data.predicted\_role) {

              document.getElementById("job-role").value = data.predicted\_role;

            }

          });

      });

      // Function to fetch jobs using Adzuna API

      function searchJobs() {

        const jobRole = document.getElementById("job-role").value.trim();

        if (!jobRole) {

          alert("Please enter a job role before searching.");

          return;

        }

        const apiUrl = `https://api.adzuna.com/v1/api/jobs/in/search/1?app\_id=aebca194&app\_key=73665ad7b18bfd0e549dee805fb6f8ee&what=${encodeURIComponent(

          jobRole

        )}`;

        document.getElementById("job-listings").innerHTML = "";

        fetch(apiUrl)

          .then((response) => response.json())

          .then((data) => {

            const listingsDiv = document.getElementById("job-listings");

            if (data.results && data.results.length > 0) {

              data.results.forEach((job) => {

                const jobElement = document.createElement("div");

                jobElement.classList.add("job-item");

                jobElement.innerHTML = `

                  <h3>${job.title}</h3>

                  <p><strong>Company:</strong> ${job.company.display\_name}</p>

                  <p><strong>Location:</strong> ${job.location.display\_name}</p>

                  <p><strong>Salary:</strong> ${

                    job.salary\_min

                      ? "£" + job.salary\_min + " - £" + job.salary\_max

                      : "Not specified"

                  }</p>

                  <p><a href="${

                    job.redirect\_url

                  }" target="\_blank">View Job</a></p>

                `;

                listingsDiv.appendChild(jobElement);

              });

            } else {

              listingsDiv.innerHTML =

                "<p>No job listings found for this role.</p>";

            }

          })

          .catch((error) => console.error("Error fetching data:", error));

      }

    </script>

  </body>

</html>

* **Appendix C**: Flask integration and Database setup (SQLPlus)

**app.py**

from flask import Flask, request, render\_template, redirect, url\_for, session, jsonify

from sqlalchemy import create\_engine, Column, Integer, String, Text

from sqlalchemy.orm import declarative\_base, sessionmaker

from sqlalchemy.exc import IntegrityError

import joblib

import re

app = Flask(\_\_name\_\_)

app.secret\_key = 'your\_secret\_key'

# Hardcoded login credentials

USERNAME = 'admin'

PASSWORD = 'password123'

# Database configuration

DATABASE\_URL = 'oracle+oracledb://SYSTEM:SYSTEM@localhost:1521/XE'

engine = create\_engine(DATABASE\_URL)

Session = sessionmaker(bind=engine)

db\_session = Session()

# Define the database model

Base = declarative\_base()

class Journey(Base):

    \_\_tablename\_\_ = 'journeys'

    id = Column(Integer, primary\_key=True, autoincrement=True)

    year = Column(String(50))

    company = Column(String(100))

    name = Column(String(100))

    email = Column(String(100))

    gender = Column(String(20))

    skills = Column(Text)

    salary = Column(String(50))

    jobrole = Column(String(100))

    projects = Column(Text)

    suggestions = Column(Text)

# Create the table if it doesn't exist

Base.metadata.create\_all(engine)

# Load model and vectorizer

model = joblib.load('random\_forest\_model.pkl')

vectorizer = joblib.load('tfidf\_vectorizer.pkl')

POOR\_SKILLS = ["ms office", "finance", "basic computer knowledge", "management", "word", "ppt", "mongo", "powerpoint", "excel","tableau","power bi"]

IMPORTANT\_TOOLS = ["agile", "jira", "scrum", "trello", "kanban", "confluence"]

# Login route

@app.route('/')

def login():

    return render\_template('login.html')

@app.route('/job\_role')

def job\_role():

    return render\_template('job\_role.html')

@app.route('/login', methods=['POST'])

def login\_submit():

    username = request.form['username']

    password = request.form['password']

    if username == USERNAME and password == PASSWORD:

        session['logged\_in'] = True

        return redirect(url\_for('home'))

    else:

        error = "Invalid username or password"

        return render\_template('login.html', error=error)

@app.route('/home')

def home():

    if 'logged\_in' in session and session['logged\_in']:

        return render\_template('home.html')

    else:

        return redirect(url\_for('login'))

@app.route('/logout')

def logout():

    session.pop('logged\_in', None)

    return redirect(url\_for('login'))

@app.route('/share', methods=['GET', 'POST'])

def share\_journey():

    if request.method == 'POST':

        journey = Journey(

            year=request.form['year'],

            company=request.form['company'],

            name=request.form['name'],

            email=request.form['email'],

            gender=request.form['gender'],

            skills=request.form['skills'],

            salary=request.form['salary'],

            jobrole=request.form['jobrole'],

            projects=request.form['projects'],

            suggestions=request.form['suggestions']

        )

        try:

            db\_session.add(journey)

            db\_session.commit()

        except IntegrityError as e:

            db\_session.rollback()  # Roll back transaction in case of error

            return f"An error occurred while saving: {e}"

        return redirect(url\_for('view\_journey'))

    return render\_template('share\_journey.html')

@app.route('/view\_others')

def view\_journey():

    journeys = db\_session.query(Journey).all()

    return render\_template('view\_journey.html', journeys=journeys)

@app.route('/upload\_skills')

def upload\_skills():

    return render\_template('index.html')

@app.route('/predict', methods=['POST'])

def predict():

    skills = request.form['Skills'].lower()

    skills\_list = re.split(r'\W+', skills)

    # Directly predict "Java Developer" if the skill "java" is in the input

    if "java" in skills\_list:

        predicted\_role = "Java Developer"

        session['predicted\_role'] = predicted\_role

        return jsonify({'prediction': predicted\_role})

    # Filter out poor skills and retain important tools

    filtered\_skills = " ".join(skill for skill in skills\_list if skill not in POOR\_SKILLS or skill in IMPORTANT\_TOOLS)

    if not filtered\_skills.strip():

        return jsonify({'prediction': "Please add more specific skills relevant to the job role."})

    # Proceed with model prediction if "java" was not in the skills

    input\_data = vectorizer.transform([filtered\_skills]).toarray()

    predicted\_role = model.predict(input\_data)[0]

    if predicted\_role == "Mobile App Developer":

        return jsonify({'prediction': "The skills provided may not match a specific role. Please add more relevant skills to improve the prediction."})

    # Store prediction in session for later use

    session['predicted\_role'] = predicted\_role

    return jsonify({'prediction': predicted\_role})

@app.route('/get\_predicted\_role')

def get\_predicted\_role():

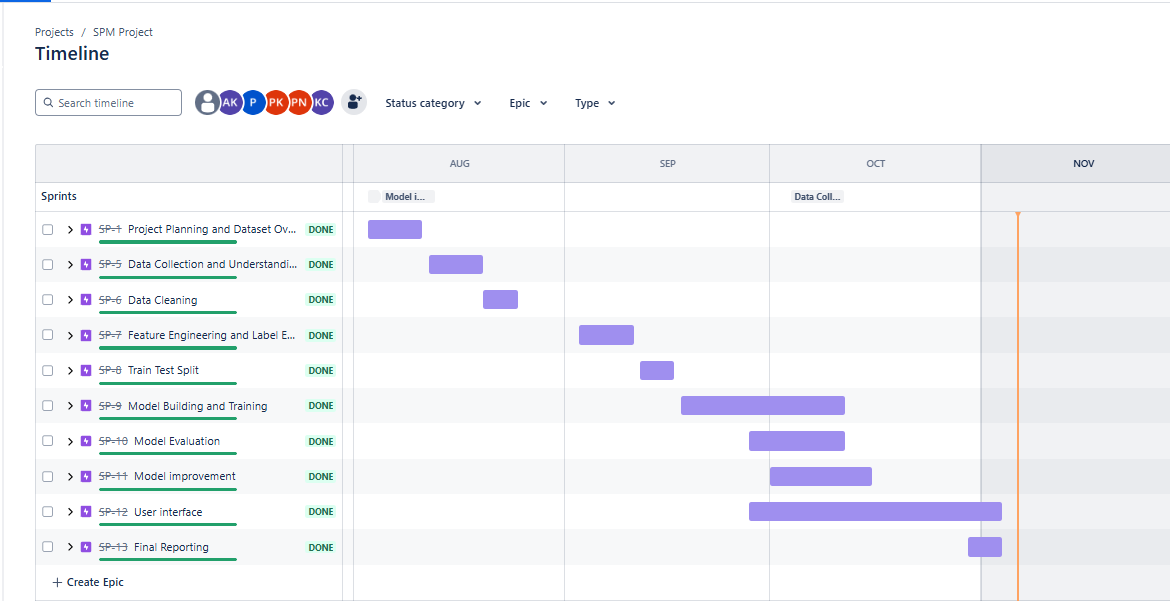
    predicted\_role = session.get('predicted\_role', None)

    return jsonify({'predicted\_role': predicted\_role})

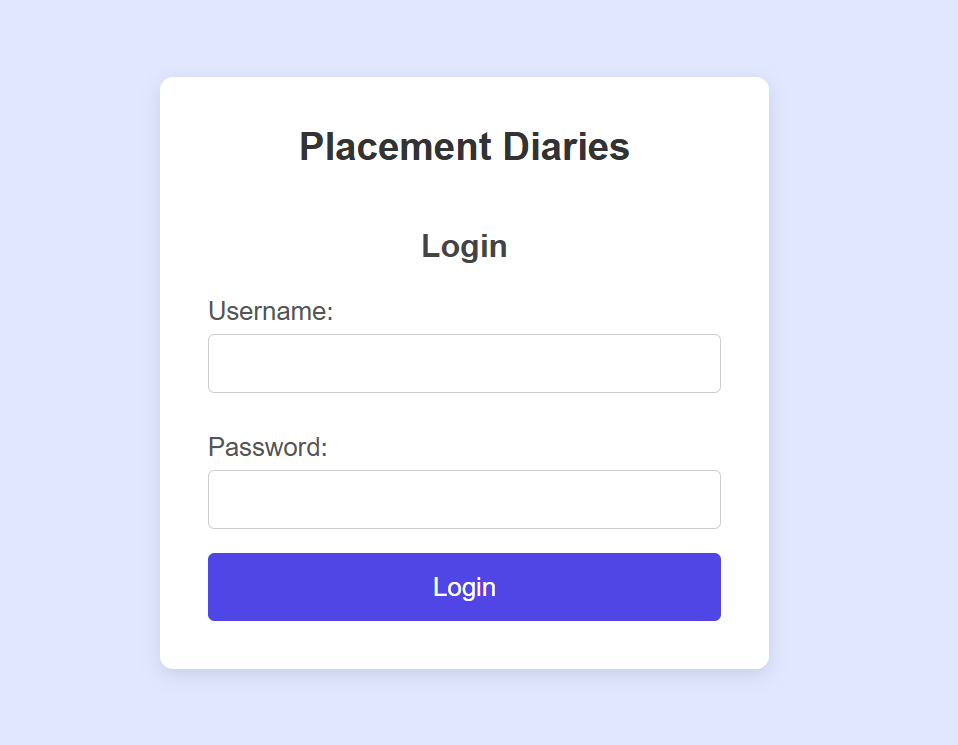
if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug=True)

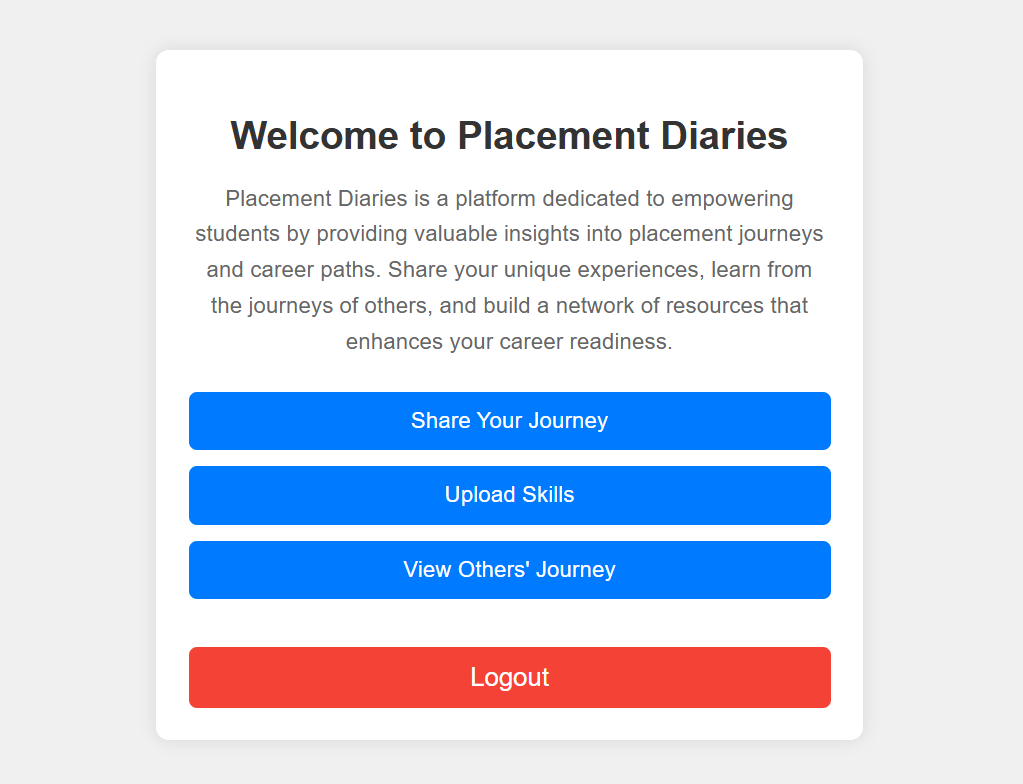
* **Appendix D**: Screenshots of JIRA Board and Frontend Interfaces

****

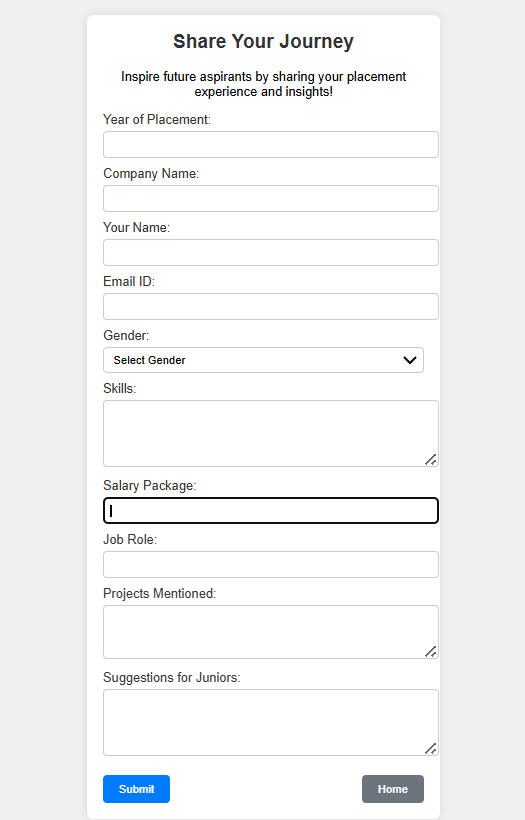
**LOGIN PAGE**

****

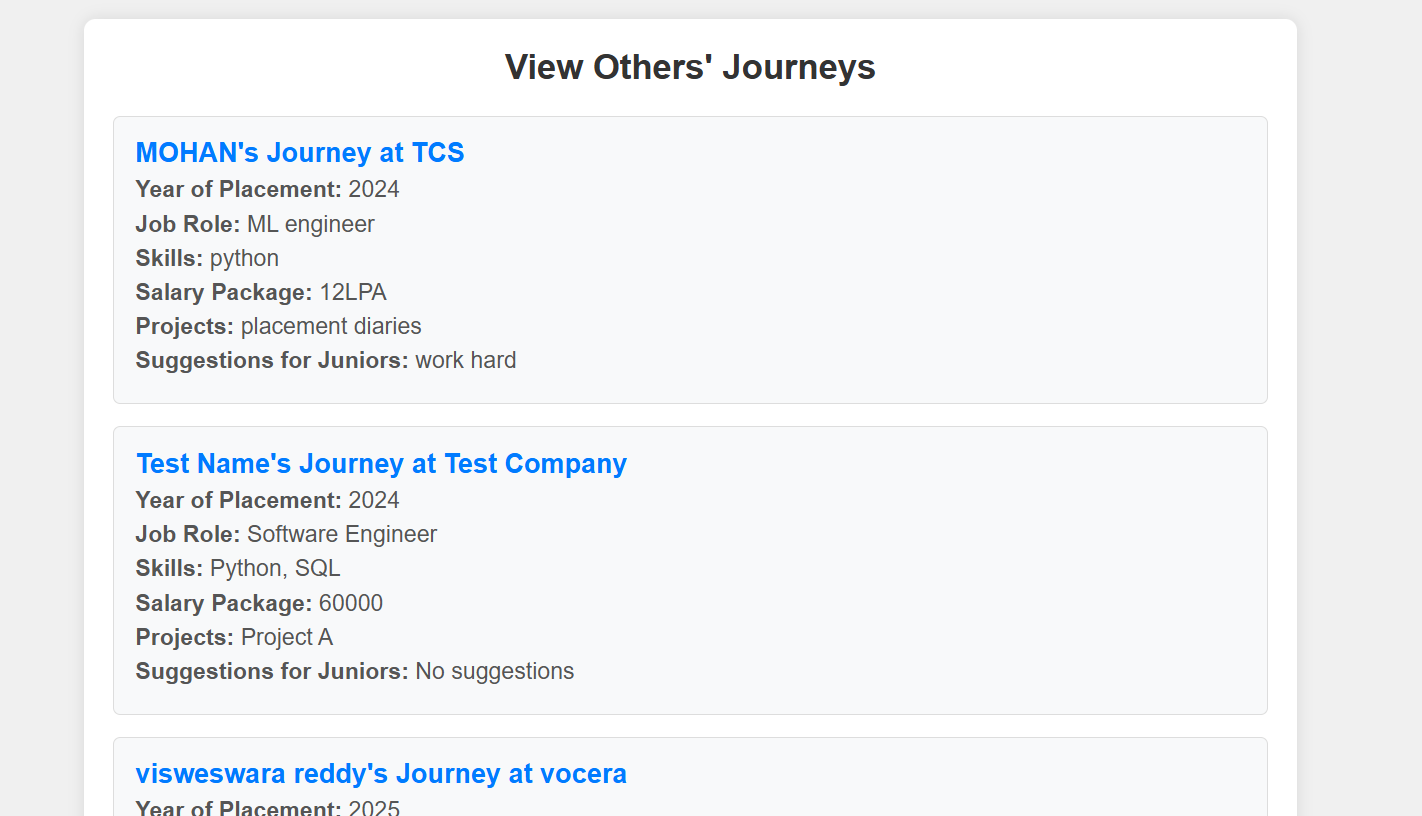
**HOME PAGE**

****

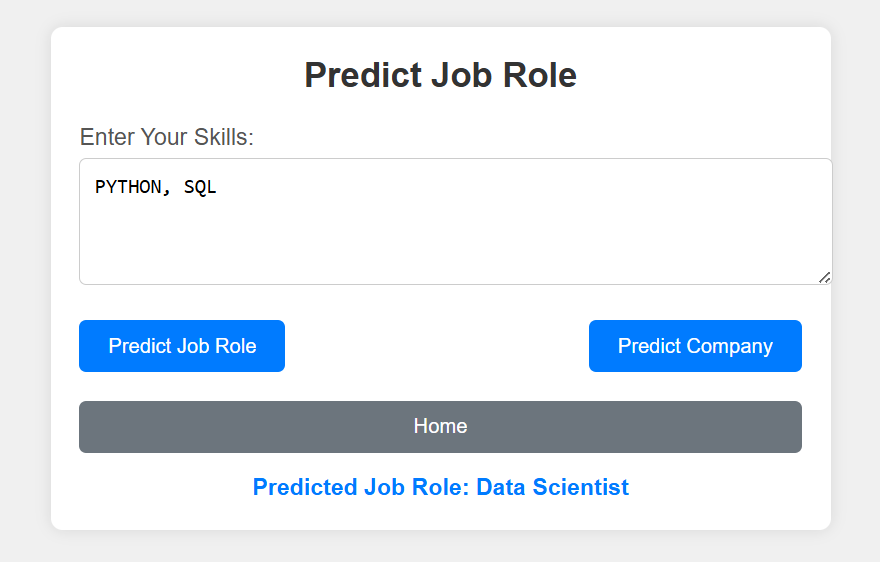
**SHARE JOURNEY PAGE**

****

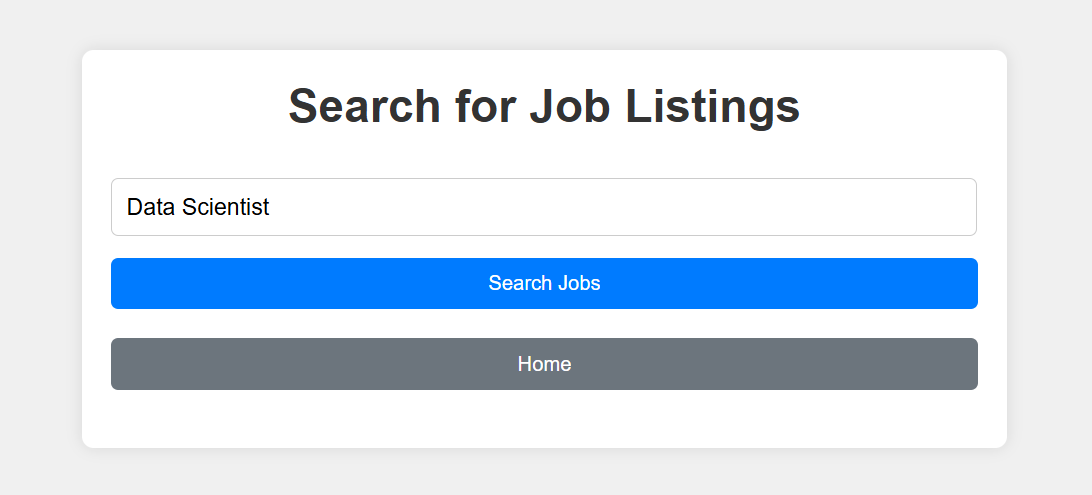
**VIEW JOURNEY PAGE**

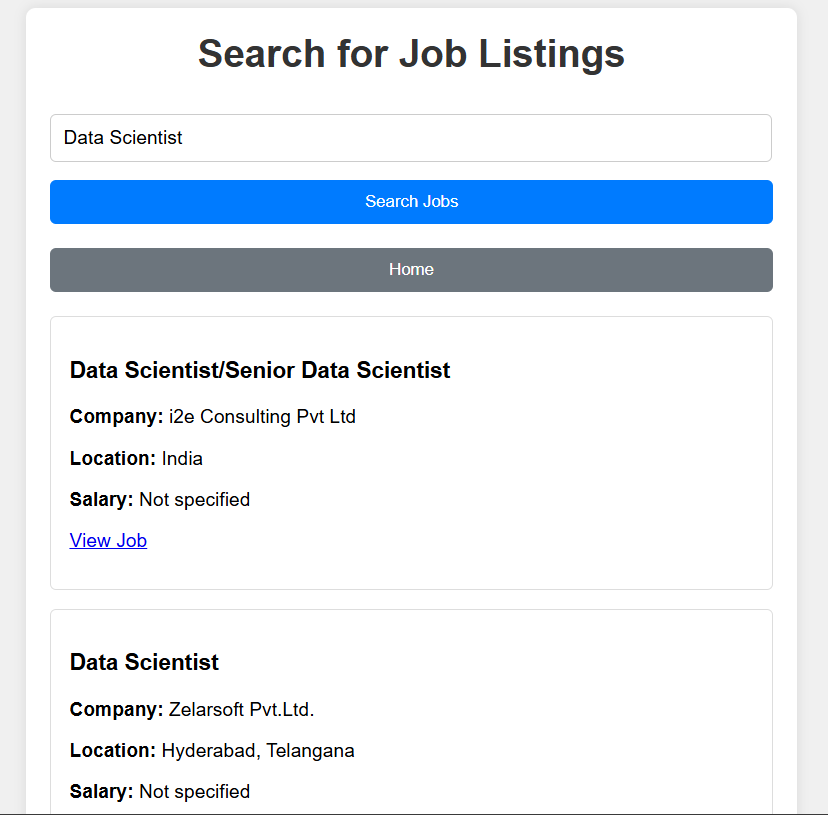
****

**SKILLS PAGE(INDEX)**

****

**COMPANY PAGE (JOB ROLE)**

****

****