CERTIFICATE

This is to certify that the Project Report submitted by **JITENDRA CHAUDHARY (22SE02ML095)** to the **P P SAVANI UNIVERSITY** for the partial fulfilment of the subject credit requirements is a bonafied work carried out by the student.

This is to further certify that I have been supervising the Major/Minor Project of **JITENDRA CHAUDHARY (22SE02ML095).**

The contents of this report, in full or in parts, have not been submitted to any other Institute or University for award of any degree, diploma or titles.

Sign of Faculty Mentor :

Name of Faculty Mentor:

Date:

ACKNOWLEDGEMENT

This report would not have been possible without my teachers who were always there when I needed them the most. I take this chance to acknowledge them and extend my sincere gratitude for helping me make this Report a possible.

I wish to thank my faculty mentor **Ms. Shruti Mishra,** Assistant Professor, School of Engineering. It has been an honor to learn under their mentorship.

As my mentor, she has constantly motivated me to remain focused on achieving my goal. Their observations and guidance helped me to establish the overall direction of the report and to move forward with learning in depth. Their vital support at each juncture, which culminated in successful completion of my project work. I express my sincere gratitude to them for constant support during the project work.

I am also thankful to faculty members of the department for constant support and guidance.

I am thankful to Dean, School of Engineering for his initiative of imparting Project during tenure of your study making us learn new things and to help us in expanding our horizons.

Name of Student : **JITENDRA CHAUDHARY**
Enrollment No : **22SE02ML095**

ABSTRACT

The project “Personalised Career Recommendation and Skills Growth Tracker using AI” aims to assist individuals in identifying the most suitable career paths and skill development opportunities based on their personal interests, skills, experience, and salary expectations. The system leverages Artificial Intelligence (AI) and Natural Language Processing (NLP) to analyse both user input and real-world job data. Using the Sentence Transformer model (all-MiniLM-L6-v2), the system generates embeddings for job descriptions and user profiles to measure similarity through cosine similarity. A Random Forest Classifier is then used to predict job suitability and recommend the most relevant roles.

The model evaluates various parameters such as job titles, industries, experience levels, required skills, and salary ranges. Users interact with the system through a simple web interface, and the backend processes their input to generate personalized recommendations. This AI-driven approach bridges the gap between users’ current skills and career goals, helping them make informed decisions while promoting continuous professional growth.

Table of Contents

1. Introduction

1.1 Objective

Choosing the right career has become difficult due to the growing number of job roles and fast-changing industry demands. Many students and professionals struggle to match their skills and interests with suitable job opportunities. Traditional counselling or keyword-based systems cannot provide accurate, personalized guidance.

This project, "Personalized Career Recommendation and Skills Growth Tracker using AIML," aims to solve this problem by using Artificial Intelligence and Machine Learning. The system analyzes the user's interests, skills, experience, profession preference, and salary expectations. It then compares this information with a job dataset using Sentence Transformer embeddings, cosine similarity, and Random Forest classification to recommend the most suitable career paths. The system also highlights required skills for each recommended role, helping users understand their strengths and areas of improvement.

1.2 Scope

The scope of this project includes collecting and preprocessing a career-related dataset, training a recommendation model using machine learning techniques, and integrating the model into a web-based platform. Users can input their skills and receive personalized job role suggestions along with skill improvement guidance. The system focuses on entry-level to mid-level job recommendations and is designed for students, fresh graduates, and individuals exploring new career options. The scope does not include real-time market analysis or advanced HR-level recruitment features.

1.3 Project Goals

- Develop a functional machine learning model that predicts suitable career roles based on user input.
- Design a user-friendly web interface where users can submit their details easily.
- Provide skill improvement suggestions for each recommended job role.
- Create a backend system that connects the trained model with the website.
- Ensure accurate predictions by preprocessing and training on a relevant dataset.
- Offer a smooth user experience with instant, personalized recommendations.
- Deliver a complete project report and documentation following academic guidelines.

2. System Analysis

2.1 Identification of Need

Students and professionals often struggle to select the right career path because of limited guidance and the vast number of available job roles. Manual counselling is time-consuming and subjective, while traditional job portals provide generic suggestions. Therefore, a system is needed that can analyze user skills, interests, and experience and match them with real job data using AI for accurate, personalized career recommendations.

2.2 Preliminary Investigation

During the initial research stage, it was found that no reliable or practical career recommendation system existed for students or freshers. Most available tools online were either too generic or only suggested jobs based on simple keyword matching.
Further investigation showed that some students had created similar projects, but:

- They were developed only for academic purposes.
- The systems were poorly trained with very small datasets.
- Recommendations were not accurate or personalized.
- Many models failed to consider skills, experience, or salary expectations.
- A few projects were incomplete and did not work properly when tested.

This highlighted the need for a more advanced, usable, and intelligent system that uses real datasets, proper NLP embeddings, and machine learning for accurate predictions.

2.3 Feasibility Study

a. Technical Feasibility

- Python and libraries like Sentence Transformer, Scikit-learn, Flask are easily available and suitable.
- Dataset is structured and ready for preprocessing.
- Model can run on normal systems without GPU.

b. Operational Feasibility

- System is easy to use through a simple web interface.
- Users only need to input interests, skills, profession, salary, and experience.

c. Economic Feasibility

- Uses open-source tools → no extra cost.
- Low operational expense as system can run locally or on low-cost servers.

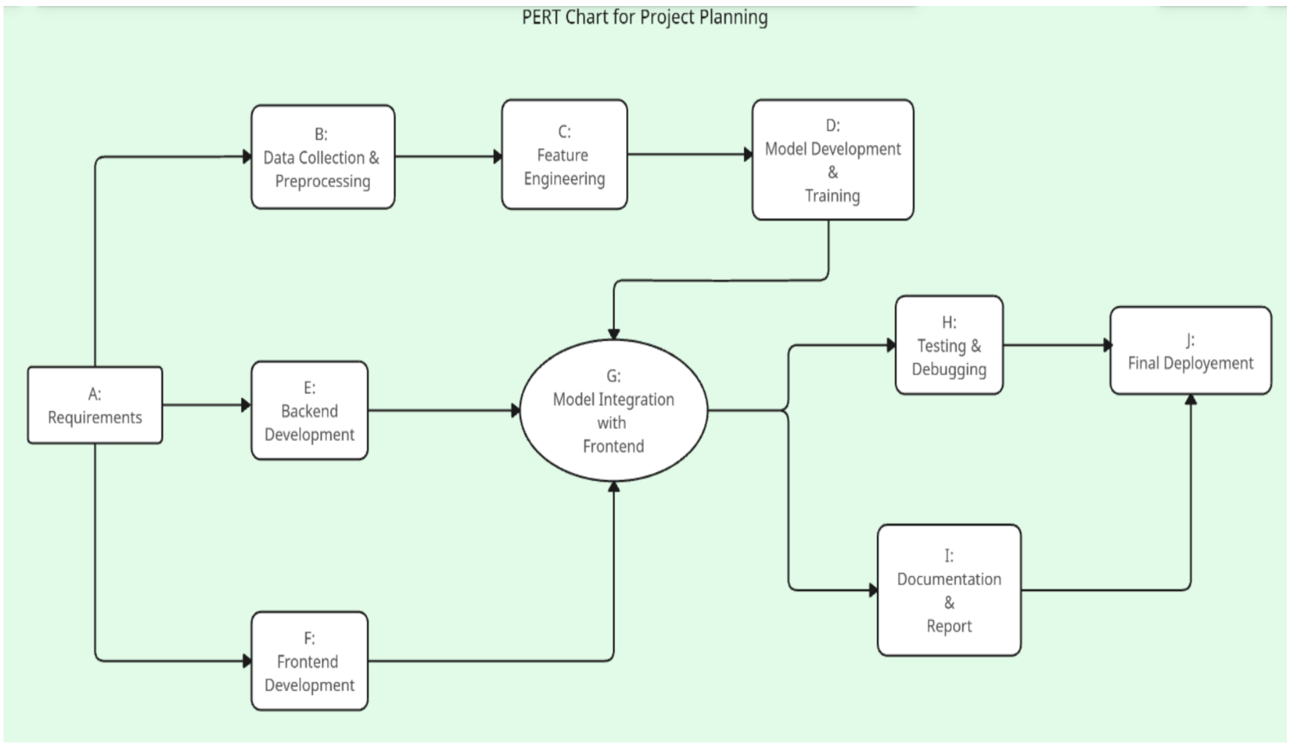
2.4 Project Planning

The project was planned in the following phases:

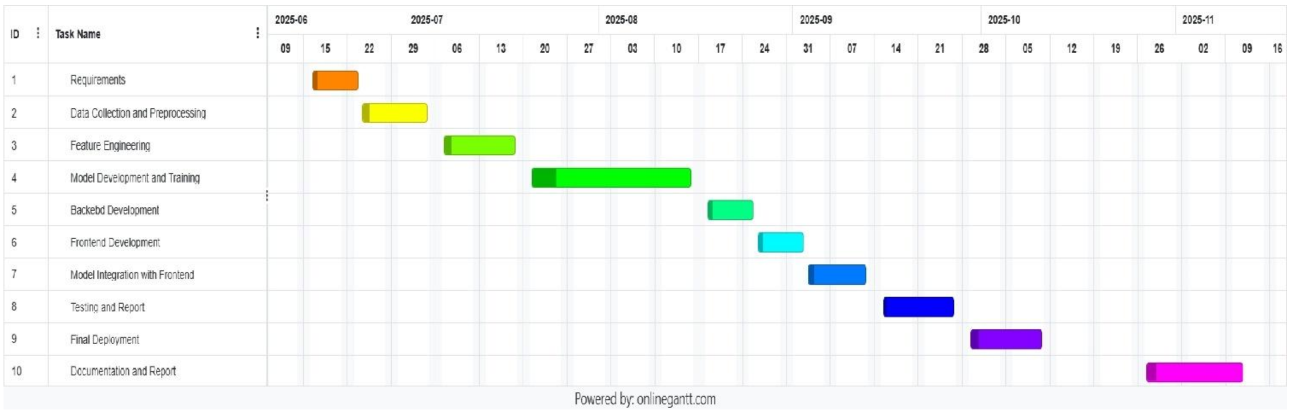
- Dataset Collection & Cleaning
- Embedding Generation using Sentence Transformer
- Model Training using Random Forest
- Similarity Calculation & Hybrid Scoring
- Backend Development (Python + Flask)
- Frontend UI Development (HTML, CSS, JS)
- Testing & Refinement
- Documentation and Report Writing

2.5 Project Scheduling

PERT Chart-


PERT Chart for Project Planning
A: Requirements
B: Data Collection & Preprocessing
C: Feature Engineering
D: Model Development & Training
E: Backend Development
F: Frontend Development
G: Model Integration with Frontend
H: Testing & Debugging
I: Documentation & Report
J: Final Deployement

Gantt Chart-



2.6 Software requirement specifications (SRS)

a. Functional Requirements

- User can enter interests, skills, profession, experience, and salary.
- System generates embeddings for user input.
- Model compares user data with job dataset.
- System provides top 5 job recommendations.
- System displays required skills for each job role.

b. Non-Functional Requirements

- **Performance**: Fast inference (text embeddings computed quickly).
- **Accuracy**: Hybrid scoring (ML + cosine similarity).
- **Usability**: Simple and clean web interface.
- **Security**: Limited access to stored model and data.

c. Software Requirements

- Python 3.x
- Sentence Transformer
- Scikit-learn
- Flask
- Pandas, NumPy
- HTML, CSS, JavaScript

2.7 Software Engineering Paradigm applied

For this project, the Iterative Software Development Model was used. This model was ideal because the system involved machine learning, data preprocessing, and a web interface—features that require continuous testing and improvement.

In each iteration, a small but complete part of the system was developed, tested, and refined. Early iterations focused on dataset cleaning and generating embeddings. Later iterations worked on training the machine learning model, integrating it with the website, testing recommendations, and improving user experience.

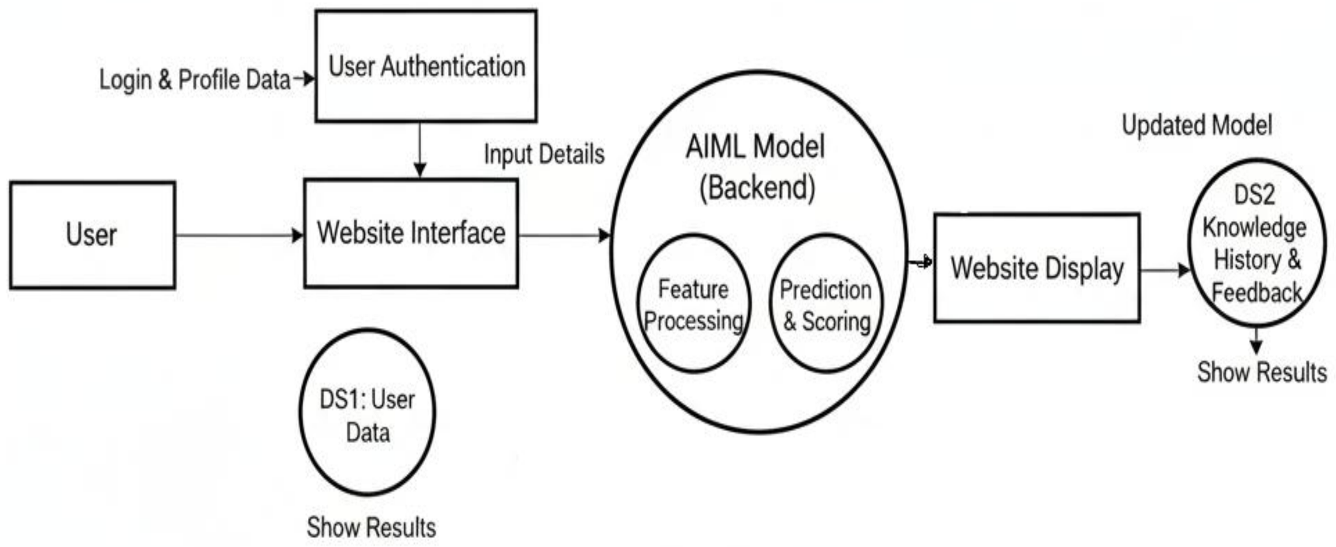
The iterative approach allowed:

- Step-by-step improvements
- Easy debugging and refinement
- Better accuracy of the ML model
- Flexibility to modify features based on feedback

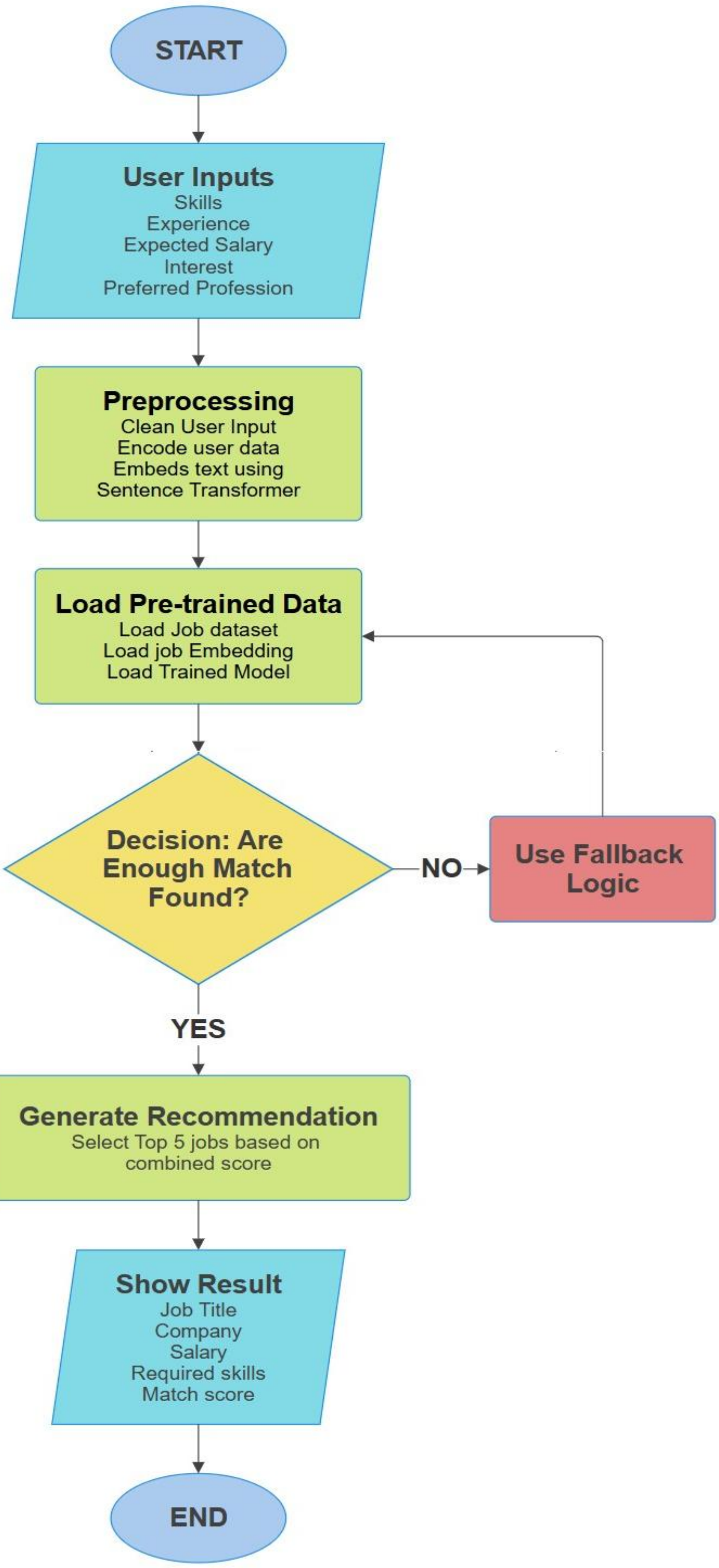
Overall, the paradigm ensured a stable, accurate, and user-friendly career recommendation system.

2.8 Data models and Diagrams

1. Data Flow Diagram



2. Flow Chart


START
User Inputs
Skills
Experience
Expected Salary
Interest
Preferred Profession
Preprocessing
Clean User Input
Encode user data
Embeds text using
Sentence Transformer
Load Pre-trained Data
Load Job dataset
Load job Embedding
Load Trained Model
Decision: Are Enough Match Found?
NO
Use Fallback Logic
YES
Generate Recommendation
Select Top 5 jobs based on combined score
Show Result
Job Title
Company
Salary
Required skills
Match score
END

3. System Design

3.1 Modularization details

Your project is divided into clear, independent modules that work together to provide career recommendations and skill-growth tracking.

Modules:

1. **User Management Module**
   - Handles registration, login, authentication.
   - Stores user details and preferences.
2. **Data Collection Module**
   - Collects user inputs such as skills, interests, education, experience.
3. **Machine Learning Module**
   - Loads trained model.
   - Predicts the best career fields based on user profile.
   - Generates confidence scores.
4. **Recommendation Module**
   - Career suggestions
   - Required skills for chosen career
   - Learning resources
5. **Database Management Module**
   - Stores user data, skills, recommended careers, progress.
6. **Skill Progress Tracking Module**
   - Tracks improvements in skills.
   - Updates learning progress.
7. **Web Interface Module**
   - Frontend UI for user interaction with the system.

Each module is designed so if one part is updated (e.g., ML model), the rest remains unaffected.

3.2 Data integrity and constraints

Data integrity ensures that all information stored in the system remains accurate, consistent, and reliable. The project maintains integrity through a combination of database-level constraints and application-level validations.

Key constraints applied include:

- **Primary Keys:**
  Every main table (Users, Skills, Careers) has a unique identifier to prevent duplicate entries.
- **Foreign Keys:**
  Used in linking tables such as *UserSkills* and *Recommendations* to maintain relationships between users and careers.
- **Domain Constraints:**
  Fields like skill level (0–100), experience (in years), and email format follow strict validation rules.
- **Not-Null Constraints:**
  Critical fields such as user name, email, and selected skills must be filled before submission.

In short, these constraints ensure that only clean, meaningful, and validated data is stored, which improves the performance and accuracy of the AI/ML recommendation model.

3.3 Database design

The database for this project is designed using SQLite with Flask-SQLAlchemy (ORM), which provides a structured and scalable way to store user information and their career-related details. The design follows a relational model, ensuring that data is organized, connected, and easy to retrieve.

At the core, the system contains two main tables:

1. Users
2. Career Profiles

These two tables are linked using a one-to-one relationship, where each user can have only one career profile. This separation keeps authentication data and professional details independent and well-organized.

3.4 User Interface Design

The user interface of the *Personalized Career Recommendation and Skills Growth System* is designed to be simple, clean, and easy to navigate. Since the target users include students and job seekers, the UI focuses on clarity and smooth flow from registration to receiving recommendations. The system consists of five main screens: Home Page, Create Account Page, Login Page, Get Recommendation Page, and the Dashboard for Profile Completion.

1. Home Page

This is the first screen the user sees, and it introduces the purpose of the system. Key elements include:

- A simple title and description of the project
- Buttons for Login and Create Account
- A visually clean layout to guide first-time users

The Home Page sets the tone of the application and helps users understand what the system offers even before logging in.

2. Create Account Page

This page allows new users to register before accessing the system. It typically contains:

- Input fields such as Name, Email, Password
- A “Create Account” button
- A small link redirecting to Login if the user already has an account

The UI focuses on clarity and error-free input, ensuring smooth onboarding.

3. Login Page

Registered users can log into the system through this interface. The page includes:

- Email and Password fields
- A Login button
- A clean minimal UI to reduce distractions

This page acts as a secure entry point ensuring only authenticated users can access the system.

4. **Start Getting Recommendation Page**

After logging in, users are taken to the *Get Recommendation* screen. This page consists of:

- A brief message introducing how recommendations will be generated
- A button such as "Start Recommendation" or "Proceed"
- A simple layout that encourages users to continue to the dashboard

This page acts as the transition point from login to skill/career evaluation.

5. **Dashboard (Profile Completion Page)**

The Dashboard is one of the most important UIs, as users provide details required for generating personalized career recommendations.
The interface includes:

- Fields for entering personal details, current skills, education, and interests
- A submit/update button
- Profile completion indicators (optional)

The dashboard is designed to be user-friendly so that the user can easily provide all necessary information without confusion. Once the details are completed, the system uses this data for generating recommendations.

3.5 Test Cases

1. Unit Testing

Unit testing focuses on testing individual components or modules of the system to ensure each part works independently.
In your Personalized Career Recommendation System, unit tests check small parts like:

- Input validation functions
- Login function
- Registration function
- Recommendation logic
- Profile update logic
- Form validation (email, password strength)

The goal is to ensure each function behaves correctly before integrating them together.

Test ID	Component	Test Description	Expected Output
UT-01	Registration Form	Test with empty username/email/password	System shows error: “All fields are required”
UT-02	Login Function	Test login with correct credentials	User successfully logged in
UT-03	Login Function	Test login with incorrect password	Error message displayed
UT-04	Profile Form	Test saving profile with valid data	Profile information stored in database
UT-05	Recommendation Module	Test recommendation function with valid user input	System generates career suggestions

2. System Test Cases

System testing validates the entire system as a whole.
Here, the goal is to ensure:

- All modules work together properly
- The workflow is smooth
- There are no integration issues
- User journey is consistent

System testing covers complete operations from registration → login → profile setup → recommendation generation.

Test ID	**Scenario**	**Input / Action**	**Expected Result**
ST-01	User Registration Flow	User completes registration form	Account created and redirected to login
ST-02	Complete Workflow	Register → Login → Fill Profile	System works smoothly and moves to dashboard
ST-03	Profile Completion	User submits incomplete profile	System shows: “Please complete all required fields”
ST-04	Recommendation Generation	User submits complete profile	System displays recommended career options
ST-05	Logout	User clicks logout	User session ends and redirected to home/login

4. Coding

The coding phase of the *Personalized Career Recommendation System* involves implementing the backend logic, machine learning model, database operations, and frontend web interface. The project follows a modular structure where each functionality is separated into different Python modules and HTML templates, ensuring clarity, maintainability, and reusability.

4.1 Database Creation and Management

Even though the project uses a lightweight database structure, it plays an essential role in storing and managing user information. The database creation process involves defining the required tables, applying essential constraints, and ensuring reliable data access.

Key elements included in database creation:

- **User Table:** Stores user credentials, ensuring unique email constraints and secure password handling.
- **Profile Table:** Contains skills, education, experience, and other inputs needed for recommendation.
- **Constraints:**
    - Email must be unique
    - Mandatory fields must be filled
    - Valid data formats must be maintained
- **Data Insertion:**
    - Occurs during user registration
    - Happens when users update their profile
- **Access Rights:**
    - Only authenticated users can access the dashboard and recommendations
    - Sensitive information is protected through backend-level access control

Overall, the database layer supports the proper functioning of user registration, login, and career recommendation workflows.

4.2 Complete Project Coding

Project coding is divided into three major components:

1. **Frontend Development**
   The user interface is developed using HTML, CSS, and supportive scripts.
   It includes pages for:

   - Home/landing page
   - Account creation
   - Login
   - Dashboard for profile editing
   - Recommendation module

   The layout follows a clean, simple, and user-friendly design to ensure smooth navigation.

2. **Backend Development**
   The backend manages:

   - User registration and authentication
   - Input validation
   - Communication with the database
   - Processing of profile information
   - Generating career recommendations using the ML model

   The backend follows a structured and modular architecture, separating routes, logic, and database operations clearly.

3. **Machine Learning Module**
   A machine learning model is prepared using a structured dataset.
   The model performs:

   - Data preprocessing
   - Feature extraction
   - Training and evaluation
   - Prediction of suitable job roles

   The trained model is integrated into the backend so that the user receives recommendations instantly through the dashboard.

4.3 Comments and Description of Coding segments

The codebase is documented with meaningful comments explaining the purpose and function of each segment. This improves understanding and simplifies future updates.

Documentation Highlights:

- User authentication logic is commented to explain validation flow.
- Profile processing section includes comments describing data extraction for the ML model.
- Recommendation module is documented to explain prediction handling.
- UI templates include comments for layout structure and dynamic data rendering.

The presence of clear comments ensures that other developers or evaluators can easily understand how each part of the system works.

5. Standardization of the coding

The coding for the Personalized Career Recommendation System follows a structured, modular, and optimized development methodology. The aim is to ensure that every component—backend logic, machine learning integration, database operations, and user interface interaction—maintains consistency, reliability, and performance. The system is designed using clean coding principles, predictable architecture, and reusable functions that make the entire project easier to scale and maintain.

5.1 Code Efficiency and Error Handling

The complete system is built with efficiency and robustness in mind. The backend, machine learning module, and user interface are designed such that response time remains minimal while maintaining accuracy and stability.

1. **Code Efficiency**
   - **Modular Architecture:**
     Every major functionality—user authentication, profile processing, recommendation generation, and database interaction—is separated into independent modules. This reduces complexity and makes the code scalable and easy to debug.

   - **Optimized Database Interaction:**
     Queries are executed using efficient ORM operations, reducing repeated database calls and ensuring fast access to user and profile data.

   - **Model Preloading for Recommendations:**
     The career recommendation model and preprocessing steps are initialized once and reused.
     This avoids repeated loading and ensures instant response when the user requests recommendations.

   - **Lightweight Backend Processing:**
     The model runs on CPU-friendly algorithms, eliminating the need for expensive hardware while still maintaining good prediction quality.

   - **Efficient Data Flow:**
     User inputs are validated, processed, and passed through the ML pipeline in a streamlined manner without unnecessary conversions or redundant computations.

2. **Error Handling**

To ensure that the system never crashes unexpectedly, multiple layers of error handling are applied throughout the application.

- **Backend-Level Error Handling:**
  - Login errors (invalid credentials)
  - Registration errors (duplicate username/email)
  - Missing or incomplete profile data
  - Failed database updates
  - Server errors such as incorrect routing or missing templates
- **ML Pipeline Error Handling:**
  - Prevents failures due to empty user inputs
  - Handles unexpected data types or missing fields
  - Ensures model outputs are safely returned even if certain values are missing
- **User-Friendly Messages:**
  When errors occur, users see helpful and meaningful messages rather than technical traces. This improves overall usability and smoothens the user experience.
- **Fallback Handling:**
  If the recommendation model encounters invalid or insufficient data, the system provides a safe fallback response rather than terminating the session.

5.2 Parameter Calling / Passing

The project maintains a clean and structured parameter flow. Each part of the system communicates with others through explicit, well-defined parameters, reducing dependency and improving readability.

1. **Structured Function Interaction**

- **Profile Data Collection → Preprocessing → ML Model → Prediction**
  All data moves smoothly from the user interface to the machine learning model without redundant transformations.

- **Clear Function Signatures:**
  Backend functions accept only the required parameters and return predictable outputs, such as:
  - User credentials
  - Career profile fields
  - Processed inputs for recommendation generation
  - Prediction results (recommended roles)
- **No Use of Unnecessary Global Variables:**
  This improves testability and prevents conflicts between components.

2. Inter-Module Communication

- Authentication module passes the user ID to the profile module
- Profile module passes cleaned profile data to ML module
- ML module returns recommended job roles back to the backend
- Backend renders results into the UI templates

This streamlined flow ensures that each component handles only its designated task, making the system maintainable and easy to extend.

5.3 Validation Checks

Validation is a critical part of ensuring reliable recommendations and preventing system misuse. Both frontend and backend validations are applied consistently throughout the system.

1. User Input Validation

- **Registration & Login Validation:**
  - Username length constraints
  - Valid email format
  - Password strength checks
  - Prevention of duplicate accounts

- **Career Profile Validation:**
    - Required fields such as name and education
    - Valid age and experience ranges
    - Acceptable text length limits for skill and goals fields
    - Salary expectations checked for valid numeric ranges
- **Backend Validation**
    - Null-value checks
    - Type checks (integer, string, boolean)
    - Range validations
    - Constraints defined in the database schema
- **ML Validation**
    - Required profile inputs exist
    - Cleaned and formatted data is passed to the model
    - Missing fields do not break the prediction flow
- **System-Level Validation**
    - Only authenticated users can access the dashboard
    - User cannot access another user’s data
    - Corrupted data is prevented from entering the ML pipeline

Overall, validation ensures the system remains secure, predictable, and accurate.

6. Testing

1. Testing Techniques and Testing Strategies Used

To ensure the reliability, accuracy, and smooth functioning of the Personalized Career Recommendation System, multiple testing techniques and strategies were applied. The goal was to validate every component—from user registration to AI-generated career suggestions.

- **Unit Testing**

  Each module was tested individually, including:

  - **User Registration & Login Module**

    – Email/password validation

    – Session handling
  - **Profile Completion Module**

    – Skills, interests, academic details input validation

    – Mandatory field checking
  - **AI Recommendation Engine**

    – Processing user profile input

    – Mapping skills/interests to career paths
  - **UI Components**

    – Dashboard tiles

    – Form components

    – Button actions and navigation
- **Integration Testing**

  Checked the interaction between connected modules:

  - Registration → Login → Dashboard flow
  - Profile inputs → Recommendation engine → Output suggestions
  - Data passing between frontend and backend
  - Error handling flow from backend to UI
- **System Testing**

  End-to-end testing of the entire application:

  - Creating an account → logging in → completing profile → generating recommendations
  - Checking system response for incomplete profiles

  - Verifying stability under repeated navigation
  - Observing UI smoothness and correctness of displayed content
- **Manual Testing**

  The system was tested manually by:
  - Entering random skills, unrealistic skills, missing inputs
  - Testing invalid emails, weak passwords
  - Refreshing pages to check session persistence
  - Observing recommendation quality and UI responsiveness

2. Testing Plan Used

A structured, phased testing plan was followed.

Phase 1 – Unit Testing

- Conducted after building each feature
- Ensured that login, profile forms, and recommendation functions worked correctly on their own

Phase 2 – Integration Testing

- Executed after connecting frontend screens with backend logic
- Verified flow between modules like:
  - Sign-up → Login → Dashboard
  - Dashboard → Profile → Recommendation Engine

Phase 3 – System Testing

Performed after complete project assembly to evaluate:

- Accuracy and relevance of AI-generated career suggestions
- UI navigation speed and clarity
- Handling of edge-case user inputs
- Overall stability of the web application

Tools & Environment Used

- Browser Developer Tools (UI debugging)
- Python backend environment
- Custom AI model (Recommendation logic)
- Localhost testing on Windows Machine
- Console logs for tracking user input flow and errors

3. Test Reports for Unit Test Cases and System Test Cases

Unit Test Case Report

Test Case ID	Module	Test Scenario	Expected Output	Actual Output	Result
UT_001	Registration	Enter valid/invalid email	Accept valid, show error for invalid	Works as expected	Pass
UT_002	Login	Correct and incorrect credentials	Login success or meaningful error	Works correctly	Pass
UT_003	Profile Input	Missing or incomplete skills/interests	Show validation error	Correct validation	Pass
UT_004	AI Engine	Provide complete profile	Generate career recommendations	Recommendations displayed	Pass
UT_005	UI Components	Click buttons / navigate	Proper page redirection	All UI actions work well	Pass

System Test Case Report

Test Case ID	Scenario	Steps	Expected Output	Actual Output	Result
ST_001	New user complete workflow	Register → Login → Profile → Recommendations	Completed profile leads to career options	Works perfectly	Pass
ST_002	Incomplete profile	Skip mandatory fields	System prevents generation	Shows proper message	Pass
ST_003	Recommendation Accuracy	Add specific skills (e.g., Python, data analytics)	Relevant jobs (Data Analyst, ML Engineer)	Accurate suggestions	Pass
ST_004	Invalid login attempt	Wrong password	Display error	Shows correct error	Pass
ST_005	Navigation flow	Move between all pages	Smooth transitions	No UI lag	Pass

4. Debugging and Code Improvement

Issues Identified

- Some incorrect inputs in profile form caused breaks in recommendation generation.
- UI responsiveness reduced when user entered multiple skills.
- Validation messages were missing for certain fields.
- Recommendation output formatting was initially inconsistent.

Fixes Done

- Improved input validation for skills, interests, and academic data.
- Added missing error messages and protective checks.
- Optimized profile-to-AI mapping logic to reduce delays.
- Cleaned up UI components for better clarity and navigation.

Improvements

- Modularized backend code for easier maintenance.
- Improved accuracy of recommendation logic by refining skill–career mapping.
- Enhanced UI structure for better user experience.
- Better error handling across login and profile modules.
- Added safety checks for empty or unrealistic input values.

7. System Security Measures

The project includes several security practices to ensure safe handling of user inputs, system operations, and future scalability. While the current prototype does not use a persistent database, the design follows proper security guidelines to maintain confidentiality, integrity, and controlled access.

1. **Database / Data Security**

- User images and text inputs are processed temporarily in memory, ensuring that no sensitive data is stored on the device or server.
- All incoming data is validated and sanitized to prevent harmful inputs such as code injections or corrupted files.
- In a full-scale version, encrypted storage, secure data handling policies, and scheduled backups would be implemented to protect long-term stored data.
- Communication between modules can be secured with HTTPS in production to prevent data interception.

2. **Creation of User Profiles and Access Rights**

- The system design supports the addition of user accounts to manage personalized data and access levels.
- Role-based access control (RBAC) can be implemented so only authorized users can access sensitive features, modify system settings, or use specific tools.
- Authentication mechanisms, such as login credentials or API keys, ensure that only verified users can use the system.
- Session management and logout controls help maintain user privacy and prevent unauthorized access.

8. Cost Estimation of the Project

The Personalized Career Recommendation System was developed entirely using free and open-source tools, along with the developer's personal computer. Therefore, the project does not involve any direct financial investment. However, a basic cost estimation is still prepared to understand the overall effort and resources utilized during development.

1. Cost Components

Even though the project did not require monetary spending, the following non-financial cost elements were considered:

- **Development Hardware:** Personal laptop/PC used for coding and testing.
- **Software & Tools:** All development tools such as Python, frameworks, libraries, and editors are open-source and free to use.
- **Development Effort:** Time invested in coding, GUI design, model training, testing, and documentation.
- **Electricity & Internet Usage:** Minimal operational cost, but no separate financial investment was made.

2. Cost Estimation Model Used

For academic documentation, the project follows a simple Work-Based Cost Estimation approach (WBS):

- Planning and Requirement Understanding
- UI/UX Design
- Model Development (ML logic + data processing)
- Backend Logic for Recommendation
- Testing and Debugging
- Documentation and Report Preparation

Since all work was carried out individually on a personal system, the only cost involved is developer effort, not money.

9. Reports

During the development and testing stages of the Personalised Career Recommendation and Skills Growth Tracker, various internal reports were prepared to continuously evaluate system performance, model accuracy, user experience, and overall stability. These reports helped in early issue detection, model refinement, and improvement of system reliability.

- **Unit Test Reports**

Prepared for individual modules, including:

  - User input validation (skills, interests, experience)
  - Data preprocessing and vectorization
  - Career recommendation engine
  - Skills gap analysis module
  - GUI/Frontend component behavior

These reports ensured that each module performed correctly before integration.

- **Integration Test Reports**

Documented the performance and correctness of interactions between major components:

  - User input → preprocessing → ML model
  - Model recommendation output → UI display
  - Skills gap analysis → roadmap generation
  - Frontend communication with backend logic

Integration reports confirmed that data flowed correctly throughout the pipeline.

- **System Test Reports**

Generated after full application testing to evaluate:

  - Accuracy of career recommendations
  - Correctness of skill-gap analysis
  - Response time for predictions
  - User interface consistency and reliability

These reports summarized overall system behavior under normal and edge-case conditions.

10. Future Scope and Further Enhancements

The Personalised Career Recommendation and Skills Growth Tracker has strong potential for expansion. The following enhancements can significantly improve accuracy, usability, and real-world impact:

- **Real-Time Job Market Integration**

  The system can connect to platforms like LinkedIn and Naukri to fetch trending job roles, required skills, and salary insights, making recommendations more industry-aligned.

- **Full Database-Backed User Profiles**

  Users will be able to save their data, view past recommendations, update skills, and maintain a long-term career progress dashboard.

- **Hybrid AI Recommendation Engine**

  Using multiple ML/LLM models or domain-specific modules can improve recommendation accuracy and provide more reliable results.

- **Mobile App Version**

  A dedicated mobile application can offer better accessibility, notifications, and a more convenient user experience.

- **Multilingual Support**

  Adding Indian languages can make the system accessible to a wider audience, especially students from non-English backgrounds.

- **Advanced Analytics**

  Future dashboards can display learning progress, trending careers, and personalised weekly/monthly insights.

11. Bibliography

A comprehensive bibliography is included to acknowledge the references, datasets, tools, and learning resources used throughout the research, design, and development of the Personalised Career Recommendation AIML. These sources supported key areas such as machine learning, data preprocessing, classification algorithms, user profiling, recommendation systems, and web-based interface design.

The bibliography covers research papers, online documentation, open-source libraries, and developer resources that guided the development of the AI model pipeline—especially in data cleaning, feature engineering, Random Forest model training, and generating personalised career predictions. These references provided the theoretical and technical backing required to build a functional, data-driven, and user-friendly career recommendation system.

Key References:

- **Scikit-Learn Documentation** – Machine learning algorithms (Random Forest, model evaluation, preprocessing)
- **Pandas & NumPy Documentation** – Data cleaning, formatting, and transformation
- **Streamlit Documentation** – Interface workflows for model interaction (if used)
- **Python Official Documentation** – Standard functions, libraries, and development syntax
- **Career Recommendation Research Papers** – Studies on AI-driven guidance systems and skill-based occupation matching
- **Online Machine Learning Tutorials (Kaggle, Medium, Towards Data Science)** – Reference guides for dataset handling and building career prediction models
- **Open-Source GitHub Repositories** – Examples of recommendation system implementations and ML project structures

12. Appendices

The appendices provide supplementary documentation and supporting materials related to the *Personalised Career Recommendation and Skills Growth Tracker using AI*. These materials offer deeper insights into the system's architecture, model training workflow, and overall development process. They serve as extended reference documents for developers, reviewers, and stakeholders who want a deeper understanding of the project's implementation and functionality.

The appendices include the following:

- **Software Requirement Specifications (SRS)**
  A detailed record of functional and non-functional requirements, including user registration/login behavior, profile completion steps, model interaction, and expected system outputs such as personalised career recommendations and skill suggestions.

- **System Design Diagrams**
  Visual diagrams including the Flowchart, Data Flow Diagram (DFD Level 0), and Entity–Relationship (ER) Model.
  These diagrams explain how user data flows through the application, how the career recommendation model interacts with user input, and how various system components are structured.

- **Machine Learning Workflow & Architecture**
  Architecture diagrams showing:
    - Data preprocessing and cleaning steps
    - Feature extraction and encoding
    - Model training using Random Forest
    - Prediction processing for career recommendations
    - User profiling and dashboard integration

  This illustrates the end-to-end ML pipeline used in the project.

- **Test Cases and Test Reports**
  A collection of unit and system-level test cases prepared to evaluate module correctness, input handling, page navigation, ML model accuracy, and response consistency.
  The reports summarise test outcomes and help validate system reliability.

- **Project Code Repository Link**

Instead of embedding full code in the report, a Google Drive link (provided in the report) includes:

  - Python backend implementation
  - Model training script
  - Preprocessed dataset
  - Frontend HTML/CSS pages (Home, Login, Create Account, Dashboard)
  - JavaScript used for UI validation (email/password checks)

- **Project Timeline Artifacts**

The PERT Chart and Gantt Chart used for planning and tracking development phases, from requirement gathering and dataset preparation to UI creation, model training, integration, and testing.

13. Glossary

This glossary defines the technical terms, acronyms, and abbreviations used throughout the AI-Based Photo Relevance Verification System documentation. It serves as a quick reference for concepts related to multimodal AI, image captioning, text similarity scoring, and application development.

Term	**Definition**
AI (Artificial Intelligence)	Technology enabling machines to perform human-like tasks such as understanding images and text.
Sentence-BERT (SBERT)	A transformer-based model that converts sentences into embeddings and computes semantic similarity.
Embedding	A vector representation of text used by the system to compare captions with user-entered descriptions.
Semantic Similarity	A measure of how closely two text inputs match in meaning.
Similarity Score	A numerical value (percentage or decimal) indicating how relevant the image is to the user-provided description.
Relevance Classification	Categorizing the image as *Relevant*, *Partially Relevant*, or *Not Relevant* based on similarity thresholds.
Streamlit	A Python-based framework used to build the interactive user interface of the system.
Frontend	The user-facing interface for uploading images, entering text, and viewing results.
Backend	The processing layer responsible for AI model execution, caption generation, and similarity computation.
Model Inference	Running the BLIP and SBERT models to generate outputs such as captions or similarity scores.

Term	Definition
Image Preprocessing	Steps applied to an uploaded image (resizing, conversion) before sending it to the captioning model.
Base64 Encoding	A method of converting image files into text format for secure and lightweight processing.
Thresholding	Predefined cutoff values used to determine the relevance category of an image.
Confidence Score	The final score representing the model's confidence in the relevance of the image to the given text.
Prototype	An early working version of the system used for testing accuracy and workflow.
Data Validation	Ensuring uploaded inputs (images/text) are safe, properly formatted, and usable.
Hugging Face Transformers	The library used to load and run BLIP, SBERT, and other AI models.
Multimodal Learning	AI that processes multiple inputs—image + text—to perform joint tasks like relevance verification.
User Input Text	The location or scene description provided by the user for comparison with the model-generated caption.

14. References

1. Bengio, Y., Goodfellow, I., & Courville, A. (2016). Deep Learning. MIT Press.
2. Russell, S., & Norvig, P. (2021). *Artificial Intelligence: A Modern Approach.* Pearson.
3. Zhang, Z., & Zhao, L. (2020). "Career Guidance Based on Machine Learning and Big Data Analysis." *Journal of Intelligent Systems*, 29(3).
4. Ramesh, V., & Rao, G. (2018). "Predicting Suitable Careers Using Machine Learning Techniques." *International Journal of Advanced Research in Computer Science*.
5. Hugging Face Documentation. *Transformers Library & Pretrained Models.* https://huggingface.co/docs
6. Scikit-Learn Documentation. *Machine Learning Algorithms, Model Training & Evaluation.*
   https://scikit-learn.org
7. TensorFlow Documentation. *Neural Networks and Deep Learning Models.* https://www.tensorflow.org
8. Pandas Documentation. *Data Cleaning, Preprocessing and Analysis.* https://pandas.pydata.org
9. NumPy Documentation. *Numerical Computation and Feature Engineering.* https://numpy.org
10. Streamlit Documentation. *Building Interactive AI-Powered Web Applications.* https://streamlit.io
11. O*NET Online. *Career Skills, Job Categories & Competency Datasets.* https://www.onetonline.org
12. LinkedIn Economic Graph Insights. *Skills-Demand Trends & Job Market Analytics.*
13. Kaggle Datasets. *Career Prediction, Personality Assessment, and Skill-Based Recommendation Data Sources.*
    https://www.kaggle.com