**1. INTRODUCTION**

**1.1 ASTRACT**

The **Resume Analyzer using ML and NLP Techniques** is implemented as a **web-based application**, providing a user-friendly interface for recruiters and job seekers. This automated approach eliminates manual effort, reduces bias, and ensures **a faster and more accurate recruitment process**. With the ability to process large volumes of resumes efficiently, the system The **Resume Analyzer using ML and NLP Techniques** is an advanced AI-powered system designed to automate and optimize resume screening, parsing, and job recommendations. In today's competitive job market, recruiters receive thousands of resumes, making manual screening time-consuming, inefficient, and prone to human biases. This system leverages **Machine Learning (ML)** and **Natural Language Processing (NLP)** to classify resumes into relevant job categories, extract essential details such as **contact information, skills, and education**, and recommend suitable job opportunities based on the extracted data.

The system applies **various NLP techniques**, including **TF-IDF, Word2Vec, and Count Vectorizer**, to process resume content effectively. It then utilizes **classification algorithms** like **Random Forest, Support Vector Machine (SVM), Decision Trees, and AdaBoost** to categorize resumes with high accuracy. The **job recommendation engine** further enhances candidate-job matching by analyzing key features extracted from resumes.

**2. SYSTEM STUDY**

**2.1 EXISTING SYSTEM**

The traditional recruitment process relies on **manual resume screening**, where HR professionals and recruiters **review resumes one by one** to shortlist candidates. However, this approach has multiple drawbacks:

**LIMITATIONS**

* **Time-Consuming Process**: Recruiters spend significant time reading and analyzing each resume, making it **slow and inefficient**, especially for companies handling **bulk hiring**.
* **High Chances of Human Errors**: Manual screening is **prone to mistakes**, where important details may be overlooked or misclassified.
* **Human Bias**: Personal biases can impact decision-making, leading to unfair candidate evaluations. **AI-driven automation eliminates this issue.**
* **Lack of Scalability**: Traditional methods cannot handle large volumes of resumes efficiently, making it difficult for companies to process **thousands of applications**.
* **No Automated Job Recommendation**: The current system **does not provide job recommendations** based on resume content, making the recruitment process less efficient.
* **Unstructured Resume Data**: Resumes come in different formats (PDF, DOCX, etc.), and extracting structured data manually is **challenging**.

**2.2 PROPOSED SYSTEM**

The **Resume Analyzer using ML and NLP Techniques** overcomes these challenges by **automating** the resume screening, parsing, and job recommendation processes using **Artificial Intelligence (AI).**

**FEATURES**

1. **Automated Resume Classification**:

* Uses **Machine Learning models** to **automatically classify resumes** into different job categories such as **Software Engineer, Data Scientist, Marketing Manager**

1. **Efficient Resume Parsing**:

* Extracts **structured information** such as:
  + **Candidate Name**
  + **Contact Information (Phone & Email)**
  + **Skills**
  + **Education**
  + **Work Experience**

1. **Job Recommendation Engine**:

* Analyzes the extracted resume details and **suggests the best job roles** based on the candidate’s **qualifications and skills**.

1. **NLP-Based Resume Processing**:

* Uses **TF-IDF, Word2Vec, Count Vectorizer,** and other NLP techniques to analyze **resume content efficiently.**

1. **Integration with Machine Learning Models**:

* Employs **classification algorithms** like:
  + **Random Forest**
  + **Support Vector Machine (SVM)**
  + **Decision Tree**
  + **Logistic Regression**
  + **AdaBoost**

1. **Scalability & Speed**:

* Processes **large volumes of resumes quickly**, reducing screening time from **hours to minutes**.

1. **Bias-Free Screening**:

* **Eliminates human bias** by making data-driven, **objective** resume classifications and recommendations.

1. **User-Friendly Web Application**:

* Provides a **Flask-based web interface** where users can **upload resumes and view categorized results instantly.**

With these features, the **Resume Analyzer using ML and NLP Techniques** ensures **higher efficiency, accuracy, and fairness** in the recruitment process.

**3. SYSTEM REQUIREMENTS**

The system requirement deals with hardware requirement and software requirement to install and run this web application.

**3.1 HARDWARE REQUIREMENTS**

Processor : Intel(R) Core(TM) i3-2120 CPU@3.30GHz

RAM : 4GB

Hard disk : 500GB

Monitor : HP

**3.2 SOFTWARE REQUIREMENTS**

Operating System : Windows10 (64 bit)

Frontend : Python

Backend : CSV file, MYSQL

Libraries : Flask, Scikit-learn, PyPDF2

IDE : PyCharm / Jupyter Notebook

Documentation &

Presentation : Microsoft word 2007

**4. SOFTWARE DESCRIPTION**

The **Resume Analyzer using ML and NLP Techniques** is developed using various software tools and libraries to ensure **efficient resume processing, classification, and job recommendation**. Below are the key components:

### ****4.1 Flask:****

* Flask is a **lightweight and flexible web framework** used for building the web-based interface of the Resume Analyzer.
* It enables easy integration of **Machine Learning models** with a user-friendly **resume upload and result display system.**

### ****4.2 Scikit-learn:****

* A popular **Machine Learning library** that provides tools for **classification, regression, clustering, and model evaluation**.
* Used in the Resume Analyzer to implement **Random Forest, SVM, Decision Trees, and other classification models.**

### ****4.3 PyPDF2:****

* A Python library for **extracting text from PDF resumes**.
* Helps in processing **unstructured resume data** and converting it into a structured format for analysis.

### ****Regular Expressions (re module):****

* Used for **pattern recognition and text extraction**, such as:
* Extracting **email addresses** and **phone numbers** from resumes.
* Identifying **skills, education, and experience details**.

### ****4.5 Pickle:****

* A Python module for **serializing and deserializing models**, allowing trained ML models to be **saved and loaded efficiently**.

## ****4.6 PyCharm – Integrated Development Environment (IDE):****

PyCharm is a **powerful IDE developed by Jet Brains**, specifically designed for **Python development.** It provides an **intuitive coding environment** with advanced tools for **debugging, code navigation, and project management**. It is widely used by **data scientists, software developers, and ML engineers** due to its **efficient code writing, debugging, and project structuring capabilities.**

### ****Key Features of PyCharm:****

* **Smart Code Completion** – Provides intelligent **auto-suggestions** for functions, classes, and modules.
* **Syntax Highlighting** – Makes the code more readable by using **different colors for keywords, variables, and functions.**
* **Advanced Debugger** – Allows **step-by-step execution, breakpoints, and variable tracking**for efficient debugging.
* **Integrated Version Control** – Supports **Git, GitHub, SVN**, and other version control systems for **seamless project management**.
* **Virtual Environment Management** – Enables the creation of **separate virtualenvironments** for managing dependencies efficiently.
* **Database Integration** – Direct connection to **SQLite, MySQL, PostgreSQL**, and other databases, allowing database management from the IDE.
* **Unit Testing Support** – Allows running and debugging **unit tests** to ensure **error-free** code execution.

### ****Advantages of PyCharm:****

* **Boosts Productivity** – Provides an **efficient coding environment** that reduces development time.
* **Cross-Platform Compatibility** – Available for **Windows, Linux, and macOS**, making it accessible for all developers.
* **Code Quality Enhancement** – Helps identify **syntax errors, unused variables, and redundant code** through built-in **linting tools.**
* **Customizable Interface** – Developers can **customize themes, key bindings, and plugins** for a personalized experience.

### ****Disadvantages of PyCharm:****

* **High System Resource Usage** – Consumes **a lot of RAM and CPU**, which can slow down lower-end systems.
* **Long Startup Time** – Takes **longer to load** compared to lightweight editors like **VS Code.**
* **Expensive Professional Edition** – The **free Community Edition** lacks some advanced features, requiring a **paid license** for full functionality.
* **Complex for Beginners** – Initial setup, especially with **virtual environments and remote debugging**, can be overwhelming for beginners.

### ****Why We Use PyCharm for the Resume Analyzer Project?****

PyCharm is chosen for this project because of its **efficient debugging, built-in ML support, and excellent code management features.** It allows developers to easily integrate **Flask, scikit-learn, and NLP libraries** while providing a **smooth coding experience**.

## ****4.7 Jupyter Notebook – Interactive Development Environment:****

### ****What is Jupyter Notebook?****

Jupyter Notebook is an **open-source, interactive computing environment** that allows users to write and execute Python code in a **cell-based format**. It is widely used for **Machine Learning, Data Science, and NLP-based projects** due to its flexibility and ease of visualization.

### ****Key Features of Jupyter Notebook:****

* **Interactive Coding Environment** – Allows developers to **write, execute, and debug code in separate cells.**
* **Supports Markdown & Rich Text** – Enables **inline documentation** using Markdown, LaTeX, and HTML.
* **Data Visualization** – Compatible with **Matplotlib, Seaborn, and Plotly**, making it ideal for data exploration.
* **Supports Multiple Languages** – Although Python is the primary language, Jupyter also supports **R, Julia, and Scala**.
* **Remote Accessibility** – Can be hosted on a **server or cloud environment**, allowing users to access notebooks from anywhere.
* **Built-in Kernel Management** – Supports different Python environments and allows running multiple kernels for various projects.

### ****Advantages of Jupyter Notebook:****

* **Best for Exploratory Data Analysis (EDA)** – Ideal for **testing small code snippets** and **visualizing data instantly.**
* **Easy to Use** – Beginners can **quickly start coding** without the need for extensive setup.
* **Supports Interactive Widgets** – Provides **interactive charts, forms, and data visualization tools**.
* **Good for Documentation** – Inline Markdown support makes it easy to document code while writing it.

### ****Disadvantages of Jupyter Notebook:****

* **Not Ideal for Large Projects** – Managing **complex applications** inside Jupyter can be difficult compared to full-fledged IDEs.
* **Version Control Challenges** – Jupyter notebooks are stored in **JSONformat**, making it difficult to track changes using **Git**.
* **Higher Memory Consumption** – Running multiple **heavy datasets** can slow down the system.
* **Limited Code Structure Support** – Unlike PyCharm, it does not support **project-based coding structures**, making it less suitable for **large-scale projects.**

### ****Why Use Jupyter Notebook for the Resume Analyzer Project?****

Jupyter Notebook is used in this project for:

* **Exploratory Data Analysis (EDA)** – Helps in **understanding the dataset** before training models.
* **Machine Learning Model Development** – Used for **testing and training models before deployment.**
* **NLP Feature Extraction** – Applies **TF-IDF, Word2Vec, and Count Vectorizer** for resume text processing.
* **Visualization & Debugging** – Helps analyze **classification results and confusion matrices** interactively.

### ****Combining PyCharm and Jupyter Notebook:****

For the **Resume Analyzer using ML and NLP Techniques**, we use:

* **Jupyter Notebook** for **data pre-processing, feature extraction, and model training.**
* **PyCharm** for **integrating models into the Flask web application and building a production-ready system.**

This hybrid approach **enhances development efficiency** by leveraging **the best features of both tools.** In the **Resume Analyzer using ML and NLP Techniques**, both **PyCharm** and **Jupyter Notebook** play crucial roles in different stages of the project. While **JupyterNotebook** is primarily used for **data exploration, model training, and experimentation**, **PyCharm** is used for **building, integrating, and deploying the final application**.

By combining these two powerful tools, we ensure an **efficient and structured workflow** for **Machine Learning development, debugging, and deployment**.

#### ****1. Jupyter Notebook for Research & Model Development:****

* Used for **data exploration, visualization, and testing different NLP techniques.**
* Supports **Markdown**, making it easier to document findings, test hypotheses, and iterate on model improvements.
* Ideal for **prototyping ML models**, performing **feature extraction (TF-IDF, Word2Vec, etc.),**and**evaluating classification algorithms.**
* Allows **step-by-step execution** of code, making debugging and error handling more manageable during the model-building phase.

#### ****2. PyCharm for Code Structuring & Deployment:****

* Helps organize the project into a **structured format** with multiple scripts, modules, and dependencies.
* Provides a **robust debugging environment** to ensure seamless model integration into the Flask web application.
* Facilitates **version control with Git**, allowing efficient team collaboration.
* Enables **virtual environment management**, ensuring all dependencies are properly installed and managed.

### ****Workflow:****

1. **Data Pre-processing in Jupyter Notebook**
   * Load and clean the dataset.
   * Apply **text pre-processing techniques** such as tokenization, stop word removal, and stemming.
   * Convert text into numerical representations using **TF-IDF, Word2Vec, or Count Vectorizer.**
   * Perform **Exploratory Data Analysis (EDA)** using **Matplotlib and Seaborn** for visualization.
2. **Model Training & Evaluation in Jupyter Notebook**
   * Train **Machine Learning models** such as Random Forest, SVM, and Decision Trees.
   * Evaluate models using **classification reports and confusion matrices**.
   * Fine-tune hyper parameters for better performance.
3. **Code Refactoring & Integration in PyCharm**
   * Transfer the **best-performing model** from Jupyter Notebook to PyCharm.
   * Convert **notebook-based code** into well-structured Python scripts.
   * Integrate the model into a **Flask web application** for real-time resume analysis.
4. **Testing & Debugging in PyCharm**
   * Use PyCharm’s **debugger** to check for errors and optimize code performance.
   * Perform **unit testing** to validate different modules (resume parsing, classification, job recommendation).
   * Ensure seamless execution of the model within the Flask web application.
5. **Deployment & Scalability**
   * Deploy the **Resume Analyzer application** on a local or cloud-based server.
   * Use **PyCharm for version control (Git)** to track changes and ensure code stability.
   * Implement additional features such as **database integration** and API connectivity for enhanced functionality.

### ****MySQL****

MySQL is a widely used **open-source relational database management system (RDBMS)** that allows for efficient storage, retrieval, and management of structured data. It is based on Structured Query Language (SQL), and it supports multiple users, making it ideal for both small-scale applications and large enterprise-level systems.

MySQL is known for its speed, reliability, and flexibility. It is cross-platform and integrates well with web-based technologies and programming languages like **Python**, which is used in the Resume Analyzer Project.

In this project, MySQL plays a key role in managing **admin login credentials** and storing user data. It serves as the backbone for data operations, enabling secure and scalable access to records such as usernames and passwords.

### ****Advantages of MySQL****

1. **Open Source and Free**
   * MySQL is free to use and open-source, which makes it accessible for developers, start-ups, and large organizations alike.
2. **High Performance**
   * MySQL offers fast data access and query processing, making it suitable for real-time applications like web apps and dashboards.
3. **Cross-Platform Support**
   * It runs on major operating systems like Windows, macOS, and Linux, making it highly portable
4. **Scalability**
   * It can handle databases from small to very large sizes and is capable of managing thousands of users simultaneously.
5. **Data Security**
   * Offers robust access control, user authentication, and secure connections to ensure data confidentiality and integrity.
6. **Community Support**
   * MySQL has a large user community, excellent documentation, and many tutorials, making it easy to get help and learn.
7. **Integration with Python**
   * It works seamlessly with Python through libraries like MySQL-connector-python, making it suitable for web frameworks like Flask and Django

### ****Disadvantages of MySQL****

1. **Basic Support for Complex Transactions**
   * While MySQL handles simple transactions well, it may fall short in high-complexity scenarios compared to databases like PostgreSQL

.

1. **Limited Advanced Features**
   * Certain advanced features (like full-text search or materialized views) are either limited or less powerful than other database systems.
2. **Case Sensitivity Can Be Confusing**
   * Table and field names can behave differently across platforms depending on case sensitivity settings, which can lead to unexpected bugs.
3. **Concurrency Limitations**
   * In high-write environments or large enterprise applications, concurrency handling may require careful configuration or tuning.
4. **Not Ideal for Unstructured Data**
   * Since MySQL is a relational database, it is not optimal for handling large volumes of unstructured or semi-structured data like JSON, multimedia, or documents — which NoSQL databases like MongoDB handle better.

### Tools Used with MySQL

* **MySQL Workbench**: GUI tool used to create schemas, manage tables, and run queries visually.
* **MySQL Server**: The core RDBMS engine running on the local host.
* **MySQL-connector-python**: Python library for executing SQL queries and managing connections from the Flask application.

### Purpose of Using MySQL

The main purpose of integrating MySQL into the Resume Analyzer Project is to provide **persistent, structured, and secure data storage** for various system components. While the resume parsing and machine learning functionalities are handled by Python-based services, MySQL handles:

* **Admin login authentication**
* **User signup data (if implemented)**
* **Storing parsed resume results and logs** (as a future enhancement)

### Benefits of Using MySQL

* **Structured Data Management**: Handles tabular data like usernames, passwords, and resume logs efficiently.
* **Cross-Platform**: Runs on Windows, macOS, and Linux with ease.
* **High Performance**: Fast data access and optimized for real-time querying.
* **Scalable**: Can support additional features like multi-user access, analytics logging, and job tracking.
* **Community & Support**: Well-documented with large community support

**HTML:**

HTML, or Hypertext Markup Language, is the standard markup language used to create and structure content on the web. It provides the basic building blocks of web pages by defining elements such as headings, paragraphs, links, images, tables, and forms. HTML is not a programming language; rather, it is a markup language that tells a web browser how to display content.

**Advantages of HTML:**

* **Simplicity**: HTML is easy to learn and use, making it beginner-friendly.
* **Platform Independent**: Works across all platforms and browsers.
* **Free and Open**: HTML is not tied to any vendor or platform and is open for everyone to use.
* **Integrates Well**: HTML integrates seamlessly with CSS and JavaScript for enhanced styling and interactivity.
* **SEO-Friendly**: Proper use of HTML tags helps improve website visibility on search engines.
* **Multimedia Support**: HTML5 allows embedding of audio and video directly into web pages without plugins.

**Future of HTML:**

The evolution of HTML continues to align with the growing demands of modern web applications. Key trends shaping the future include:

* **Progressive Web Apps (PWAs)**: HTML5 is a foundation for PWAs, which offer app-like experiences directly in the browser.
* **Accessibility Improvements**: Ongoing development is enhancing support for assistive technologies.
* **Web Components**: HTML now supports reusable custom elements, making code modular and maintainable.
* **Native Support for APIs**: HTML5 works with modern web APIs for geolocation, offline storage, notifications, and more.
* **Integration with AI and IoT**: HTML will continue playing a role in interfaces for intelligent and interconnected devices.

**CSS:**

CSS, or Cascading Style Sheets, is a style sheet language used to control the presentation and design of web pages written in HTML. While HTML provides the structure of the content, CSS defines how that content looks—such as its layout, colors, fonts, and spacing. CSS helps make websites more visually appealing and user-friendly.

**Advantages of CSS:**

* **Separation of Concerns**: Keeps content (HTML) and design (CSS) separate for easier maintenance.
* **Reusability**: Styles can be reused across multiple pages using external style sheets.
* **Consistency**: Ensures a uniform look and feel throughout the website.
* **Improved Load Times**: Cleaner code and caching of external style sheets lead to faster performance.
* **Accessibility and Responsiveness**: Enables responsive design through media queries and flexible layouts.

**The Future of CSS:**

CSS continues to evolve to meet modern web design requirements. The future includes:

* **CSS Grid and Flex box**: Advanced layout models that simplify complex designs.
* **Custom Properties (CSS Variables)**: Allow for more flexible and dynamic styling.
* **Sub grid and Container Queries**: Give finer control over nested and component-based layouts.
* **Integration with JavaScript Frameworks**: CSS is increasingly integrated with React, Vue, and Angular using CSS-in-JS libraries.
* **Enhanced Performance and Features**: Browsers continue to optimize CSS rendering and introduce new styling capabilities.

**JavaScript:**

JavaScript is a high-level, interpreted programming language primarily used to create dynamic and interactive effects within web browsers. It allows developers to implement complex features on web pages, such as interactive forms, real-time updates, animations, and client-side validations. JavaScript runs in the browser without the need for server-side processing, making it essential for modern web development.

**Core Features of JavaScript:**

* **Client-Side Scripting**: JavaScript runs in the user’s browser, enabling interactive behavior on web pages.
* **Event Handling**: Detects and responds to user actions like clicks, hovers, and form submissions.
* **DOM Manipulation**: Allows developers to dynamically change the structure and content of HTML documents.
* **AJAX Requests**: Facilitates communication with the server without refreshing the page.
* **Compatibility**: Supported by all major browsers without any need for additional plugins.

**Advantages of JavaScript:**

* **Interactivity**: Enhances user engagement by allowing real-time updates and dynamic content.
* **Speed**: JavaScript runs immediately in the browser, improving response times and performance.
* **Versatility**: Can be used on both the front-end (with frameworks like React or Vue) and the back-end (with Node.js).
* **Rich Ecosystem**: Offers a wide range of libraries and frameworks that speed up development.
* **Community Support**: Large and active community with extensive resources and tutorials.

**5. PROBLEM DESCRIPTION**

Recruitment is a **time-intensive and error-prone** process when resumes are manually screened. Organizations receive **hundreds to thousands of resumes** for each job opening, making it difficult to efficiently evaluate candidates. Traditional resume screening is **slow, subjective, and inconsistent**, leading to challenges such **as human bias, errors in shortlisting, and lack of automation.**

The **Resume Analyzer using ML and NLP Techniques** addresses these issues by automating the **resume screening, parsing, classification, and job recommendation** process. It **extracts key details, categorizes resumes based on job roles, and suggests relevant jobs,** making hiring **faster, fairer, and more efficient**.

### ****System Features and Sections:****

To enhance the user experience and streamline recruitment, the Resume Analyzer includes the following sections:

**1. Homepage Section:**

**Purpose:** The **homepage** of the Resume Analyzer serves as the central hub for users to interact with the system. Its primary purpose is to provide a seamless and user-friendly interface that allows users to upload their resumes and instantly receive valuable insights. It also acts as a gateway for administrators to access the backend dashboard for resume management.

**Features:**

* Enable users to upload resumes in a convenient and intuitive way
* Provide real-time extraction and analysis of resume content
* Present category predictions and job recommendations clearly
* Allow secure admin login access for backend management

### ****2. About Section:****

**Purpose:** Provides an overview of the **Resume Analyzer system**, its functionality, and benefits.

**Features:**

* Explains how **ML and NLP** are used to analyze resumes.
* Describes the **resume parsing, classification, and job recommendation** process.
* Highlights the **advantages of automation** in recruitment.
* Includes **user testimonials or success stories** to build credibility.

### ****3. Upload Section:****

**Purpose:** Allows users to **upload resumes** for automated analysis.

**Features:**

* Supports **multiple file formats** (PDF, DOCX, TXT).
* **Drag-and-drop functionality** for easy file uploads.
* Displays **real-time progress** while processing the resume.
* Extracts and **displays structured information**, such as:Name, Contact Information, Skills, Education, and Work Experience.
* Provides **instant results** with **predicted job category** and **recommended jobs**.

### ****4. Signup Section:****

**Purpose:** Enables **users to create an account** for personalized job recommendations and resume tracking.

**Features:**

* **User Registration Form** with fields for: Name, Email, Password, Phone Number, Preferred Job Category.
* Allows users to **save uploaded resumes** for future access (In future).
* Enables **job tracking and application history**.
* Provides a **dashboard** with user-specific recommendations.

### ****5. Login Section:****

#### ****Purpose:**** The Login Section provides ****secure access**** for registered users to the Resume Analyzer system. It ensures that only authorized users can upload resumes, view results, and receive personalized job recommendations.

#### ****Features:****

* **User Authentication** via email and password (MySQL-based).
* **Validation and Feedback** for successful or failed login attempts.
* **Secure Access** to resume analysis and job recommendation features.
* **Clean and Simple UI** for quick login.
* **Prepares system for user-specific functionalities** like saved results and session handling (future scope).

### ****6. Contact Section:****

**Purpose:** Allows users to **reach out for support, inquiries, or feedback**.

**Features:**

* Contact form with fields for: Name, Email, Subject, and Message.
* Live **chat support** for instant assistance.
* Displays **company contact details**, including email and phone number.
* **FAQ section** to answer common questions about resume parsing and job recommendations.

The **Resume Analyzer using ML and NLP Techniques** provides an **intelligent, automated** approach to resume screening and job recommendation. With dedicated sections like **Homepage**, **About, Upload, Signup, Login and Contact**, it ensures a **seamless user experience** for both job seekers and recruiters. By **eliminating manual effort, reducing bias, and improving accuracy,** this system revolutionizes the recruitment process.

**6. SYSTEM DESIGN**

**6.1 USE CASE DIAGRAM:**

Homepage

Sign up

Login

About

Upload Resume

Contact

Extracted information’s

USER

Contact

Contact

**6.2 ACTIVITY DIAGRAM:**

Homepage

Signup

Login

Upload Resume

No

Display error message

Validate

Format

Yes

Parse Resume

Classify Resume

Parse Resume

Parse Resume

Display Result

|  |  |
| --- | --- |
| **Shape** | **Description** |
|  | Attribute |
|  | Multi-valued Attribute |
|  | Primary-key attribute |
|  | Strong entity |
|  | Relationship |
|  | Link |
|  | Total participation |
|  | Composite Attribute |

**6.3 TABLE DESIGN**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **FIELD NAME** | **DATA TYPE** | **DESCRIPTION** |
| 1 | ID | Int | Unique id for the user |
| 2 | EMAIL | Varchar(100) | Email of the user |
| 3 | USERNAME | Varchar(100) | The name of the user |
| 4 | PASSWORD | Varchar(100) | Password created by user |

**1. USER SIGNUP TABLE**

**2. DATASET TABLE FOR CATEGORY MODEL:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **FIELD NAME** | **DATA TYPE** | **DESCRIPTION** |
| 1 | ID | Int | Unique id for the user |
| 2 | CATEGORY | Varchar(50) | The categories in the resume |
| 3 | FEATURES | Varchar(1000) | Features in the resume to parse |

**3. DATASET TABLE FOR JOB RECOMMENDATION MODEL:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **FIELD NAME** | **DATA TYPE** | **DESCRIPTION** |
| 1 | ID | Int | Unique id for the user |
| 2 | CATEGORY | Varchar(100) | Unique category to recommend job |
| 3 | FEATURES | Varchar(1500) | Unique features includes (skill, education, name, contact, e-mail) |

**7. TESTING AND IMPLEMENTATION**

**UNIT TESTING**

Unit testing is a level of software testing where individual or components of software are tested. The purpose is to validate that each unit of software platforms as designed. A unit is a smallest testable part of any software it usually has one or a few inputs and usually a single output.

**INTEGRATION TESTING**

Integration testing is a level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and stubs are used to assist in integration testing. The main objective of integrating is to reduce risks and finding defects of the software components.

**SYSTEM TESTING**

System testing is a level of testing that validates the complete and fully integrated software product. The purpose of system is to evaluate the end to end system specifications. Usually, the software is only one element large computer based system. System testing tests is not only the design, but also the behaviour and even they believed expectations of the customers.

**VALIDATION TESTING**

At the culmination of integration testing, software is completely assembled as a package, interfacing error have been uncovered and corrected and finals series of software tests begin-validation test begins. Validation testing can be defined in many ways.

**OUTPUT TESTING**

After performing the validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the require output in the specified format. Asking the users about the format required by them, tests the outputs generated are displayed by this system under consideration. The output format is considered in two ways. One is on screen and another is printed format. The output format on this screen is found to be correct as the format was designed in the system design phase according to the user needs.

**USER ACCEPTANCE TESTING**

User acceptance of a system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with the perspective system users at the time of developing and making changes were ever require preparation test data place viral role in the system testing. After the preparation test data place vital role in the system testing after the preparing test data the system under study is tested using the test data. While testing the testing by using test data errors are again uncovered. These errors are noted for future use.

**IMPLEMENTATION PLAN**

An importation aspect of a system a lust job is to make sure that the new design is implemented to the established standards. The tem implementation has different meaning, ranging from the conversion of a basic application to a complete replacement of a computer system. Implementation used here is to means the process of converting a new system design into an operational one. Conversion is one aspect of implementation. There are three types of implementation.

Asking the users about the format required by them, tests the outputs generated are displayed by this system under consideration. The output format is considered in two ways. One is on screen and another is printed format. The output format on this screen is found to be correct as the format was designed in the system design phase according to the user needs.

Implementation of a new computer system to replace an existing one this is usually a difficult conversion. If not properly planned there can be many problems. Some large computer system has taken as year to convert. Implementation of a modified application to replace an existing one using same computers this type of conversion is relatively easy to handle provided there are no major changes in the files.

Implementation is the process of converting a new or revised system design into an operation one. It is the key stage in achieving a successful new system, because usually it involves a lot of up-level in the user department.

Therefore it must be carefully planned and controlled apart from the users and testing of the system. Educations of users should really have taken design work training has to been given to the staff regarding the new system, once the staff is trained, the system can be tested.

### ****CROSS-BROWSER COMPATIBILITY TESTING****

The Resume Analyzer is a web-based application and must function correctly across different browsers. This testing ensures that UI elements such as resume upload, login, signup, and result display work consistently on:

* Google Chrome
* Mozilla Firefox
* Microsoft Edge
* Safari

All visual elements, interactive forms, and resume result sections were tested for **responsive behaviour and consistent rendering**.

### ****SECURITY TESTING****

Given that the system handles **personal information (names, emails, phone numbers, and resumes)**, security is a critical aspect. The following areas were tested:

* **Input Validation**: All user inputs (login, signup, file upload) are validated to prevent **SQL injection** and **form tampering**.
* **Authentication Control**: Login attempts with incorrect credentials are handled gracefully, and successful logins are validated against stored hashed passwords.
* **Session Management**(Planned enhancement): Secure session handling is planned to maintain user state post-login.

### ****MACHINE LEARNING MODEL PERFORMANCE TESTING****

The core functionality of this system depends on ML-based categorization and job recommendation.

Testing focused on:

* **Accuracy** of classification models using metrics like **precision, recall, and F1-score**.
* **Confusion Matrix Evaluation** for identifying misclassification patterns.
* **Speed of Prediction** to ensure near real-time response during resume uploads.

All models were tested on **training, validation, and test datasets** to avoid over fitting and ensure generalization.

### ****FILE HANDLING AND FORMAT TESTING****

The upload module supports **PDF and TXT files**. The following tests were conducted:

* Upload of **valid files** for resume parsing.
* Rejection of **unsupported formats** with proper error messages.
* Uploads of **empty or corrupt files** to check system stability.

This ensures robustness and prevents application crashes due to invalid input.

### ****DATABASE INTEGRATION TESTING****

Since MySQL is used for **user authentication and data storage**, all queries related to:

* **Signup**
* **Login**
* **Data retrieval**

Were tested with valid and invalid inputs.  
This ensures smooth communication between the Flask application and the MySQL database.

### ****IMPLEMENTATION BEST PRACTICES****

#### ****Modular Design:****

All functionalities such as resume parsing, classification, job recommendation, and contact extraction are implemented in **separate functions/modules** to ensure maintainability and reusability.

#### ****Progressive Enhancement Plan:****

Future enhancements such as **user dashboards, saved resume history, and email notifications** are planned with the current modular backend in mind, making future upgrades easier.

#### ****Deployment:****

The application is deployed locally using **Flask** and can be extended to platforms like **Heroku, AWS, or Python Anywhere** for broader accessibility.

**8. SAMPLE SOURCE CODE**

**APP CODE:**

from flask import Flask,request,render\_template

import pickle

from PyPDF2 import PdfReader

import re

import mysql.connector

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

# Connect to MySQL

db = mysql.connector.connect(

host="localhost",

user="root", # use your MySQL username

password="180378", # replace with your password

database="resume\_analyzer\_db"

)

cursor = db.cursor()

#load models \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

rf\_classifier\_categorization = pickle.load(open('models/rf\_classifier\_categorization.pkl', 'rb'))

tfidf\_vectorizer\_categorization = pickle.load(open('models/tfidf\_vectorizer\_categorization.pkl', 'rb'))

rf\_classifier\_job\_recommendation = pickle.load(open('models/rf\_classifier\_job\_recommendation.pkl', 'rb'))

tfidf\_vectorizer\_job\_recommendation = pickle.load(open('models/tfidf\_vectorizer\_job\_recommendation.pkl', 'rb'))

#helpers function \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

def cleanResume(txt):

cleanText = re.sub('http\S+\s', ' ', txt)

cleanText = re.sub('RT|cc', ' ', cleanText)

cleanText = re.sub('#\S+\s', ' ', cleanText)

cleanText = re.sub('@\S+', ' ', cleanText)

cleanText = re.sub('[%s]' % re.escape("""!"#$%&'()\*+,-./:;<=>?@[\]^\_`{|}~"""), ' ', cleanText)

cleanText = re.sub(r'[^\x00-\x7f]', ' ', cleanText)

cleanText = re.sub('\s+', ' ', cleanText)

return cleanText

# Prediction and Category Name

def predict\_category(resume\_text):

resume\_text = cleanResume(resume\_text)

resume\_tfidf = tfidf\_vectorizer\_categorization.transform([resume\_text])

predicted\_category = rf\_classifier\_categorization.predict(resume\_tfidf)[0]

return predicted\_category

def job\_recommendation(resume\_text):

resume\_text= cleanResume(resume\_text)

resume\_tfidf = tfidf\_vectorizer\_job\_recommendation.transform([resume\_text])

recommended\_job = rf\_classifier\_job\_recommendation.predict(resume\_tfidf)[0]

return recommended\_job

def pdf\_to\_text(file):

reader = PdfReader(file)

text = ''

for page in range(len(reader.pages)):

text += reader.pages[page].extract\_text()

return text

# resume parsing

def extract\_contact\_number\_from\_resume(text):

contact\_number = None

# Use regex pattern to find a potential contact number

pattern = r"\b(?:\+?\d{1,3}[-.\s]?)?\(?\d{3}\)?[-.\s]?\d{3}[-.\s]?\d{4}\b"

match = re.search(pattern, text)

if match:

contact\_number = match.group()

return contact\_number

def extract\_email\_from\_resume(text):

email = None

# Use regex pattern to find a potential email address

pattern = r"\b[A-Za-z0-9.\_%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}\b"

match = re.search(pattern, text)

if match:

email = match.group()

return email

def extract\_name\_from\_resume(text):

name = None

# Use regex pattern to find a potential name

pattern = r"(\b[A-Z][a-z]+\b)\s(\b[A-Z][a-z]+\b)"

match = re.search(pattern, text)

if match:

name = match.group()

return name

**APP ROUTE CODE:**

#route path \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@app.route('/')

def resume():

return render\_template('resume.html')

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

def signup():

name = request.form.get('name')

email = request.form.get('email')

password = request.form.get('password')

if not name or not email or not password:

return render\_template('resume.html', signup\_message="All fields are required!")

try:

cursor.execute("INSERT INTO users (name, email, password) VALUES (%s, %s, %s)", (name, email, password))

db.commit()

return render\_template('resume.html', signup\_message="Signup successful!")

except mysql.connector.Error as err:

return render\_template('resume.html', signup\_message="Error: " + str(err))

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

def login():

email = request.form.get('email')

password = request.form.get('password')

cursor.execute("SELECT \* FROM users WHERE email = %s AND password = %s", (email, password))

user = cursor.fetchone()

if user:

return render\_template('resume.html', login\_message="Login successful!")

else:

return render\_template('resume.html', login\_message="Invalid credentials!")

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

def pred():

if 'resume' not in request.files:

return render\_template("resume.html", message="No resume file uploaded.")

file = request.files['resume']

filename = file.filename

if filename == '':

return render\_template("resume.html", message="No selected file.")

if filename.endswith('.pdf'):

text = pdf\_to\_text(file)

eliffilename.endswith('.txt'):

text = file.read().decode('utf-8')

else:

return render\_template('resume.html', message='Invalid file format. Please upload a PDF or TXT file.')

predicted\_category = predict\_category(text)

recommended\_job = job\_recommendation(text)

phone = extract\_contact\_number\_from\_resume(text)

email = extract\_email\_from\_resume(text)

extracted\_skills = extract\_skills\_from\_resume(text)

extracted\_education = extract\_education\_from\_resume(text)

name =extract\_name\_from\_resume(text)

return render\_template('resume.html', predicted\_category=predicted\_category, recommended\_job=recommended\_job,

phone=phone, name=name, email=email, extracted\_skills=extracted\_skills,

extracted\_education=extracted\_education)

#py main

if \_\_name\_\_=="\_\_main\_\_":

app.run(debug=True)

def extract\_skills\_from\_resume(text):

# List of predefined skills

skills\_list = [

'Python', 'Data Analysis', 'Machine Learning', 'Communication', 'Project Management', 'Deep Learning', 'SQL',

'Tableau',

'Java', 'C++', 'JavaScript', 'HTML', 'CSS', 'React', 'Angular', 'Node.js', 'MongoDB', 'Express.js', 'Git',

'Research', 'Statistics', 'Quantitative Analysis', 'Qualitative Analysis', 'SPSS', 'R', 'Data Visualization',

'Matplotlib',

'Seaborn', 'Plotly', 'Pandas', 'Numpy', 'Scikit-learn', 'TensorFlow', 'Keras', 'PyTorch', 'NLTK', 'Text Mining',

'Natural Language Processing', 'Computer Vision', 'Image Processing', 'OCR', 'Speech Recognition',

'Recommendation Systems',

'Collaborative Filtering', 'Content-Based Filtering', 'Reinforcement Learning', 'Neural Networks',

'Convolutional Neural Networks',

'Recurrent Neural Networks', 'Generative Adversarial Networks', 'XGBoost', 'Random Forest', 'Decision Trees',

'Support Vector Machines',

'Linear Regression', 'Logistic Regression', 'K-Means Clustering', 'Hierarchical Clustering', 'DBSCAN',

'Association Rule Learning',

'Apache Hadoop', 'Apache Spark', 'MapReduce', 'Hive', 'HBase', 'Apache Kafka', 'Data Warehousing', 'ETL',

'Big Data Analytics',

'Photoshop', 'Illustrator',

'InDesign', 'Figma', 'Sketch', 'Zeplin', 'InVision', 'Product Management', 'Market Research',

'Customer Development', 'Lean Startup',

'Business Development', 'Sales', 'Marketing', 'Content Marketing', 'Social Media Marketing', 'Email Marketing',

'SEO', 'SEM', 'PPC',

'Google Analytics', 'Facebook Ads', 'LinkedIn Ads', 'Lead Generation', 'Customer Relationship Management (CRM)'

skills = []

for skill in skills\_list:

pattern = r"\b{}\b".format(re.escape(skill))

match = re.search(pattern, text, re.IGNORECASE)

if match:

skills.append(skill)

return skills

def extract\_education\_from\_resume(text):

education = []

# List of education keywords to match against

education\_keywords = [

'Computer Science', 'Information Technology', 'Software Engineering', 'Electrical Engineering', 'Mechanical Engineering', 'Civil Engineering',

'Chemical Engineering', 'Biomedical Engineering', 'Aerospace Engineering', 'Nuclear Engineering', 'Industrial Engineering', 'Systems Engineering',

'Environmental Engineering', 'Petroleum Engineering', 'Geological Engineering', 'Marine Engineering', 'Robotics Engineering', 'Biotechnology',

'Biochemistry', 'Microbiology', 'Genetics', 'Molecular Biology', 'Bioinformatics', 'Neuroscience', 'Biophysics', 'Biostatistics', 'Pharmacology',

for keyword in education\_keywords:

pattern = r"(?i)\b{}\b".format(re.escape(keyword))

match = re.search(pattern, text)

if match:

education.append(match.group())

return education

def extract\_name\_from\_resume(text):

name = None

# Use regex pattern to find a potential name

pattern = r"(\b[A-Z][a-z]+\b)\s(\b[A-Z][a-z]+\b)"

match = re.search(pattern, text)

if match:

name = match.group()

return name

**HTML, CSS, JS CODE:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>ML Resume Analyzer</title>

<link href="https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;600&display=swap" rel="stylesheet">

<style>

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

font-family: 'Poppins', sans-serif;

}

body {

background: url("{{ url\_for('static', filename='backg.jpg') }}") center/cover no-repeatfixed;

background-size: cover;

background-position: center;

background-attachment: fixed;

color: #fff;

}

/\* Navigation Bar \*/

.navbar {

display: flex;

justify-content: space-between;

align-items: center;

padding: 15px 50px;

background: rgba(15, 32, 39, 0.9);

box-shadow: 0 4px 15px rgba(0, 0, 0, 0.5);

}

.navbar a {

color: white;

text-decoration: none;

padding: 10px 20px;

transition: 0.3s;

}

.navbar a:hover {

background: #ff4757;

border-radius: 5px;

}

.navbar img {

width: 30px;

vertical-align: middle;

}

/\* Hero Section \*/

.hero {

height: 90vh;

background: rgba(0, 0, 0, 0.5);

display: flex;

flex-direction: column;

justify-content: center;

align-items: center;

text-align: center;

}

.hero h1 {

font-size: 48px;

font-weight: 600;

text-shadow: 2px 2px 10px rgba(255, 255, 255, 0.2);

}

.hero p {

font-size: 18px;

margin-top: 10px;

max-width: 600px;

opacity: 0.9;

}

.hero button {

margin-top: 20px;

padding: 12px 25px;

border: none;

background: #ff4757;

color: white;

font-size: 16px;

cursor: pointer;

border-radius: 5px;

transition: 0.3s;

}

.hero button:hover {

background: #ff6b81;

transform: scale(1.05);

}

/\* Sections \*/

.section {

background: rgba(26, 26, 46, 0.9);

padding: 60px 50px;

text-align: center;

border-radius: 12px;

margin: 20px auto;

max-width: 90%;

box-shadow: 0 4px 15px rgba(0, 0, 0, 0.6);

}

**HTML, CSS, JS ROUTE CODE:**

#route path \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@app.route('/')

def resume():

return render\_template('resume.html')

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

def signup():

name = request.form.get('name')

email = request.form.get('email')

password = request.form.get('password')

if not name or not email or not password:

return render\_template('resume.html', signup\_message="All fields are required!")

try:

cursor.execute("INSERT INTO users (name, email, password) VALUES (%s, %s, %s)", (name, email, password))

db.commit()

return render\_template('resume.html', signup\_message="Signup successful!")

except mysql.connector.Error as err:

return render\_template('resume.html', signup\_message="Error: " + str(err))

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

def login():

email = request.form.get('email')

password = request.form.get('password')

cursor.execute("SELECT \* FROM users WHERE email = %s AND password = %s", (email, password))

user = cursor.fetchone()

if user:

return render\_template('resume.html', login\_message="Login successful!")

else:

return render\_template('resume.html', login\_message="Invalid credentials!")

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

def pred():

if 'resume' not in request.files:

return render\_template("resume.html", message="No resume file uploaded.")

file = request.files['resume']

filename = file.filename

if filename == '':

return render\_template("resume.html", message="No selected file.")

if filename.endswith('.pdf'):

text = pdf\_to\_text(file)

eliffilename.endswith('.txt'):

text = file.read().decode('utf-8')

else:

return render\_template('resume.html', message='Invalid file format. Please upload a PDF or TXT file.')

predicted\_category = predict\_category(text)

recommended\_job = job\_recommendation(text)

phone = extract\_contact\_number\_from\_resume(text)

email = extract\_email\_from\_resume(text)

extracted\_skills = extract\_skills\_from\_resume(text)

extracted\_education = extract\_education\_from\_resume(text)

name = extract\_name\_from\_resume(text)

return render\_template('resume.html', predicted\_category=predicted\_category, recommended\_job=recommended\_job,

phone=phone, name=name, email=email, extracted\_skills=extracted\_skills,

extracted\_education=extracted\_education)

#py main

if \_\_name\_\_=="\_\_main\_\_":

app.run(debug=True)

**MODEL CODE (CATEGORY):**

**Import** numpy **as** np

**import** pandas **as** pd

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

**from** sklearn.model\_selection **import** train\_test\_split

**from** sklearn.feature\_extraction.text **import** TfidfVectorizer

**from** sklearn.ensemble **import** RandomForestClassifier

**from** sklearn.metrics **import** accuracy\_score,confusion\_matrix

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

df**=**pd**.**read\_csv('UpdatedResumeDataSet.csv')

clean\_df**=**pd**.**read\_csv('clean\_resume\_data.csv')

resume**=**pd**.**read\_csv('Resume.csv')

clean\_df['Category']**.**value\_counts()

plt**.**figure(figsize**=**(15,5))

sns**.**countplot(clean\_df['Category'])

plt**.**xticks(rotation**=**90)

plt**.**show()

counts**=**clean\_df['Category']**.**value\_counts()

labels**=**clean\_df['Category']**.**unique()

plt**.**figure(figsize**=**(15,10))

plt**.**pie(counts,labels**=**labels,autopct**=**'%1.1f%%',shadow**=True**,colors**=**plt**.**cm**.**plasma(np**.**linspace(0,1,3)))

plt**.**show()

# Balance Dataset:

**From** sklearn.utils **import** resample

*# Define the maximum count among all categories*

max\_count**=**clean\_df['Category']**.**value\_counts()**.**max()

*# Resample each category to match the maximum count*

balanced\_data**=**[]

**for**category**in**clean\_df['Category']**.**unique():

category\_data**=**clean\_df[clean\_df['Category']**==**category]

**if**len(category\_data)**<**max\_count:

*# Perform oversampling for categories with fewer samples*

balanced\_category\_data**=**resample(category\_data,replace**=True**,n\_samples**=**max\_count,random\_state**=**42)

**else**:

*# Perform undersampling for categories with more samples*

balanced\_category\_data**=**resample(category\_data,replace**=False**,n\_samples**=**max\_count,random\_state**=**42)

balanced\_data**.**append(balanced\_category\_data)

*# Concatenate the balanced data for all categories*

balanced\_df**=**pd**.**concat(balanced\_data)

# Train-Test Split:

X**=**balanced\_df['Feature']

y**=**balanced\_df['Category']

X\_train,X\_test,y\_train,y\_test**=**train\_test\_split(X,y,test\_size**=**0.2,random\_state**=**42)

# Encoding (TF-IDF):

tfidf\_vectorizer**=**TfidfVectorizer()

X\_train\_tfidf**=**tfidf\_vectorizer**.**fit\_transform(X\_train)

X\_test\_tfidf**=**tfidf\_vectorizer**.**transform(X\_test)

# Train Random Forest Classifier:

**From** sklearn.metrics **import** classification\_report

rf\_classifier**=**RandomForestClassifier()

rf\_classifier**.**fit(X\_train\_tfidf,y\_train)

*# Step 4: Accuracy Evaluation*

y\_pred**=**rf\_classifier**.**predict(X\_test\_tfidf)

accuracy**=**accuracy\_score(y\_test,y\_pred)

print("Accuracy:",accuracy)

print(classification\_report(y\_test,y\_pred))

*# Confusion Matrix*

conf\_matrix**=**confusion\_matrix(y\_test,y\_pred)

plt**.**figure(figsize**=**(10,8))

sns**.**heatmap(conf\_matrix,annot**=True**,fmt**=**'d',cmap**=**'Blues',xticklabels**=**rf\_classifier**.**classes\_,yticklabels**=**rf\_classifier**.**classes\_)

plt**.**xlabel('Predicted')

plt**.**ylabel('True')

plt**.**title('Confusion Matrix')

plt**.**show()

# Predictive System:

*# Clean resume*

**Import** re

**Def** cleanResume(txt):

cleanText**=**re**.**sub('http\S+\s',' ',txt)

cleanText**=**re**.**sub('RT|cc',' ',cleanText)

cleanText**=**re**.**sub('#\S+\s',' ',cleanText)

cleanText**=**re**.**sub('@\S+',' ',cleanText)

cleanText**=**re**.**sub('[%s]'**%**re**.**escape("""!"#$%&'()\*+,-./:;<=>?@[\]^\_`{|}~"""),' ',cleanText)

cleanText**=**re**.**sub(r'[^\x00-\x7f]',' ',cleanText)

cleanText**=**re**.**sub('\s+',' ',cleanText)

**return** cleanText

*# Prediction and Category Name*

**def**predict\_category(resume\_text):

resume\_text**=**cleanResume(resume\_text)

resume\_tfidf**=**tfidf\_vectorizer**.**transform([resume\_text])

predicted\_category**=**rf\_classifier**.**predict(resume\_tfidf)[0]

**return** predicted\_category

*# Example Usage*

resume\_file**=**'information technology manager network engineer systems management certificate project management certificate'

predicted\_category**=**predict\_category(resume\_file)

print("Predicted Category:",predicted\_category)

# Save Files:

**Import** pickle

pickle**.**dump(rf\_classifier,open('models/rf\_classifier\_categorization.pkl','wb'))

pickle**.**dump(tfidf\_vectorizer,open('models/tfidf\_vectorizer\_categorization.pkl','wb'))

**MODEL CODE (JOB RECOMMENDATION):**

import pandas as pd

df**=**pd**.**read\_csv("Resume Job Recommendation.csv")

df**.**head(2)

df**.**columns

df['Work Type']**.**value\_counts()

df**=**pd**.**read\_csv("jobs\_dataset\_with\_features.csv")

df**.**shape

df**.**head()

*# Dropping classes with less than 6500 instances*

min\_count**=** 6500

role\_counts**=**df['Role']**.**value\_counts()

dropped\_classes**=**role\_counts[role\_counts**<**min\_count]**.**index

filtered\_df**=**df[**~**df['Role']**.**isin(dropped\_classes)]**.**reset\_index(drop**=**True)

# Balance Dataset:

*# Checking the updated role counts*

filtered\_df['Role']**.**value\_counts()

len(filtered\_df['Role']**.**value\_counts())

df**=**filtered\_df**.**sample(n**=**10000)

df**.**head()

**TFIDF:**

from sklearn.model\_selection import train\_test\_split

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score

# Train-Test Split:

*# Splitting the data into features (X) and target (y)*

X **=**df['Features']

y **=**df['Role']

*# Train-test split*

X\_train, X\_test, y\_train, y\_test**=**train\_test\_split(X, y, test\_size**=**0.2, random\_state**=**42)

# Encoding (TF-IDF):

*# TF-IDF vectorization*

tfidf\_vectorizer**=**TfidfVectorizer()

X\_train\_tfidf**=**tfidf\_vectorizer**.**fit\_transform(X\_train)

X\_test\_tfidf**=**tfidf\_vectorizer**.**transform(X\_test)

*# RandomForestClassifier*

rf\_classifier**=**RandomForestClassifier()

rf\_classifier**.**fit(X\_train\_tfidf, y\_train)

*# Predictions*

y\_pred**=**rf\_classifier**.**predict(X\_test\_tfidf)

*# Accuracy*

accuracy **=**accuracy\_score(y\_test, y\_pred)

print("Accuracy:", accuracy)

**Recommendation:**

*# Clean resume*

import re

def cleanResume(txt):

cleanText**=**re**.**sub('http\S+\s', ' ', txt)

cleanText**=**re**.**sub('RT|cc', ' ', cleanText)

cleanText**=**re**.**sub('#\S+\s', ' ', cleanText)

cleanText**=**re**.**sub('@\S+', ' ', cleanText)

cleanText**=**re**.**sub('[%s]' **%**re**.**escape("""!"#$%&'()\*+,-./:;<=>?@[\]^\_`{|}~"""), ' ', cleanText)

cleanText**=**re**.**sub(r'[^\x00-\x7f]', ' ', cleanText)

cleanText**=**re**.**sub('\s+', ' ', cleanText)

return cleanText

*# Prediction and Category Name*

def job\_recommendation(resume\_text):

resume\_text**=**cleanResume(resume\_text)

resume\_tfidf**=**tfidf\_vectorizer**.**transform([resume\_text])

predicted\_category**=**rf\_classifier**.**predict(resume\_tfidf)[0]

return predicted\_category

*# Example Usage*

resume\_file**=** """Objective:

Dedicated and results-oriented Banking professional with a strong background in

Education:

- Bachelor of Business Administration in Finance, XYZ University, GPA: 3.8/4.0

Skills:

- Proficient in financial modeling and analysis using Excel, Bloomberg Terminal, and other financial software

Experience:

Financial Analyst | ABC Bank

- Conducted financial analysis and risk assessment for corporate clients, including credit analysis, financial statement analysis, and cash flow modeling

Certifications:

- Certified Financial Planner (CFP)

- Series 7 and Series 63 Securities Licenses

Languages:

- English (Native)

- Spanish (Proficient)

"""

predicted\_category**=**job\_recommendation(resume\_file)

print("Predicted Category:", predicted\_category)

# Save Files:

import pickle

pickle**.**dump(rf\_classifier,open('rf\_classifier\_job\_recommendation.pkl','wb'))

pickle**.**dump(tfidf\_vectorizer,open('tfidf\_vectorizer\_job\_recommendation.pkl','wb'))

**MODEL CODE(EXTRACTING INFO):**

import re

from pdfminer.high\_level import extract\_text

def extract\_text\_from\_pdf(pdf\_path):

return extract\_text(pdf\_path)

resume\_path = "info resume.pdf"

text = extract\_text\_from\_pdf(resume\_path)

text

**Function to Extract:**

import re

def extract\_contact\_number\_from\_resume(text):

contact\_number = None

*# Use regex pattern to find a potential contact number*

pattern = r"\b(?:\+?\d{1,3}[-.\s]?)?\(?\d{3}\)?[-.\s]?\d{3}[-.\s]?\d{4}\b"

match = re.search(pattern, text)

if match:

contact\_number = match.group(

return contact\_number

phone = extract\_contact\_number\_from\_resume(text)

phone

**Extracting Email Address:**

import re

def extract\_email\_from\_resume(text):

email = None

*# Use regex pattern to find a potential email address*

pattern = r"\b[A-Za-z0-9.\_%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}\b"

match = re.search(pattern, text)

if match:

email = match.group()

return email

email = extract\_email\_from\_resume(text)

email

**Extracting Skills:**

import re

def extract\_skills\_from\_resume(text, skills\_list):

skills = []

for skill in skills\_list:

pattern = r"\b{}\b".format(re.escape(skill))

match = re.search(pattern, text, re.IGNORECASE)

if match:

skills.append(skill)

return skills

*# List of predefined skills*

skills\_list = [

'Python', 'Data Analysis', 'Machine Learning', 'Communication', 'Project Management', 'Deep Learning', 'SQL', 'Tableau',

'Java', 'C++', 'JavaScript', 'HTML', 'CSS', 'React', 'Angular', 'Node.js', 'MongoDB', 'Express.js', 'Git',

'Research', 'Statistics', 'Quantitative Analysis', 'Qualitative Analysis', 'SPSS', 'R', 'Data Visualization', 'Matplotlib','Smart Contracts', 'Web3', 'Non-Fungible Tokens (NFTs)']

extracted\_skills**=**extract\_skills\_from\_resume(text, skills\_list)

if extracted\_skills:

print("Skills:", extracted\_skills)

else:

print("No skills found")

**Extracting Education:**

import re

def extract\_education\_from\_resume(text):

education = []

*# List of education keywords to match against*

education\_keywords = [

'Computer Science', 'Information Technology', 'Software Engineering', 'Electrical Engineering', 'Mechanical Engineering', 'Civil Engineering',

'Chemical Engineering', 'Biomedical Engineering','Conservation Biology', 'Wildlife Biology', 'Zoology']

for keyword in education\_keywords:

pattern = r"(?i)\b{}\b".format(re.escape(keyword))

match = re.search(pattern, text)

if match:

education.append(match.group())

return education

extracted\_education = extract\_education\_from\_resume(text)

if extracted\_education:

print("Education:", extracted\_education)

else:

print("No education information found")

**Extracting Name Using spaCy:**

def extract\_name\_from\_resume(text):

name = None

*# Use regex pattern to find a potential name*

pattern = r"(\b[A-Z][a-z]+\b)\s(\b[A-Z][a-z]+\b)"

match = re.search(pattern, text)

if match:

name = match.group()

return name

name = extract\_name\_from\_resume(text)

if name:

print("Name:", name)

else:

print("Name not found")

# Balance Dataset:

from sklearn.utils import resample

*# Define the maximum count among all categories*

max\_count**=**clean\_df['Category']**.**value\_counts()**.**max()

*# Resample each category to match the maximum count*

balanced\_data**=** []

for category in clean\_df['Category']**.**unique():

category\_data**=**clean\_df[clean\_df['Category'] **==** category]

if len(category\_data) **<**max\_count:

*# Perform oversampling for categories with fewer samples*

balanced\_category\_data**=** resample(category\_data, replace**=**True, n\_samples**=**max\_count, random\_state**=**42)

else:

*# Perform undersampling for categories with more samples*

balanced\_category\_data**=** resample(category\_data, replace**=**False, n\_samples**=**max\_count, random\_state**=**42)

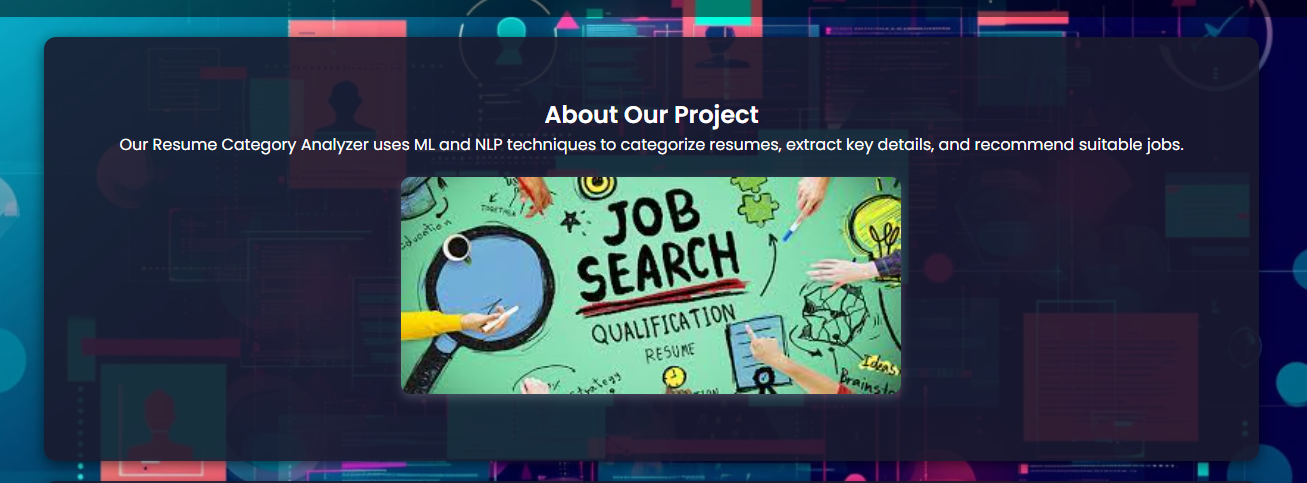
balanced\_data**.**append(balanced\_category\_data)

**9. SAMPLE SCREENSHOTS**

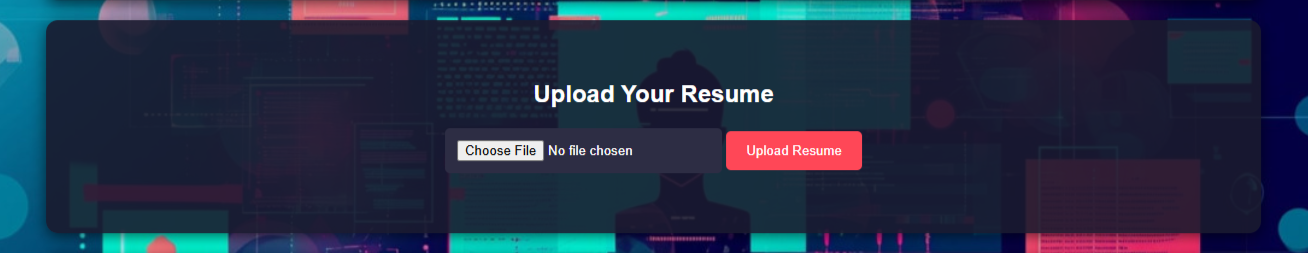
**1. HOMEPAGE:**



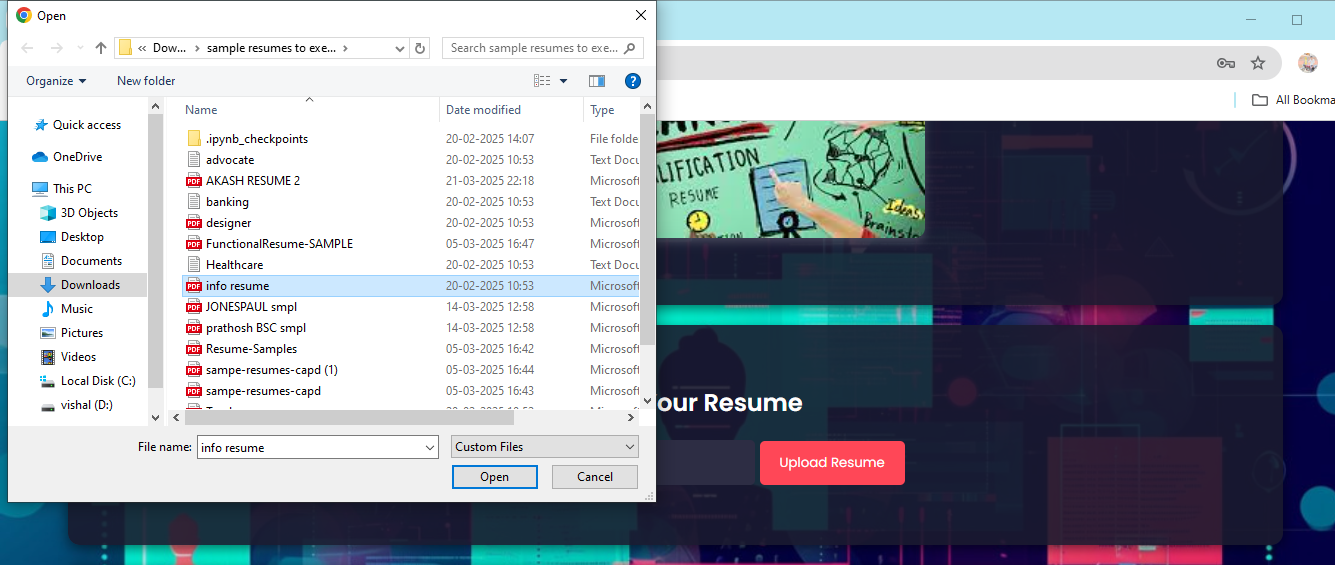
**2. ABOUT SECTION:**



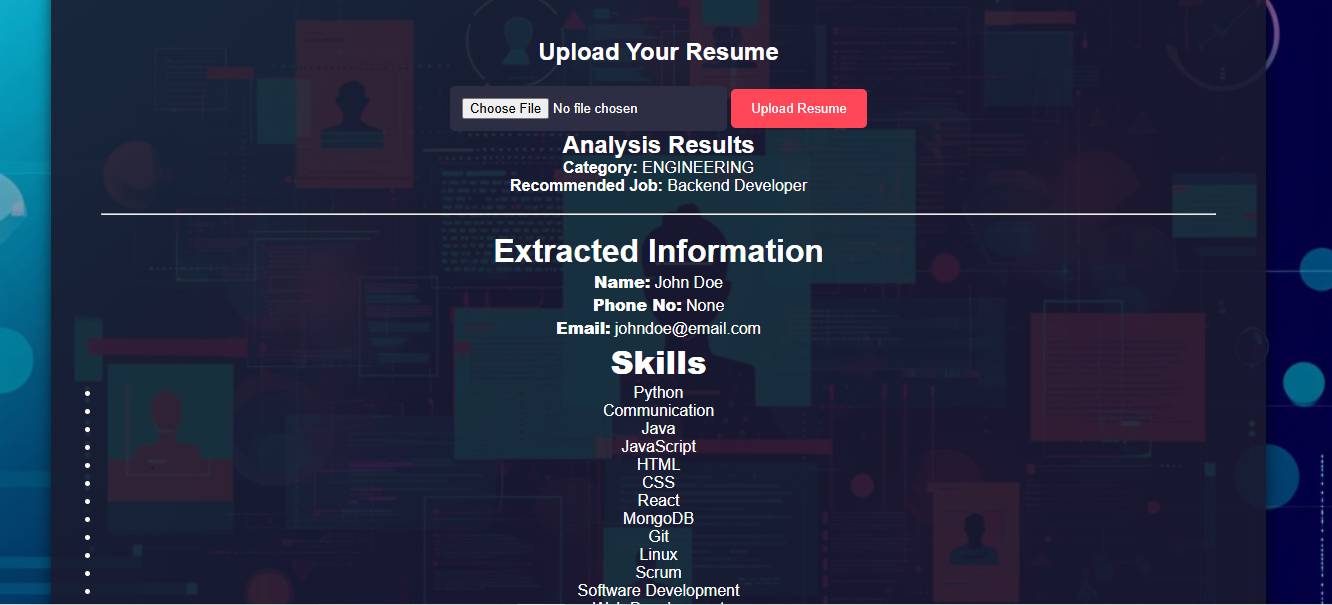
**3. UPLOAD SECTION:**



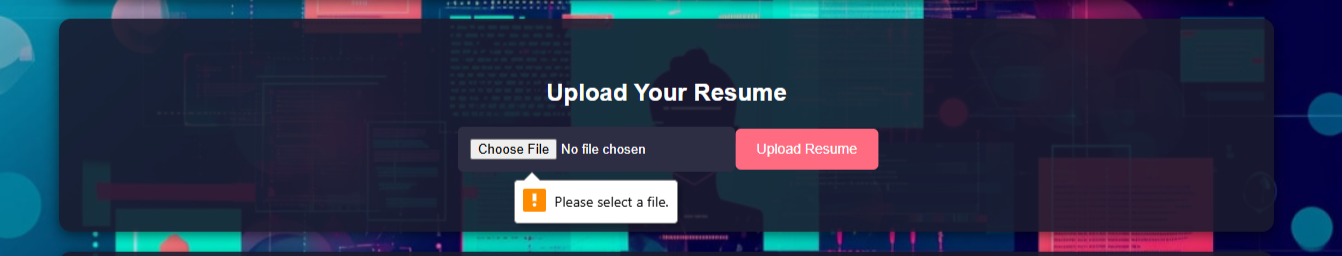
**UPLOAD RESUME:**



**PREDICTED RESULT:**



**UPLOAD RESUME (ERROR PAGE):**



**SIGNUP SECTION:**



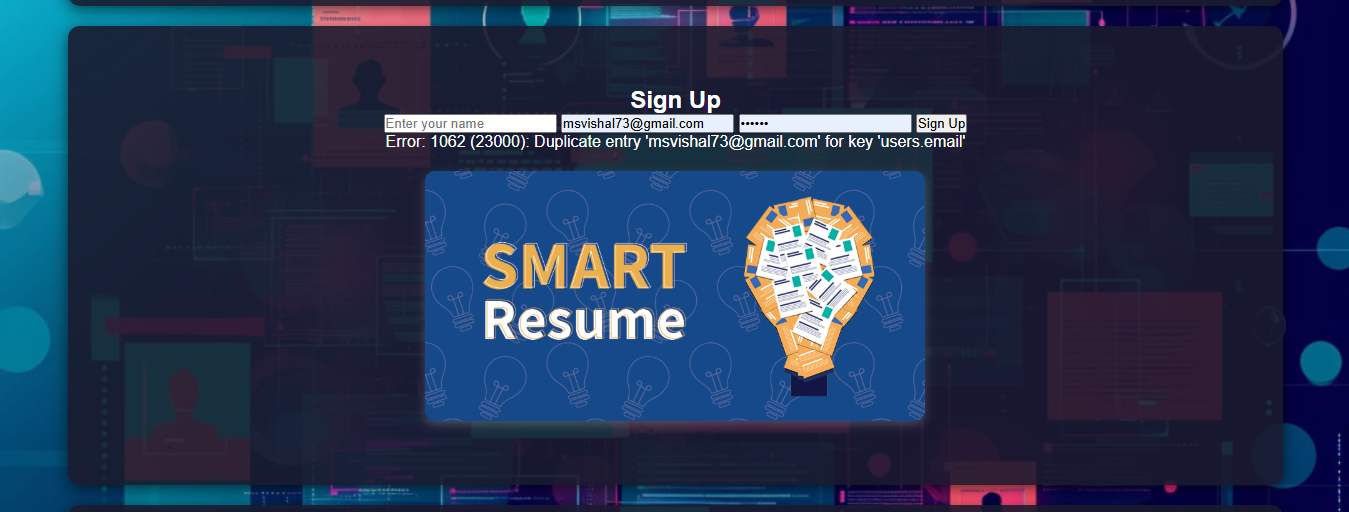
**SIGNUP (ERROR PAGE1):**



**SIGNUP (ERROR PAGE 2):**



**SIGNUP (ERROR PAGE 3):**



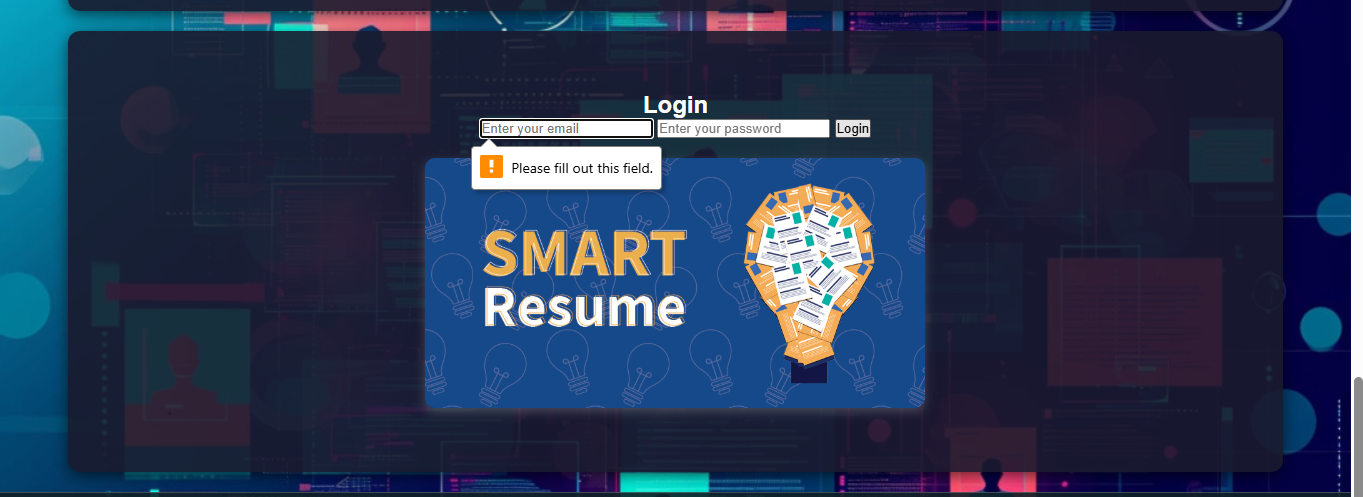
**LOGIN SECTION:**



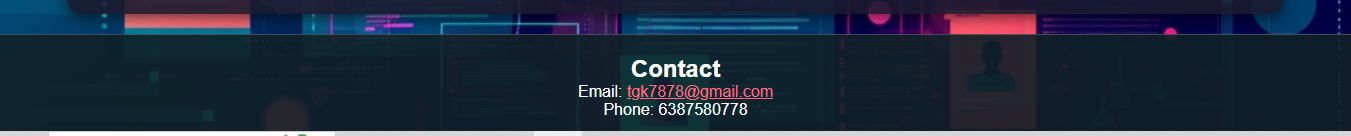
**LOGIN (ERROR PAGE 1):**



**LOGIN (ERROR PAGE 2):**



**CONTACT SECTION:**



**10. CONCLUSION**

The **Resume Analyzer using ML and NLP Techniques** successfully addresses key challenges in traditional recruitment by introducing automation, intelligence, and efficiency into the resume screening process. By integrating Machine Learning for resume classification and Natural Language Processing for extracting structured information, the system ensures faster, fairer, and more accurate hiring decisions.

The web-based application enables users to upload resumes, view real-time analysis results, and receive job recommendations based on their qualifications and skills. With the use of Flask, Scikit-learn, PyPDF2, and MySQL, the system provides a robust backend that is capable of handling large volumes of resumes with minimal human intervention.

Incorporating key modules like Signup, Login, Resume Upload, and Contact Support, the platform delivers a seamless and interactive user experience. Its flexible design allows for easy maintenance, upgrades, and scaling, making it highly adaptable to the evolving needs of recruiters and job seekers.

Overall, the Resume Analyzer stands out as a **cost-effective, scalable, and smart solution** to modern-day recruitment challenges.

**FUTURE ENHANCEMENTS:**

To improve the system's functionality and scalability, the following **enhancements** are planned for future versions:

### ****1. User Dashboard****

* Track resume uploads
* View past analysis reports
* Receive saved job recommendations
* Update profile and preferences

### ****2. Admin Panel****

* View and manage uploaded resumes
* Track user activity
* Manage categories and job roles
* Monitor system logs and performance

### ****3. Resume Feedback & Scoring System****

* Provide a **resume score** based on parameters like skill relevance, education, and formatting.
* Offer suggestions for improving resumes (e.g., missing keywords, formatting tips).

**BIBLOGRAPHY**

**Books:**

1. Sebastian Raschka, & Vahid Mirjalili (2019). *Python Machine Learning* (3rd ed.). Packt Publishing.  
   ➤ Focuses on building ML models with scikit-learn and deep learning using Keras.
2. Christopher Bishop (2006). *Pattern Recognition and Machine Learning*. Springer.  
   ➤ A classic in understanding machine learning fundamentals.

**Online Resources / Links:**

1. **Scikit-learn: Machine Learning in Python**  
   Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825–2830  
   <https://scikit-learn.org>
2. **JupyterLab Documentation**  
   Project Jupyter. (2023). JupyterLab: The Next Generation Interface for Project Jupyter.  
   <https://jupyterlab.readthedocs.io>
3. **Pickle Module — Python Serialization**  
   Python Software Foundation. (2023). pickle — Python object serialization.  
   <https://docs.python.org/3/library/pickle.html>