

Name : Budagum Keerthan Varma  
College : IIT Gandhinagar  
Roll Number : 23110068  
Department : Artificial Intelligence  
[GitHub](#) [LinkedIn](#) [Mail](#)

# System Design Document: AI JOB FINDER

## The Personalized Job Searcher



# JobFinder

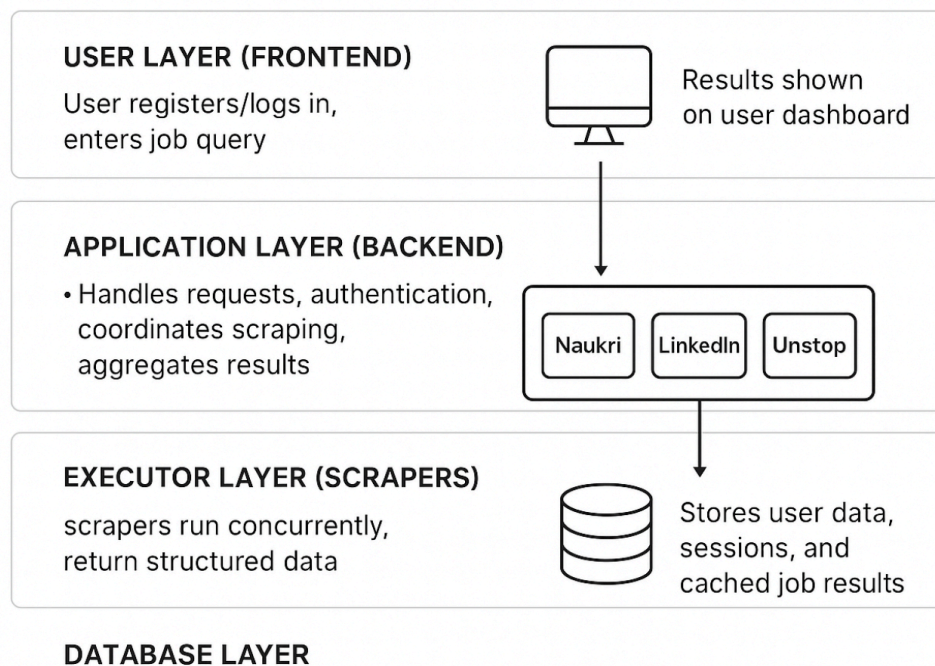
CareerFlow

---

## Introduction

AI JobFinder is a personalized job search platform designed to simplify the process of finding relevant career opportunities. With the increasing number of job portals and scattered listings, job seekers often face challenges in accessing consolidated and accurate results. JobFinder addresses this by integrating multiple sources into a single platform, allowing users to search, filter, and explore job postings efficiently. The system focuses on providing tailored recommendations based on user preferences, ensuring that candidates save time and effort while identifying opportunities that best match their skills and interests.

## System Architecture



**Front End:** The AI Job Finder features a clean and responsive web dashboard built with Flask, HTML, CSS and JavaScript providing an intuitive interface for users to register, log in, and search for jobs effortlessly across multiple platforms.

---

---

**Job Search Automation:** Using Selenium WebDriver, the application automates navigation and data extraction from job portals like Naukri, LinkedIn, and Unstop, ensuring accurate and up-to-date job listings.

**Robust Backend:** Powered by Flask, the backend efficiently handles HTTP requests, manages user authentication, orchestrates concurrent job searches, and aggregates results from multiple sources.

**Robust Data Storage:** User credentials, session information, and search data are securely stored in SQLite, providing a lightweight yet reliable database solution for seamless application performance.

## Components and Modules

### BackEnd:

- **App:** This is the main entry point of the backend. It sets up the Flask application, defines all the routes, and handles HTTP requests from the frontend. It also manages user authentication, sessions, and orchestrates the execution of job scrapers. After the scrapers fetch job listings, `app.py` aggregates the results and sends them to the frontend dashboard for display.
- **Models:** This file defines the database schema using SQLAlchemy. It primarily contains the User model, which stores user credentials and session information securely. `models.py` is responsible for all interactions with the SQLite database, including creating tables, adding new users, and querying user information for login and authentication purposes.
- **Executors folder:** This folder contains all the job scraping modules, each focused on a specific job portal. It includes `Naukri_executor.py`, `Linkedin_executor.py`, `unstop_executor.py`. Each executor uses Selenium WebDriver to perform browser automation and returns structured data to the backend.
- **Data folder:** The backend manages CSV files containing lists of companies, job roles, and locations used to guide searches. It also allows exporting aggregated job results from Naukri, LinkedIn, and Unstop into CSV format, giving users a structured, local copy for analysis or record-keeping.

---

## FrontEnd:

- **Templates folder:** This folder contains all HTML templates rendered by Flask. It includes `base.html` for the landing page, `login.html` and `signup.html` for user authentication, `dashboard.html` for entering job queries, `results.html` for displaying job search results, and `loading.html` which shows a waiting page while job results are being fetched. These templates work dynamically with the backend to provide a smooth and responsive user experience.
- **Static folder:** The static folder holds all frontend assets such as CSS stylesheets, JavaScript files, and images/icons.

## Workflow

The AI Job Finder follows a well-defined workflow to efficiently provide users with aggregated job listings from multiple platforms.

1. **User Input:** Users begin by accessing the dashboard and entering a job query, such as a specific job title, company, or location. Upon clicking the search button, the request is sent to the backend for processing.
2. **Backend Processing:** The Flask backend receives the query, validates the input, and prepares to invoke the job scraping modules. It ensures secure handling of user data and coordinates the subsequent steps.
3. **Concurrent Job Scraping:** Dedicated executors for Naukri, LinkedIn, and Unstop run concurrently using Selenium WebDriver. Each executor automates browser navigation, performs searches on the respective platform, logs in if necessary, and extracts relevant job details such as title, company, location, and job description. Running the scrapers concurrently significantly reduces the total search time.
4. **Data Aggregation and Structuring:** Once the scrapers complete their tasks, the backend collects results from all executors. Duplicate entries are removed, and data is structured in a unified format to ensure consistency across different platforms.
5. **Frontend Display:** While the scraping and aggregation are in progress, the user is shown `loading.html`, which provides visual feedback that the search is running.

Once the results are ready, `results.html` presents the job listings in a clean, organized dashboard format, making it easy for users to browse and filter opportunities.

## Demo

 Job Finder

### Login

Email

Password

Please fill out this field.

Login

Don't have an account? [Sign up here](#)

 Job Finder

### Sign Up

Username

Email

Password

Sign Up


Already have an account? [Login here](#)


Login successful! ✕


# Welcome to Job Finder

- Your personalized job search assistant



 Frontend Developer

 Select company

 Hyderabad

 Search Jobs



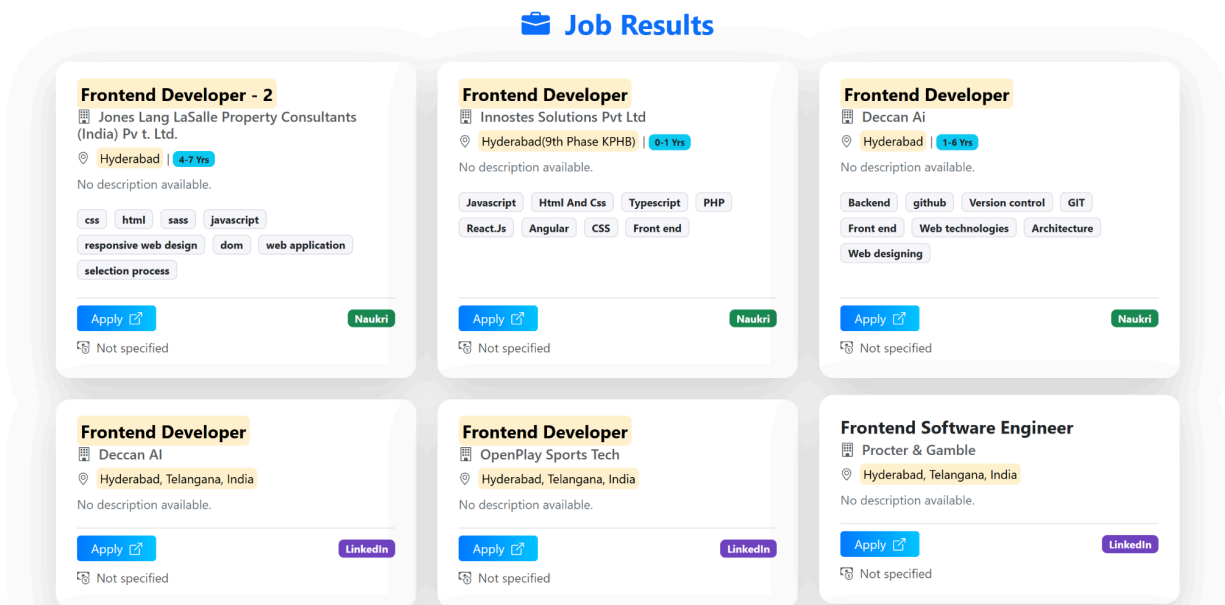
## Fetching jobs, please wait...

We are carefully searching multiple trusted job sources to bring you the most relevant opportunities.

This may take a few moments, but we promise it's worth the wait!

Accuracy matters more than speed — the best matches are coming your way.

Connecting to job platforms...



## Technical Highlights

The AI Job Finder is built using a combination of robust technologies and design principles to deliver an efficient and user-friendly job aggregation platform.

- **Flask-Based Backend:** The backend is powered by Flask, providing a lightweight yet powerful framework to handle HTTP requests, user authentication, session management, and coordination of scraping modules. Its modular structure ensures maintainability and scalability.
- **Concurrent Web Scraping:** The application uses dedicated **executors** for Naukri, LinkedIn, and Unstop, implemented with Selenium WebDriver. These scrapers run concurrently, significantly reducing job search time while fetching data from multiple platforms simultaneously.
- **Structured Data Management:** User credentials, session information, and search metadata are securely stored in **SQLite**, enabling reliable and fast access. The system also uses **CSV files** to maintain lists of companies, job roles, and locations, ensuring accurate and structured job matching.
- **Responsive Frontend:** The frontend leverages HTML, CSS, and JavaScript along with Flask templates ([templates/](#)) and static assets ([static/](#)) to deliver a clean,

---

intuitive, and responsive interface. Pages like **dashboard, results, and loading** ensure smooth user interactions and clear feedback during searches.

- **Real-Time Feedback:** The [loading.html](#) page provides visual cues while scraping is in progress, and [results.html](#) presents job listings in an organized, easy-to-read format, enhancing the user experience.
- **Modular and Extensible Design:** The project's modular architecture allows easy integration of additional job portals, new features, or improved scraping logic without disrupting the existing workflow.

## Limitations

While the AI Job Finder provides a unified platform for job searching across multiple portals, there are certain limitations to consider:

- **Dependence on Web Scraping:** The system relies on Selenium-based scraping. Any structural changes in the job portals' HTML may break the scrapers, requiring updates and maintenance.
- **LinkedIn Delay:** LinkedIn job results often require longer load times due to dynamic content and additional authentication steps, which may slow down searches compared to other portals.
- **Authentication Constraints:** Some portals enforce strict login or captcha verifications, limiting the ability to scrape without manual user input.
- **Scalability Limits:** Since searches are performed in real-time using Selenium, handling a very large number of concurrent users may affect performance.
- **Data Consistency:** As data is fetched live from portals, there may occasionally be duplicate or incomplete job listings.
- **Local Hosting:** In its current form, the system is primarily designed for local use. Deployment to production servers requires additional optimization and configuration.



---

## Future Work

- **Integration of More Job Portals:** Expanding beyond Naukri, LinkedIn, and Unstop to include platforms like Indeed, Glassdoor, and company career pages for broader coverage.
- **Automated Notifications:** Enabling email or SMS alerts for users when new jobs matching their preferences are available.
- **AI-Powered Recommendations:** Leveraging machine learning to recommend jobs based on a user's past searches, profile, and application patterns.
- **Advanced Filters:** Allowing users to refine results by salary range, experience level, location, job type (remote, hybrid, onsite), and company size.
- **User Profiles & History:** Building personalized dashboards where users can save preferences, search history, and favorite jobs for quick access.
- **Scalability through Cloud Deployment:** Hosting the application on cloud platforms (AWS, Azure, GCP) with containerization (Docker) for high availability and multi-user scalability.
- **Real-Time Data Pipeline:** Transitioning from Selenium scraping to APIs or headless browser automation to improve speed and reliability.
- **Data Analytics & Insights:** Providing visual analytics like job trends, top hiring companies, and demand for specific roles using charts and dashboards.
- **Mobile Application:** Extending the platform as a Flutter/React Native mobile app for on-the-go job searching.
- **Enhanced Security:** Implementing OAuth for secure authentication and ensuring data protection with encryption.

---

## Error Handling and Logging

- **Error Handling:**
  - Each executor (Naukri, LinkedIn, Unstop) includes `try-except` blocks to gracefully handle exceptions such as element not found, timeouts, or login failures.
  - User inputs are validated on both frontend and backend to prevent invalid queries or empty searches.
  - If a portal fails to respond, the system continues with results from other portals, ensuring partial but usable outputs.
- **Logging:**
  - Logs are maintained to track scraping activities, including successful searches, errors encountered, and execution times.
  - Critical errors are recorded for further analysis, enabling quick resolution when a portal's structure changes.
  - Logging also helps in monitoring user activity and backend performance, which is crucial when scaling the system.

These mechanisms ensure that the platform remains reliable, transparent, and easy to maintain, even when dealing with the dynamic nature of external job portals.

## Interaction Logs

[Chat History](#)

[Chat History 2](#)

[Chat History 3](#)

---

## Conclusion

The AI Job Finder project addresses one of the most pressing challenges for job seekers today: the need to navigate multiple portals, each with different interfaces and scattered opportunities. By combining the power of Flask for backend management, Selenium for concurrent job scraping, and a clean, responsive frontend, the system delivers a unified platform where users can efficiently access opportunities from Naukri, LinkedIn, and Unstop in one place.

This project not only demonstrates technical proficiency in web automation, data handling, and modular system design but also emphasizes usability by providing a seamless experience with clear workflows, real-time feedback, and organized results. While there are current limitations such as reliance on dynamic web scraping and performance constraints, these challenges form the stepping stones for future enhancements.

Looking ahead, the system holds strong potential to evolve into a comprehensive career assistance platform. Features such as AI-driven recommendations, automated notifications, advanced filters, mobile accessibility, and cloud deployment can transform it into a scalable, intelligent, and globally useful solution for job seekers.

## THANK YOU

---