

TECH JOB PORTAL BY SCRAPING INDEED AND GLASSDOOR

Name : Dilli Ganesh B

Date : 13.10.2025

1. ABSTRACT

The “Tech Job Portal by Scraping Indeed and Glassdoor” project is designed to simplify the process of searching for technology-related job opportunities. It uses web scraping to automatically extract and update job listings from trusted sources such as Indeed and Glassdoor. The system focuses exclusively on jobs in the software and IT sectors, providing users with filtered, relevant, and up-to-date listings.

The project is built using **Flask** as the backend framework, **Python** for data processing and scraping, and **MySQL** for storing user and job-related information. The platform includes user registration, login, job search with filters, saved jobs, and notification functionalities. Periodic scraping ensures that users always have access to the latest job postings. This project not only automates the job search process but also delivers a specialized experience for developers and IT professionals seeking employment in the tech industry.

2. INTRODUCTION

In the modern digital era, job portals have become a vital part of recruitment and employment. However, most existing platforms are generalized, catering to every industry, which results in an overwhelming number of listings for users who are specifically interested in technology-related roles. Job seekers often spend a significant amount of time filtering and sorting through irrelevant postings, which reduces efficiency and increases frustration.

The **Tech Job Portal by Scraping Indeed and Glassdoor** addresses this issue by providing a **dedicated job search platform exclusively for the tech domain**. It

leverages web scraping to collect real-time data from two major job sites — **Indeed** and **Glassdoor** — ensuring users have access to accurate and updated job listings.

The main objective of the project is to create an automated, easy-to-use platform where developers can quickly search, save, and apply for jobs that match their skills and preferences. It also offers features like advanced filters and job alerts, helping users stay informed about new opportunities as soon as they are posted.

3. OBJECTIVES

The primary objectives of this project are as follows:

1. To build a **dedicated job portal** focusing solely on technology and IT-related job roles.
2. To implement **automated web scraping** from Indeed and Glassdoor to gather job listings.
3. To allow users to **register, log in, and manage** their job search preferences.
4. To enable **job search with filters** such as title, company, location, and salary.
5. To maintain a **database** of job listings and user data using MySQL.
6. To provide **notifications or alerts** when new jobs matching user preferences are found.
7. To integrate a **scheduler** that performs periodic scraping to keep listings up to date.
8. To deliver a **clean, responsive, and user-friendly interface** for smooth interaction.

4. SYSTEM ANALYSIS

4.1 Problem Definition

Traditional job portals display thousands of listings from multiple sectors, making it difficult for software professionals to find relevant jobs efficiently. Users must apply manual filters repeatedly, and even then, results may

not be updated or accurate. Additionally, the lack of automation means that users miss opportunities that are posted between their manual searches

4.2 Proposed Solution

The proposed system is an intelligent, tech-specific job portal that automates the process of job searching by scraping data from trusted online sources. The platform filters out irrelevant listings and presents only technology-oriented jobs. Users can create accounts, search for jobs, save interesting postings, and receive notifications. Automated scheduling ensures that the job database remains current without manual intervention.

4.3 Advantages of the Proposed System

- **Automation:** The portal automatically fetches job data at regular intervals.
- **Relevance:** Only technology-related jobs are displayed.
- **Efficiency:** Reduces the user's time spent in searching and filtering.
- **User Convenience:** Allows login, saved jobs, and alerts.
- **Data Freshness:** Regular scraping keeps listings up to date.

5. SYSTEM REQUIREMENTS

5.1 Hardware Requirements

- Processor: Intel Core i3 or higher
- RAM: Minimum 4 GB
- Hard Disk: Minimum 500 MB free space
- Display: Standard 15-inch or higher

5.2 Software Requirements

- Programming Language: Python

- Framework: Flask
- Libraries Used: BeautifulSoup, Requests, SQLAlchemy, APScheduler
- Database: MySQL
- Tools: Visual Studio Code, XAMPP, Postman
- Operating System: Windows or Linux

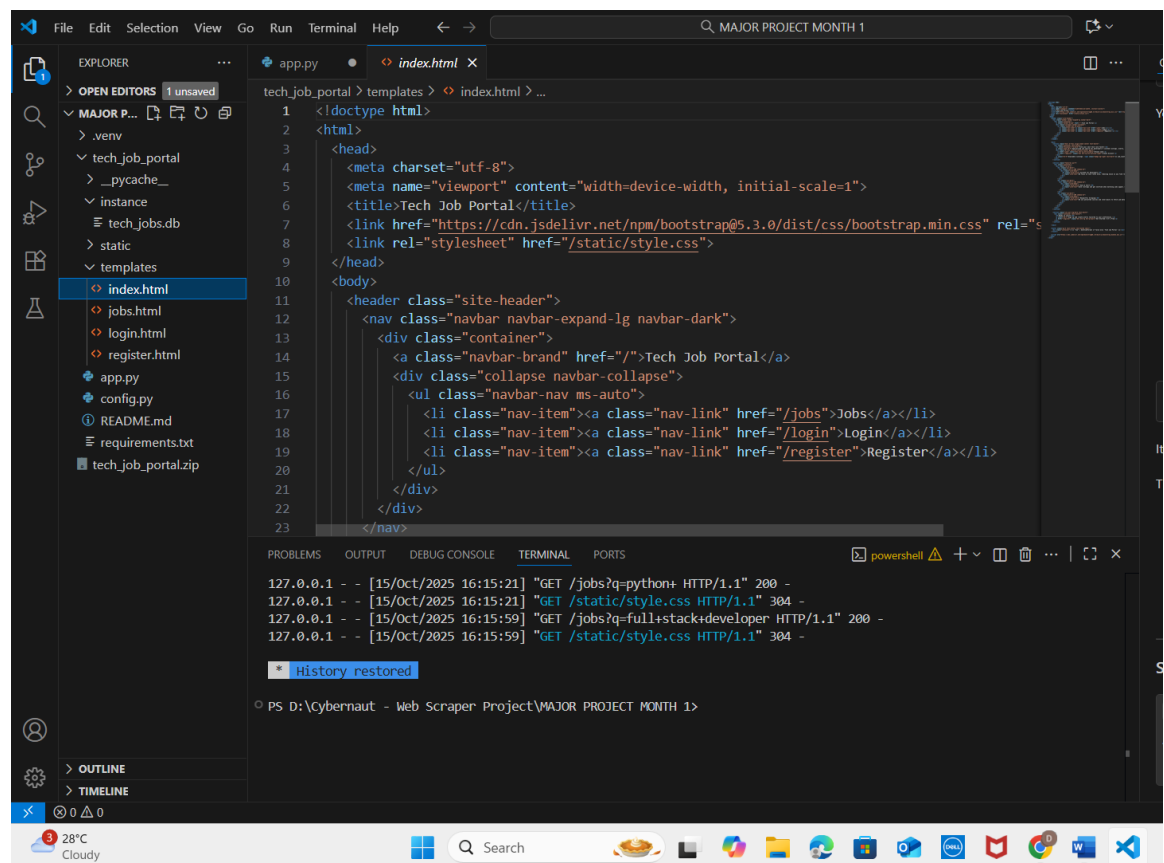
6. SYSTEM DESIGN

The project is structured using a **three-tier architecture**:

1. Frontend Layer:

Handles user interaction using HTML, CSS, and JavaScript. Provides forms for login, registration, and job search.

Index.html



The screenshot shows the Visual Studio Code editor with the `index.html` file open. The file contains the following HTML code:

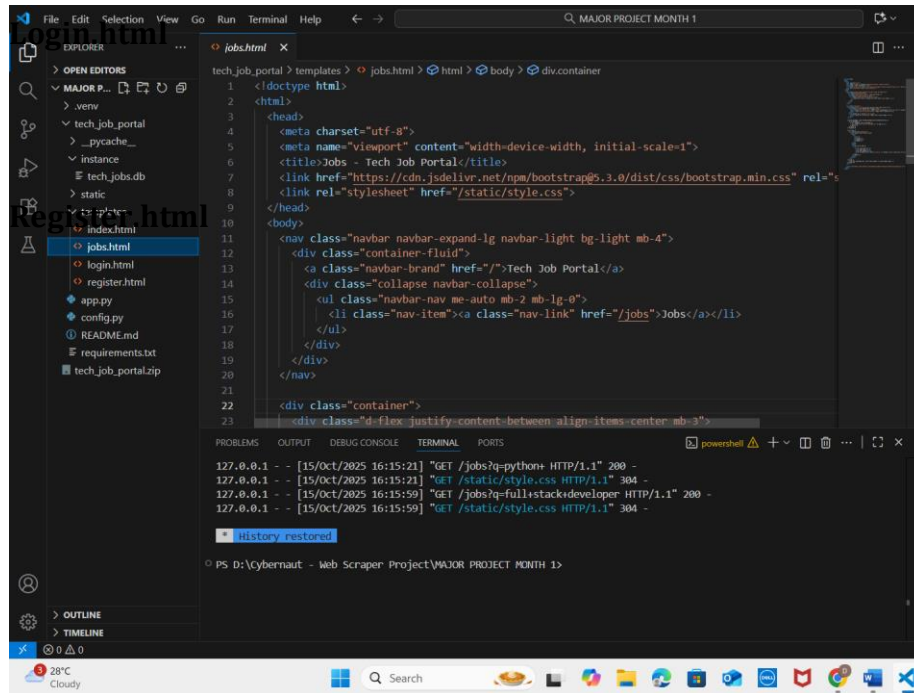
```
1 <!doctype html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1">
6     <title>Tech Job Portal</title>
7     <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">
8     <link rel="stylesheet" href="/static/style.css">
9   </head>
10  <body>
11    <header class="site-header">
12      <nav class="navbar navbar-expand-lg navbar-dark">
13        <div class="container">
14          <a class="navbar-brand" href="/">Tech Job Portal</a>
15          <div class="collapse navbar-collapse">
16            <ul class="navbar-nav ms-auto">
17              <li class="nav-item"><a class="nav-link" href="/jobs">Jobs</a></li>
18              <li class="nav-item"><a class="nav-link" href="/login">Login</a></li>
19              <li class="nav-item"><a class="nav-link" href="/register">Register</a></li>
20            </ul>
21          </div>
22        </div>
23      </nav>
24    </header>
25  </body>
26 </html>
```

The terminal window at the bottom shows the following HTTP requests and responses:

```
127.0.0.1 - - [15/Oct/2025 16:15:21] "GET /jobs?q=python+ HTTP/1.1" 200 -
127.0.0.1 - - [15/Oct/2025 16:15:21] "GET /static/style.css HTTP/1.1" 304 -
127.0.0.1 - - [15/Oct/2025 16:15:59] "GET /jobs?q=full+stack+developer HTTP/1.1" 200 -
127.0.0.1 - - [15/Oct/2025 16:15:59] "GET /static/style.css HTTP/1.1" 304 -
```

A message "History restored" is also visible in the terminal.

Jobs.html



The screenshot shows a Visual Studio Code editor with the file `jobs.html` open. The file content is as follows:

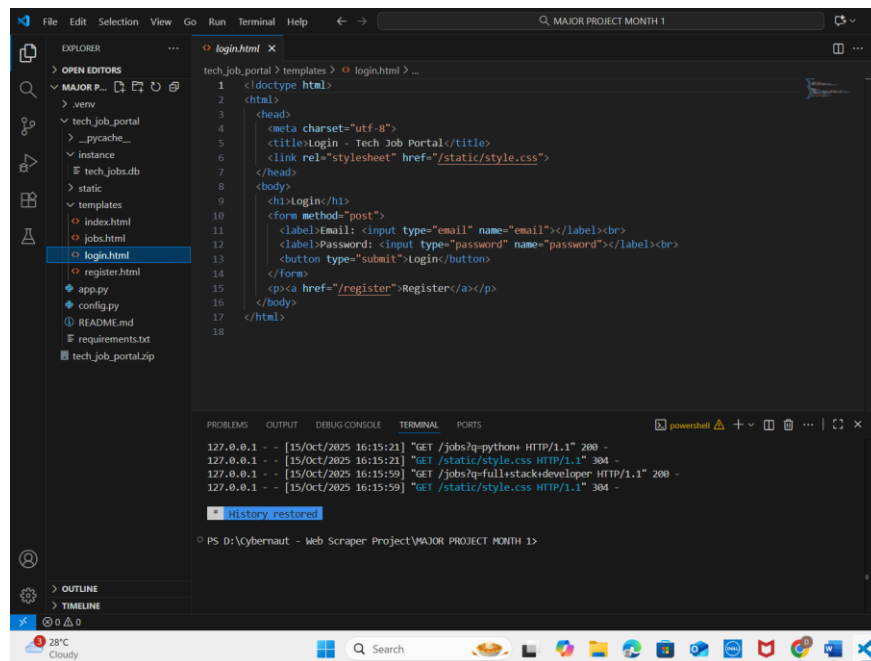
```
1 <!doctype html>
2 <html>
3
4 <head>
5   <meta charset="utf-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1">
7   <title>Jobs - Tech Job Portal</title>
8   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">
9 </head>
10 <body>
11   <nav class="navbar navbar-expand-lg navbar-light bg-light mb-4">
12     <div class="container-fluid">
13       <a class="navbar-brand" href="/">Tech Job Portal</a>
14       <div class="collapse navbar-collapse">
15         <ul class="navbar-nav me-auto mb-2 mb-lg-0">
16           <li class="nav-item"><a class="nav-link" href="/jobs">Jobs</a></li>
17         </ul>
18       </div>
19     </div>
20   </nav>
21
22   <div class="container">
23     <div class="d-flex justify-content-between align-items-center mb-3">
```

The terminal at the bottom shows the following output:

```
127.0.0.1 - - [15/Oct/2025 16:15:21] "GET /jobs?q=python+ HTTP/1.1" 200 -
127.0.0.1 - - [15/Oct/2025 16:15:21] "GET /static/style.css HTTP/1.1" 304 -
127.0.0.1 - - [15/Oct/2025 16:15:59] "GET /jobs?q=fullstackdeveloper HTTP/1.1" 200 -
127.0.0.1 - - [15/Oct/2025 16:15:59] "GET /static/style.css HTTP/1.1" 304 -
```

A message "History restored" is visible in the terminal. The status bar at the bottom indicates the file path: `PS D:\Cybernaut - Web Scraper Project\MAJOR PROJECT MONTH 1>`.

Login.html



The screenshot shows a Visual Studio Code editor with the file `login.html` open. The file content is as follows:

```
1 <!doctype html>
2 <html>
3
4 <head>
5   <meta charset="utf-8">
6   <title>Login - Tech Job Portal</title>
7   <link rel="stylesheet" href="/static/style.css">
8 </head>
9 <body>
10   <div>Login</div>
11   <form method="post">
12     <label>Email: <input type="email" name="email"></label><br>
13     <label>Password: <input type="password" name="password"></label><br>
14     <button type="submit">Login</button>
15   </form>
16   <p><a href="/register">Register</a></p>
17 </body>
18 </html>
```

The terminal at the bottom shows the following output:

```
127.0.0.1 - - [15/Oct/2025 16:15:21] "GET /jobs?q=python+ HTTP/1.1" 200 -
127.0.0.1 - - [15/Oct/2025 16:15:21] "GET /static/style.css HTTP/1.1" 304 -
127.0.0.1 - - [15/Oct/2025 16:15:59] "GET /jobs?q=fullstackdeveloper HTTP/1.1" 200 -
127.0.0.1 - - [15/Oct/2025 16:15:59] "GET /static/style.css HTTP/1.1" 304 -
```

A message "History restored" is visible in the terminal. The status bar at the bottom indicates the file path: `PS D:\Cybernaut - Web Scraper Project\MAJOR PROJECT MONTH 1>`.

2. Backend Layer (Flask):

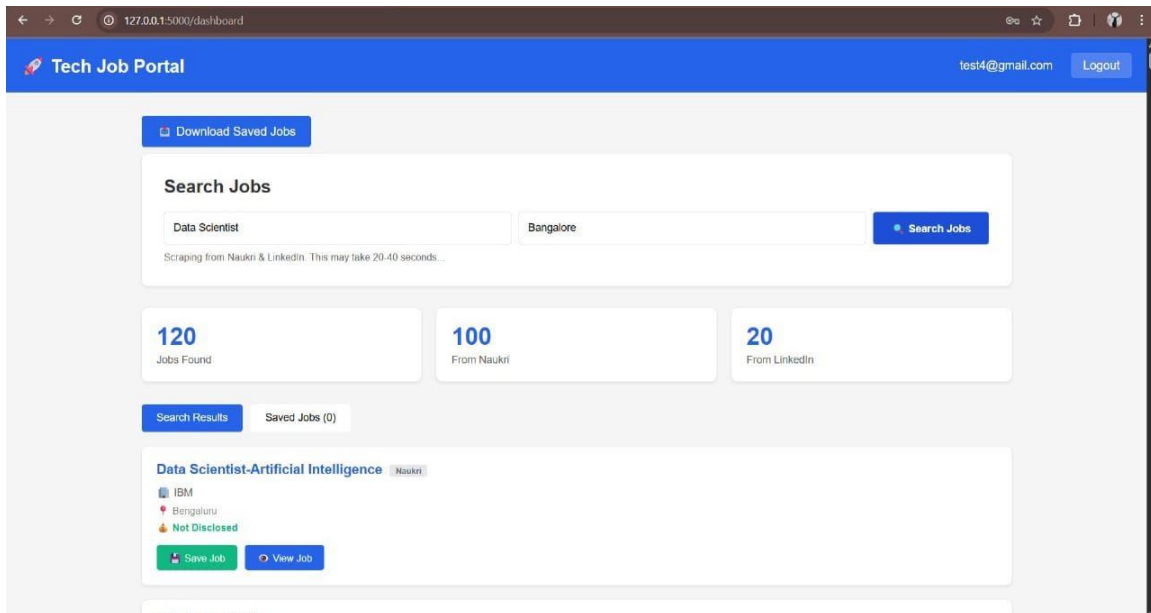
Manages routes, server-side logic, and data processing. Handles web scraping requests, job search operations, and communication with the database.

3. Database Layer (MySQL):

Stores user information, job details, and saved job data. Ensures structured data management and easy retrieval.

Flow Description:

When a user visits the portal, they can register or log in. The backend fetches job listings from the database, which are populated by the scraper. The user can apply filters, view job details, or save them. The scheduler periodically runs in the background to fetch new job data automatically.



7. MODULE DESCRIPTION

7.1 User Module

- Handles user registration, login, and authentication.
- Stores user credentials securely in the database.
- Allows users to manage saved jobs and preferences.

7.2 Job Scraper Module

- Uses BeautifulSoup and Requests to scrape data from Indeed and Glassdoor.
- Extracts relevant information such as job title, company, salary, and description.
- Cleans and structures the data before storing it in the database.

7.3 Database Module

- Responsible for storing job data and user information.
- Provides easy access and retrieval for job searches and saved jobs.

7.4 Scheduler Module

- Periodically runs scraping scripts using APScheduler.
- Ensures that the database remains updated with the latest job listings.

7.5 Notification Module

- Alerts users about new job listings via email or in-app notifications.
- Uses email APIs like SendGrid or Flask-Mail for message delivery.

7.6 Search and Filter Module

- Enables users to search by job title, company name, salary range, or location.
- Displays only relevant results, improving user satisfaction and usability.

8. IMPLEMENTATION

The project was implemented in several stages:

1. **Environment Setup:**

Python, Flask, and MySQL were installed and configured. The project structure was organized with separate folders for templates, static files, and scripts.

2. **Web Scraping:**

Using BeautifulSoup, the scraper was programmed to extract job data directly from Indeed and Glassdoor's public listings. The data included job title, company name, salary, and job URL.

3. **Database Integration:**

SQLAlchemy ORM was used to connect Flask to MySQL. Separate tables were designed for storing user accounts, job listings, and saved jobs.

4. **User Interface:**

The frontend pages were built using HTML and CSS, providing a simple and intuitive layout. Job listings were displayed in a card-like format, making them easy to browse.

5. **Automation:**

The APScheduler library was integrated to schedule scraping tasks multiple times a day, ensuring that the listings remained fresh and relevant.

6. **Notifications:**

A basic notification system was created to inform users about new listings that matched their criteria.

9. TESTING

Comprehensive testing was conducted to ensure that all modules worked together seamlessly:

- **Unit Testing:** Checked individual functions such as login, scraping, and filtering.
- **Integration Testing:** Verified interactions between Flask routes, scraper, and database.

- **Functional Testing:** Confirmed that the main functionalities (search, save, alert) performed as expected.
- **User Testing:** Feedback from trial users helped refine the interface and usability.

The results confirmed that the application is stable, reliable, and performs as intended.

10. RESULTS AND DISCUSSION

The final system provides a fully functional web portal capable of:

- Automatically scraping job listings from multiple platforms.
- Displaying only tech-related job opportunities.
- Allowing user registration, login, and preference saving.
- Sending timely alerts for new job postings.

The system performed efficiently during testing, successfully automating the job search process while maintaining accuracy and relevance.

11. CONCLUSION

The **Tech Job Portal by Scraping Indeed and Glassdoor** project successfully achieves its objective of delivering a domain-specific, user-friendly, and automated job search platform. By focusing on technology-related roles and integrating automation, it simplifies the job-hunting experience for developers and IT professionals.

The system's modular architecture allows future expansion, and its automated scraping mechanism ensures data freshness without manual updates. It stands as a practical and scalable solution for modern recruitment needs in the technology sector.

12. FUTURE ENHANCEMENTS

To improve and expand the platform further, the following enhancements can be implemented:

1. **AI-Powered Job Recommendations** based on user skills and past searches.
2. **Resume Upload and Parsing** for better profile-job matching.
3. **Mobile Application** for Android/iOS users.
4. **Integration with LinkedIn and GitHub APIs** for extended networking.
5. **Chatbot Assistance** to guide users in real time.
6. **Admin Dashboard** for managing users, jobs, and analytics.

13. REFERENCES

1. <https://www.indeed.com>
2. <https://www.glassdoor.com>
3. <https://flask.palletsprojects.com>
4. <https://docs.python.org>
5. <https://beautiful-soup-4.readthedocs.io>