Core Functions:

**a. User Registration and Profiles:**

Allow users to create accounts as either job seekers or employers.

Collect basic user information like name, email, and password.

Create and edit user profiles, including personal information, job history, skills, and

company

details.

**b. Job Listings:**

Allow employers to post job listings.

Include job details such as title, company, location, job description, and application

instructions.

Enable job seekers to browse and search for job listings.

**c. Job Search and Filters:**

Implement a search bar to help job seekers find relevant job listings.

Provide basic search filters, such as location, job type (full-time, part-time, contract), and

industry.

**d. Application Submission:**

Allow job seekers to apply for jobs directly through the platform.

Include the option to upload and manage resumes and cover letters.

**e. User Dashboard:**

Create a user dashboard for both job seekers and employers.

Allow users to manage their profiles, job listings, and job applications.

**f. Notifications:**

Send email or push notifications to inform users about important updates, such as new job

listings or application responses.

Web Architecture :

We will employ a client-server architecture.

The diagram illustrates the end-to-end flow of data in the job board MVP. The web client communicates with the web server through a RESTful API. The web server interacts with a database to store and retrieve data. Here's an overview:

1. Front-End (Web Client):
   * User Interface (UI): Built using HTML, CSS, and JavaScript.
   * Front-end Framework, Flask and Java Script: Provides a responsive and interactive user interface.
   * User Registration and Login Components.
   * User Profile Management Components.
   * Job Listing Pages.
   * Job Search and Filtering Components.
   * Application Submission Forms.
   * User Dashboard Components.
   * Notifications System (may use WebSocket or other technologies for real-time updates).
2. Back-End (Web Server):
   * Web Application Framework, Flask:
     + Handles HTTP requests and routing.
     + Implements middleware for authentication and validation.
   * User Authentication and Authorization:
     + Manages user accounts and sessions.
     + Utilizes tokens for user sessions.
   * RESTful API Endpoints:
     + User Management: Registration, Login, Profile Editing.
     + Job Listings: Create, Read, Update, Delete (CRUD) operations.
     + Job Search and Filtering.
     + Application Submission.
     + User Dashboard Data Retrieval.
     + Notifications Handling.
   * Data Validation and Sanitization: Protects against security vulnerabilities.
   * Integration with Third-Party Services: Email notifications and push notifications
   * Security Measures: Protection against common web application vulnerabilities.
3. Database:
   * Database System, MySQL:
     + Stores user profiles, job listings, applications, and other data.
   * Database Schema:
     + Tables for users, job listings, job history, profile info, notifications and applications, structured according to data model shown in the data model section.
     + Indexing for efficient data retrieval.
4. APIs and Communication:
   * RESTful APIs for communication between the web client and server.
   * Secure Communication: HTTPS to protect data in transit.
5. Scalability and Performance:
   * Horizontal Scalability: Allows room for deployment across multiple servers or cloud-based scaling.
   * Optimization: Database query optimization, caching, and use of CDNs for static assets.
6. Security:
   * Security Best Practices: Protection against SQL injection, XSS, CSRF, and other vulnerabilities.

APIs and Methods

For communication between the web client and the web server, we need the following API routes:

* User Registration and Profiles:
  + POST /api/users/register: Allows users to create accounts as either job seekers or employers
  + POST /api/users/login: Handles user login.
  + POST /api/users/logout: Handles user logout.
  + GET /api/users/:id/profile: Retrieves user profiles.
  + PUT /api/users/:id/profile: Enables users to create and edit their profiles, including personal information, job history, skills, and company details..
* Job Listings:
  + POST /api/job-listings: Allows employers to post job listings.
  + GET /api/job-listings/:id: Allows job seekers to view job details for a specific job listing..
  + GET /api/job-listings: Provides a list of job listings.
  + PUT /api/job-listings/:id: Allows employers to edit job listings.
  + DELETE /api/job-listings/:id: Allows employers to delete job listings.
* Job Search and Filters:
  + GET /api/job-listings/search: Allows job seekers to search for job listings based on location and job type
  + GET /api/job-listings/filters: Provides filter options (e.g., location, job type).
* Application Submission:
  + POST /api/applications: Allows job seekers to apply for jobs directly through the platform and upload resumes and cover letters.
  + GET /api/applications/:id: Retrieves application details.
  + GET /api/applications/job/:jobId: Retrieves applications for a specific job listing.
* User Dashboard:
  + GET /api/users/:id/dashboard: Retrieves data for user dashboards (profile, job listings, job applications).
* Notifications:
  + GET /api/notifications: Allows users to receive email or push notifications about important updates, such as new job listings or application responses.
  + Implement email and push notification systems to notify users about relevant updates (e.g., new job listings, application responses).
* The following API endpoints or function/methods will be created to allow other clients to use:
  + createUser(): Creates a new user account.
  + loginUser(): Logs in a user.
  + logoutUser(): Logs out a user.
  + createProfile(): Creates a new user profile.
  + updateProfile(): Updates an existing user profile.
  + createJobListing(): Creates a new job listing.
  + getJobListing(): Retrieves job details for a specific job listing.
  + searchJobListings(): Searches for job listings based on location, job type, and industry.
  + applyForJob(): Allows job seekers to apply for jobs directly through the platform and upload resumes and cover letters.
  + getDashboard(): Retrieves user dashboard data.
  + sendNotification(): Sends email or push notifications to users.

Data Models:

The above diagram illustrates the data model for the job board MVP. The database consists of six tables: users, profiles, job\_listings,job\_history,notifications, and applications.

* The users table stores user account information.
* The profiles table stores user profile information.
* The job\_listings table stores job listing information.
* The applications table stores job application information.
* The job\_history table stores job histories information for a user.
* The notifications table stores notification information.
* users

| Column Name | Data Type | Description |
| --- | --- | --- |
| user\_id | integer | Unique identifier for each user |
| email | string | User’s email address |
| password | string | User’s password |
| account\_type | string | User’s account type (job seeker or employer) |

* job\_history

| Column Name | Data Type | Description |
| --- | --- | --- |
| job\_history\_id | integer | Unique identifier for each job history entry |
| user\_id | integer | Foreign key referencing the user who owns the job history |
| job\_title | string | Job title |
| company\_name | string | Company name |
| start\_date | date | Start date of the job |
| end\_date | date | End date of the job |

* job\_listings

| Column Name | Data Type | Description |
| --- | --- | --- |
| job\_id | integer | Unique identifier for each job listing |
| employer\_id | integer | Foreign key referencing the employer who posted the job |
| job\_title | string | Job title |
| company\_name | string | Company name |
| location | string | Job location |
| job\_description | string | Job description |
| application\_instructions | string | Application instructions |

* applications

| Column Name | Data Type | Description |
| --- | --- | --- |
| application\_id | integer | Unique identifier for each job application |
| job\_id | integer | Foreign key referencing the job listing to which the application is submitted |
| user\_id | integer | Foreign key referencing the job seeker who submitted the application |
| resume | string | File path to the job seeker’s resume |
| cover\_letter | string | File path to the job seeker’s cover letter |

* profiles

| Column Name | Data Type | Description |
| --- | --- | --- |
| profile\_id | integer | Unique identifier for each profile |
| user\_id | integer | Foreign key referencing the user who owns the profile |
| first\_name | string | User’s first name |
| last\_name | string | User’s last name |
| job\_history | string | User’s job history |
| skills | string | User’s skills |
| company\_details | string | User’s company details |
| Personal \_info | string | User’s personal information |

* **notifications:**

notification\_id (PK)

user\_id (FK)

message

timestamp

status (read/unread)

User stories:

Here are some detailed user stories that the job board MVP will satisfy:

1. As a job seeker, I want to be able to create a profile that includes my personal information, job history, skills, and company details so that employers can find me easily.
2. As an employer, I want to be able to post job listings that include job details such as title, company, location, job description, and application instructions so that job seekers can apply for the job.
3. As a job seeker, I want to be able to search for job listings based on location, job type, and industry so that I can find relevant job openings.
4. As a job seeker, I want to be able to apply for jobs directly through the platform and upload resumes and cover letters so that I can easily apply for jobs.
5. As a user, I want to be able to manage my profiles, job listings, and job applications through a user dashboard so that I can easily keep track of my activity on the platform.
6. As a user, I want to be able to receive email or push notifications about important updates, such as new job listings or application responses so that I can stay up-to-date with the latest information.

**User Registration:**

As a new user, I can register an account by providing my name, email, and password.

As a job seeker, I can provide additional personal information during registration.

J**ob Posting:**

As an employer, I can post a job listing with details like title, company, location, description, and application instructions.

**Job Search and Filters:**

As a job seeker, I can search for job listings using keywords and apply filters like location, job type, and industry.

**Application Submission:**

As a job seeker, I can apply for a job by uploading my resume and cover letter through the platform.

**User Dashboard:**

As a user, I can log in and access my dashboard to manage my profile, view my job listings, and track my job applications.