#### Description of Website and How It Works

**1. Overview**

This document describes the purpose, structure, and functionality of the job management website, along with detailed instructions on how it works.

**2. Purpose**

The website is designed to facilitate job posting and management. It allows job seekers to find jobs and enables service providers to post, update, and delete job listings. The website ensures secure user authentication and authorization using JWT.

**3. Structure**

The website's backend is structured using the Model-View-Controller (MVC) pattern, which separates the application logic into distinct parts:

* **Models**: Represent the data structure. In this case, jobs and workers data are stored in JSON files.
* **Controllers**: Handle the request logic and interact with the models.
  + **Jobs Controller**: Manages all job-related operations (CRUD).
  + **Workers Controller**: Manages worker-related operations (view details).
* **Routes**: Define the endpoints and map them to controller functions.

**4. Functionality**

**4.1. Authentication**

* **Login**: Users can log in using their username and password. On successful login, a JWT token is provided.
  + **Endpoint**: POST /login
  + **Request Body**: {"username": "1", "password": "1"}
  + **Response**: {"token": "your\_jwt\_token"}

**4.2. Job Management**

* **Get All Jobs**
  + **Endpoint**: GET /jobs
  + **Headers**: Authorization: Bearer <your\_token>
* **Get Job by ID**
  + **Endpoint**: GET /jobs/:id
  + **Headers**: Authorization: Bearer <your\_token>
* **Add Job**
  + **Endpoint**: POST /jobs
  + **Headers**: Authorization: Bearer <your\_token>
  + **Request Body**: {"title": "New Job", "description": "Job Description", "worker-id": 1}
* **Update Job**
  + **Endpoint**: PUT /jobs/:id
  + **Headers**: Authorization: Bearer <your\_token>
  + **Request Body**: {"title": "Updated Job", "description": "Updated Description", "worker-id": 1}
* **Delete Job**
  + **Endpoint**: DELETE /jobs/:id
  + **Headers**: Authorization: Bearer <your\_token>
* **Get Jobs by Worker ID**
  + **Endpoint**: GET /jobs/workers/:id
  + **Headers**: Authorization: Bearer <your\_token>

**4.3. Worker Management**

* **Get All Workers**
  + **Endpoint**: GET /workers
  + **Headers**: Authorization: Bearer <your\_token>
* **Get Worker by ID**
  + **Endpoint**: GET /workers/:id
  + **Headers**: Authorization: Bearer <your\_token>

**5. How It Works**

1. **User Authentication**: Users log in to obtain a JWT token.
2. **Authorization**: All subsequent requests must include the JWT token in the Authorization header.
3. **Job and Worker Management**: Users can manage jobs and view worker details using the provided endpoints.

**6. Conclusion**

The website leverages the MVC pattern to separate concerns, making the application more modular and maintainable. The use of controllers for job and worker management ensures a clear separation of logic, enhancing the application's scalability and ease of maintenance.