Portfolio backend template

Introduction:

A template with the minimal functions to make a portfolio; The template comes with 4 automated CRUD basic functions which makes those functions reusable among the portfolio system/template and modules that makes the functionalities controllable over the whole portfolio system.

Automation and modularity are used in this project to enhance testability, code structure, flexibility, readability and maintainability with taking the performance and speed into consideration too.

Schema:

Starting from the schema we have one SQL database with 9 tables which are:

1. Project table
   schema:
   ```
   CREATE TABLE IF NOT EXISTS Project(
     id INT PRIMARY KEY AUTO_INCREMENT,
     name VARCHAR(255) NOT NULL,
     describtion TEXT NOT NULL,
     stack_used TEXT,
     github_link VARCHAR(360)
   );
   ```
   Name -> Project name
   Description -> describe what is the project and what did you do in it
   Stack used -> skills and concepts you used in this project
   GitHub link -> attach the GitHub link for your project if any or if you want to elaborate in other application

2. Skill:

   Schema:

   ```
   CREATE TABLE IF NOT EXISTS Skill(
      id INT PRIMARY KEY AUTO_INCREMENT,
      name VARCHAR(100) NOT NULL,
      level ENUM('Beginner', 'intermediate', 'above intermediate',
   'advanced', 'expert') NOT NULL,
      type ENUM('Technical', 'Concept') NOT NULL,
      is_learning BOOLEAN NOT NULL,
      added_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
      learnt_date DATE
   );
   ```

   Name -> skill name

   Level -> One of the values inside the Enum

   Type -> pick whether it's technical/practical skill or a concept

   Is_learning -> true or false with no default value

   Added_at -> it has a default value to know when was it added to the portfolio

   Learnt_date -> the date you earned that skill or got to that level

3. Article

   Schema:

   ```
   CREATE TABLE IF NOT EXISTS Article(
      id INT AUTO_INCREMENT PRIMARY KEY,
      title VARCHAR(255) NOT NULL,
      body TEXT NOT NULL,
      link VARCHAR(350),
      description TEXT,
      still_writing BOOLEAN NOT NULL,
      hidden BOOLEAN DEFAULT True,
      created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
   ```

updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
);
Title -> article title
Body -> article body
Link -> If you have uploaded it on any website/platform share it here
Description -> Briefly what is that article about
Still_writing -> set that to true if you are still working on it
Hidden -> this can help you if you want to publish it or not yet
Created_at -> added automatically
Updated_at -> added automatically

4. Contact
   Schema:
   CREATE TABLE IF NOT EXISTS Contact(
   id INT AUTO_INCREMENT PRIMARY KEY,
   name VARCHAR(100) NOT NULL,
   info VARCHAR(350),
   photo_url VARCHAR(255)
   );

   Name -> add the name of the contact way

   Info -> how can they contact with you

   Photo_url -> a URL to load the photo from

5. Resume
   Schema:
   CREATE TABLE IF NOT EXISTS Resume(
   id INT AUTO_INCREMENT PRIMARY KEY,
   file_url VARCHAR(255),
   uploaded_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
   );

File_url -> a URL to the resume you uploaded to make it downloadable if wanted

Uploaded_at -> added automatically

6. Experience
   Schema:

```
CREATE TABLE IF NOT EXISTS Experience(
   id INT PRIMARY KEY AUTO_INCREMENT,
   type ENUM('Education', 'Work') NOT NULL,
   name TEXT NOT NULL,
   duration INT NOT NULL,
   link TEXT,
   duties TEXT
);
```

   Type -> Educational experience or work experience
   Name -> could be company name, institute name or even the employer/system name if it was a contract job
   Duration -> preferred to be in days then computed into other forms
   Link -> A reference or website link if found
   Duties -> what did you do there
7. OpenTo:
   Schema:

```
CREATE TABLE IF NOT EXISTS OpenTo(
   id INT PRIMARY KEY AUTO_INCREMENT,
   employmentType ENUM('Part-time', 'Full-time', 'Contract', 'Project-based') NOT NULL,
   flexible_hours BOOLEAN NOT NULL,
   duties TEXT,
   position VARCHAR(255) NOT NULL,
   locationType ENUM('Remote', 'On-site', 'Hybrid') NOT NULL
);
```

   employmentType -> one of the values in the ENUM
   flexible_hours -> true or false

duties -> what do you expect to do in that position
position -> the position name
locationType -> one of the values in the ENUM

CRUD functions:

They are handled in the databaseHandlers.js and the connection pool is handled through the conncetion.js by the mysql2/promise createPool() and the parameters are handled inside the .env (to define the parameters create and .env file and define the host, user, password and database).

Inside the databaseHandlers.js you can find the 4 basic CRUD functions that are reused among the whole project insertIntoDatabase, deleteFromDatabase, getFromDatabase, updateDatabase.

The module also comes with 2 filters for the skills one by level (check them in the database schema or the list in the module for validation) and the other by the type (technical or concepts).

The get function does not send the ID for security of the system.

Service.js:

Inside the pageLoaders folder you can find the APIs of the services each one has the delete, load, update and insert operations except for the skill has an extra function which is filterByExperience and validators when needed besides the code structure and the usage of the mysql2 placeholders that lowers the SQL injections risk.

Server.js:

It has the API end points as it uses helmet for security and the router for easier and more automated and dynamic routing ways each

service has it’s endpoints structured in a way that makes it easier for a next dev to customize.

Security:

This template was meant to be basic so no authentication, logs, rate limiters will be found but there is basic validation and type checking.