## TECHNOLOGIES NEEDED

### Languages And Frameworks

* HTML: Used for structuring the web pages.
* CSS: For styling the web pages to ensure they are visually appealing.
* JavaScript: For client-side scripting to handle interactive elements like modal pop-ups.
* Node.js: A JavaScript runtime built on Chrome's V8 JavaScript engine, necessary for running JavaScript on the server.
* Express.js: A fast, unopinionated, minimalist web framework for Node.js, used to handle server-side logic.

### Database

* MySQL: Used to store user data and other necessary information.

### Development Tools

* IDE: Integrated Development Environment (IDE) like Visual Studio Code, Sublime Text, or Atom , which ever environment you are used to for coding purposes.
* Postman: Useful for testing server endpoints during development.
* Git: Version control system to manage changes to the project files.
* Node Package Manager (npm): Used for managing Node.js packages.

### Libraries and Middleware:

* dotenv: For loading environment variables from a .env file into process.env to manage sensitive configuration settings securely.
* bcrypt: For hashing and securing passwords.

## INSTALLATION STEPS

1. **Install Node.js and npm**

Node.js is bundled with npm (Node Package Manager), which is used for managing dependencies.

* Visit [Node.js official website](https://nodejs.org/en) and download the installer for your operating system.
* Follow the installation instructions. To verify that Node.js and npm were installed successfully, open your command prompt or terminal and run:
* node -v
* npm -v
* These commands should print the version numbers of Node.js and npm, respectively.

1. **Install Git**

* Windows
* Download the installer from [Git's official site](https://git-scm.com/).
* Run the downloaded installer and follow the instructions, making sure to include Git in your PATH.

git --version

* To verify installation, run:
* This command should print the installed version of Git.
* Mac
* Git might already be installed. You can check by running ‘**git –version**’ in the terminal.
* If it's not installed, you can install it via [Homebrew](https://brew.sh/) (if you have Homebrew installed) by running:

brew install git

* Linux
* Use your package manager to install Git. For Ubuntu, for example:
* sudo apt update
* sudo apt install git
* Verify the installation with ‘**git –version**’.

1. **Clone the Repository**

Once Git is installed, you can clone the project repository.

* Open your command prompt or terminal.
* Navigate to the directory where you want to store the project.
* Execute the clone command:

git clone https://github.com/mviswan34/Env\_Awareness\_Website.git

1. Install Project Dependencies

Navigate to the project directory and install the required npm packages.

* Open your command prompt or terminal.
* Change directory to your project’s root directory.

cd path\to\your\project

* Install all dependencies defined in ‘package.json’:

npm install …

1. Set Up Environment Variables

* Open .env in a text editor and fill in the necessary details like API Secret key and database credentials.

1. Install External Libraries:

Install dotenv which is essential since it manages environment variables securely.

* Open your command prompt or terminal.
* Navigate to your project's Home Page directory.
* Install dotenv using npm. This command will download dotenv :

npm install dotenv

1. Install SQL and make sure you are connected to the database.
2. **Download the MySQL Installer**

* Visit the MySQL official website: [MySQL Downloads](https://dev.mysql.com/downloads/)
* Select the MySQL Community Server edition, which is free.
* Choose the installer version suitable for your operating system (Windows, macOS, Linux).

1. **Run the Installer**

* For Windows: Run the downloaded installer, which is typically an .msi file. Follow the installation wizard and choose the "Developer Default" to install both the server and MySQL Workbench.
* For macOS: Open the downloaded .dmg file and follow the instructions to install MySQL. The installer will also install MySQL Workbench.
* For Linux: Depending on your distribution, you can use the package manager to install MySQL. For example, on Ubuntu, you can use:

sudo apt update

sudo apt install mysql-server

* Complete the installation process and note the username and passwords set during this process; you will need them to connect to the database by adding to .env file as mentioned before.

1. **Secure MySQL Installation**

* Run the mysql\_secure\_installation utility. This script will help you improve the security of your MySQL installation:

sudo mysql\_secure\_installation

* Follow the prompts to set a password for the MySQL root user, disable remote root access, remove anonymous user accounts, and remove the test database.

1. **Start the MySQL Service**

* For Windows, the MySQL server typically starts automatically after installation.
* For macOS and Linux, you might need to start the server manually:

sudo systemctl start mysql

1. **Verify MySQL Installation**

* Verify that MySQL is running by connecting to the database server using the command line:

mysql -u root -p

* Enter the password when prompted. If you connect successfully, you are now in the MySQL command line interface.

1. **Setting Up the Database:**

* Log into MySQL Workbench. Connect to the local server.

CREATE TABLE `user\_registration` (

`email\_id` VARCHAR(255) NOT NULL,

`password` VARCHAR(255) NOT NULL,

`zip\_code` VARCHAR(10),

PRIMARY KEY (`email\_id`)

);

* Create a table by running the command:
* Ensure database access credentials are securely stored in the dotenv file.

1. **Generate Secret Key:**

* Secret keys are used for cryptographic functions such as generating signatures or encrypting data. They ensure that only those with the correct key can access or validate secured data.
* You can generate a secret key directly in Node.js:

require('crypto').randomBytes(32, (err, buffer) => {

const secret = buffer.toString('base64');

console.log(secret);

});

* This script will output a random 256-bit key, encoded in base64. Use the .env file to store your secret keys locally.

## RUN THE APPLICATION

Start the application server using Node.js.

* Ensure you are still in the project’s home page directory in your command prompt or terminal.
* Also ensure you change the path to the absolute path names to where the files (images and such) are located in your index and css files of both home page and checklist page.
* Execute:

node server.js

* You can now double click on index.html of home page, and it should open up the website.

## MODIFICATIONS NEEDED IN CURRENT CODE

* The bee image in checklist is a clickable item. One should connect this to correct display page.
* The submit buttons in all the checklist pages are to be connected to backend. Either a new table could be created for checklist and the records could be maintained there or additional columns can be added to ‘user\_registration’.

## NEW DEVELOPMENT

* About Page, Details Page, User Tracking should all be developed fresh. Modifications can either be made in Home Page index and css files or each page can have a separate html and css files like we did in checklist, and you can simply connect them to the home page’s html file.

## HOSTING AND SECURITY GUIDE

**Hosting the Website**

1. **Select a Hosting Provider**:
   * **Shared Hosting**: Suitable for smaller websites with moderate traffic. Providers like Bluehost, HostGator, or SiteGround offer easy-to-use interfaces and one-click installations.
   * **Virtual Private Servers (VPS)**: Good for medium-sized businesses with more traffic. VPS provides dedicated server resources and better customization options. Examples include DigitalOcean, Linode, and AWS EC2.
   * **Dedicated Servers**: Best for large businesses or high-traffic websites needing full control over server resources.
   * **Cloud Hosting**: Flexible and scalable hosting solutions such as AWS, Google Cloud, or Microsoft Azure. These platforms allow the website to easily scale resources up or down based on traffic.
2. **Domain Registration**:
   * Register a domain name through registrars like GoDaddy, Namecheap, or Google Domains.
   * Ensure the domain name is easy to remember and reflects the website’s purpose.
3. **Deployment**:
   * Deploy the website using FTP/SFTP if on a traditional host, or use integrated deployment tools provided by cloud services.
   * For Node.js applications, consider using platforms like Heroku or Netlify that simplify the deployment process.

**Adopting Security Features (A Cyber Security Expert is needed)**

1. **SSL/TLS Certificate**:
   * Essential for encrypting data transmitted between the user’s browser and your server. It also boosts SEO rankings and user trust.
   * Obtain and install an SSL certificate. Many hosts offer free SSL certificates through Let's Encrypt.
2. **Secure Coding Practices**:
   * Avoid common security flaws like SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF).
   * Validate and sanitize all user inputs (Like we did for passwords).
   * Use prepared statements for database queries.
3. **Data Protection**:
   * Implement strong password policies and store passwords securely using hashing algorithms (e.g., bcrypt).
   * Ensure sensitive data is encrypted both at rest and in transit.
4. **Regular Updates and Maintenance**:
   * Keep all server software and dependencies up to date to protect against vulnerabilities.
   * Regularly update your application to fix bugs and apply security patches.
5. **Monitoring and Logging**:
   * Set up monitoring tools to track performance and detect unusual activities that could indicate security threats.
   * Implement logging to keep records of significant actions and enable auditing.
6. **Backup Strategy**:
   * Regularly back up website data and ensure backups are stored securely.
   * Test recovery procedures to confirm that data restoration works correctly.
7. **User Access Management**:
   * Implement role-based access control to restrict who can access sensitive areas of the website.
   * Use two-factor authentication (2FA) for administrative accounts.

**Additional Security Measures**

1. **Content Delivery Network (CDN)**:
   * Use a CDN like Cloudflare or Akamai to enhance website performance and security. CDNs can help mitigate DDoS attacks and reduce load times.
2. **Firewall Configuration**:
   * Configure web application firewalls (WAF) to filter, monitor, and block harmful traffic trying to exploit vulnerabilities.
3. **Compliance and Legal Requirements**:
   * Ensure compliance with regulations relevant to your client’s location and industry, such as GDPR, HIPAA, or PCI-DSS.

contact details for developers:

megha Viswanath – meghaviswanath1995@gmail.com