Sultan Qaboos University College of Science
Department of Computer Science
COMP4701: WEB APPLICATION DEVELOPMENT- FALL 2024
(section 10)

Volunteering Website

Project part 1

Group members:

Al-Mardas Saif Al Busaidi 133705 (Team Leader)
Bashar Ahmed Almuqaimi 137760
Hilal Saif AlManji 137735

Part A - PROJECT SPECIFICATION:

1) Project Information:

Volunteering is becoming more viral nowadays in Oman in different ways, and publicize these volunteering opportunities to people is a challenge in itself. We see the need to make an online platform that gathers these volunteering opportunities and delivers them to the public in an organized way. This platform definitely will help a lot of organizations, Charities, and even individuals to find people to volunteer. Also, volunteers will gain experience and even a certificate that will help them in their careers by participating in such opportunities.

In this project we use:
React for Front-End and sending API request and ASP.Net for Backend

2)Project Requirements:

Functional Requirements:

- User Login and Registration
- User dashboard
- Volunteering organizations dashboard
- List of available Volunteering applications.
- Writing complaints

Non-Functional Requirements:

Privacy : User information and the organization of the volunteering application should be save and not accessible by the other users .

Reliability : The web works as intended with no fault ,and can handle the errors .

Security: Data Encryption for users data .
Authentication & Authorization for the user and organization. To make sure everything works smoothly .

Usability : Ease of Use and Error handling

Maintainability: Clear and concise documentation should be provided to developers and users.

3)Project Design:

Login and Register Page:

- **Purpose**: Allow users to create an account or log in.
- **Workflow**:
    - Users fill out a registration form, which validates inputs such as email, password, and other user details.
    - After successful validation, the information is sent to the back-end, where it is processed and stored in the database(Note: database is not created yet, we store the information in a list).
    - Once registered, users can log in, and a session token (JWT) is issued to keep them authenticated across the site.
- **Front-End Interaction**: React form components handle input fields and validation. Upon submission, the form data is sent to the back-end API.
- **Back-End Interaction**: The ASP.NET API validates and saves user data to the `Users` table in the database(Note: database is not created yet, we store the information in a list).

Show Volunteering Posts:

- **Purpose**: Display a list of volunteering opportunities to users.
- **Workflow**:
    - The front-end sends a request to the back-end to retrieve available posts.
    - The back-end queries the database(Note: database is not created yet, we store the information in a list), and returns a structured list of volunteering opportunities.
    - The front-end displays these posts, allowing users to click on a post to view more details.
    - When a post is clicked, the user is directed to a dynamic page showing specific information about the opportunity, such as location, time, and requirements, along with a registration form.
- **Front-End Interaction**: A list of posts is rendered using React components. Each post has a button or link that opens the details page.
- **Back-End Interaction**: ASP.NET queries the `Posts` table and returns the data to the front-end.

Add Volunteering Posts Page:

- **Purpose**: Allow organizations, charities, or individuals to post new volunteering opportunities.
- **Workflow**:
    - Users (with appropriate permissions) fill out a form to create a post, including fields for title, description, date, location, and any additional information.
    - The form data is validated on the front-end to ensure required fields are filled out and follow the correct format.
    - Once submitted, the data is sent to the back-end, which further validates it before saving it to the database(Note: database is not created yet, we store the information in a list).
- **Front-End Interaction**: A form component is used for data entry and validation.
- **Back-End Interaction**: ASP.NET saves the post data to the `Posts` table.

Admin Dashboard Page:

- **Purpose**: Allow the admin to manage all registered volunteers for each opportunity.
- **Workflow**:

    - The admin logs in and accesses a dashboard showing all posted volunteering opportunities.
- **Front-End Interaction**: The dashboard page renders data from the back-end using components that show lists of opportunities and registered users.
- **Back-End Interaction**: ASP.NET retrieves opportunities that posted by the admin according to the email stored in the sessions from the `Posts` table for display and management by the admin.

User Dashboard Page:

- **Purpose**: Allow users to view their profile and the volunteering opportunities they're currently registered for.
- **Workflow**:
- After logging in, the user can view their account details and a list of their active volunteering engagements.
- **Front-End Interaction**: A user dashboard and opportunity list are rendered using React components.
- **Back-End Interaction**: ASP.NET retrieves the user's data and related opportunities from the `Users` and `posts` tables.

About us Page:

Dynamic Content Loading:

- The About Us page uses Razor syntax to dynamically render content requested from the back-end. When a user navigates to this page, the front-end sends an HTTP request to an API endpoint on the server.
- The back-end processes this request and fetches the content for the About Us page.
- The fetched data is then returned as a JSON response, which the Razor page processes to display the content.

Add Complaints page:

Dynamic Page Loading:

- When a user navigates to the Complaints page, the front-end sends a request to an API endpoint to retrieve the page content and form structure.
- The back-end serves this request by providing the Razor page, ensuring the latest version of the complaint form and any relevant instructions or messages are displayed.

Information Flow Between Layers

1. **User Registration/Login:**
    - Front-End: User submits the registration/login form.
    - Back-End: Receives data, validates it, and interacts with the database. Issues a JWT token on successful login.
    - Database: Stores user data in the `Users` table.

      (Note: database is not created yet, we store the information in a list)

2. **Viewing Volunteering Posts:**
    - Front-End: Sends a request to fetch all posts.
    - Back-End: Retrieves posts from the `Posts` table.
    - Database: Supplies the necessary data for each volunteering opportunity.

      (Note: database is not created yet, we store the information in a list)

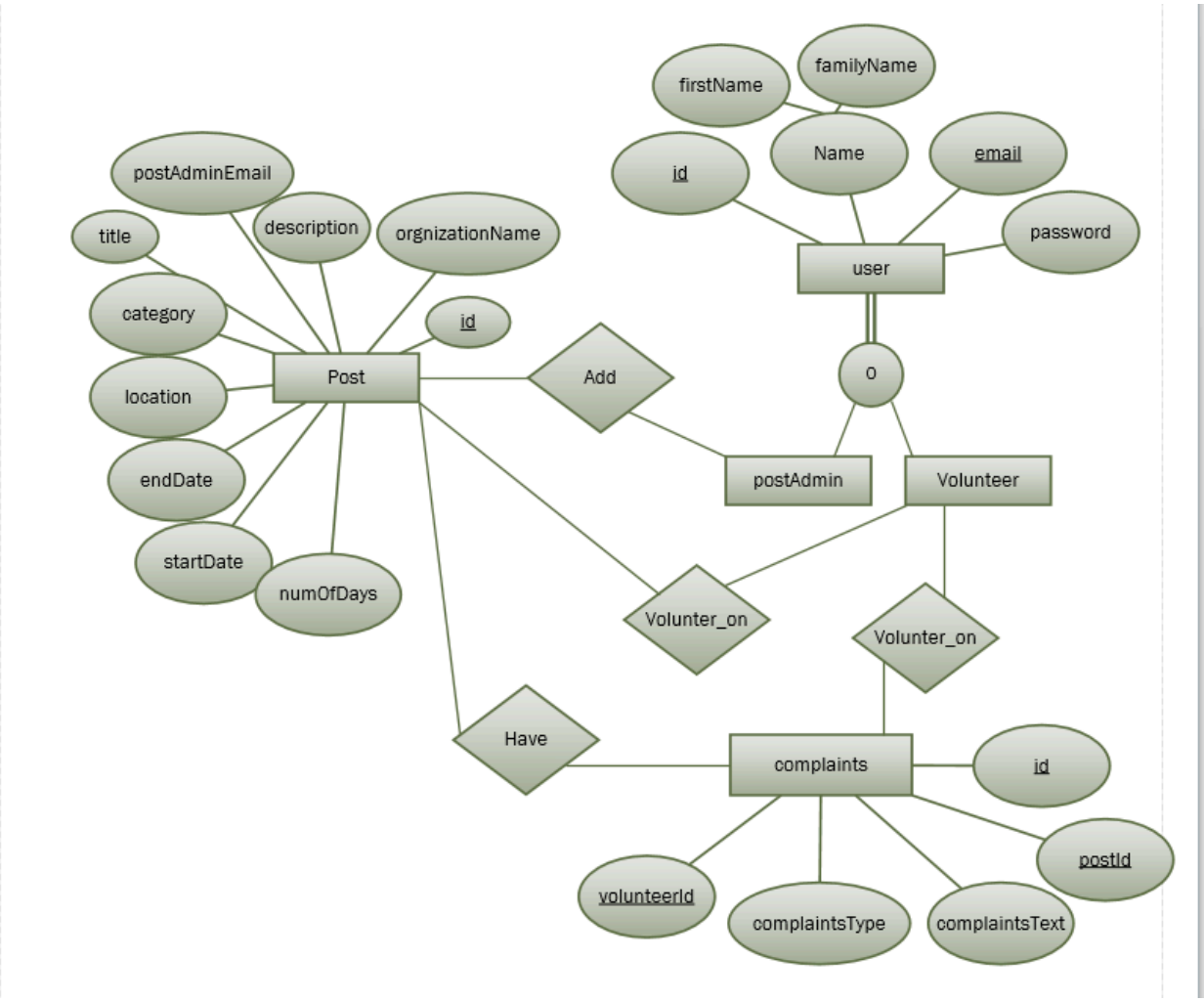
3. **Adding a Volunteering Post:**
    - Front-End: User submits the add-post form.
    - Back-End: Validates form data and saves it to the database.
    - Database: Adds a new entry in the `Posts` table.

      (Note: database is not created yet, we store the information in a list)

4. **Admin/User Dashboard Access:**
    - Front-End: Admin/user accesses their dashboard.
    - Back-End: Fetches specific data based on user roles and permissions.
    - Database: Supplies the required data for dashboard views, such as volunteer registrations.

      (Note: database is not created yet, we store the information in a list)

4)Database design:



Entities and Attributes:

1. Post

- Attributes:
  - `id` (integer): Primary key, uniquely identifies each post.
  - `postAdminEmail` (string): The email of the admin who created the post.
  - `title` (string): The title of the volunteering opportunity.

        - `category` (PostCategory): Represents the category of the post, defined by an enumeration (`PostCategory`).
        - `description` (string): A detailed description of the volunteering opportunity.
        - `numOfDays` (int): The duration of the opportunity in days.
        - `location` (string): The location where the volunteering opportunity will take place.
        - `orgName` (string): The name of the organization hosting the opportunity.
        - `startDate` (string): The start date of the volunteering opportunity.
        - `endDate` (string): The end date of the volunteering opportunity.
    - **Relationships:**
        - A Post can have multiple Complaints associated with it, representing any issues raised by users about that specific volunteering opportunity.

2. User

- Attributes:
    - `id` (integer): Primary key, uniquely identifies each user.
    - `firstName` (string): The first name of the user.
    - `familyName` (string): The family (last) name of the user.
    - `email` (string): The email address of the user, used for login and as a unique identifier.
    - `password` (string): The password for the user's account, stored securely.
- Relationships:
    - A User can create Posts if they are an admin, allowing them to manage and share volunteering opportunities.
    - Users can submit Complaints related to specific Posts they engage with.

3. Complaints

- Attributes:
    - `PostId` (integer): Foreign key linking to the Post entity, indicating which post the complaint is about.
    - `ComplaintType` (string): The type of complaint (e.g., "Content Issue," "Location Discrepancy," "Scheduling Problem").
    - `ComplaintText` (string): The actual content of the complaint, detailing the user's concerns or issues with the post.
- **Relationships:**
    - A Complaint is linked to a single Post through the `PostId` foreign key, representing the specific volunteering opportunity being complained about.

PART B - IMPLEMENTATION:

1)Enumeration

```
public enum PostCategory {
        Education, // Volunteering opportunities related to educational activities and tutoring
        Environment, // Opportunities focused on environmental conservation and sustainability
        Health, // Volunteering in healthcare, wellness, and health education
        Community, // General community service activities and events
        AnimalWelfare // Volunteering for animal care and welfare initiatives
}
```

2)Web Pages

We have created 8 web pages using react and asp.net:

- Home page
- About us page
- Dashboard page
- log in page
- Register page
- Posts page
- Add-post page
- Complaint page

all the pages are explained in the video and at the last of this report.

3)Business Logic

Flow of Information Between Layers

- **Front-end**: Users interact with the web app through forms on Razor pages. For instance, when a user submits a post or complaint, it triggers an HTTP request to the backend.

- **API Layer**: Razor pages or JavaScript calls send data to the API endpoints.
- **Business Logic Layer**: The API calls corresponding methods in the service classes, where data is validated and processed.
- **Response**: The API then responds back to the front-end, updating the user interface based on the result (e.g., showing success or error messages).

Pages and Functionality

- **Login/Register Page**: Allows users to register and log in.
- **Add Post Page**: Enables organizations to create posts. Data is processed and saved.
- **View Posts Page**: Retrieves all posts, filtered by category if needed, and displays them.
- **Complaints Page**: Lets users submit complaints about specific posts.

4)Data Sharing

In our volunteering web application, data sharing across different components and pages is crucial for providing a smooth user experience. Since you're using sessions and cookies to store the user's JWT (JSON Web Token), we can securely manage user authentication and share data as users navigate through the application. we implement data sharing using various mechanisms like sessions and cookies.

Part C - COLLABORATION AND SUBMISSION:

COLLABORATION AND SUBMISSION

Student Name	work	persentage
Hilal Saif AlManji	Index page, ,Report(Project Design ,Project Information) , Posts page	33%
Bashar Ahmed Almuqaimi	Report(Project Requirements),about us page,log in page ,Register page ,Add-post page.	34%
Al-Mardas Saif Al-Budaidi	Admin and user Dashboard, Report(Database design , video),Complaint page	33%

Note: When the team members finish their tasks they send the code to the Team Leader to upload into GitHub.

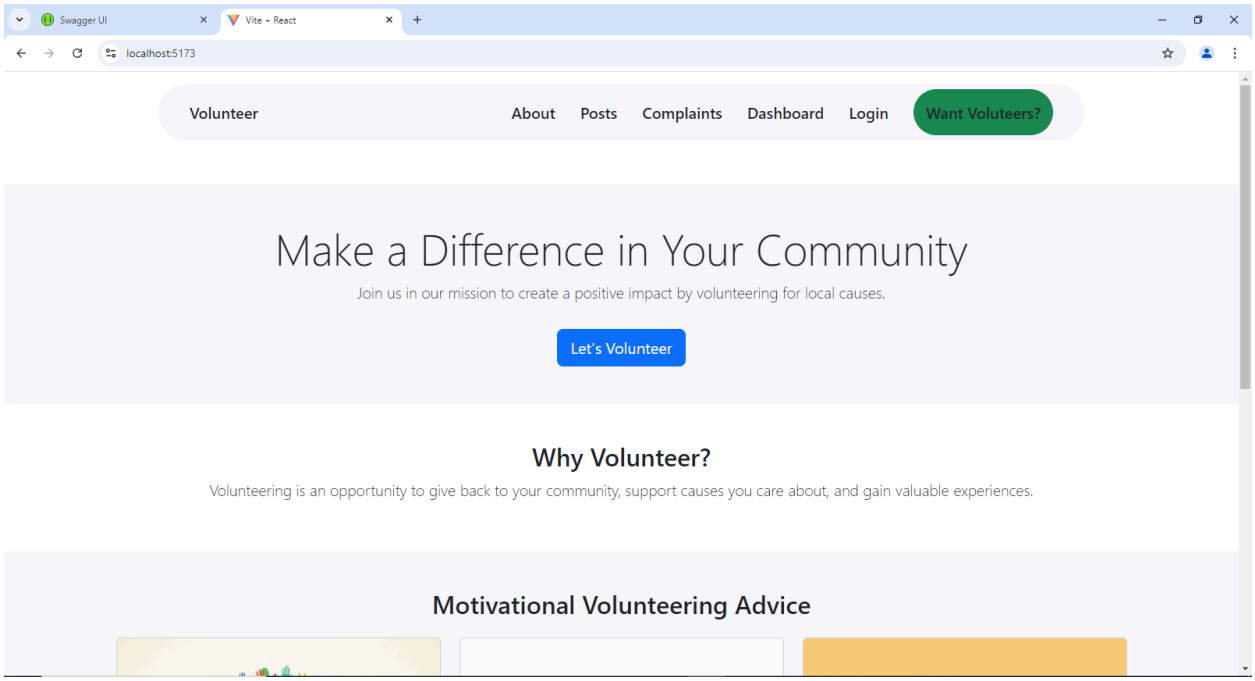
GitHub Submission

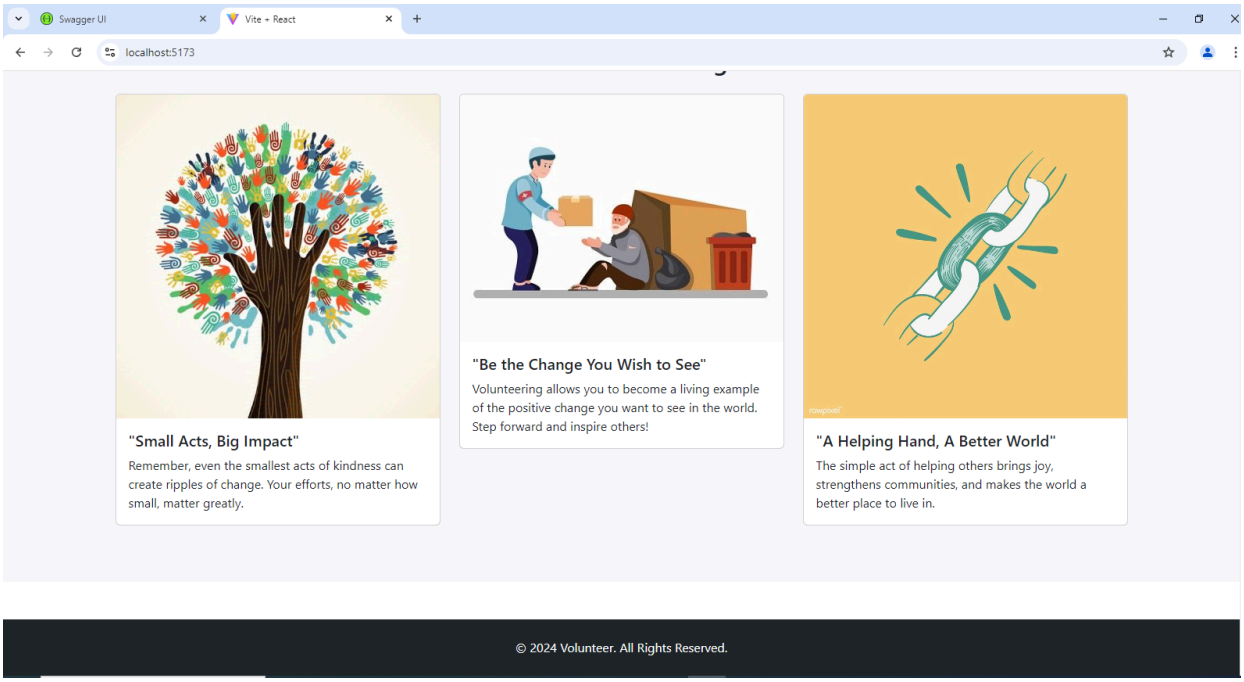
https://github.com/dark0crystal/volunteer-web.git

Discussing the website pages

1-Home Page :

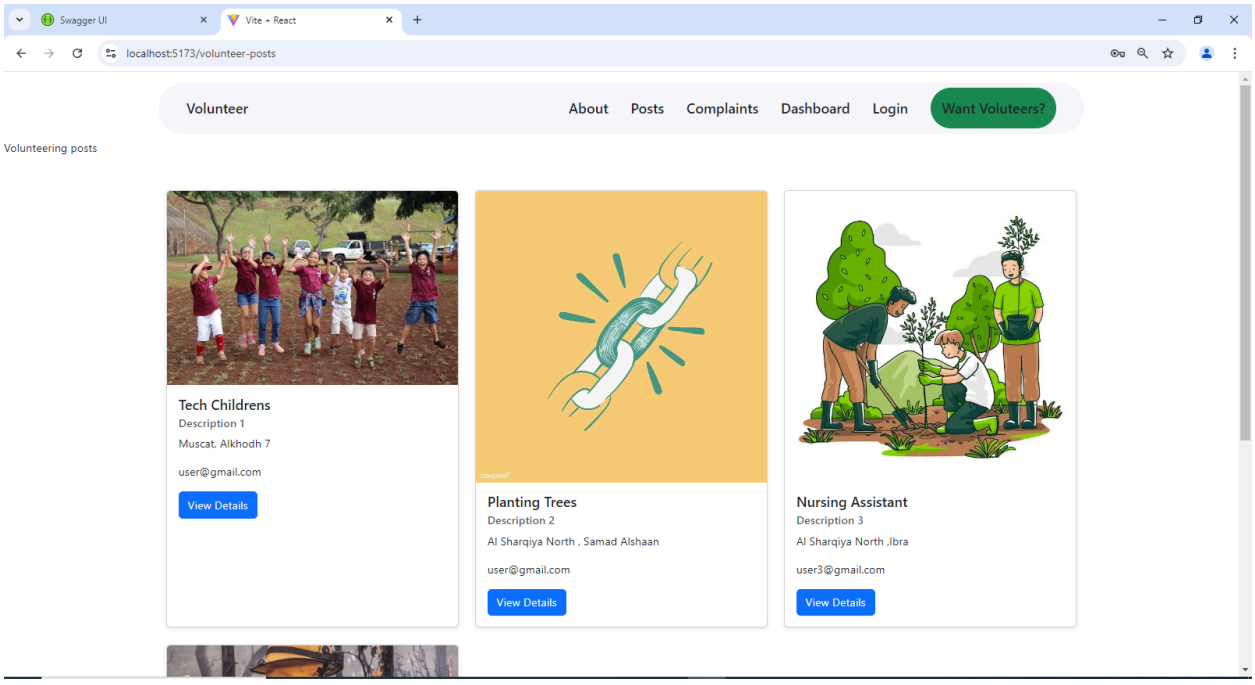
The home page introduce the idea of the website to the user whether he/she is a volunteer or an organization that wants volunteers. It contains advice and the benefits of volunteering. And a button to take the user to the posts page to see the posts.

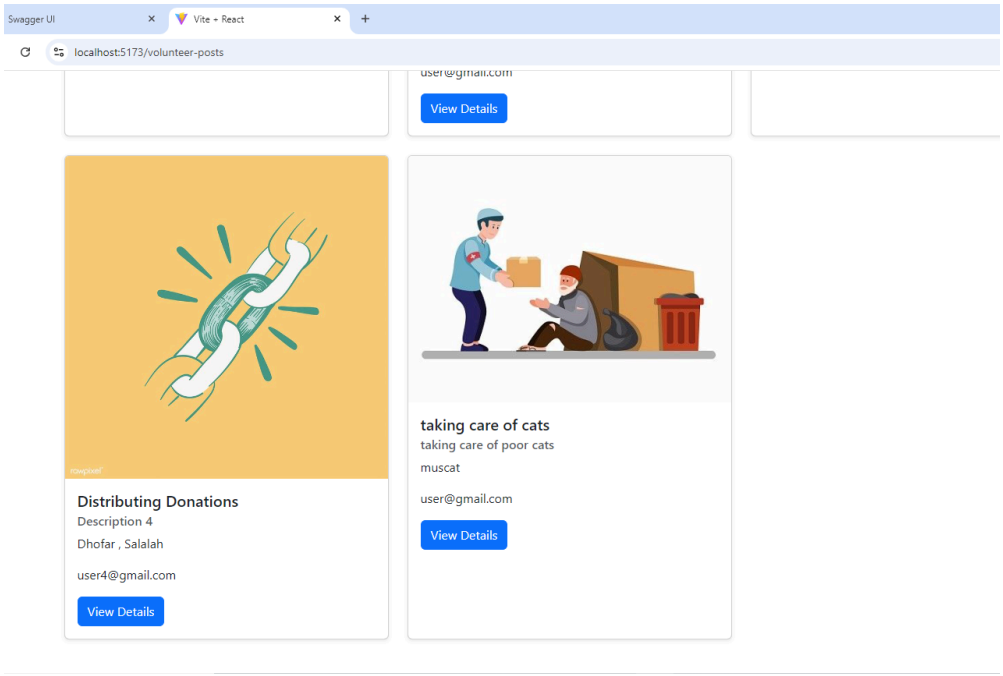




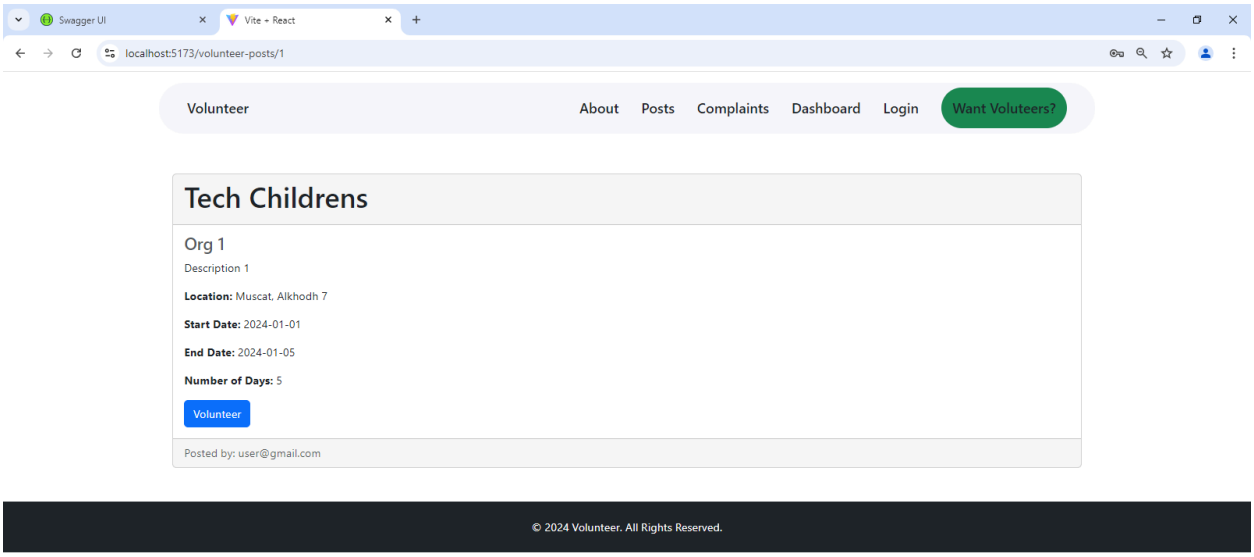
2-Volunteering Posts Page:

On the volunteering posts page the user can see the volunteering opportunities, if the user wants to volunteer or see more details about one of the opportunities he/she just can click on the "View Details" button.

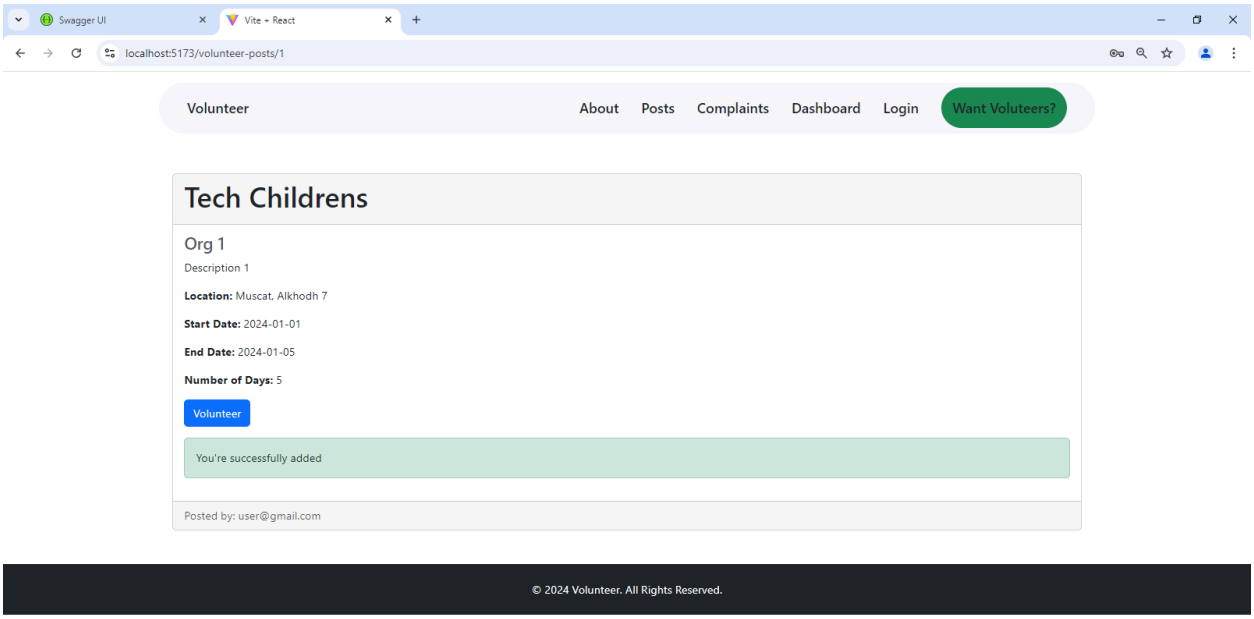




After clicking on the view details button the user can see the details of the volunteering opportunities, also the user can volunteer for the opportunity.

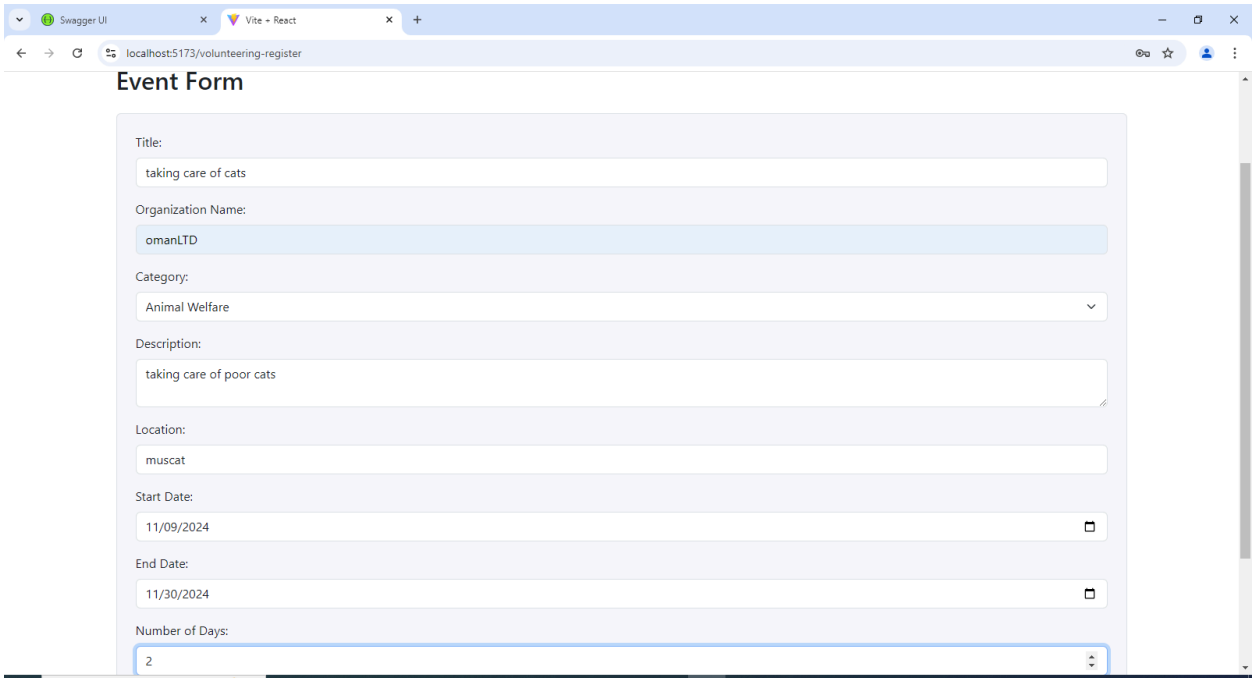


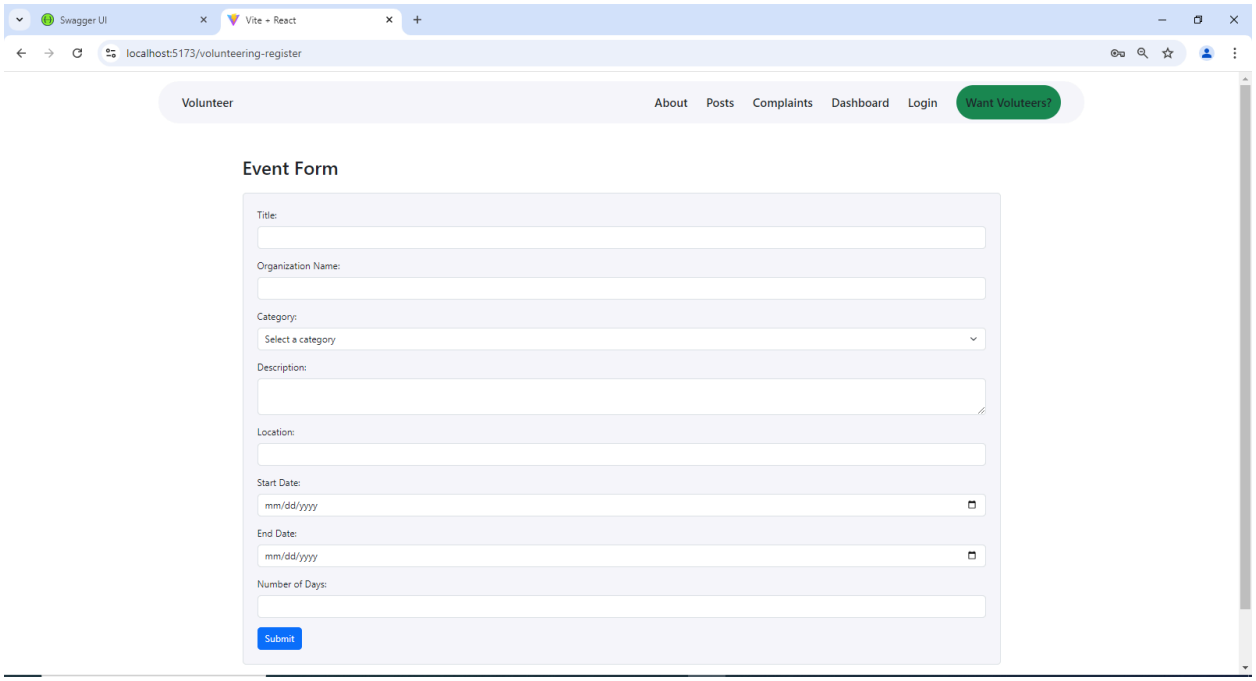
After volunteering in the opportunity, a successful message will appear.



3-Volunteering register page:

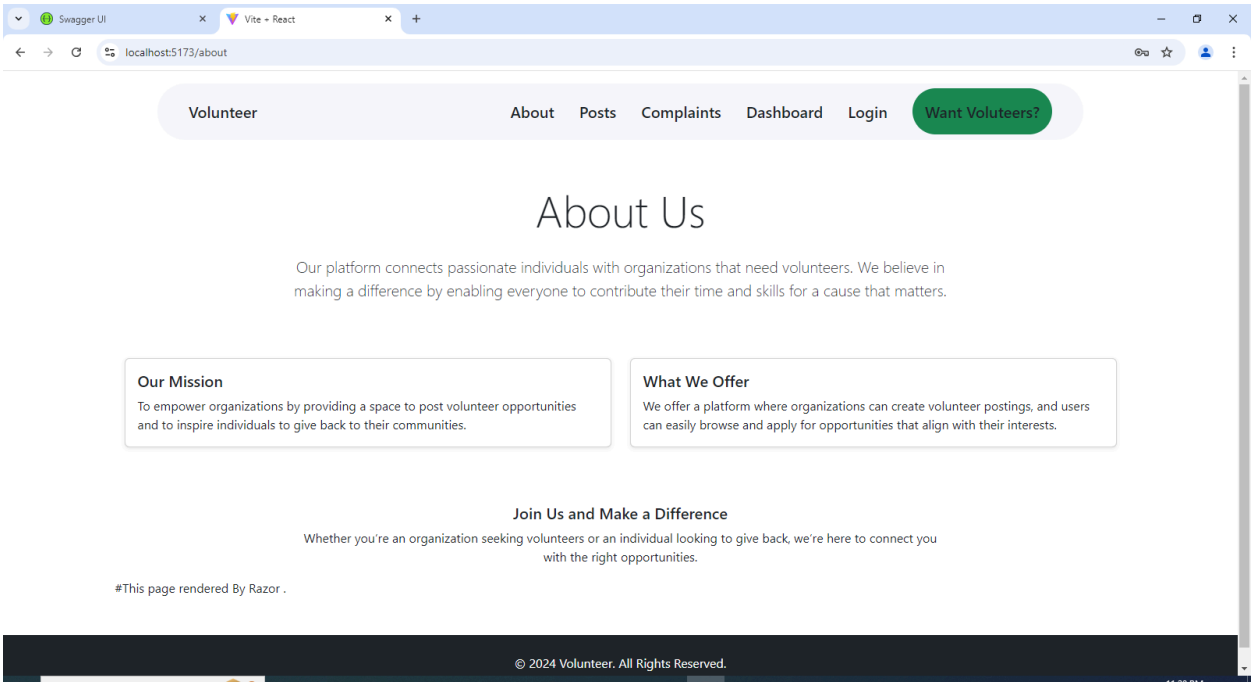
It contains a form that allows the user who is in need of volunteers to add a volunteering opportunity, it will ask the user to add details about the opportunity. When submitting the form it will be saved in the List(because we didn't create DB yet). Then the regular user can see the opportunities in the posts page.





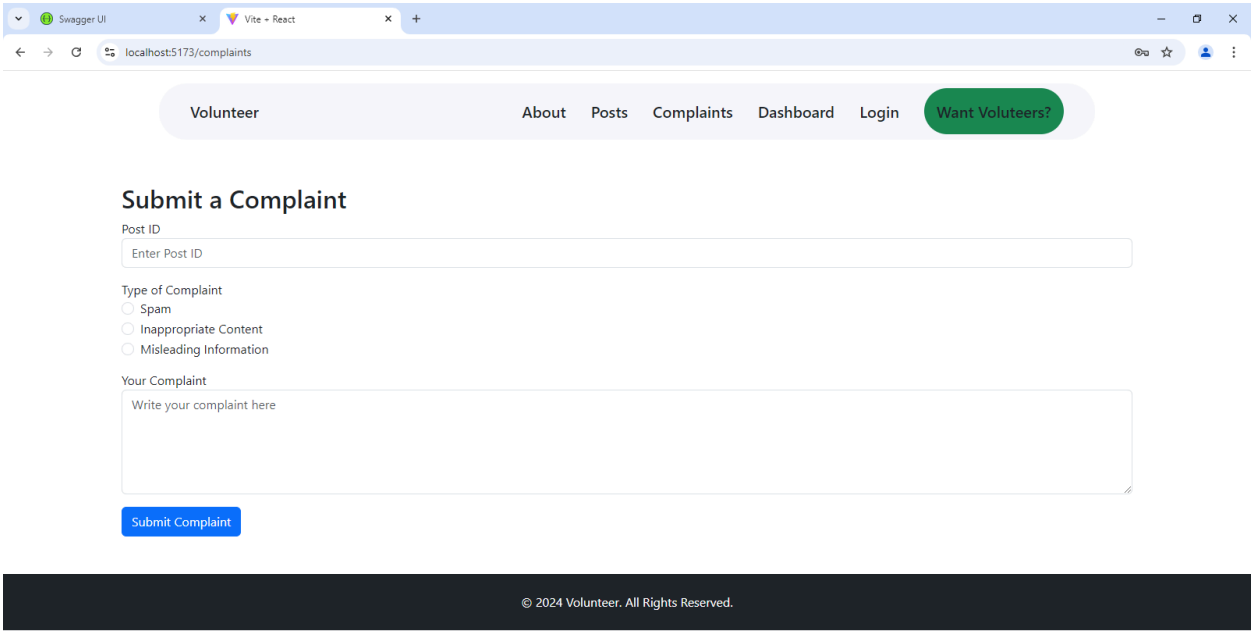
About us page:

The about us page is created using razor page , then using the controllers we create an api to get the about us page in the frontend.



Complaints page:

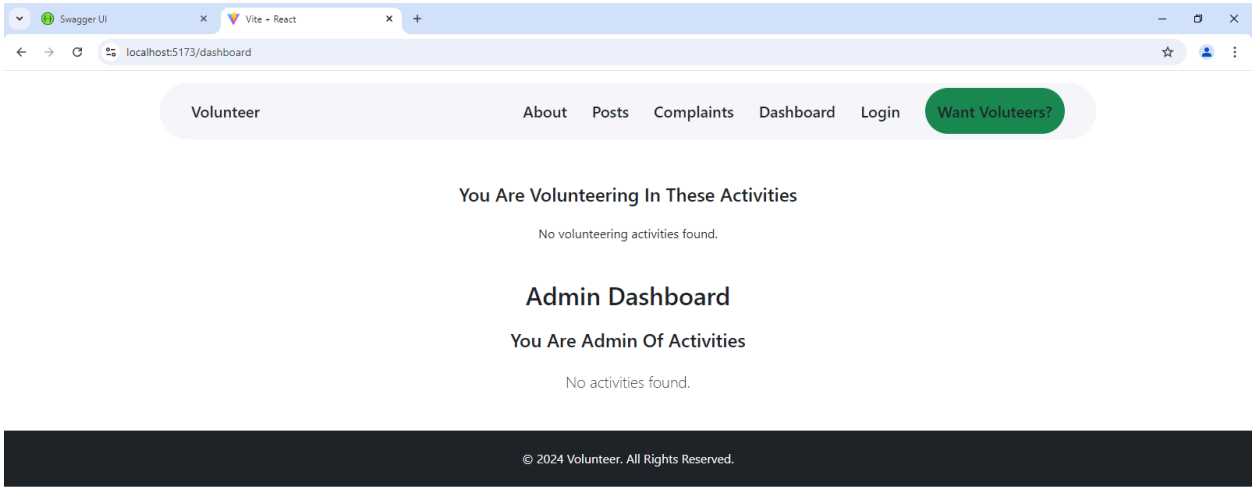
The complaints page is created using razor page , then using the controllers we create an api to get the complaints page in the frontend. It's a form to allow users to write complaints, after submitting the content will be sent to the backend again to store in the database.(list at this moment)



User And Admin Dashboard:

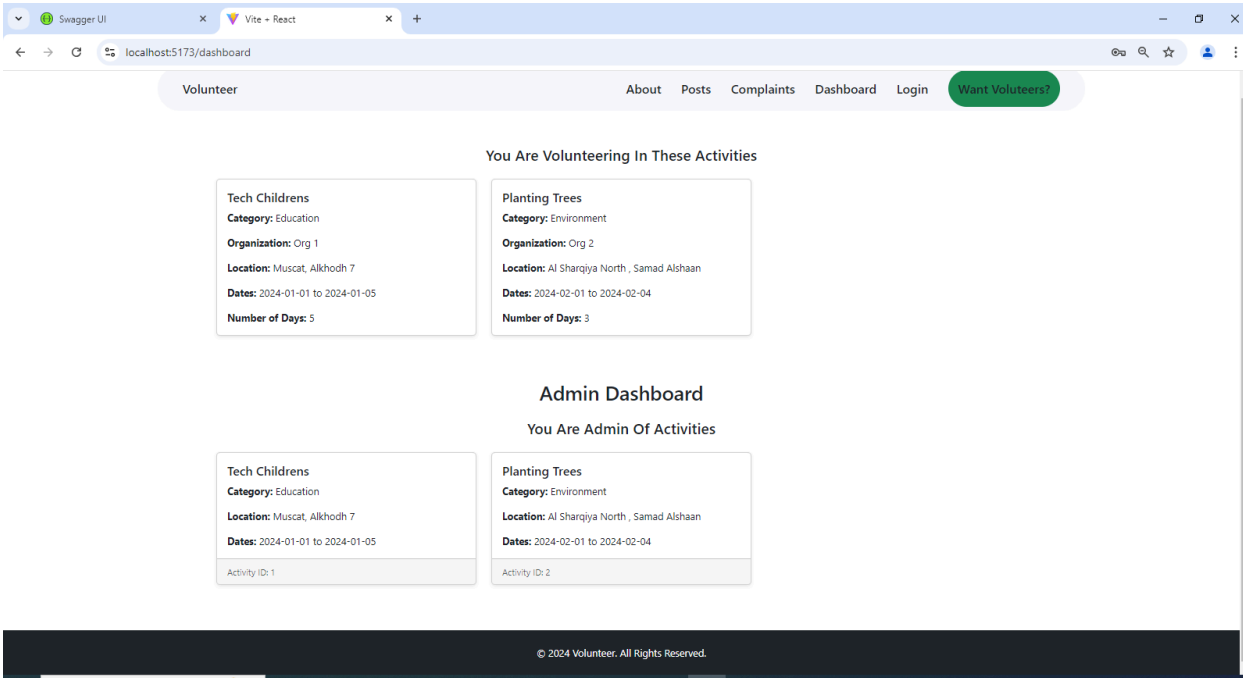
Before login:

This page shows the opportunities that the user volunteer in or the volunteering opportunities that a user or ognaization created looking for volunteers. In the image below the user can't see any posts if he/she does not logged in.



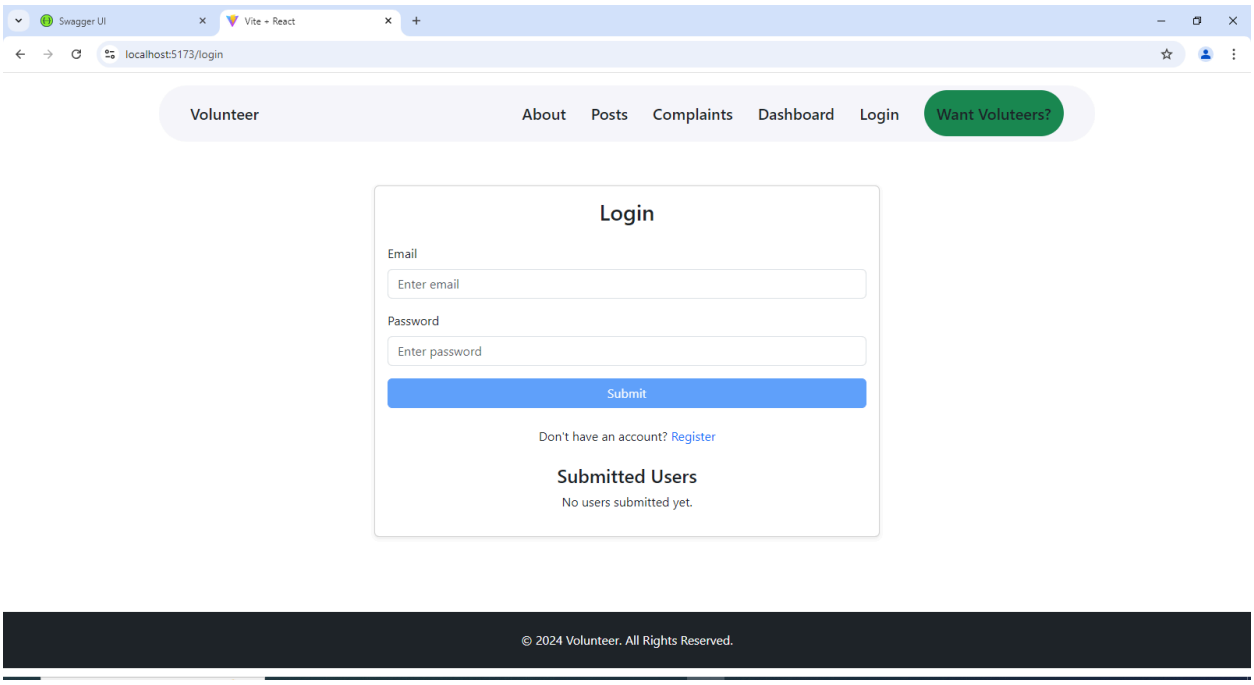
After login:

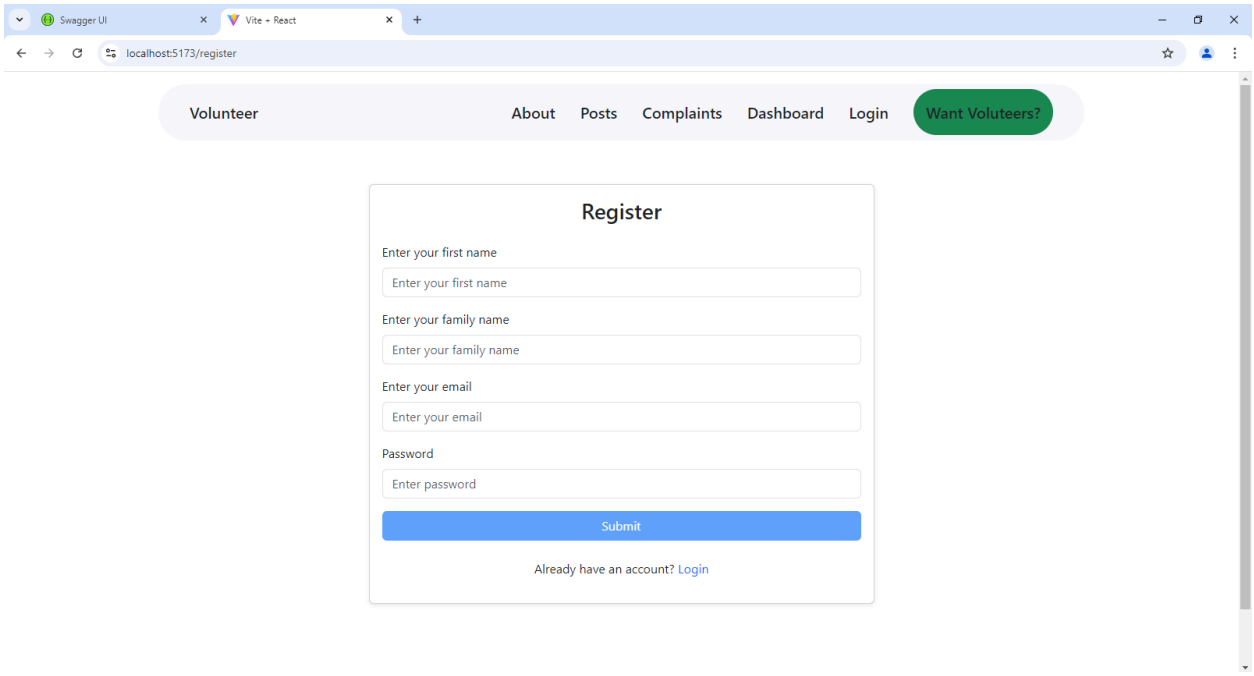
Using sessions and cookies the user can stay logged in when navigating between different pages. The user can see the opportunities that volunteers in and the opportunities user created.



Log in and Register pages:

when user register in the website his information will be sent to the database, and then we he try to login the we search about the user in the database.After login the user email will be used across the website using sessions and cookies.




Swagger UI
Vite + React
localhost:5173/register
Volunteer
About
Posts
Complaints
Dashboard
Login
Want Voluteers?
Register
Enter your first name
Enter your first name
Enter your family name
Enter your family name
Enter your email
Enter your email
Password
Enter password
Submit
Already have an account? Login