TECHNISCHE UNIVERSITÄT
CHEMNITZ

Term Paper

Path Finder
Map Application

Subject :

Datenbanken und Webtechniken

Course:

Web Engineering

By

Darshan Ashwinbhai Dhanani

Matriculation number: **813118**

Content

1 Introduction

This project delivers an interactive map application named "Path Finder", designed to enhance access to information about various facilities in Chemnitz. With the growing demand for free, non-proprietary information, Open Data portals are increasingly common at various administrative levels. The Chemnitz Open Data portal[1], for example, offers valuable resources for such applications.

"Path Finder" utilizes data from the Chemnitz Open Data Portal[1] to provide comprehensive, user-friendly access to information about these facilities. The application includes a range of features aimed at enhancing user experience and accessibility:

- **Interactive Map Interface:** A dynamic map that allows users to interact with various elements to obtain detailed information.
- **Category Facility:** Users can filter facilities based on specific categories such as schools, kindergardens, and social child project.
- **Routes Between Facilities:** The application provides the most efficient routes between two selected facilities, helping users plan their visits.
- **Pop-up Information:** Each map icon is equipped with a pop-up feature that provides detailed information about the facility.
- **User-Set Home Address:** Users can set their home address within the application for more personalized navigation.
- **Favorite Marking**: Facilities can be marked as favorites for easy access in the future.
- **Live Location Visibility:** The application can display the user's live location on the map for real-time navigation.

2 Technologies Used

The following sections outline the key technologies used in different domains to create the Path Finder website:

2.1 Frontend

The user interface of the web application is designed using HTML for structure, CSS for styling, and JavaScript for interactivity. Additionally, Jinja2 templating engine is utilized within Flask to dynamically render HTML templates.

- ***HTML (HyperText Markup Language):*** Provides the structure and content of the web pages.
- ***CSS (Cascading Style Sheets):*** Controls the presentation and layout, enhancing the visual appeal and responsiveness of the application.
- ***JavaScript:*** Adds interactivity and dynamic behavior to the user interface, facilitating client-side scripting operations.
- ***Jinja2 Templating[9]:*** Integrated with Flask, Jinja2 allows for the creation of reusable HTML templates with dynamic content rendering, promoting code reusability and maintainability.

2.2 Backend

Flask, a lightweight Python web framework, serves as the backend technology for the application.

- ***Flask:*** Chosen for its simplicity and flexibility in building web applications. Flask provides essential tools and libraries to handle routing, HTTP requests, and responses, making it suitable for rapid development and deployment.
- ***Python:*** As the programming language behind Flask, Python offers a wide range of libraries and frameworks that integrate seamlessly with Flask, enhancing functionality and supporting diverse application requirements.

2.3 Database

MongoDB is used as the database management system (DBMS) for the application.

- ***MongoDB****:* MongoDB's flexible, document-based NoSQL model supports dynamic data needs, ensuring scalability and easy adaptation to changing structures.

2.4 External Web Service

- ***Mailtrap[2]***: I used Mailtrap to test email functionality during development. It simulates an SMTP server, capturing and previewing outgoing emails without sending them to real users. This ensured reliable email testing and debugging, streamlining the process and preventing unintended emails.

- ***Leaflet[5]:*** I integrated Leaflet to create mobile-friendly interactive maps. This leading open-source JavaScript library provided the tools to build rich, dynamic map interfaces easily. Its simplicity and extensive features enhanced the usability and appeal of my application.

3 Project Workflow

3.1 User Account Management

The Path Finder website offers users two main options for accessing the system: registering as a new user or logging in as an existing user. Additionally, there is a feature to reset forgotten passwords.

3.1.1 Registration:

New users register by completing a form with required fields: first name, last name, date of birth, role (student, parent, teacher, other), email, and password.
Upon successful submission, the user's credentials are stored in the database in an encrypted format.

- ***Email Validation:*** An on-focus-out event checks the email against the database. If the email is already in use, an error message prompts the user to use a different email address.This validation ensures users provide a unique email address before final submission.

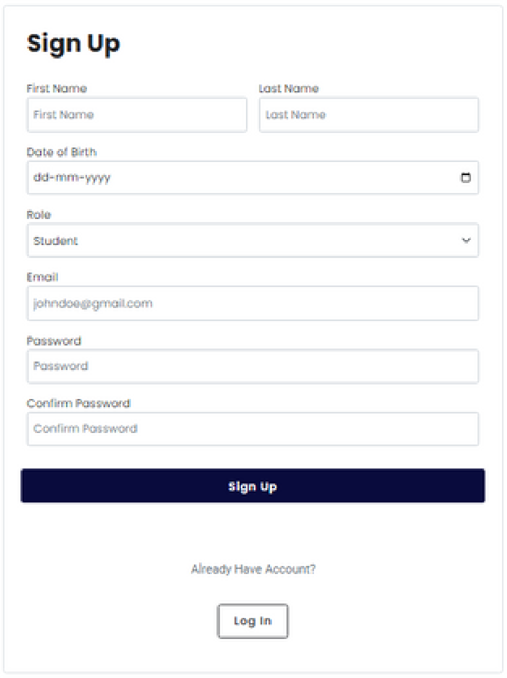


Figure 1: Registration From.

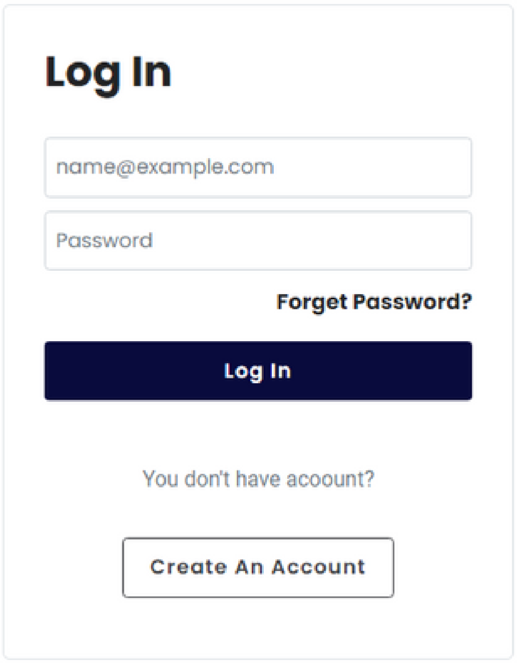


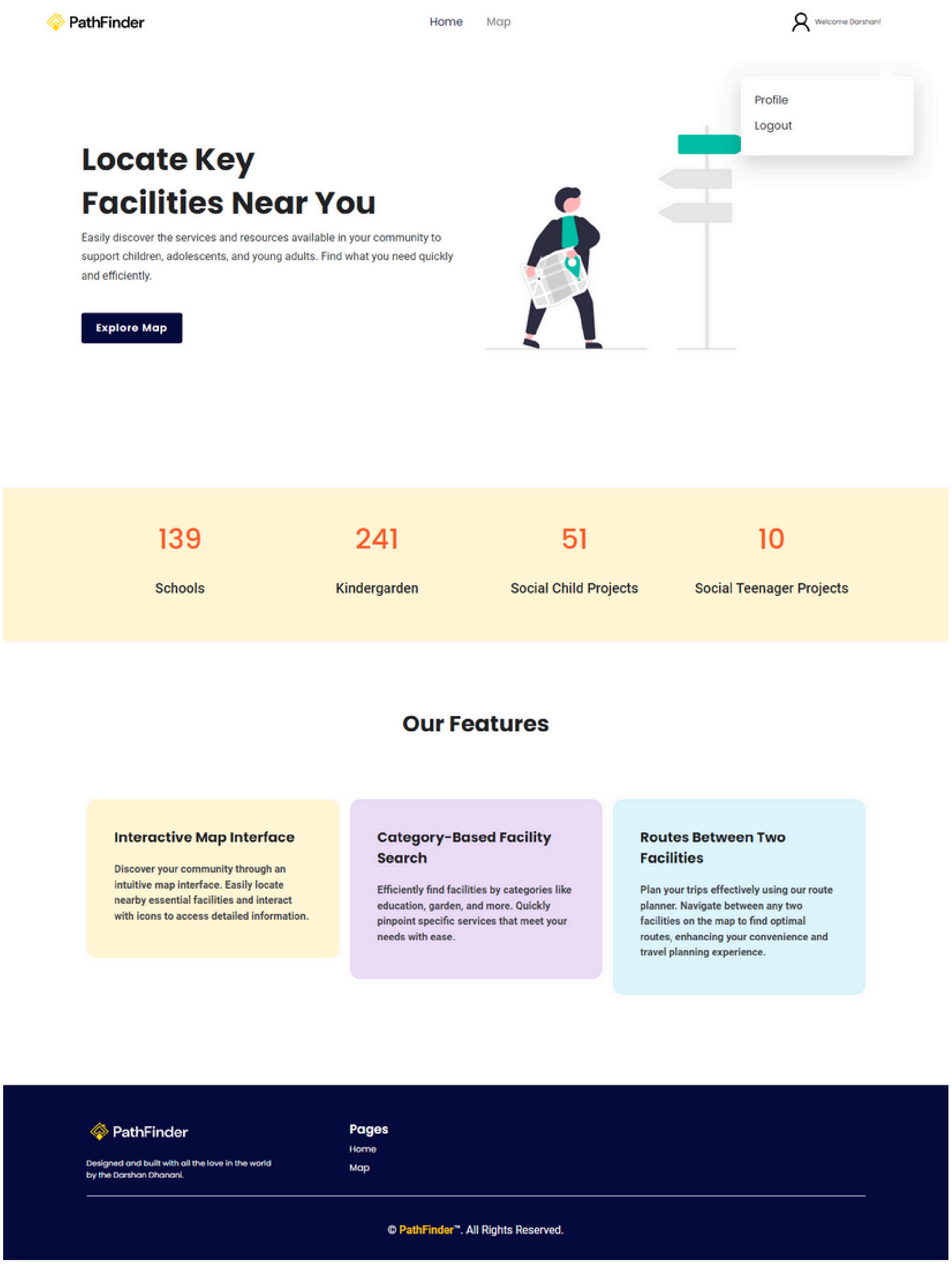
Figure 2: Login Form.

3.1.2 Login:

Registered users log in with their email and password. If the credentials match the stored records, the user gains access to the home page and all site functionalities.

3.1.3 Forgot Password:

Users click on the "Forgot Password" link and request a new password. An email with a reset password link is sent to their registered email address. By clicking this link, users can set a new password and regain access to their profile.



PathFinder

Home Map

Welcome Darshan!

Profile

Logout

Locate Key Facilities Near You

Easily discover the services and resources available in your community to support children, adolescents, and young adults. Find what you need quickly and efficiently.

Explore Map

139	241	51	10
Schools	Kindergarden	Social Child Projects	Social Teenager Projects

Our Features

Interactive Map Interface

Discover your community through an intuitive map interface. Easily locate nearby essential facilities and interact with icons to access detailed information.

Category-Based Facility Search

Efficiently find facilities by categories like education, garden, and more. Quickly pinpoint specific services that meet your needs with ease.

Routes Between Two Facilities

Plan your trips effectively using our route planner. Navigate between any two facilities on the map to find optimal routes, enhancing your convenience and travel planning experience.

PathFinder

Designed and built with all the love in the world by the Darshan Dhanani.

Pages

Home

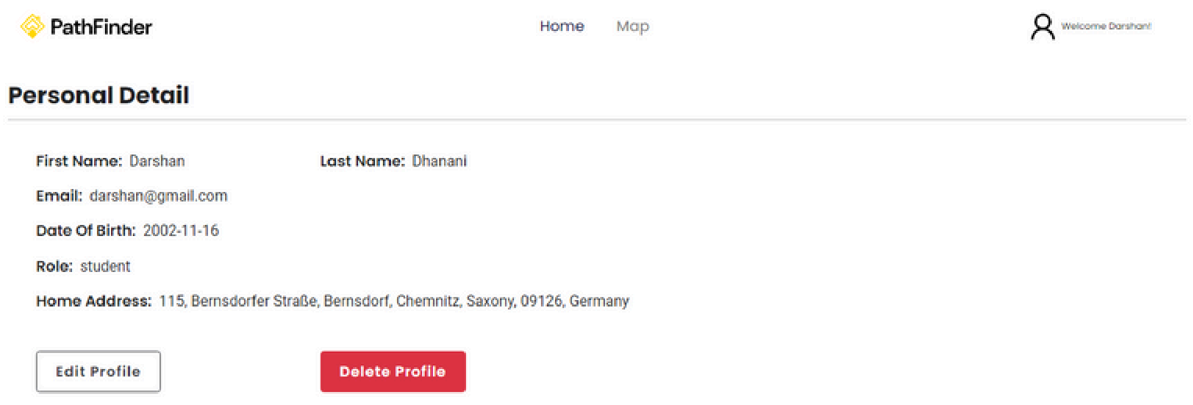
Map

© PathFinder™. All Rights Reserved.

Figure 3: Home Page After Login.

3.2 User Profile Management:

The profile displays essential personal information such as first name, last name, date of birth, role (student, parent, teacher, other), email, and any other relevant details provided during registration.


PathFinder

Home Map

Welcome Darshan!

Personal Detail

First Name: Darshan **Last Name:** Dhanani

Email: darshan@gmail.com

Date Of Birth: 2002-11-16

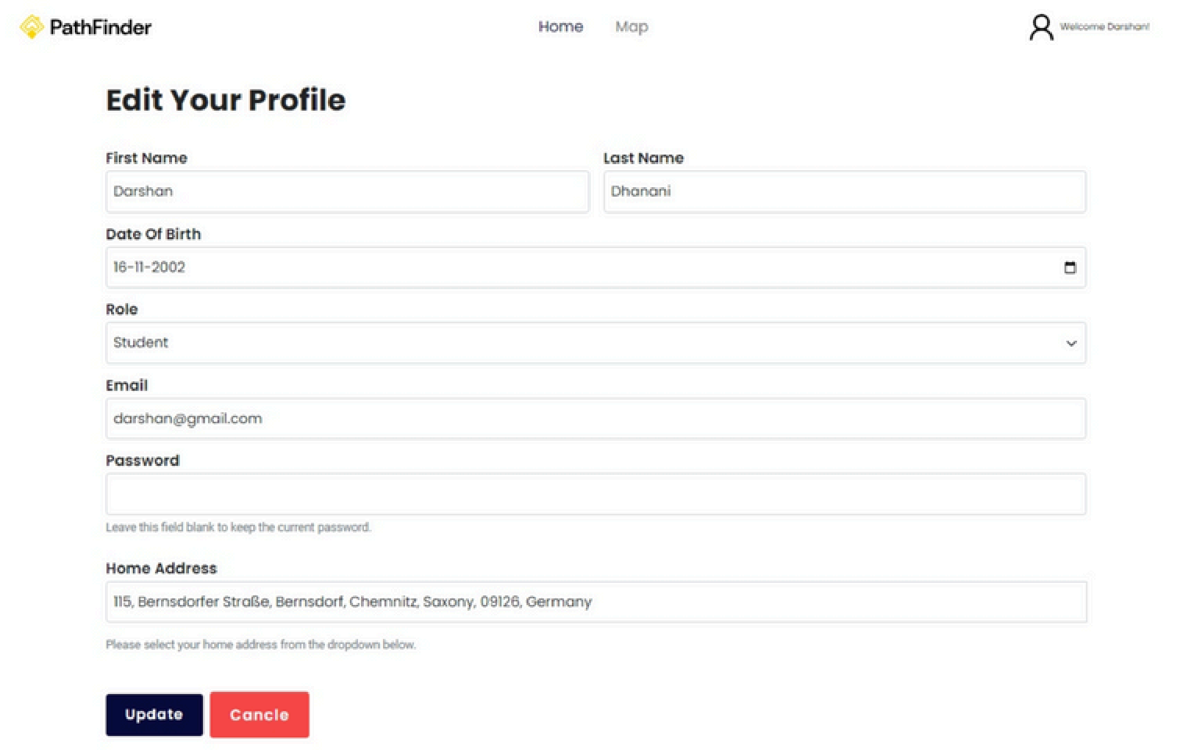
Role: student

Home Address: 115, Bernsdorfer Straße, Bernsdorf, Chemnitz, Saxony, 09126, Germany

Figure 4: User Profile.

***Edit Profile**:* Users can click the "Edit Profile" button to update their details, including their home address. Changes are securely saved in the database.

Delete Profile & Revoke Account: Clicking the "Delete Profile" button allows users to deactivate their account. Upon deletion, an email is sent with a reactivation link, allowing users to restore their account if desired.


PathFinder

Home Map

Welcome Darshan!

Edit Your Profile

First Name	Last Name
Darshan	Dhanani

Date Of Birth: 16-11-2002

Role: Student

Email: darshan@gmail.com

Password:

Leave this field blank to keep the current password.

Home Address: 115, Bernsdorfer Straße, Bernsdorf, Chemnitz, Saxony, 09126, Germany

Please select your home address from the dropdown below.

Figure 5: Edit Profile.

3.3 Map View

The map page of the Path Finder application serves as the main interface for users to interact with the map and access various functionalities. This page is designed to provide a user-friendly experience with multiple features that help users find and navigate to various facilities. Below is a detailed description of the elements and features present on this page:

3.3.1 Interactive Map Interface

The main feature of this page is the interactive map located in the center of the screen. The map is dynamic, allowing users to zoom in and out, and interact with various markers that represent different facilities. The map utilizes data from the Open Data Portal[1] to ensure up-to-date information about the facilities.

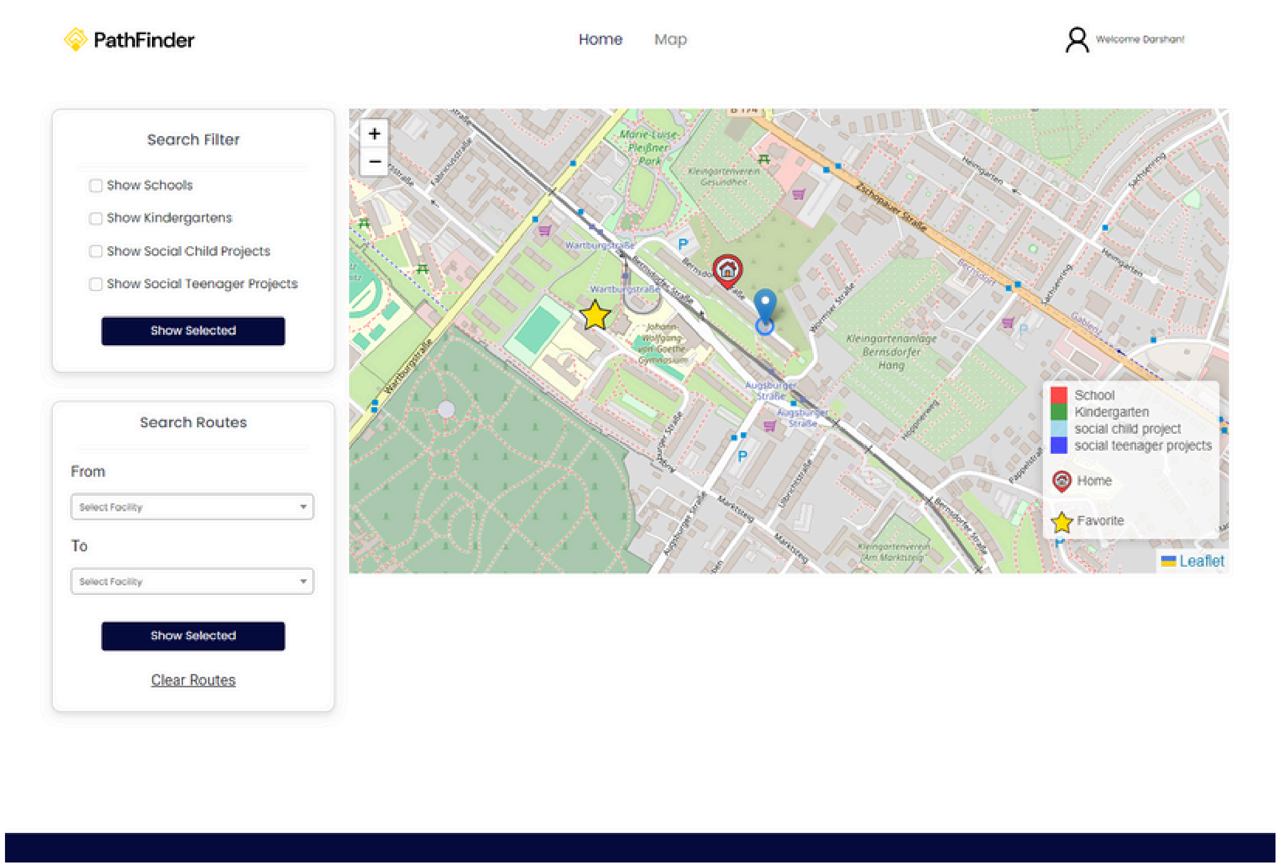


Figure 6: Map Page.

Markers and Icons:

- ***Blue Icon***: Represents the user's live location, helping them see where they are in real-time.
- ***House Icon:*** Indicates the user's set home address for personalized navigation.
- ***Star Icon:*** Marks the user's favorite facility, allowing easy access to frequently used or preferred locations.

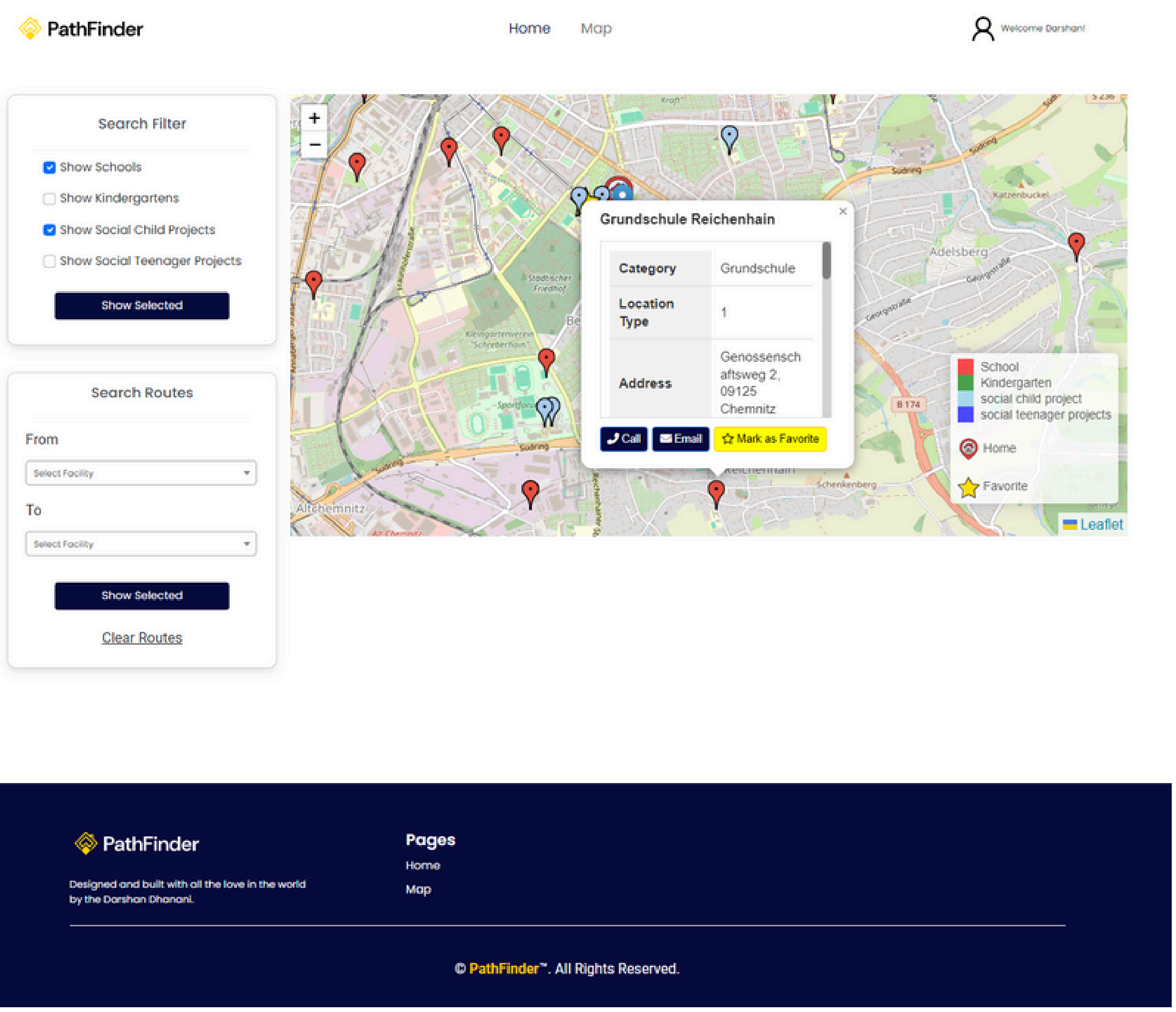
3.3.2 Sidebar Functionalities

The left sidebar provides two main functionalities: filtering facilities by category and finding routes between two selected facilities.

1. Search Filter: Users can filter the facilities displayed on the map by selecting categories such as schools, kindergardens, social child projects, and social teenager projects.

Pop-up Information: Each icon on the map is equipped with a pop-up feature. When a user clicks on a marker, a pop-up window appears, providing detailed information about the facility. This includes the facility's name, address, contact information, and any other relevant details. Additionally, the pop-up contains buttons for direct actions:

- ***Call Button****:* Allows users to call the facility directly from the application.
- ***Email Button***: Enables users to send an email to the facility directly from the application.
- ***Favorite Button***: Provides an option to mark the facility as a favorite for easy future access.



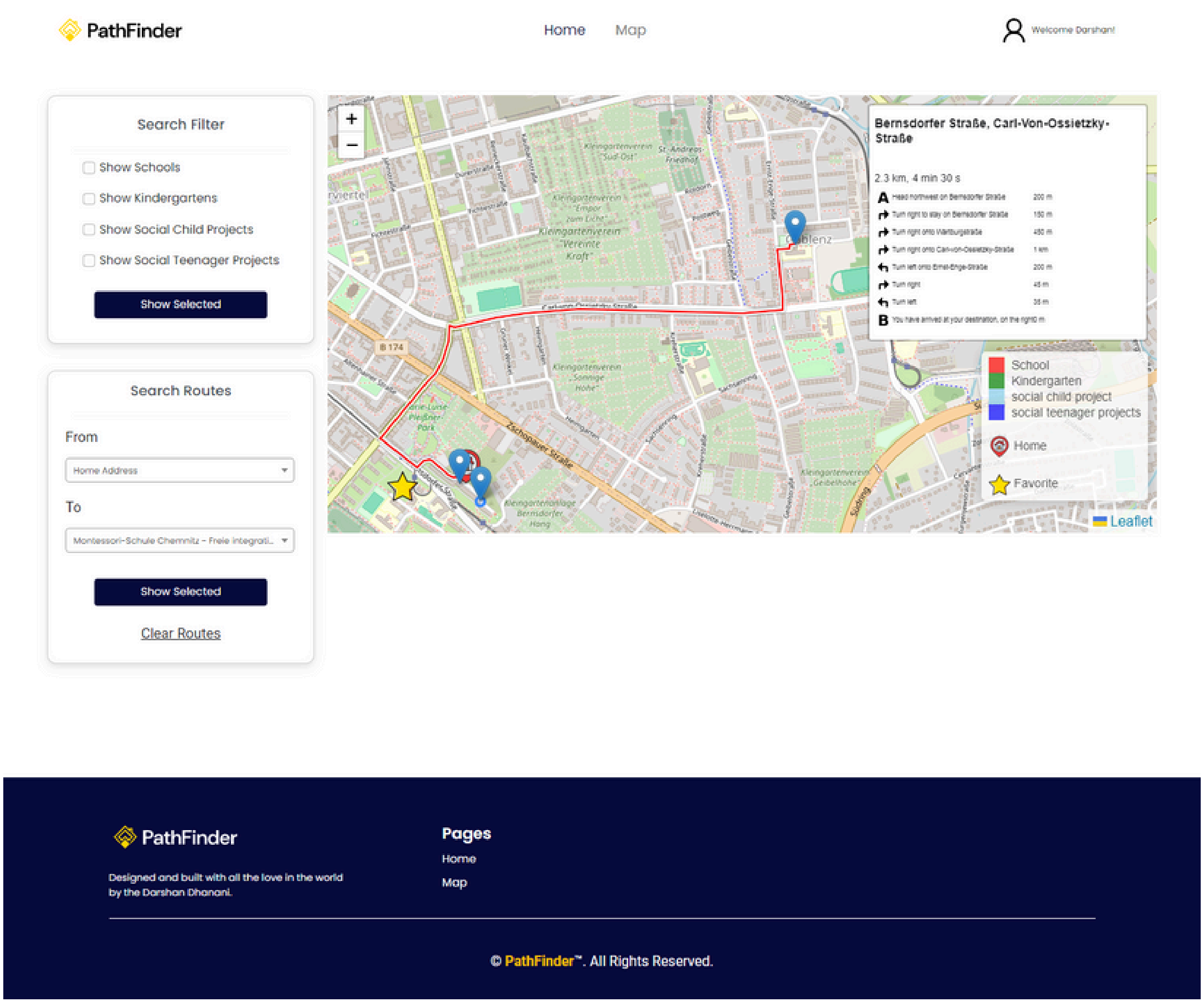
Grundschule Reichenhain

Category	Grundschule
Location Type	1
Address	Genossenschaftsweg 2, 09125 Chemnitz

Figure 7: Map Page with popup.

2. Search Routes:

- This feature allows users to find the most efficient routes between two selected facilities.
- Users can choose the starting facility and the destination facility from the drop-down menus.



Bernsdorfer Straße, Carl-Von-Ossietzky-Straße

2.3 km, 4 min 30 s

A	Head northwest on Bernsdorfer Straße	200 m
↱	Turn right to stay on Bernsdorfer Straße	150 m
↱	Turn right onto Wartburgstraße	450 m
↱	Turn right onto Carl-von-Ossietzky-Straße	1 km
↰	Turn left onto Ernst-Enge-Straße	200 m
↱	Turn right	45 m
↰	Turn left	35 m
B	You have arrived at your destination, on the right	0 m

Designed and built with all the love in the world by the Darshan Dhanani.

© PathFinder™. All Rights Reserved.

Figure 8: Map Page with route.

4 Endpoints Documentation

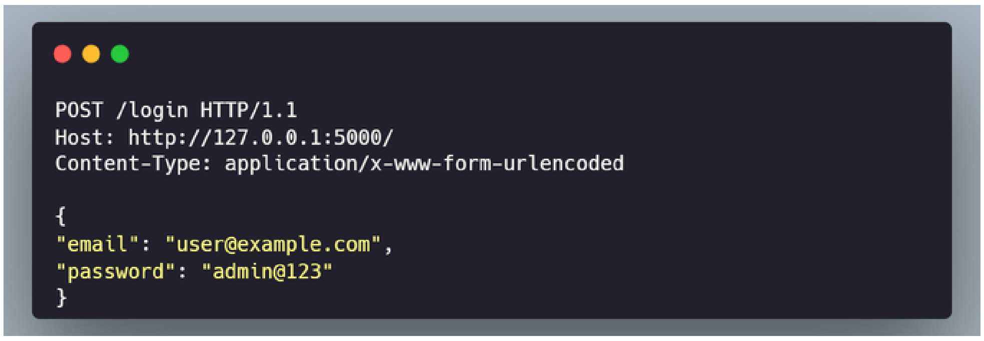
4.1 User Authentication

Login

`/login` (GET, POST)

- GET: Renders the login page.
  Template: `login.html`
- POST: Processes the login form and authenticates the user.
  Parameters:
    `email` (string) : User's email address.
    `password` (string) : User's password.
  On Success: Redirects to the home page.
  On Failure: Renders the login page with an error message.

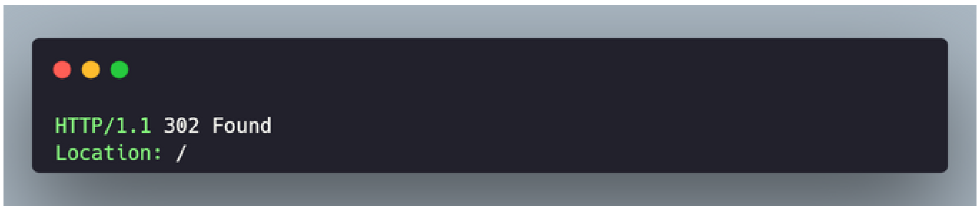
Example (POST)Request:



```
POST /login HTTP/1.1
Host: http://127.0.0.1:5000/
Content-Type: application/x-www-form-urlencoded

{
"email": "user@example.com",
"password": "admin@123"
}
```

Example (POST)Response:



```
HTTP/1.1 302 Found
Location: /
```

Register

`/register` (GET, POST)

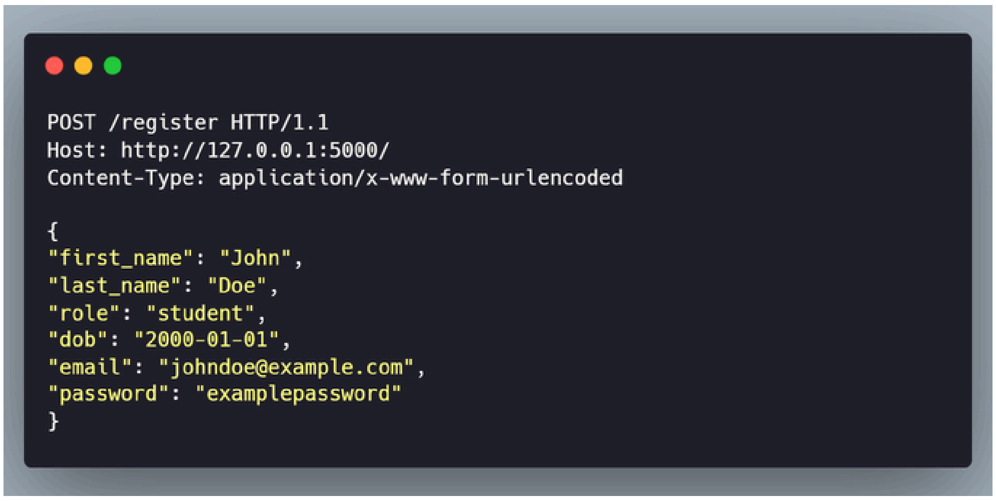
- GET: Renders the registration page.
  Template: `register.html`
- POST: Processes the registration form.
  Parameters:
    `first_name` (string) : User's first name.
    `last_name` (string) : User's last name.
    `dob` (date) :User's date of birth.
    `role` (string) : User's role (student, parent, teacher, other).

`email` (string) : User's email address.
`password` (string) : User's password.

On Success: Redirects to the home page.

On Failure: Renders the registration page with an error message.

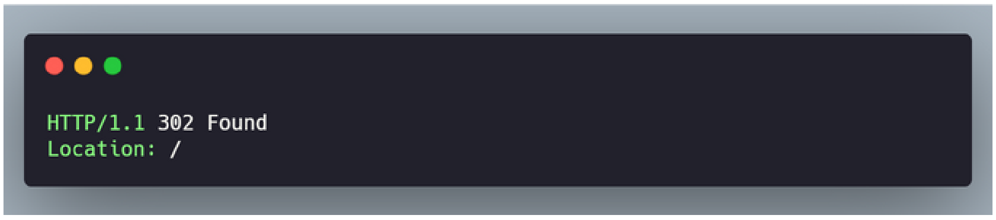
Example (POST)Request:



```
POST /register HTTP/1.1
Host: http://127.0.0.1:5000/
Content-Type: application/x-www-form-urlencoded

{
"first_name": "John",
"last_name": "Doe",
"role": "student",
"dob": "2000-01-01",
"email": "johndoe@example.com",
"password": "examplepassword"
}
```

Example (POST)Response:



```
HTTP/1.1 302 Found
Location: /
```

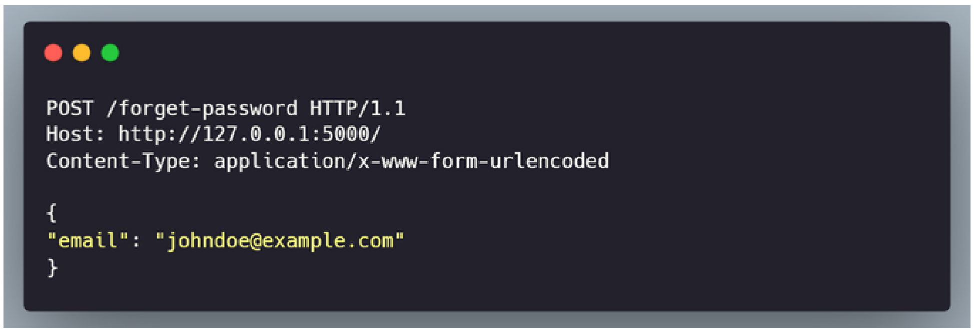
Forgot Password

`/forgot-password` (GET, POST)

- GET: Renders the forgot password page.
  Template: `forget_password.html`
- POST: Processes the forgot password form.
  Parameters:
  `email` (string) : User's registered email address.

  On Success: Sends a reset password link to the user's email and renders a confirmation page.
  On Failure: Renders the forgot password page with an error message.

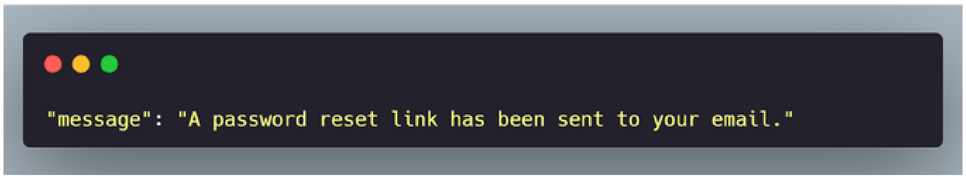
Example (POST)Request:



```
POST /forget-password HTTP/1.1
Host: http://127.0.0.1:5000/
Content-Type: application/x-www-form-urlencoded

{
"email": "johndoe@example.com"
}
```

Example (POST)Response:



```
"message": "A password reset link has been sent to your email."
```

Logout

`/logout` (GET)

- **GET**: Logs the user out by clearing the session and redirects to the home page.
  Template: `None` (redirects to the home page)

  On Success: The session is cleared and the user is redirected to the home page (index).

  On Failure: If the user is not logged in, they cannot access this route and will be redirected to the login page (handled by the @login_required decorator).

4.2 Profile Management

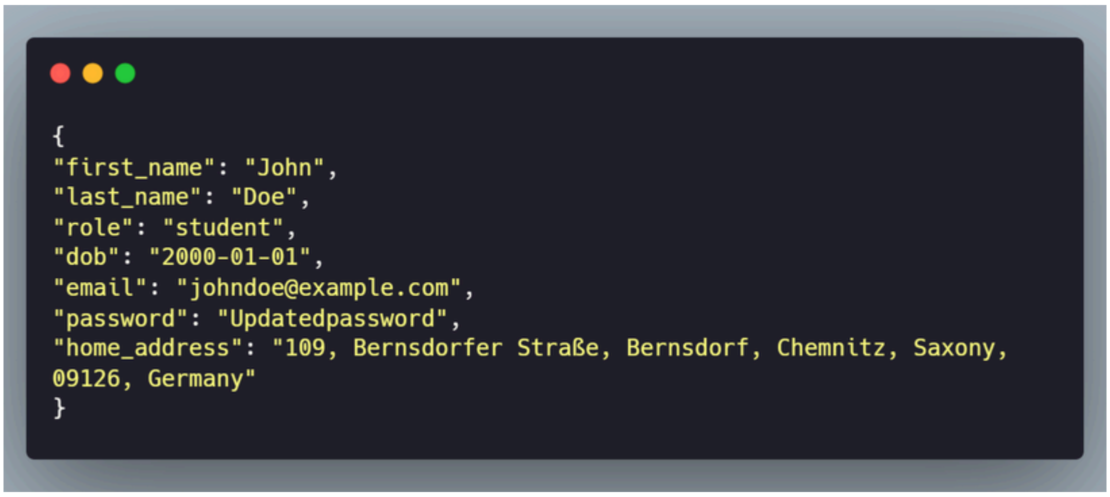
View Profile

`/user-profile` (GET)

- **GET**: Displays the user's profile page.
  Template: `user_profile.html`

  Description: Shows the user's personal information and home address.

  Data Passed to Template:



```
{
"first_name": "John",
"last_name": "Doe",
"role": "student",
"dob": "2000-01-01",
"email": "johndoe@example.com",
"password": "Updatedpassword",
"home_address": "109, Bernsdorfer Straße, Bernsdorf, Chemnitz, Saxony,
09126, Germany"
}
```

Edit Profile

`/edit-profile` (GET, POST)

- **GET**: Renders the edit profile page.
  Template: `edit_profile.html`

- POST: Processes the profile update form.

  Parameters:

  `first_name` (string) : User’s first name.

  `last_name` (string) : User’s last name.

  `dob` (date) :User’s date of birth.

  `role` (string) : User’s role (student, parent, teacher, other).

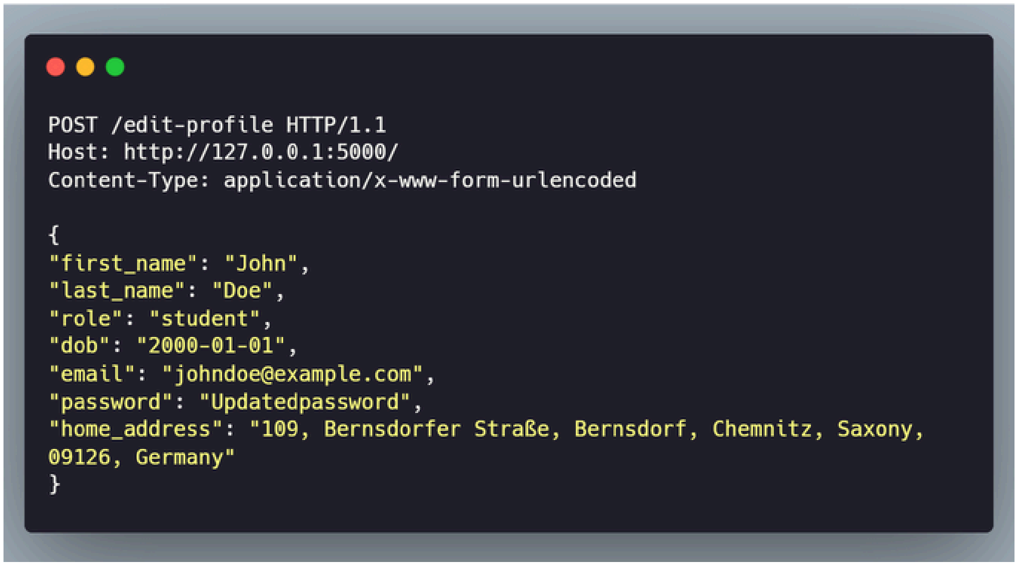
  `email` (string) : User's email address.

  `password` (string) : User's password.

  `home_address` (string) : User’s home address.

  On Success: Updates the profile in the database and redirects to the profile page.
  On Failure: Renders the edit profile page with an error message.

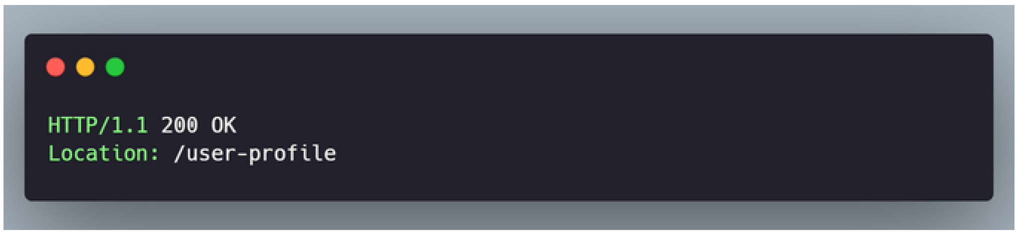
Example (POST)Request:



```
POST /edit-profile HTTP/1.1
Host: http://127.0.0.1:5000/
Content-Type: application/x-www-form-urlencoded

{
"first_name": "John",
"last_name": "Doe",
"role": "student",
"dob": "2000-01-01",
"email": "johndoe@example.com",
"password": "Updatedpassword",
"home_address": "109, Bernsdorfer Straße, Bernsdorf, Chemnitz, Saxony,
09126, Germany"
}
```

Example (POST)Response:



```
HTTP/1.1 200 OK
Location: /user-profile
```

Delete Profile & Revoke Account

`/delete-profile` (GET)

- GET: Deactivates the user’s account

  Template: `None` (redirects to the home page)

  On Success: Sends an email with a reactivation link and renders a confirmation page.

  On Failure: Renders an error page.

4.3 Map View

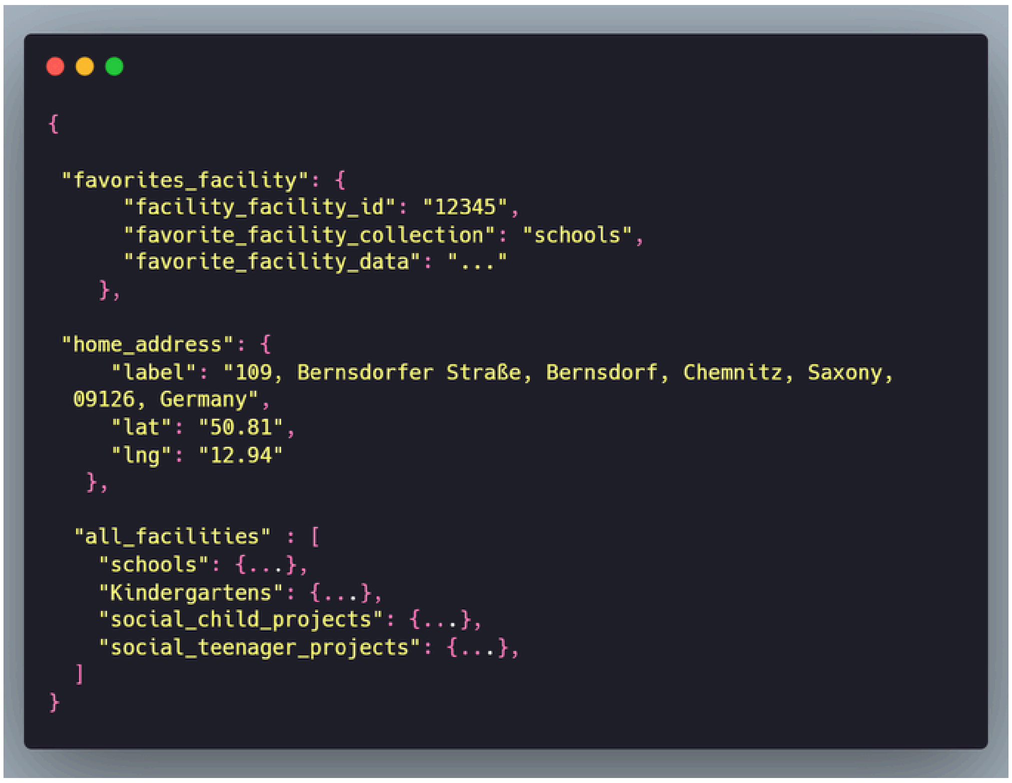
Map Page

`/map` (GET)

- **GET**: Displays the interactive map page.
  Template: `map.html`

  Description: Provides features such as searching locations, viewing points of interest, and planning routes.

  Data Passed to Template:



```
{

 "favorites_facility": {
      "facility_facility_id": "12345",
      "favorite_facility_collection": "schools",
      "favorite_facility_data": "..."
    },

 "home_address": {
     "label": "109, Bernsdorfer Straße, Bernsdorf, Chemnitz, Saxony,
  09126, Germany",
     "lat": "50.81",
     "lng": "12.94"
   },

  "all_facilities" : [
    "schools": {...},
    "Kindergartens": {...},
    "social_child_projects": {...},
    "social_teenager_projects": {...},
  ]
}
```

Filter facilities by category

`/map-json-data` (GET)

- **GET**: Specifies that the endpoint is used for searching map data by facility types (categories).
  Template: `None` (route redirect data to map page with data)

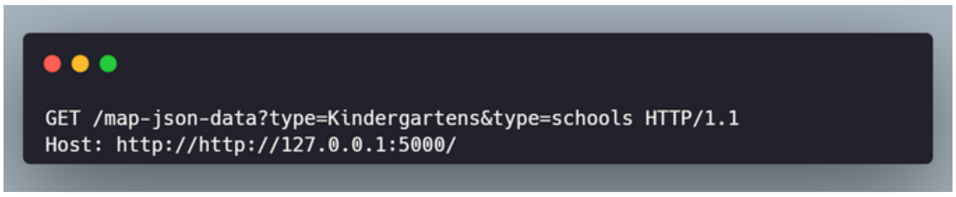
  Parameters:

  `type` (list of strings) : Specifies the types (categories) of facilities to fetch map points for.

  On Success: Server responds with the requested data.

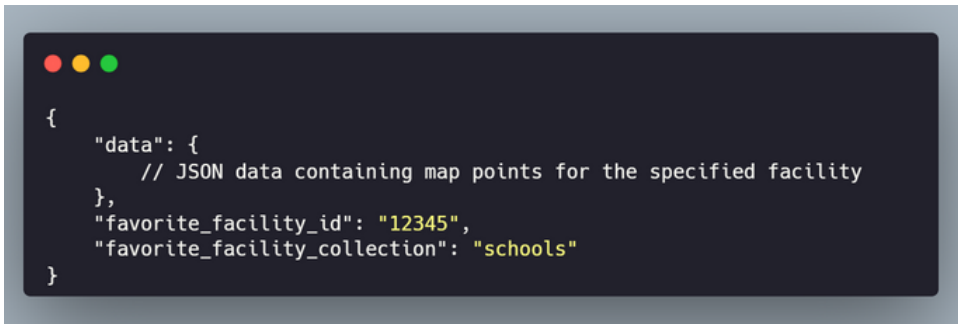
  On Failure: Server responds with an error message or relevant details may be included in the response payload to help diagnose and resolve the issue.

Example (GET)Request:



```
GET /map-json-data?type=Kindergartens&type=schools HTTP/1.1
Host: http://http://127.0.0.1:5000/
```

Example (GET)Response:



```
{
    "data": {
        // JSON data containing map points for the specified facility
    },
    "favorite_facility_id": "12345",
    "favorite_facility_collection": "schools"
}
```

Mark as Favorite Facility

`/mark-as-favorite-facility/<facility_id>/<facility_name>` (POST)

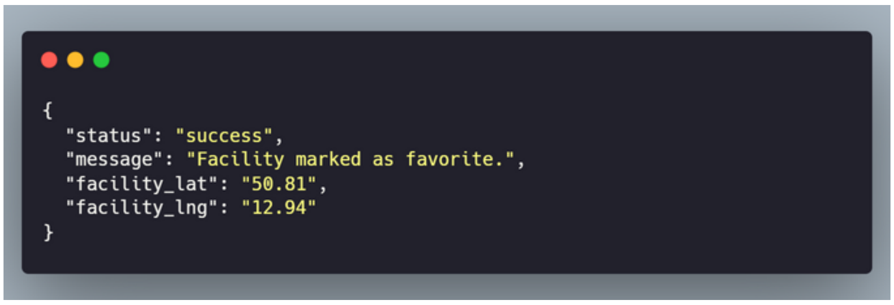
- POST: Marks a facility as a favorite for the user.

  Parameters:

  `facility_id` (string) : The ID of the facility.

  `facility_name` (string) : The name of the facility.

Example (POST)Response:



```
{
  "status": "success",
  "message": "Facility marked as favorite.",
  "facility_lat": "50.81",
  "facility_lng": "12.94"
}
```

Remove as Favorite Facility

`/remove-as-favorite-facility/<facility_id>/<facility_name>` (POST)

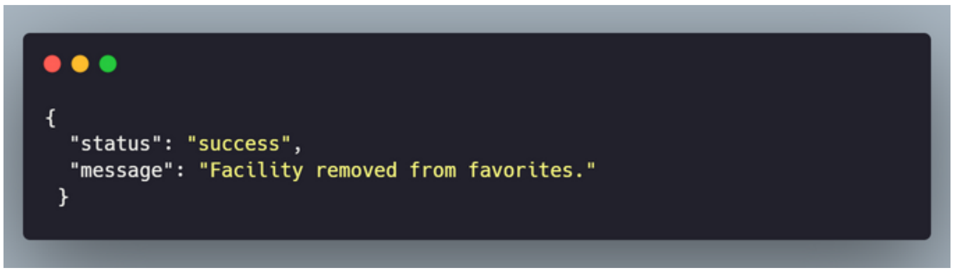
- POST: Removes a facility from the user's favorites.

  Parameters:

  `facility_id` (string) : The ID of the facility.

  `facility_name` (string) : The name of the facility.

Example (POST)Response:



```
{
  "status": "success",
  "message": "Facility removed from favorites."
 }
```

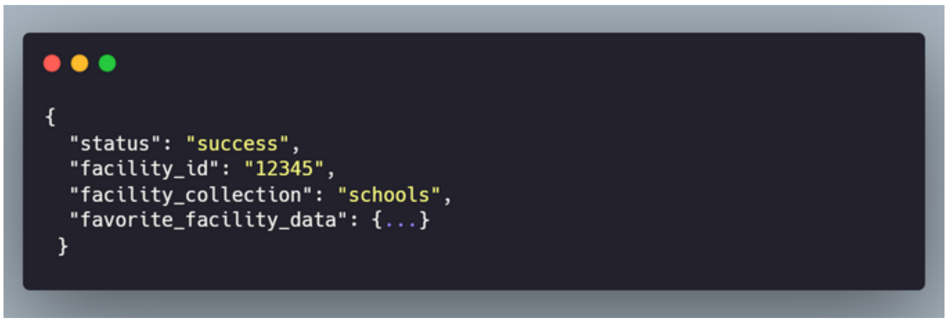
Fetch Favorite Facility

`/fetch-favorite-facility` (GET)

- GET: Fetches the user's favorite facilities.

  Template: `None` (route redirect data to map page with data)

Example (POST)Response:



```
{
  "status": "success",
  "facility_id": "12345",
  "facility_collection": "schools",
  "favorite_facility_data": {...}
 }
```

5 References

[1] https://portal-chemnitz.opendata.arcgis.com/

[2] Mailtrap: https://mailtrap.io

[3] Flask: https://flask.palletsprojects.com/en/3.0.x/

[4] MongoDB: https://www.mongodb.com

[5] Leaflet: https://leafletjs.com/reference.html

[6] Chat GPT: https://chat.openai.com/

[7] https://carbon.now.sh/

[8] https://stackoverflow.com

[9] Jinja2: https://jinja.palletsprojects.com/en/3.1.x/