Service Layer Design for GreenHaven App

The GreenHaven app will have a service layer to manage interactions between the user interface and the Firebase Firestore database. This service layer will abstract the database operations, providing a clean API for the frontend to interact with. The service layer will handle user authentication, reviews management, and user-specific data like favorites and recent searches, while leveraging the Google Places API for green space data retrieval.

Service Endpoints

Below are the service endpoints required for the MVP of GreenHaven, along with an explanation of their purpose and example request and responses.

1. **User Authentication Endpoints**
    a. *Register User:*
        i. Method: POST
        ii. Endpoint: /api/auth/register
        iii. Purpose: Register a new user in the system
        iv. Example Request:

```
{
  “email”: user@mail.com,
   "username": "user123",
   "password": "password123"
 }
```

        v. Example Response:

```
{
   "userId": "user123",
   "email": "user@example.com",
   "username": "user123",
   "createdAt": "2024-07-01T12:00:00Z"
}
```

vi. Error Response:

```
{
    "error": "Email already in use"
}
```

b. *Login User:*

i. Method: POST

ii. Endpoint: /api/auth/login

iii. Purpose: Authenticate a user and return a token

iv. Example Request:

```
{
  “email”: user@mail.com,
   "password": "password123"
 }
```

v. Example Response:

```
{
    "token": "abcdefg123456",
    "userID": "user123"
}
```

vi. Error Response:

```
{
    "error": "Invalid Email or Password"
}
```

2. Review Management Endpoints

a. *Submit Review:*

i. Method: POST

ii. Endpoint: /api/reviews

iii. Purpose: Submit a review for a green space

iv. Example Request:

```
{
   "userId": "user123",
   "placeId": "place789",
   "rating": 4.5,
   "comment": "Beautiful Park with great facilities."
}
```

v. Example Response:

```
{
   "reviewId": "review456",
   "userId": "user123",
   "placeId": "place789",
   "rating": 4.5,
   "comment": "Beautiful Park with great facilities.",
   "timestamp": "2024-07-02T10:30:00Z"
}
```

vi. Error Response:

```
{
   "error": "Failed to submit review"
}
```

b. *Get Reviews for a Place:*

i. Method: GET

ii. Endpoint: /api/reviews?placeID=place789

iii. Purpose: Retrieve all reviews for a specific green space.

iv. Example Response:

```
[
 {
    "reviewId": "review456",
    "userId": "user123",
    "placeId": "place789",
    "rating": 4.5,
```

```
    "comment": "Beautiful Park with great facilities.",
    "timestamp": "2024-07-02T10:30:00Z"
  }
]
```


v. Error Response:


```
{
  "error": "Failed to retrieve reviews"
}
```


c. *Get User Reviews:*

i. Method: GET

ii. Endpoint: /api/users/user123/reviews

iii. Purpose: Retrieve all reviews submitted by a specific user.

iv. Example Response:


```
[
  {
    "reviewId": "review456",
    "userId": "user123",
    "placeId": "place789",
    "rating": 4.5,
    "comment": "Beautiful Park with great facilities.",
    "timestamp": "2024-07-02T10:30:00Z"
  }
]
```


v. Error Response:


```
{
  "error": "Failed to retrieve user reviews"
}
```


3. User Data Management Endpoints

a. *Get User Favorites:*

i. Method: GET

ii. Endpoint: /api/users/user123/favorites

iii. Purpose: Retrieve a list of favorite green spaces for a user.

iv. Example Response:

```
[
  "place789",
  "place123"
]
```

v. Error Response:

```
{
  "error": "Failed to retrieve favorites"
}
```

b. *Add to Favorites:*

i. Method: POST

ii. Endpoint: /api/ users/user123/favorites

iii. Purpose: Add a green space to a user's favorites.

iv. Example Request:

```
{
  "placeId": "place789"
}
```

v. Example Response:

```
{
  "message": "Favorite added successfully"
}
```

vi. Error Response:

```
{
  "error": "Failed to add favorite"
}
```

c. *Get Recent Searches:*

i. Method: GET

ii. Endpoint: /api/ users/user123/recent-searches

iii. Purpose: Retrieve a list of recent searches for a user.

iv. Example Response:

```
[
   "place123",
   "place456"
]
```

v. Error Response:

```
{
   "error": "Failed to retrieve recent searches"
}
```

4. Search Endpoints

a. *Search by Address, Zip Code, or City with Radius*

i. Method: POST

ii. Endpoint: /api/places/search-by-address

iii. Purpose: Searches for green spaces by an address, zip code, or city and a specified radius

iv. Example Request:

```
{
   "address": "10024",
   "radius": 10000,   // radius in meters
   "userId": "user123"  //current authenticated user id
}
```

v. Example Response:

```
[
 {
    "place_id": "ChIJ4zGFAZpYwokRGUGph3Mf37k",
```

```
      "name": "Central Park",
      "rating": 4.6,
      "geometry": {
       "location": {
         "lat": 40.785091,
         "lng": -73.968285
       }
      }
     },
     {
      "place_id": "ChIJU0CI3H5YwokRnPhNG9oBjL8",
      "name": "Riverside Park",
      "rating": 4.5,
      "geometry": {
         "location": {
         "lat": 40.800611,
         "lng": -73.970166
       }
      }
     }
  ]
```


vi. Error Response:


```
{
   "error": "Failed to search places by address"
}
```


Communication Diagram

[UI: Create Account] <---> [POST /api/auth/register] <---> [Firestore: Users Collection]

[UI: Login] <---> [POST /api/auth/login] <---> [Firestore: Users Collection]

[UI: Write Review] <---> [POST /api/reviews] <---> [Firestore: Reviews Collection]

[UI: Green Space Listing] <---> [GET /api/reviews?placeId=place789] <---> [Firestore: Reviews Collection]

[UI: User Profile] <---> [GET /api/users/user123/reviews] <---> [Firestore: Reviews Collection]

[UI: Favorites] <---> [GET /api/users/user123/favorites] <---> [Firestore: Users Collection]

[UI: Favorites] <---> [POST /api/users/user123/favorites] <---> [Firestore: Users Collection]

[UI: Home Page] <---> [GET /api/users/user123/recent-searches] <---> [Firestore: Users Collection]

[UI: Search Page] <--> [POST /api/places/search-by-address] <--> [Firebase Function: searchByAddress] <--> [Google Places API: Geocode Address] <--> [Google Places API: Places Nearby] <--> [Firestore: Users Collection (Update Recent Searches)] <--> [Return Places Results to UI]

Summary of Service Endpoints

1. **User Authentication:**

a. POST /api/auth/register
b. POST /api/auth/login

2. **Review Management:**
    a. POST /api/reviews
    b. GET /api/reviews?placeId=place789
    c. GET /api/users/user123/reviews

3. **User Data Management:**
    a. GET /api/users/user123/favorites
    b. POST /api/users/user123/favorites
    c. GET /api/users/user123/recent-searches

4. Search Service
    a. POST/api/places/search-by-address

The service layer for the GreenHaven app is built to efficiently handle user interactions and data storage. By following RESTful principles and using well-defined endpoints, it simplifies database operations, providing a clean and easy-to-maintain API for the frontend. This setup ensures smooth and seamless user experiences, from logging in to managing reviews, making the app robust and highly responsive.