**EcoLife**

**Project Goal:**

EcoLife is a web-based platform aimed at helping users reduce their carbon footprint through personalized recommendations. EcoLife provides insights into energy use, travel habits, and waste reduction. It tailors eco-friendly suggestions to encourage sustainable living.

**Project Boundaries:**

**In-Scope:**

* User registration, login, and authentication
* Household, Transportation, and Waste Management APIs for footprint calculation
* Recommendations API for sustainability guidance
* Air Quality API for environmental data
* Secure access control for different functionalities
* Data storage for user history and calculation records

**Out-of-Scope:**

* Real-time chat or messaging features
* Geolocation-based recommendations
* Mobile app development
* Third-party service integrations beyond defined APIs

**Functional Requirements**

1. **User Registration and Login:**
   1. Users can create accounts and securely log in.
   2. Integrate secure mechanisms for password management and recovery.
2. **Carbon Footprint Calculation APIs:**
   1. **Household API:** Collects data on electricity, LPG, and CNG usage.
   2. **Transportation API:** Analyses data on personal vehicle use, public transit, and flights.
   3. **Waste Management API**: Calculates waste impact based on recycling and disposal habits.
3. **Recommendations API:**
   1. Provides customized recommendations on reducing carbon footprints, based on user habits in household energy, transportation, and waste.
4. **Air Quality API:**
   1. Retrieves real-time air quality data to inform users of environmental conditions.
5. **Authentication and Authorization:**
   1. Implement role-based access control to manage user permissions.
6. **User Dashboard and Progress Tracking:**
   1. Display calculated footprint, historical data, and recommendation history for user reference.
   2. Graphs and visual aids for easy progress tracking.

**Non-Functional Requirements**

1. **Performance:**

1.1. Ensure the platform operates efficiently, with optimized microservices communication.

1. **Security:**

2.1. Use encryption for sensitive information, ensuring robust protection against unauthorized access.

2.2. Implement regular security patches and updates.

1. **Scalability:**

3.1. Design to support increasing user demands, with scalability through microservices.

1. **Reliability:**

4.1. Maintain high availability and consistent uptime.

4.2. Redundancy for backup to prevent data loss.

1. **Usability:**

5.1. Provide a user-friendly interface with clear language, helpful guidance, and

tooltips.

**Technology Stack**

**Backend:**

* ASP.NET Core Web API in a microservices architecture.
* Each API (Household, Transportation, Waste Management, Authentication, Recommendation, and Air Quality) functions as an independent service.
* Ms SQL Server for data storage.
* JWT Authentication for secure, token-based access control.

**Frontend:**

* Angular for UI

**High-Level Design**

**Components:**

* User Service: Manages user registration, login, and profile management.
* Household Service: Collects household data and calculates footprint.
* Transportation Service: Handles data and calculations related to transportation usage.
* Waste Management Service: Manages waste data and calculates footprint impact.
* Recommendations Service: Generates personalized suggestions for reducing footprint.
* Air Quality Service: Provides real-time air quality data for users.
* Authentication and Authorization Service: Manages user roles and secure access.

**Interactions:**

* User registers or logs in.
* User inputs data for footprint calculations across household, transportation, and waste.
* EcoLife provides footprint calculation and recommendations.
* User views air quality information through the Air Quality Service.
* User tracks progress and updates personal goals.

**Low-Level Design**

**Data Model:**

* User Table: id, name, email, password, role
* Household Table: user\_id, energy\_consumption, gas\_usage
* Transportation Table: user\_id, vehicle\_type, distance
* Waste Management Table: user\_id, recycling\_habits, waste\_amount
* Recommendations Table: id, recommendation\_type, description
* Air Quality Table: id, location, air\_quality\_index

**API Endpoints:**

**1. User Service:**

* `POST /api/users/register`
* `POST /api/users/login`
* `GET /api/users/profile`

**2. Household Service:**

* `POST /api/household/calculate`
* `GET /api/household/history`

**3. Transportation Service:**

* `POST /api/transportation/calculate`
* `GET /api/transportation/history`

**4. Waste Management Service:**

* `POST /api/waste/calculate`
* `GET /api/waste/history`

**5. Recommendations Service:**

* `GET /api/recommendations`
* `POST /api/recommendations/update`

**6. Air Quality Service:**

* `GET /api/airquality/location`

**7. Authentication and Authorization Service:**

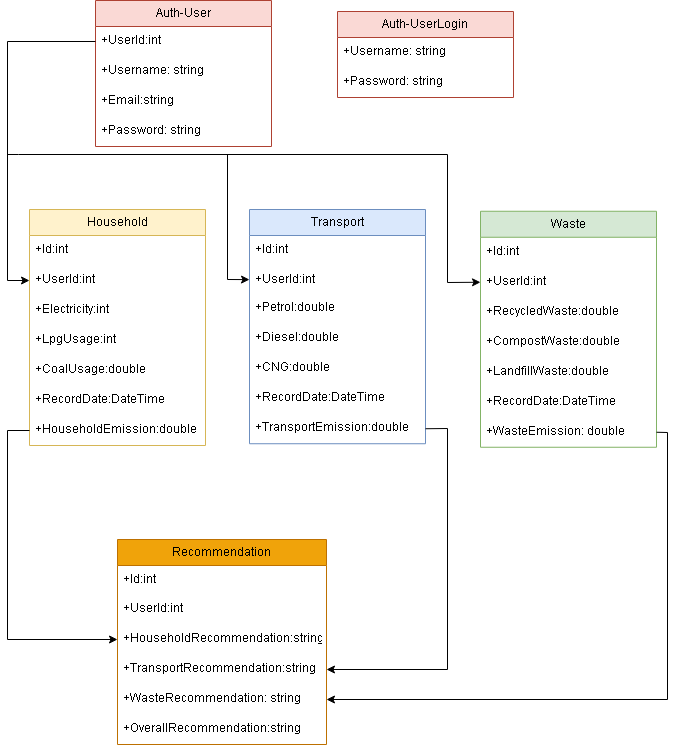
* `POST /api/authenticate/login`
* `POST /api/authenticate/logout`
* `GET /api/authenticate/validate`

**System Architecture:**

**A diagram of a software flow

Description automatically generated**

**Database Diagram:**

****

**Data Flow Diagram**

* User data flows through authentication, then to footprint services for calculations, and results are stored in the relevant tables.
* Recommendations are generated based on past data and usage patterns and are retrieved from the Recommendations Service.

A diagram of a process

Description automatically generated