**Solution**

**Assumption/Constraint**

1. Username or email can be used to login.
2. A GPSs can store several exact locations, a location can recorded by several GPSs.
3. GPS information that stored in the database are device ID (code) and device type.
4. The username and password must at least have 6 characters
5. Token will be stored in the database, if user generates new token the existing token will be changed in the database.

**Technology**

**Backend:**

* Framework : Express.js
* Database : PostgresSQL
* Library :

1. Prisma : ORM library to help querying and migrating tables.
2. Dotenv : Library to load environment variable from .env file.
3. Bcrypt : Library to encrypt and hash password and data.
4. Joi : Data validation library
5. node-jsonwebtoken: Library to encrypt and decrypt JWT Token for authentication.
6. Uuid: Library to generate and handle uuid.
7. Winston : Logger library to record logs.
8. Nodemon: library to automatically restart the application when application code changes

* Structure : Layered Architecture, the application consists of :

1. Controller : layer that handle and validate request, also to return the requested data
2. Service : layer that consists of main logic of the application.
3. Repository : layer that directly interact to the database.
4. DTO : to wrap the requested data

**Frontend**

* Framework : Angular
* Library :

1. PrimeNG: UI Component library for Angular.

* Structure :

1. Component : contains page layout and logic for each interface component
2. Interface : model interface for retrieved data from backend application.
3. Service : utility function that commonly used in the frontend, e.g. fetch API, handle cookie, etc.
4. Guards : middleware to control and manage routing navigation in the frontend application.

**Enhancement**

1. Implement unit testing to ensure the existing functionality if code changes.
2. Implement code linting to a analyse code and stylistic error.
3. Use latitude and longitude data to show location on the map.
4. Application containerization both for backend and frontend using tools like Docker, Kubernetes, etc.
5. Implement more advanced authentication protocol e.g., OAuth 2.0.
6. Implement memory caching for complex calculation e.g., dashboard.
7. Implement logging system to monitor error and issues.

**Scalability**