EPLQ (Efficient Privacy-Preserving Location-Based Query)

Overview

EPLQ is a privacy-preserving system to store, manage, and query points of interest (POIs).
The system has **Admin** and **User** modules with secure authentication and encrypted category data.

System Modules

Admin

- **Register**: Create admin account
- **Login**: Sign in to access dashboard
- **Upload Data**: Upload POI with encrypted category and location

User

- **Register**: Create user account
- **Login**: Sign in to access dashboard
- **Search Data**: Search POIs using category (requires passphrase to decrypt data)

Project Evaluation Metrics

- **Code**: Modular, readable, maintainable
- **Safe**: No harmful operations
- **Testable**: Unit testing possible
- **Maintainable**: Can be extended or modified easily
- **Portable**: Works in any OS environment
- **Database**: Firebase used for user authentication and POI storage
- **Logging**: Console logging with timestamps for all key actions
- **Deployment**: Local deployment using browser & Live Server extension
- **Optimization**: Hashing and AES encryption for secure and efficient queries

Test Cases

Admin Module

Test Case	Action	Expected Result
Admin Login	Enter valid email & password	Redirect to `admin.html` and console logs welcome with timestamp
Invalid Login	Enter wrong email/password	Alert with error message
Set Passphrase	Enter passphrase in form	Alert `Passphrase set!`

Test Case	Action	Expected Result
Logout	Click logout button	Redirect to `login.html`, console logs logout timestamp
Unauthorized Access	Open `admin.html` without login	Alert `You must log in first!` and redirect
POI Upload	Submit form with valid passphrase	Alert `POI added successfully!`, data saved in Firestore
POI Upload without passphrase	Submit form empty	Alert `You must set a Passphrase before uploading a POI!`

User Module

Test Case	Action	Expected Result
User Login	Enter valid credentials	Redirect to `user.html`, console logs login timestamp
Invalid Login	Enter wrong email/password	Alert with error message
Set Passphrase	Enter passphrase in form	Alert `Passphrase set!`
Logout	Click logout button	Redirect to `login.html`, console logs logout timestamp
Unauthorized Access	Open `user.html` without login	Alert `You must log in first!` and redirect
Search POI	Submit valid category with passphrase	Display results in results div
Search without passphrase	Submit search	Alert `You must enter a Passphrase before searching for a POI!`
Search invalid keyword	Submit non-existent category	Display `No results found for: <keyword>`

Miscellaneous

Test Case	Action	Expected Result
Logging	Perform any key action (login, logout, upload, search)	Console logs event with timestamp
Firestore Integration	Add/search POIs	Data correctly saved and retrieved
Invalid Firestore State	Users collection missing or empty	Alerts appropriate error messages

Project Structure

```
public/
 ├─html_docs/
 │   ├─admin.html      # Admin dashboard
 │   ├─index.html       # Landing/Login-Register card page
 │   ├─login.html        # Login page
 │   ├─register.html   # Registration page
 │   └─user.html        # User dashboard
 ├─ js/
 │   ├─ admin.js      # Admin dashboard logic
 │   ├─ auth.js         # Authentication helper functions
 │   ├─ crypto.js     # Encryption/decryption logic
 │   ├─ firebase.js  # Firebase initialization
 │   ├─ index.js      # Card page logic
 │   ├─ logger.js     # Console logging with timestamps
 │   ├─ login.js       # Login page logic
 │   ├─ register.js  # Registration page logic
 │   ├─ user.js       # User dashboard logic
 │   └─ utils.js       # Utility functions
 │
 ├─ styles/           # CSS files for styling
firebase.json
README.md
index.html          # Redirect to the Card page
Usage Instructions.pdf
```

Use **Google Chrome** for best compatibility.

Detailed Project Report

- **Functionality**
  - Admin: Register, Login, Upload POI (encrypted categories)
  - User: Register, Login, Search POI (decrypted using passphrase)
- **Security**
  - SHA256 hashing for query keywords
  - AES encryption/decryption for POI categories
- **Logging**
  - All key actions are logged to console with timestamps
- **Database**
  - Firebase authentication & Firestore for storing users and POIs
- **Test Cases**
  - Defined above in tables
- **Coding Standards**
  - Modular JS files
  - CSS structured with variables
- **Maintainability**
  - Clear project structure
  - Separate JS, CSS, HTML files
- **Portability**

    - Works on any machine using Chrome + Live Server

Notes

- This project is meant for **local deployment**.
- No external hosting is required.
- Before using the system, ensure **users** and **admins** exist in Firebase Authentication.
- Always set a passphrase before searching or uploading POIs.
- All functionality can be verified via console logs for action timestamps and errors.