AddWise Tech Innovations Report

Name of the Intern: **Registration No:**

Shaik. Nelofer   AP23110011470

Internship: Mern Stack Development

Duration of the Internship:

2- Months, May 19 2025 – June 19 2025

Name of the Project: Smart Device and Location Management Authentication

Organization Mentor: Mr. Adwaith

Acknowledgements

I would like to extend my heartfelt gratitude to Addwise Tech for granting me the opportunity to be a part of this enriching learning experience. The exposure to real-time problem-solving, innovative thinking, and technical skill-building offered by the organization has greatly contributed to my personal and professional development.

I am especially thankful to my mentor, Mr. Adwaith, whose continuous support, insightful feedback, and clear guidance played a vital role in shaping the direction of my project. His mentorship not only enhanced my technical knowledge but also helped me understand the importance of precision, responsibility, and creativity in software development.

I am also grateful to the entire team at Addwise Tech for creating a collaborative and encouraging environment, which made this experience both enjoyable and educational.

Introduction:

This project titled **"Smart Device and Location Management Authentication"** was developed during my internship at **Addwise Tech**, under the guidance of my mentor **Mr. Adwaith**. The primary objective of this project is to build a secure, scalable, and role-based web application that allows devices to be registered using unique QR codes and enables location tracking for each registered device.

The application supports three key user roles: Super Admin, Admin, and User. Super Admins can view and manage all registered devices and assign admin roles. Admins are responsible for generating unique 16-digit device codes, creating devices, and generating their corresponding QR codes. Regular users can scan the QR code using their device camera, upload a QR image, or enter the 16-digit code to register the device under their account.

The project implements a secure authentication system that includes registration, login via OTP, and password reset. User credentials are hashed using Bcrypt before being stored in the database. JWT (JSON Web Tokens) are used to protect routes and manage sessions securely. Role-based access control (RBAC) ensures that users only access features permitted to their role.

Once a device is registered, the user can manage its location through Start Tracking, Update Location, or Manually Set Location. The system includes a custom-built API running on a localhost server, which is tested using Postman. When a user selects a QR device, the frontend sends a request to this API, which responds with the latitude and longitude of the device in JSON format. The user can also update this location manually or through system GPS, and the updated coordinates are stored back into the database. These coordinates are then rendered on an interactive map using Leaflet.js, providing a visual representation of the device's current position.

The backend of the application is developed using Node.js and Express.js, while the frontend is built using HTML, CSS, and JavaScript. Postman was used extensively for testing endpoints and verifying API responses during development. MongoDB is used as the database to store user data, registered devices, QR codes, and associated location data for each device.

Flowchart Description:

The flowchart illustrates the complete workflow of the QR-Based Device Authentication and Location Tracking System, starting from user authentication to live location tracking.

1. Start: The system begins with user options for Login, Register, or Forgot Password.

2. Registration Flow: New users register by entering their details. After successful registration, they are redirected to the login page.

3. Login Flow: Existing users can log in using email and OTP or recover their credentials via the Forgot Password option.

4. Role Checking: Upon successful login, the system checks the user's role — Super Admin, Admin, or User — and redirects accordingly.

5. Super Admin Role:

   o Views all registered devices and manages admin accounts.

6. Admin Role:

   o Generates a 16-digit unique code for each device.

   o Creates the device and generates a QR code.

   o Saves and displays QR codes on the admin dashboard for further usage.

7. User Role:

   o Can add devices by scanning QR codes, uploading QR images, or entering the 16-digit code manually.

   o System extracts and validates the code.

   o If valid, the device is added to the user's "My Devices" section; if not, an error message is displayed.

8. Location Options:

   o Once a device is added, the user can choose from Start Tracking, Update Location, or Manually Set Location.

   o The chosen location data is stored or updated for that device.

9. API Integration:

   o When a user selects a device, a request is sent to a localhost API endpoint:

    http://localhost:5000/api/qrcodes/qr/location

    o The API responds with a JSON object containing latitude, longitude, and optionally location name.

10. Map Display:

    o Using Leaflet.js, the coordinates are visualized on an interactive map with a marker.

   o Location history, including start and end date, is shown to provide tracking context.

11. End: The flow concludes after the map and location data are displayed.
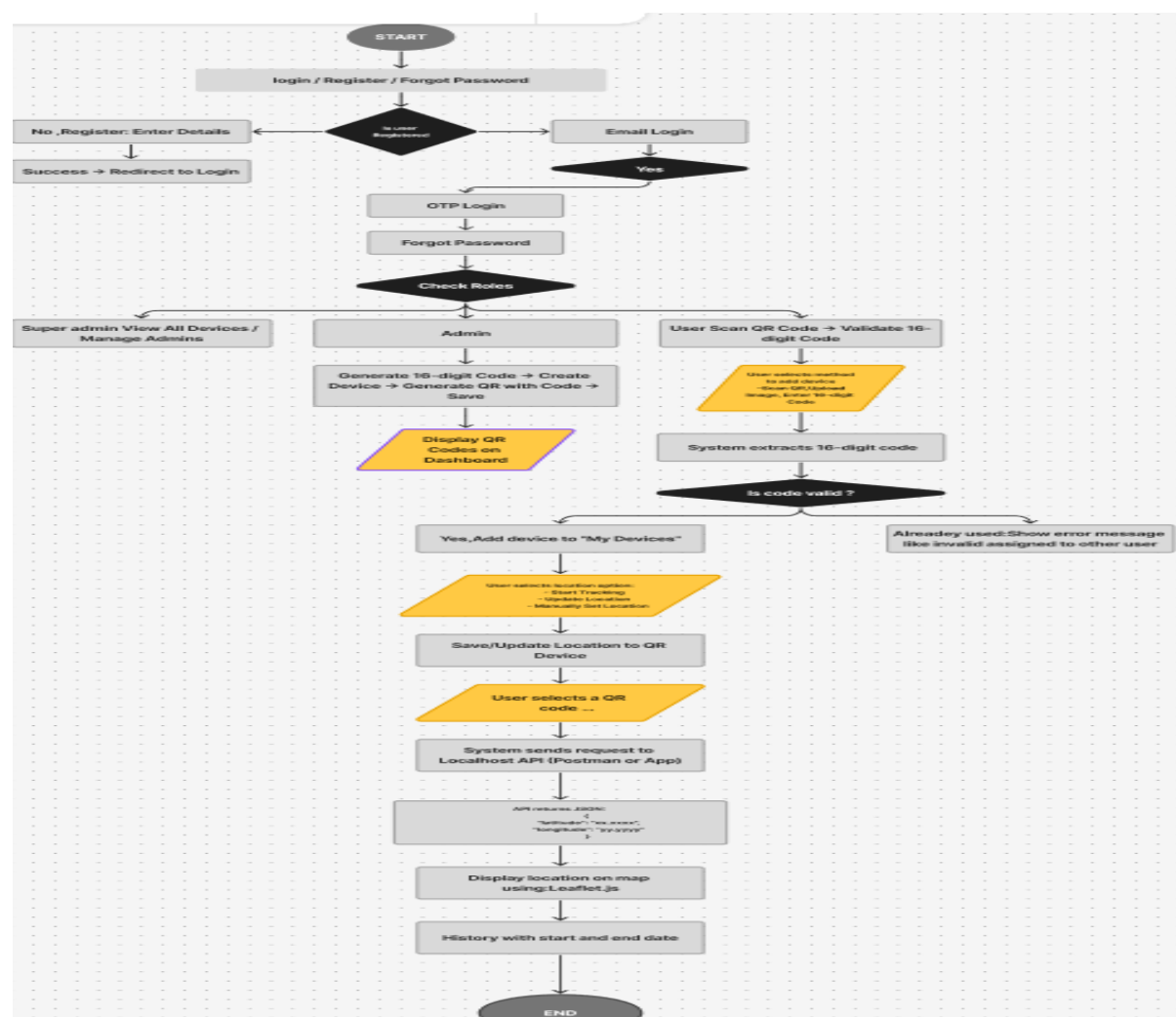
Features

Authentication & Security:

- Login / Signup / Forgot Password functionality.

- OTP-based Login for enhanced security.

- Passwords hashed using Bcrypt before storing in the database.

- JWT (JSON Web Token) implementation for secure login sessions.

- Protected routes for user/admin-only features.

- Unauthorized users are blocked from accessing restricted pages.
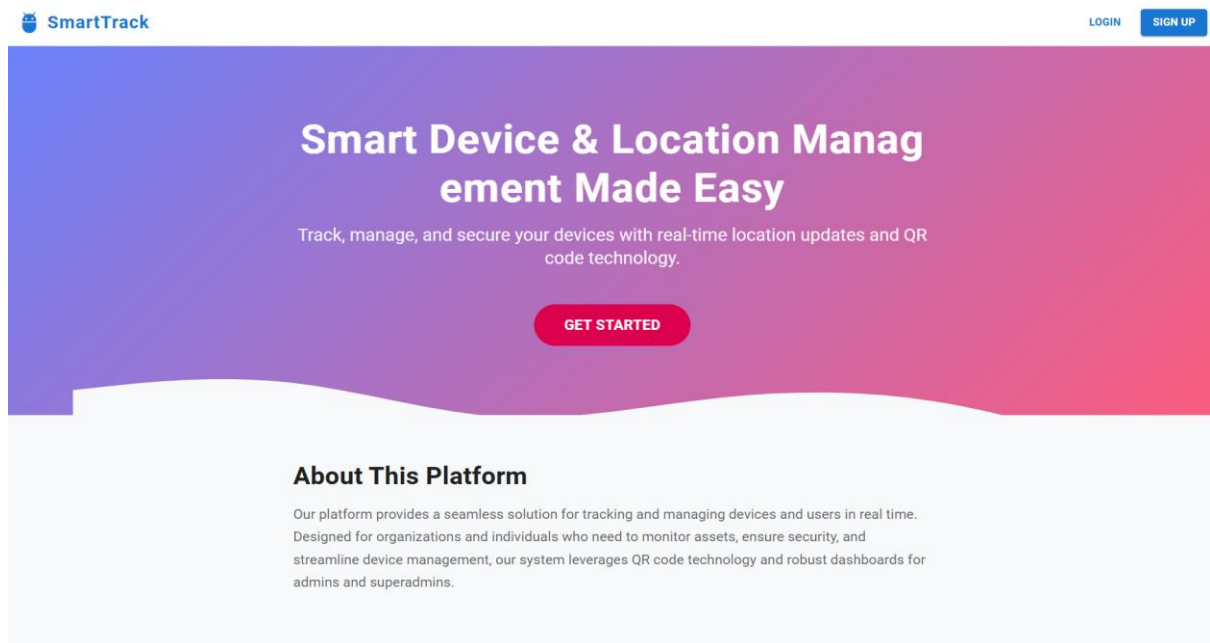
Role-Based Access Control (RBAC):

- **Three roles:**
  - Super Admin
  - Admin
  - User
- Each role sees a different dashboard UI based on permissions**.**

Dashboards and Their Features

Super Admin Dashboard

- View all devices registered in the system.

- View and manage admin accounts.

- Full system overview (user, admin, devices).

Admin Dashboard

- Generate unique 16-digit codes for each device.

- Create new device entries.

- Generate and download QR codes for each device.

- See all devices they created and manage their status.

User Dashboard

- View "My Devices" – devices linked by scanning or entering code.

- Add device through:

- QR Code Scan

- QR Image Upload

- Manual 16-digit code entry

- View detailed info of each device.

Device Management:

- Admins generate and assign **unique QR codes** to each device.

- Users can **scan, upload, or manually enter** 16-digit QR codes.

- Validation of codes:

  - Only unused and valid codes can be added.

  - Invalid/used codes show error messages.

Location Tracking & Mapping:

- After adding a device, users get **3 location options:**

  - **Start Tracking** (from system/mobile GPS)

  - **Update Location** (manually refresh coordinates)

  - **Manually Set Location** (enter lat/lng)

- Location data is **saved/updated** to each device entry in the database.

Distance Statistics

0.00	71904.74	0	15
Total User Distance (km)	Total Device Distance (km)	Online Users	Total Devices

Enter Device QR Code

SEARCH

evice Locations Map   Last updated: 9/7/2025, 5:59:12 pm   **SHOW ALL PATHS**   **Total Distance: 59730.11 km**

Leaflet | © OpenStreetMap contributors
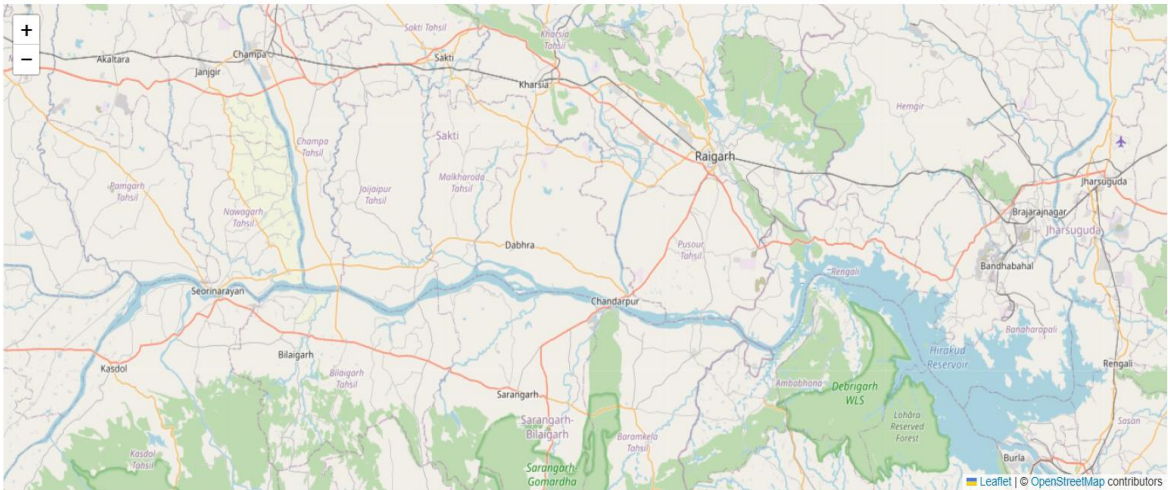
POST   http://localhost:5000/api/qrcodes/3500988322071314/location   **Send**

Params   Authorization   Headers (9)   **Body** ●   Scripts   Settings   **Cookies**

○ none   ○ form-data   ○ x-www-form-urlencoded   ● raw   ○ binary   ○ GraphQL   JSON ∨   **Beautify**

```
1  {
2      "latitude": 39.9042,
3      "longitude": 116.4074,
4      "locationName": "Beijing"
5  }
```

Body   Cookies   Headers (8)   Test Results   **200 OK** • 1.18 s • 1.54 KB   Save Response

{} JSON ∨   ▷ Preview   Visualize ∨

```
1  {
2      "message": "Device location updated",
3      "qrCode": "3500988322071314",
4      "latitude": 39.9042,
5      "longitude": 116.4074,
6      "locationName": "Beijing",
7      "lastUpdated": "2025-07-09T12:33:43.160Z",
8      "locationHistory": [
9          {
10             "latitude": 16.3426,
11             "longitude": 80.4413,
12             "locationName": null,
13             "lastUpdated": "2025-07-08T09:23:31.561Z",
14             "_id": "686ce3936380911b4b50761a"
15         },
16         {
17             "latitude": 22.5726,
18             "longitude": 88.3639,
19             "locationName": "Kolkata",
20             "lastUpdated": "2025-07-08T09:23:45.330Z",
21             "_id": "686ce3a16380911b4b50761e"
```
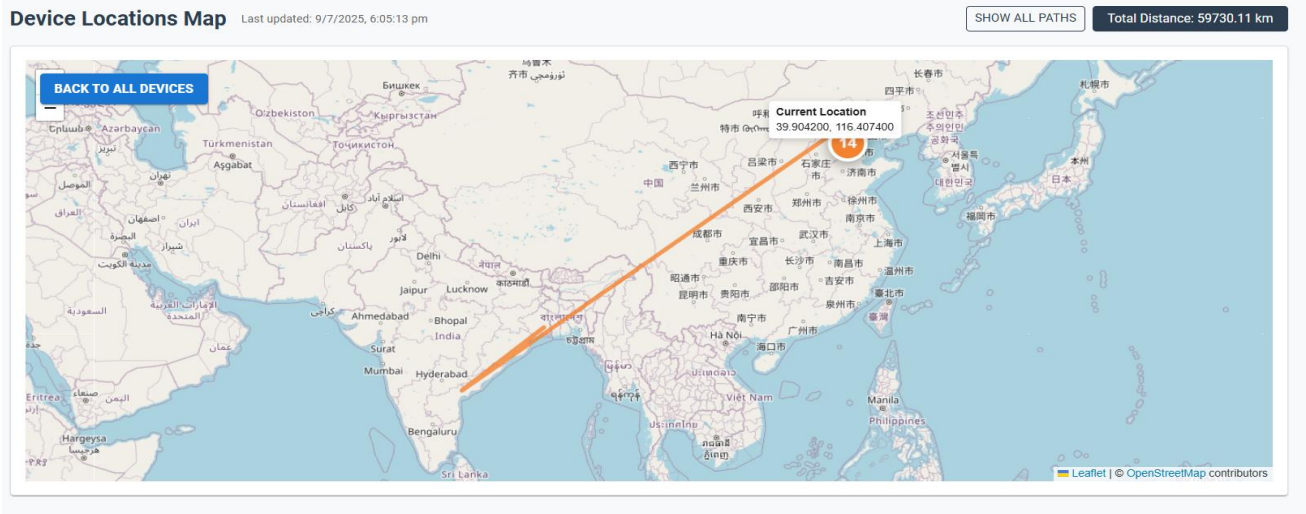
Uses a custom API endpoint:

http://localhost:5000/api/qrcodes/qr/location

Accepts device ID/code

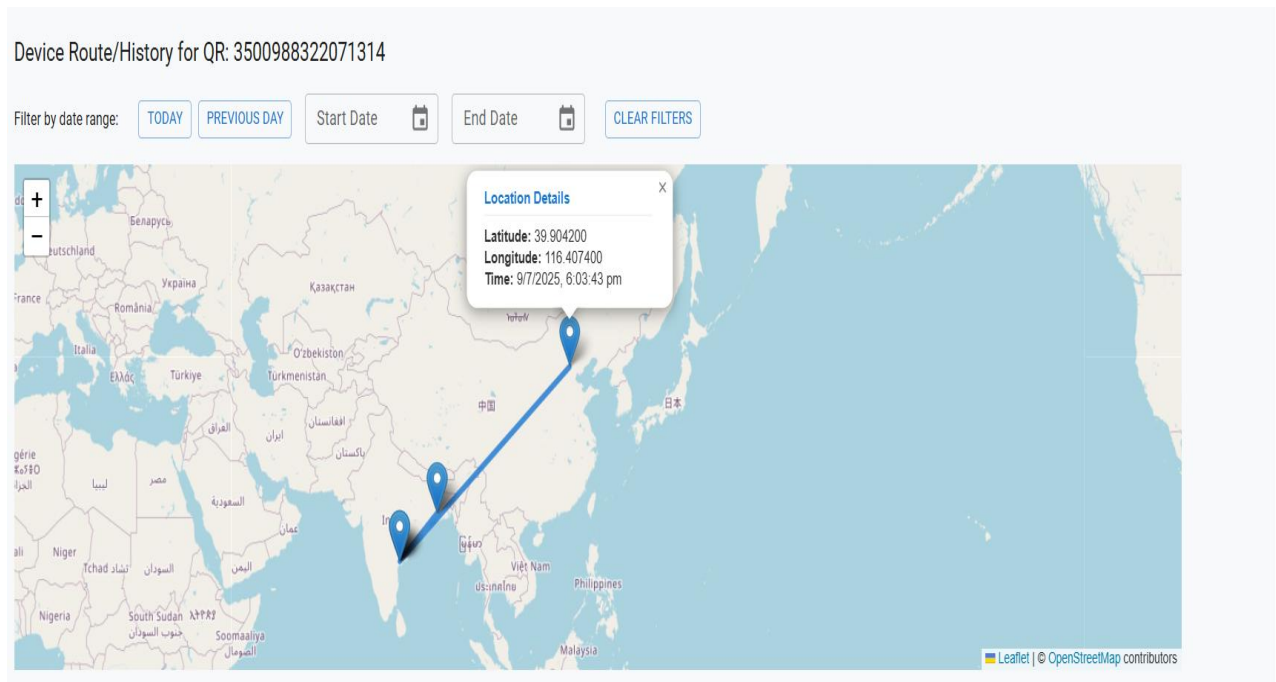**Returns latitude and longitude in JSON format:**

```
{
  "latitude": "17.3850",
  "longitude": "78.4867",
  "locationName": "Hyderabad"
}
```

Data is fetched on selection and rendered on a map using Leaflet.js:

- o Device's current position shown with a marker.
- o Map auto-centers on device location.

Tracking History (Start and End Time)

- Users can view location tracking history of their devices.

- Shows start date, end date, and previous tracking sessions (if implemented).

Device Route/History for QR: 3500988322071314

Filter by date range: TODAY | PREVIOUS DAY | Start Date | End Date | CLEAR FILTERS

Location Details

Latitude: 39.904200
Longitude: 116.407400
Time: 9/7/2025, 6:03:43 pm

Technology Stack

Frontend: HTML, CSS, JavaScript

- Backend: Node.js, Express.js

- Database: MongoDB

- API Testing: Postman

- Mapping Library: Leaflet.js

- Security: JWT for auth, Bcrypt for password hashing

Conclusion:

The QR-Based Device Authentication and Location Tracking System successfully integrates secure user authentication, role-based access control, and real-time device tracking into a unified web application. By assigning specific roles such as Super Admin, Admin, and User, the system ensures that functionalities are organized and controlled according to user responsibilities. Admins can efficiently generate and manage device QR codes, while users have flexible options to add devices and track their locations accurately.

The inclusion of a custom-built API tested via Postman, combined with JSON responses containing latitude, longitude, and location name, provides a robust and scalable backend for tracking. The frontend, developed with HTML, CSS, and JavaScript, integrates seamlessly with Leaflet.js to visualize device locations on a map. Security has been a top priority throughout the project, with JWT-based route protection and Bcrypt-hashed passwords ensuring data integrity and user safety.

This project has provided hands-on experience with full-stack web development and practical problem-solving related to QR handling, geolocation, API interaction, and secure system design. It is a strong example of how modern web technologies can be combined to build real-world applications with meaningful impact.