  **HACKATHON PHASE-I**

**COLLEGE CODE : 6128**

**COLLEGE NAME : Varuvan Vadivelan Institute of Technology**

**DEPARTMENT : B.Tech IT**

**STUDENT NM-ID : E23521BCB6E8A4581294E675F778E543**

**ROLL NO : 612823205003**

**DATE : 07-10-2025**

**TECHNOLOGY : Front End/Node JS**

**PROJECT NAME : Real-Time Location Tracking App**

**SUBMITTED BY:**

**Name : ALLAUDDIN K {TL}**

**Mobile No : 9566581173**

**MEMBERS:**

**Name : MARTINDURAIRAJ M**

**Mobile No:9080377312**

**Name : Ajay K**

**Phase 1: Real-Time Location Tracking App**



* **Project Overview & Objectives**

**Problem Statement:**

* **Users need a simple way to track multiple participants or vehicles in real-time.**
* **Existing solutions are heavy, require login, or manual refresh, making live tracking slow.**

**Key Features**

* **Real-time location tracking of all connected users.**
* **Uses geolocation API in the browser to get live coordinates.**
* **Broadcasts location data via Socket.io to all clients.**
* **Display users’ positions on a Leaflet.js map.**
* **Smooth map panning and marker updates.**
* **Handles user disconnections by removing their marker.**

**Expected Outcome:**

**A working web app where multiple users can see each other moving on a map in real-time without refreshing.**

A map of a city

AI-generated content may be incorrect.

* **Technology Stack & Environment Setup**

**Backend:**

* Node.js → server-side runtime
* Express.js → HTTP requests and static file serving
* Socket.io → real-time, bidirectional communication

**Frontend:**

* HTML, CSS, JavaScript → interface and interaction
* Leaflet.js → interactive map
* Socket.io-client → send/receive real-time data

**Tools & Environment:**

* EJS → templating engine for map page
* CSS → full-page map styling
* OpenStreetMap tiles → map display
* Local server → port 3000
* **API Design & Data Flow**

**Socket Events:**

* send-location → client sends {latitude, longitude}
* recieve-location → server broadcasts location with unique socket id
* user-disconnected → server notifies clients to remove marker

**HTTP Routes:**

* GET / → renders index.ejs (map page)

**Data Flow:**

* User opens webpage → browser connects to Socket.io
* Browser sends location updates via send-location
* Server receives → emits recieve-location to all clients
* Map updates markers or pans smoothly
* On disconnect → marker removed via user-disconnected
* **Front-End UI/UX Plan**
* Single-page interface showing full-screen interactive map
* Users’ markers update in real-time without page reload
* Smooth panning ensures new locations are always visible
* Minimal UI for simplicity and performance
* **Development & Deployment Plan**

**Development Steps:**

* Setup Node.js + Express project
* Integrate Socket.io for real-time communication
* Implement frontend map with Leaflet.js
* Connect geolocation API to send location data
* Handle multiple users and disconnect events
* Style map for full-screen view

Deployment:

* Run locally for testing
* For public use → deploy on ngork with HTTPS for geolocation
* Map tiles via OpenStreetMap or Google Maps API