CSCI-4250-002 Software Engineering Project Design Document

Project: Student Location Tracker
Repository: https://github.com/Jakehebert201/CSCI4250-ScrumProject
Version: 1.0
Date: Nov 24, 2025
Authors: Group 7 (Shane Austin, Ishwari Patel, Jacob Hebert, Deep Desai, Eloghosa Erhabor)

Purpose

This document aims to provide a clear blueprint for development, testing, and future maintenance of the student tracker application. It is intended for developers, testers, project managers, and any stakeholders who need to understand how the system is put together.

Scope

The Student Location Tracker is a mobile/web system that allows authorized users (e.g. instructors/faculty) to view the real-time or recent locations of students (with their consent) on a map. Key features include:

- Student device location reporting
- Real-time/periodic location updates
- Map-based visualization of student locations
- Role-based access: students and instructors
- Data privacy/compliance controls
- Notification/alert system (beta)

The system consists of a mobile client (for students to report location), a dashboard for instructors to manage their classes and review location reports/clock-in's, a backend service, and a database.

High-Level System Overview

The system is built using a client-server architecture:

- Client (mobile) - Students run the mobile app which periodically captures their GPS (Global Positioning System) location and sends it to the server.
- Backend API - A RESTful service that receives location updates, stores them, and provides endpoints for retrieving data.
- Database - Stores users, classes, enrollments, locations, clock events, and notifications.
- Dashboard - For instructors to view student locations on a map in real time or historically.
- Notification Service (beta, non-functional) - To alert when certain conditions are met (e.g., student clocks in/out or leaves a designated area)

Functional Requirements

Here are the major functional requirements, mapped to features:

1. User Authentication & Authorization
    a. Students and instructors must log in.
    b. Instructors have higher privileges (view all, manage classes, etc.).

2. Location Reporting
    a. Student app collects GPS periodically (or in near real time).
    b. Sends location updates to the server securely.

3. Location Storage and History
    a. Backend stores each location update with timestamp.
    b. Ability to query recent location or historical data.

4. Map Visualization (Dashboard)
    a. Web dashboard shows student positions on a map.

Non-Functional Requirements

- Performance: The backend should handle frequent location updates (e.g., every 30 seconds for many users) without degrading.
- Scalability: Support for scaling hundreds or thousands of students.
- Security:
    - Secure storage of location data.
    - User token-based authentication.
- Availability: High availability for the dashboard; backend should be resilient.
- Privacy compliance: Ensure data is only stored and accessed per consent; data retention policies.

Data Model/Entities

Here is a simplified ER model:

- User
    - User_id (PK)
    - Name
    - Email
    - Role (enum: STUDENT, INSTRUCTOR)
- LocationRecord
    - Record_id (PK)
    - User_id (FK - User)
    - Latitude (decimal)
    - Longitude (decimal)
    - Timestamp (datetime)

Data Storage Considerations

- Time-series data: Location updates are time-stamped, so consider optimizing DB for time-series or indexing on timestamp.

- Retention policy: Define how long location history is kept (e.g., 30 days, semester, etc.).
- Privacy: Access control on who can read which location data.

Mobile App

- Location Service Module
    - Uses device GPS/location provider
    - Configurable interval (e.g., every 30 seconds or on movement)
    - Can run in background (depending on OS)
- Auth Module
    - Login screen
    - Token storage/refresh
- Networking Module
    - API client to send location data (REST)
    - Handles retries, batching if needed
- Settings Module
    - Allow students to pause tracking
    - Show history (optional)

Backend API

- AuthService
    - Login/logout
    - Verify tokens
- LocationService
    - Endpoint to receive location updates (POST /locations)
    - Validate data, reject invalid or malicious requests
- QueryService
    - Endpoint for dashboard to query recent or historical location (GET /locations?user=&range=)
- AlertService
    - Trigger alerts to ping location or when manually broadcast
    - Store alert events

Dashboard (Web)

- Map UI Component
    - Integrate with mapping library (Leaflet)
    - Plot student markers
- Alert Management UI
    - View alert history
- User Management
    - List users (students) and metadata

User Interface Design (Wireframes) - Mobile (Student)

- Login Screen

    - Fields: email/password or SSO button
- Main Screen
    - Student ID and metadata, classlist, date of account creation, and account type
    - Current location on map and location history
    - Clock-in/clock-out buttons
    - Dark mode toggle
    - Logout button

User Interface Design (Wireframes) - Web (Instructor Dashboard)

- Login Screen
    - Email/password or SSO
- Map View Screen
    - Map showing student markers
    - Option to show live or historical locations
- Alert History Screen
    - Table of updates (e.g., location, class/enrollment, and manually broadcast emergencies) with options to filter by each (or by unread)
    - Buttons to test the system, send broadcast alerts, mark all as read, or clear all

Security and Privacy Considerations

- Encryption: Use HTTPS/TLS for all communication.
- Authentication: Use secure token-based auth with expiration and refresh.
- Data Minimization: Only collect necessary data; aggregate/anonymize for some views.
- Retention Policy: Define how long location data is stored, and delete older data according to policy.
- Logging and Auditing: Log access to sensitive data (who viewed which student's location).

Testing Strategy

- Unit Tests:
    - Backend: test AuthService and LocationService logic
    - Mobile: mock GPS, test clock-in/clock-out
- Integration Tests
    - End-to-end: mobile app, API, DB, dashboard
    - Location/manually broadcast alert scenarios
- Security Testing
    - Test login/logout with email and password vs. SSO
    - Check data access from different roles

Maintenance and Future Work

- Maintenance

    - Monthly review of logs and alerts
    - Database cleanup/archival of old data
    - Security audits regularly
- Future Enhancements
    - Add mobile push notifications when a student enters/exits a geofenced zone
    - Provide heatmap views on the dashboard (density of student locations)
    - Implement offline mode (store data locally, sync when online)
    - Add anonymized analytics (e.g., how many students congregate in certain areas)