Detailed Design Documentation

---

1. Overview

The Student Tracker Application is a Blazor/.NET 8 application designed to track student check-ins/check-outs, track realtime location, and support private messaging between users. The system stores user information, location logs, and messages in an SQL database.
This document describes:

- System architecture
- Domain model
- Repositories
- Class responsibilities
- UML diagram

2. System Architecture

Frontend

- Blazor Server (.NET 8)
- UI components for:
    - Student check-in/out
    - Messaging center
    - Admin/teacher dashboards
    - Relationship management

Backend

- C# domain models
- Repository pattern for database access
- Entity Framework Core (SQL Server)

Data Storage

- EF Core DbContext (ApplicationDbContext)
- Tables:
    - Users
    - Students (inherits from Users)
    - CheckInLogs
    - Messages
    - StudentTeachers

3. Domain Model Design

3.1 User Class

Represents any user in the system.
Attributes

- Id
- FirstName
- LastName
- Email
- Password
- UserType (e.g., Student, Teacher, Admin)

Responsibilities

- Acts as a base class for Student
- Primary identity object for relationships, messages, and check-ins

3.2 Student Class

Inherits from User and adds additional tracking attributes.
Attributes

- Location
- CheckInTime
- CheckOutTime

Responsibilities

- Customize user entity specifically for students
- Provide constructors to convert a User to Student

3.3 CheckInLog Class

Represents a check-in and check-out event.
Attributes

- Id
- UserId
- CheckInLatitude / CheckInLongitude
- CheckInTime
- CheckOutLatitude / CheckOutLongitude (nullable)
- CheckOutTime (nullable)
- WithinRange (flag for 5-mile rule)
- Notes
- User navigation property

Responsibilities

- Store geolocation and timestamps for student check-in/out
- Enforce 500 feet check-out validation

3.4 Message Class

Represents a private message between two users.

Attributes

- Id
- SenderId
- RecieverId
- MessageContent
- ReadByReciever
- DateTime

Responsibilities

- Allow communication between students, teachers, or admins
- Provide structure for inbox, outbox, and message persistence

3.5 StudentTeacher Class

Join table for a many-to-many relationship between students and teachers.

Attributes

- Id
- StudentId
- TeacherId

Responsibilities

- Connect users of type Student to users of type Teacher
- Support multi-teacher and multi-student registrations

4. Repository Design

Each repository abstracts EF Core interactions, promoting cleaner separation of concerns.

4.1 DbCheckInRepository

Handles all CheckInLog operations.

Methods

- CreateCheckInAsync(CheckInLog log)
- GetActiveCheckInAsync(int userId)
- UpdateCheckOutAsync(CheckInLog log)

- GetLogsByUserIdAsync(int userId)

Responsibilities
- Manage check-in to check-out lifecycle
- Return only active logs (where check-out is null)
- Support ordered log retrieval

4.2 DbMessageRepository

Handles CRUD for messages.
Methods
- CreateMessageAsync()
- DeleteMessageAsync()
- ReadAllMessagesByUserIdAsync()
- ReadMessageAsync()
- UpdateMessageAsync()

Responsibilities
- Allow inbox/outbox querying
- Update read state
- Remove messages

4.3 DbStudentTeacherRepository

Handles student-teacher linking.
Methods
- CreateStudentTeacherRelationAsync()
- DeleteStudentTeacherRelationAsync()
- GetStudentsByTeacherIdAsync()
- GetTeachersByStudentIdAsync()
- ReadStudentTeacherByIdAsync()

Responsibilities
- Maintain many-to-many relations
- Fetch related IDs for dashboards

4.4 DbUserRepository

Handles User management.
Methods
- CreateUserAsync()
- DeleteUserAsync()
- ReadAllUsersAsync()
- ReadUserAsync()

- UpdateUserAsync()
- GetUserByEmailAndPasswordAsync()

Responsibilities
- Authentication lookup
- General user CRUD
- Admin-level account management

6. UML Class Diagram

- Click Here