CS7CS3 Advanced Software Engineering Group Project

Requirements/Use Cases

Project Name: *SafeRoute*

Group: ***4***

Name	***Student ID***
Deepika Nag	24360112
Haiqing Ji	*25335332*
Yuanchen Fan	*25332247*
Jai Nagle	*23372127*
Maolan Mierkamili	*25356575*
Sarthak Srivastava	*25338417*
Sayan Mondal	*24377372*
Tao-Sen Chang	*24371560*
Xiaoxuan Duan	*24338892*
Ziwei Zhao	*24332500*

Table of Contents

1. Use Case Diagram

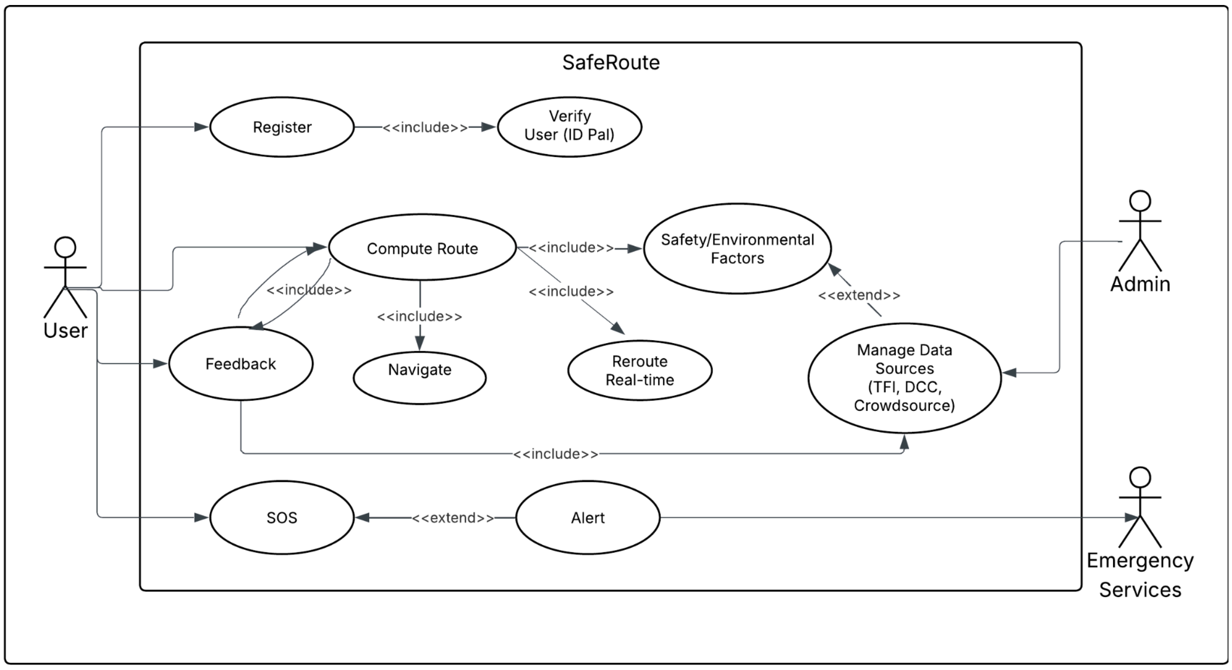

Fig 1: Overall use-case diagram of SafeRoute application

2. SafeRoute User Flows

2.1 Use Case Name: User Registration

2.1.1 Description

The goal of this use case is to enable new users to securely create an account on the SafeRoute platform. After registration, users will be able to access safety features such as safe route navigation, SOS alerts, and feedback-based recommendations.

The system will be

- Able to verify user credentials
- Support ID-Pal for verification
- Ensure user data privacy protection

2.1.2 Actors

- Users

2.1.3 Triggers and Inputs

Launch Application

- Trigger
  - User downloads and launches the SafeRoute application
- Input

- Taps on Register.

Registration

- Trigger
    - User fills the registration form
- Input
    - User enters Full Name, Username, Email Address, Mobile Number, Password, Profile Picture (Optional)

2.1.4 Flow of Events

Basic Flow			
User		System	
1	User launches the app and selects "Register"		
2	User fills out the required details – (Name, Profile Picture (Optional), Username, Email, Mobile, Password) & Captcha for rate-limiting		
		3	System validates the input fields for format and completeness, and ensures rate-limiting and DDoS protection.
		4	System checks for existing accounts with the same email / phone number.
		5	System initiates the ID-Pal verification
		6	Upon successful verification, the system encrypts and stores user data in the database.
		7	System sends a confirmation email/SMS to the user.
8	User receives confirmation and is redirected to the **Login** page.		

Alternative Flow 1: User enters invalid input			
User		System	
1	User enters an invalid input in the fields present in the registration form		
		2	The system detects missing or invalid fields (e.g., weak password, invalid email).

		3	The system prompts the user to correct the input before proceeding.
4	User receives an error message that the respective input field is incorrect		

Alternative Flow 2: Duplicate account			
User		System	
1	User enters duplicate phone number or email address		
		2	The system compares the original database and finds a duplicate phone number or email address, and suggests users to try another phone number or email address.

Alternative Flow 3: ID Pal verification failed			
User		System	
1	User relaunches the application with pending verification		
		2	IDPal verification is/was incomplete or times out.
		3	The system notifies the user to retry verification or contact IDPal support.

2.1.5 Special Requirements

- Handles user authentication using external authentication provider, authorization, and identity management through secure protocols such as OAuth 2.0 and OpenID Connect
- Dependency on IDPal services for user registration
- A primary data store for persistent user details, including profiles (PostgreSQL).
- Password Security: All passwords must be hashed and salted before storage. Communication with IDPal and the form submission process must use secure (HTTPS) protocols.
- GDPR: The system must comply with GDPR. A link to the Privacy Policy must be provided on the registration form.

2.1.6 Platform

Natively compile it to Objective-C for iOS and gradle for Android

- iOS
- Android

- IDPal

2.1.7 Preconditions

- The user has installed the SafeRoute app.
- The device must have an active internet connection.
- IDPal verification services app must be downloaded.
- Database services must be up and running.

2.1.8 Postconditions

- A new verified user account is successfully created and stored in the system.
- The user receives confirmation and can now log in.
- System updates the active user registry for future interactions.

2.2 Use Case Name: Login & Forgot Password

2.2.1 Description

The goal of this use case is to securely authenticate users and grant them access to the SafeRoute application. In cases where users forget their credentials, the alternate flow provides a secure method to recover their username or reset their password, ensuring uninterrupted access to the system.

2.2.2 Actors

- Users
- Supporting Actors
  - Authentication Provider
  - Caching System

2.2.3 Triggers and Inputs

Login

- Triggers:
  - The registered user initiates the login process by entering their credentials and tapping the “Login” button.

- Inputs:
  - Email ID and Password entered by the user in the login form.
  - Login button action to submit credentials for authentication.

Forgot Password

- Triggers:
  - The user is unable to log in due to forgotten or incorrect credentials and taps the “Forgot Password” button.

    - Tapping on the magic link to redirect to the reset password page.

- Inputs:
    - New Password and Confirm Password fields for setting updated credentials.

2.2.4 Flow of Events

Basic Login Flow

Basic Flow (Login Flow)			
User		System	
1	The user enters their email ID and password.		
2	The user taps the “Login” button.		
		3	The authentication provider receives a request to authenticate the user using the provided credentials.
		4	The system validates the user’s credentials against the records in the database and also checks if the user is activated with IDPal
		5	Upon successful validation, the system generates a JWT token with a defined time-to-live (TTL) to maintain the user’s authenticated session.
6	The JWT token is stored securely on the client side.		
7	The user is successfully logged in and redirected to the homepage.		

Negative Login Flow 1 (Forgot Password)

Alternative Flow 1 (Forgot Password)			
User		System	
1	The user enters a correct email ID but an incorrect password.		
2	The user taps the “Login” button.		
		3	The authentication provider receives a request to authenticate the user using the provided credentials and validates the credentials against the database.

		4	Validation fails, and the system returns an error message indicating invalid credentials.
5	The user receives an error message stating that the credentials are incorrect.		
6	The user taps the "Forgot Password" button.		
		7	The system sends an email to the user containing a password reset (magic) link.
8	The user taps the password reset link in the email.		
		9	The system prompts the user to create a new password.
10	The user enters a new password and confirms the new password and saves it.		
		11	The system updates the user's credentials in the database.
12	The user receives a confirmation message and an email indicating that the password has been successfully reset.		

Negative Login Flow 2 (Incorrect Email)

Alternative Flow 2 (Incorrect Email)			
User		System	
1	The user enters an unmapped email ID and password		
2	The user taps the "Login" button.		
		3	The authentication provider receives a request to authenticate the user using the provided credentials.
		4	The system validates the user's credentials against the records in the database.
		5	Validation fails, and the system returns an error message indicating email does not exist.
6	The user receives an error message stating that the email does not exist.		

7	The user is successfully redirected to the Register page		

2.2.5 Platform

Natively compile it to Objective-C for iOS and gradle for Android

- iOS
- Android
- Authentication Provider (Auth0)

2.2.6 Special Requirements

The login use case depends on several external systems to ensure secure, efficient, and reliable authentication:

- Handles user authentication using external authentication provider, authorization, and identity management through secure protocols such as OAuth 2.0 and OpenID Connect.
- Use a form of caching to improve login performance by temporarily storing session data and frequently accessed user information for faster retrieval.
- A primary data store for persistent user details, including profile.

2.2.7 Preconditions

- The user must already be registered in the system.
- The user provides the necessary credentials (email/password, API key, OAuth token, etc.) for authentication.
- The device must have a working network connection.
- The authentication service must be responsive.

2.2.8 Postconditions

- Session integrity post authentication is ensured
- SSL certificates, API keys, or tokens must be valid and not expired
- The UI or backend now recognizes the user as logged in, enabling authenticated features such as saving routes, editing layers, or syncing data.

2.3 Use Case Name: Provide a route from origin to destination

2.3.1 Description

The goal of this use case is to calculate the route based on the origin, destination and safety/environmental factors. The calculated route will prioritize user safety based on the factors provided by the data source and routing algorithm.

The system will be able to:

- Calculate a route based on origin and destination inputs.
- Integrate and analyze real-time and static safety data from data sources.
- Provide clear turn-by-turn navigation guidance to the user.

2.3.2 Actors

- Users
- Supporting Actors
  - Routing Algorithm
  - System Admin (Data sources)

2.3.3 Triggers and Inputs

Finding the route

- Triggers:
  - The user opens the safety route mapping application on their device.
  - The user types the starting point and the intended destination into the corresponding search fields.
- Input:
  - The user enters the origin address into the designated input field.
  - The user enters the destination address into the designated input field.

Selection of the route

- Trigger
  - The user selects their preferred route from multiple safe route options generated by the system based on safety factors such as crime rate, lighting, and traffic conditions.
- Input
  - The user taps the “Directions” button to view the detailed navigation for the selected route.

Start the navigation

- Trigger
  - The user selects the preferred route from the available safe route options displayed on the map.
  - Once the route is selected, the application initiates turn-by-turn navigation to guide the user to their destination
- Input
  - The user taps the “GO” or “Start” button to begin navigation along the selected route.

2.3.4 Flow of Events

Basic Flow			
User		System	
1	User types in origin		
2	User types/selects Destination		
		3	System queries the environmental data from the data source pertaining to the origin and destination
		4	System compares all routes with the routing algorithm to calculate the safe route
		6	The resulting route is displayed to the user
7	User starts the navigation by tapping the button “Go”		
		8	System begins to guide the route
9	User reaches the destination		
		10	System ends the navigation

Alternative Flow 1(Destination and origin is same)			
User		System	
1	User types the origin		
2	User tries to type the same destination		
		3	System blocks the user from typing the same destination

Alternative Flow 2(User types the destination not the origin)			
User		System	
1	User types the destination without specifying an origin		
		2	System assigns the current location as the origin and computes the route
		3	System displays the route from Origin to Destination

Alternative Flow (User deviates from the path)			
User		System	
1	User deviates from the navigation route		
		2	System follows the Re-routing Flow

2.3.5 Special Requirements

To ensure real-time responsiveness and situational accuracy, the application will implement live data integrations through multiple external APIs and open data frameworks.

- **TFI Live API**: Provides live updates on public transportation timetables, delays, and service alerts across the network.
- **Google Business API**: Supplies contextual business and point-of-interest data, enabling the system to identify nearby safe or essential locations for example police stations, hospitals, or well-lit establishments.
- **DCC (Data Communications Centre)**: Facilitates secure communication and synchronization of dynamic datasets, enhancing data reliability for decision-making.
- **OpenRouteService API**: Supports route calculation and optimization based on live geospatial data, offering the most efficient and safest routes available in real time.

2.3.6 Platform

- External Mapping Platform
- OpenRouter Routing Engine

2.3.7 Preconditions

- Have an active internet connection
- Have the respective permissions (location, notification, speaker)
- Database should be up and running

2.3.8 Postconditions

- User reach the destination using the specified path

2.4 Use Case Name: Re-route while navigating

2.4.1 Description

The goal of the system is to actively re-route the predicted path from origin to destination based on external parameters when a user is navigating. These external parameters could be anything from a closed business to a faulty street light to an inactive public transport service when the user is navigating.

2.4.2 Actors

- Users
- Supporting Actors
  - System Admin (Datasources, City Council, Crowdsourced User Data)
  - Emergency Responders
  - Map Routing Engine

2.4.3 Triggers and Inputs

- The user deviates from the planned route.

- The system detects a blockage, unsafe route possibility, SOS alert or road closure ahead.
- The user manually requests a new route (e.g., “Avoid tolls” or “Change destination”).
- The user receives a tip from crowdsourced data/user feedback that leads them to a new route.

2.4.4 Flow of Events

Basic Flow (Reroute path)			
User		System	
1	User is actively navigating along a route.	1	System is actively monitoring GPS and route alignments
2	User takes an incorrect turn/notices a safety concern and decides to re-route		
		3	System update new parameter, suggest re-route and notify users
		4	The routing engine calculates an optimal alternative path from the user’s current position to the destination.
5	User agrees to the suggested re-route		
		6	System starts the navigation
		7	Updated route details (ETA, distance, route highlights) are displayed.
8	User reaches the destination		
		9	The system logs the reroute event for analytics or user feedback.

Alternative Flow 1: Network Failure			
User		System	
1	The user is mid-navigation and takes a wrong turn/wants to re-route based on a safety concern.		
		2	The system attempts to fetch updated route data.
		3	System detects no network connection and checks for cached map tiles or previously stored routes.
		4	If cache is available, the system recalculates based on the stored

			GPS data OR If cache is unavailable the system maintains the last known route and displays a message: “Cannot reroute at this time.”

2.4.5 Special Requirements

- Collecting external parameters when a user is navigating. These external parameters could be anything from a closed business to a faulty street light to an inactive public transport service when the user is navigating.
- Real time update external parameters on time.

2.4.6 Platform

- External Mapping Platform
- OpenRouter Routing Engine

2.4.7 Preconditions

- Real time parameters, like a closed business to a faulty street light to an inactive public transport service
- Have the respective permissions (location, notification, speaker)

2.4.8 Postconditions

- Records the user's reroute event and logs it for analytics.

2.5 Use Case Name: Feedback

2.5.1 Description

The goal of this use case is to collect user’s safety experiences like ratings, text based feedback or tags (regarding a specific category including tags like street lighting, business, etc) regarding the routes they have taken, which will be used to improve subsequent safety ratings, especially in blacklisted areas.

2.5.2 Actors

- Users
- Crowdsourced Data Contributors
- Supporting Actors
    - System Admin (Datasources, City Council, Crowdsourced User Data)
    - Emergency Responders

2.5.3 Triggers and Inputs

- Trigger:
    - User giving Feedback - Report unsafe streets or incidents

    - User tap **Send** to submit feedback
    - System updates data source
- Input fields:
    - Rating severity from 1 - 10 (1 is safest and 10 is the most dangerous)
    - Tags
    - Comment

2.5.4 Flow of Events

Basic Flow			
User		System	
1	User inputs origin and destination		
2	User taps on button "Search"		
		3	System computes the route based on Safety Environmental condition
		4	System displays available routes
5	User selects the desired route		
6	User taps on the button "Go"		
		7	System starts the navigation
8	User encounters or sees a dangerous situation		
9	User want to report the situation by feedback feature		
10	User taps feedback button		
		11	System shows feedback form
12	User rates the route (1–10 stars), selects the tag and optionally adds comments		
13	User completes the form and taps "Submit"		
		14	System receives the feedback, evaluates it for its efficacy and if accepted this information is stored in the database
		15	System reroute a safer route based on the feedback

Alternative Flow 1(Feedback without navigation)			
User		System	
1	User sees the map		
2	User tap one area and want to input feedback		
3	User tap feedback button		
		4	System shows feedback form

5	User rates the route (1–10 stars) and optionally adds comments		
6	User completes the form and taps “Submit”		
		7	System receives the feedback, evaluates it for its efficacy and if accepted this information is stored in the database

Alternative Flow 4(Network error while submitting the feedback form)			
User		System	
1	User inputs origin and destination		
2	User taps on button “Search”		
		3	System computes the route based on Safety Environmental condition
		4	System displays available routes
5	User selects the desired route		
6	User taps on the button “Go”		
		7	System starts the navigation
8	User encounters or sees a dangerous situation		
9	User want to report the situation by feedback feature		
10	User taps feedback button		
		11	System shows feedback form
12	User completes the form and taps “Submit”		
		13	System is unable to successfully resolve a network connection
		14	System shows warning and blocks the submit because of the network error or Internal Server Error

2.5.5 Platform

- External Mapping Platform
- OpenRouter Routing Engine

2.5.6 Special Requirements

- User Experience and Accessibility - Feedback submission should be available both during and after navigation, through a simple, mobile-friendly interface.
- The form should be easy to understand, with clear rating options and short, optional comment fields.

2.5.7 Preconditions

- The user must already be registered and logged in the system with valid credentials.
- Since feedback systems always depend on map systems (tiles, geolocation APIs), the device must have a working network connection.
- The authentication service (e.g., backend server or OAuth provider) must be up and responsive.
- The feedback feature is available and visible in the app interface.
- The user is either currently navigating or has recently completed a route within the application.

2.5.7 Postconditions

- Safety feedback stored into database
- Retraining the route and new routes are shown on the map
- When other users navigate the same route, the system will incorporate previous feedback to refine safety ratings and route suggestions.

2.6 Use Case Name: SOS Feature

2.6.1 Description

When a user encounters an emergency, he or she can send a distress message to the local police or emergency contact, along with the location and user information.

2.6.2 Actors

- Emergency Responders
- Users
- Supporting Actors
  - System Admin (Gardai)

2.6.3 Triggers and Inputs

- Triggers:
  - Long-press the SOS button (button in app/lock screen widget/Action button on phone)
  - Alert sent to the emergency service.
  - The system sends user location and user information.
  - Option to mark no threat anymore.

- Inputs:
    - Real-time location: (GPS/Wifi/cell tower) location accuracy, recent movements.
    - User information: name, gender, age, emergency contact number,(users need to explicitly authorize)
    - Location clue: Photo/short audio (2–5s)
    - Undo input: User taps "Safe" and selects the reason (accidental touch/escaped)

2.6.2 Flow of Events

Basic Flow			
User		System	
1	User long press SOS button		
		2	The system immediately enters the "Help Countdown (default 5 seconds)" page; if in silent mode, the screen will not light up.
3	User is prompted with an option to slide "Cancel" during the countdown to prevent accidental touches.		
		4	System prefetches location, user data, voice, photos in the background.
		5	When the countdown ends, the system sends data to police and emergency contacts
		6	The app enters a continuous update state, sending the latest location to the server every second
		7	System sends status updates to users, including notifying the police and emergency contacts. This is a silent notification, requiring users to actively check the status.
8	User receives the notification and can update their status as threat resolved (status could be marked as accidental touch, out of danger, or resolved by the police)		

Alternative Flow 1: Network Failure	
User	System

1	The user is trying to use the SOS feature and loses network/internet connection.		
		2	The system detects no network connection and returns an error message that there is no network connection and falls back to calling the emergency service.
3	User receives the response and uses the manual calling service to contact emergency services.		

2.6.4 Platform
- Inhouse SOS Platform
- OpenRouter Routing Engine

2.6.5 Special Requirements
- Emergency services interface system and contact.
- Audit compliance: Sending user information to the police and emergency contacts is subject to GDPR requirements, and user data is only stored for 60 days.

2.6.6 Preconditions
- The user has logged in, collected minimum user information, set the default route, and notified the police or emergency contacts.
- Authorized to read location and microphone permissions.
- The phone needs to have a network signal and active internet connection.

2.6.7 Postconditions
- At least one of the police or emergency contacts must confirm receipt of the emergency message.
- After notifying the police or emergency contacts, a page will be provided to view the real-time location.