Project Specification

No-Drown Transmission Speed Prototype

By

Troels Kaldau

Overview

Project Context

This document is written on behalf of LittleGiants for the No-Drown project. The project aims to develop an application capable of detecting falls near bodies of water with a heightened risk of drowning. Upon detecting a fall, the application will send an alert to a remote server, which includes the user's location, enabling the server to dispatch help.

Problem Statement

For the application to be viable, it must detect the fall and transmit a distress signal containing the user's location before the phone comes into contact with water. A target alert delay of less than 100 ms has been established.

Prototype Description

The prototype will comprise a mobile application connected to a backend system. The application will be designed to send data to the backend upon user request or at specified intervals. This data will simulate the information which the final product would send to signal a fall, with the objective of measuring the transmission delay.

Additionally, the prototype will feature a demonstration mode to showcase the concept. In this mode, the application will utilize the accelerometer to detect acceleration exceeding a certain threshold, triggering a message to the backend. This message will then be relayed to a third-party receiver, which falls outside the scope of this project.

Requirements

Functional Requirements

Users must be able to:

FR1. Send a data message.

FR2. Set an interval for data message transmission.

FR3. View a list of sent requests.

FR4. Access details of requests, which include:

- FR4.1. Process timestamps.
- FR4.2. Viewing received location coordinates.
- FR4.3. Seeing detected acceleration.

FR5. Customize the data included in the alert message, such as:

- User ID.
- Accelerometer data.
- Timestamp.
- Battery status.
- Network status.

FR6. Adjust the precision of location data.

FR7. Toggle the show-mode (fall-detection) on and off.

FR8. Modify the threshold for fall detection.

FR9. Export all sent requests in a CSV format.

Success Criteria

1. The data message must be transmitted from the client device within 100 ms.
2. The provided location must be within 100 meters of the client device's actual location.

Design

UI Design

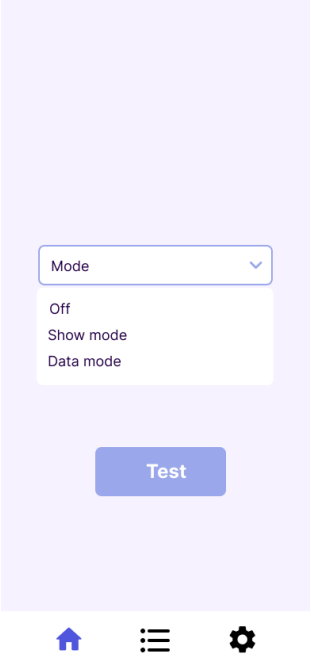

Figure 1: Home

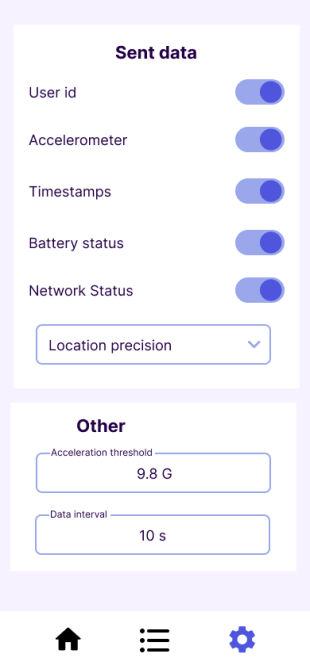

Figure 2: Settings

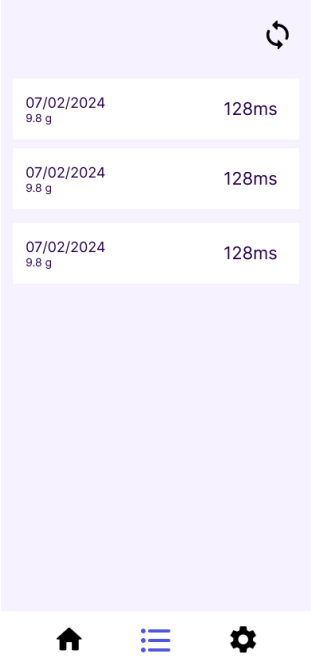

Figure 3: List

Figure 4: Details

System Design

The application requires the device's location at the time a fall is detected. Triangulating the location at the moment of detection would take too long; hence, the device must continuously update its location. While most phones already determine location continuously, the accuracy is often compromised to conserve battery life. An appropriate strategy must be devised to ensure sufficient location accuracy while minimizing battery consumption and background processing.

In the prototype's demonstration mode, falls are detected based solely on a predefined acceleration threshold. Future versions may utilize acceleration patterns for fall detection. These patterns could be processed either on the phone or streamed to the backend for cloud-based analysis. To determin the most suitable method, both approaches will be implemented for comparative analysis.

Various frameworks and patterns exist for WebSocket communication. To enhance speed, the transfer of binary messages is preferred over JSON-formatted messages.

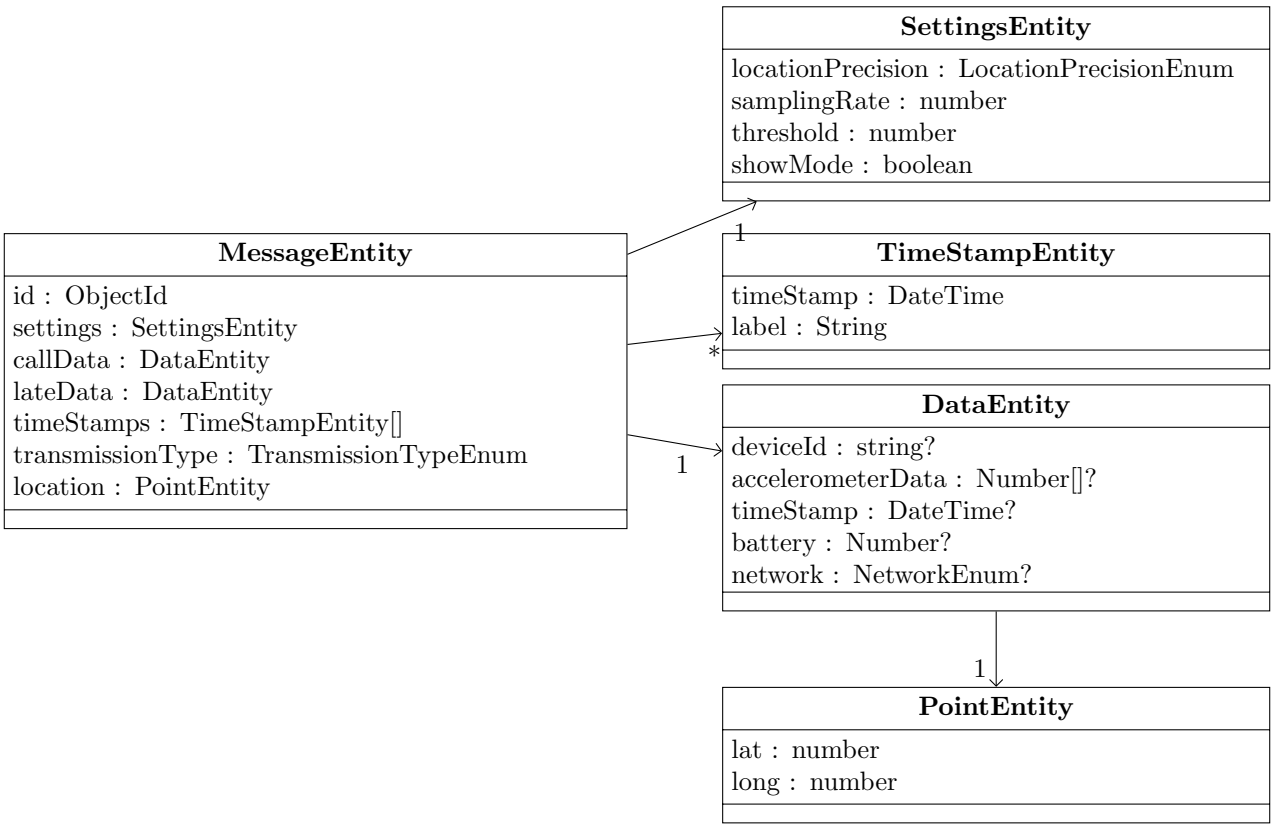

Figure 5: Detection entities

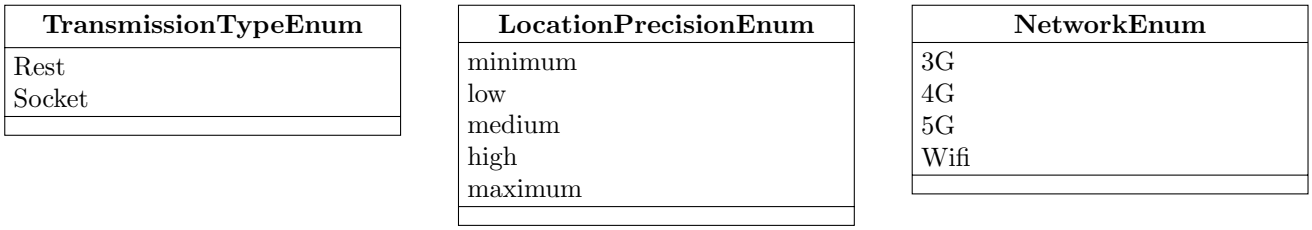

Figure 6: Enum models

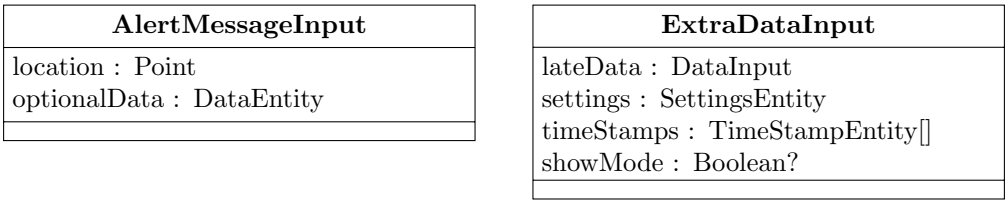

Figure 7: Endpoint input models

Property	Description
Method	POST
URI Path	/message
Description	Sends an alert message.
Request Body	JSON object matching the AlertMessageInput model.
Response	ObjectId of the created message.
Error Codes	'400 Bad Request' for invalid input.

Property	Description
Method	PATCH
URI Path	/message/{message-id}
Description	Adds extra data to an existing alert message without delaying the initial transmission.
Request Body	JSON object conforming to the ExtraDataInput model.
Response	Confirmation of the update.
Error Codes	'400 Bad Request' for invalid input. '404 Not Found' for a non-existent message ID.

Property	Description
Method	GET
URI Path	/message/recent
Description	Retrieves a list of the most recent alert messages.
Request Body	N/A
Response	List of MessageEntity models.
Error Codes	N/A

Property	Description
Method	GET
URI Path	/message/data
Description	Downloads a CSV file containing all messages.
Request Body	N/A
Response	CSV file with all messages
Error Codes	N/A

Estimation

TOTAL: 120 hours

- Backend: 40h
  - ◇ Backend setup: 14h
  - ◇ Data structure setup: 8h
  - ◇ Alert message endpoint: 4h
  - ◇ Extra data endpoint: 4h
  - ◇ Recent data endpoint: 6h
  - ◇ CSV data endpoint: 4h
- Frontend: 80h
  - ◇ Frontend setup: 16h
  - ◇ Home page: 4h
  - ◇ Settings page: 8h
  - ◇ Recent page: 4h
  - ◇ Data detail page: 4h
  - ◇ Location setup: 12h
  - ◇ Accelerometer setup: 8h
  - ◇ Alert Message call: 12h
  - ◇ Interval alert message call: 4h
  - ◇ Show-mode setup: 8h