Requirements and Analysis Document for . . .

Lucas Karlsson, Joakim Ohlsson, Seif Bourogaa,
Joakim Tubring, Filip Hanberg

2020-09-16
version

1 Introduction

The applications purpose is to report current position of ticket controllers in real-time. The reports in this case will create alerts for when ticket controllers are in the nearby area or on the same route as the user. The application purpose is also to gather information from the reports and store that information for the future.

The application will consist of an android GUI connected to a graph based system. The model will map reports to geographical coordinates and position in the public transportation network.

1.1 Definitions, acronyms, and abbreviations

Model: The part of the program that will manage functionality for the application. GUI Node

2 Requirements

2.1 User Stories

2.1.1 Story Identifier: US1

Story Name: Make Report

Description As a User, I need to report when I see controllants so that other Users can see where the controllants are.

Confirmation A Report is made. The report is connected to an Incident. If there is no Incident then create a new one. The Incident is updates/shown in the GUI.

Functional Can I change my report? Can i make different type of reports?

Non-functional availability... security..

2.1.2 Story Identifier: US2

Story Name: Endorse Incident

Description As a User, I need to endorse a Incident so that other Users can see that it is more trustworthy.

Confirmation The Incident increases its trust factor. GUI is updated with new trust factor.

Functional Can i repel an Incident? Non-functional availability... security...

2.1.3 Story Identifier: US3

Story Name: Search and browse

Description As a User, I need to search and see where the controllants are so that I can be more carfeul.

Confirmation Latest Incidents shows up in GUI. A search function of Incidents.

Functional

Non-functional

availability... security...

2.2 Definition of Done

The entire code should be commented using Javadoc standard, methods and variables should be declared in such a way they are easy to understand and the code should have been tested and peer-reviewed by the group. After every feature branch has been approved they should be merged to master.

2.3 User interface

Our very, very, early GUI sketches seen in Figure 1 and Figure 2 below.

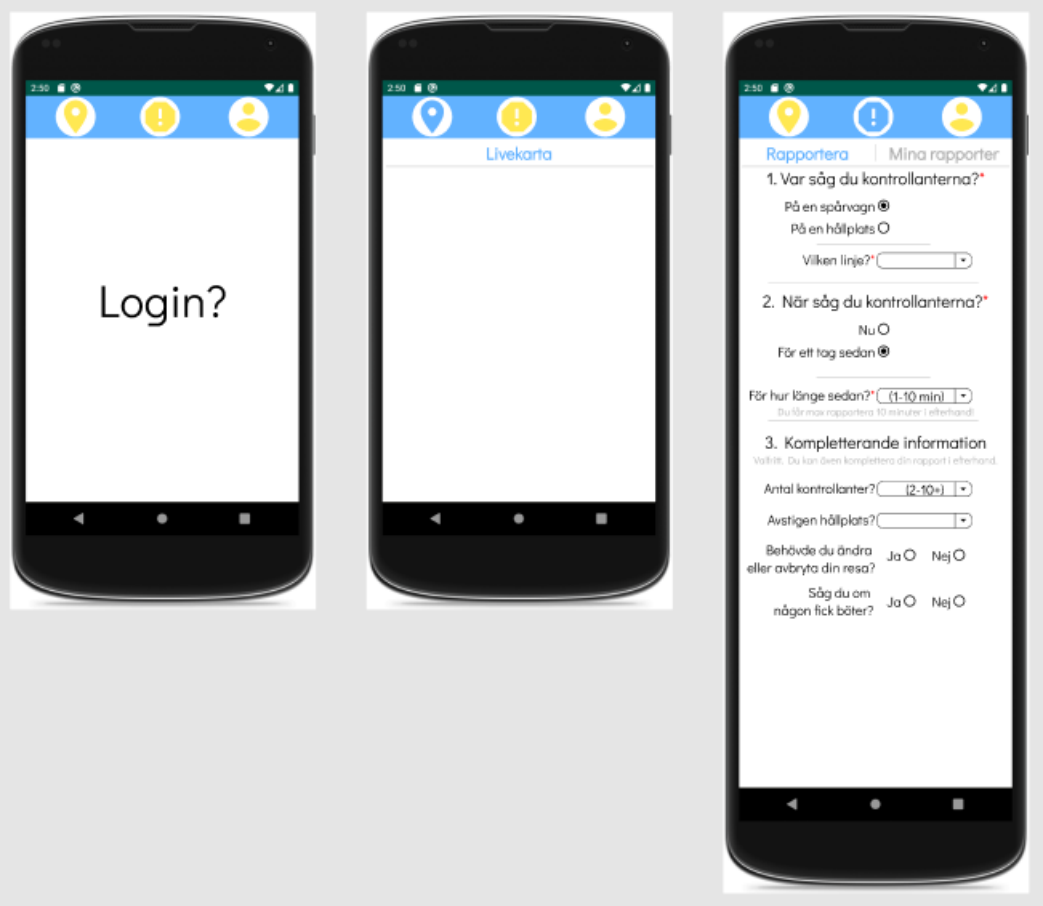

Figure 1: Preliminary GUI

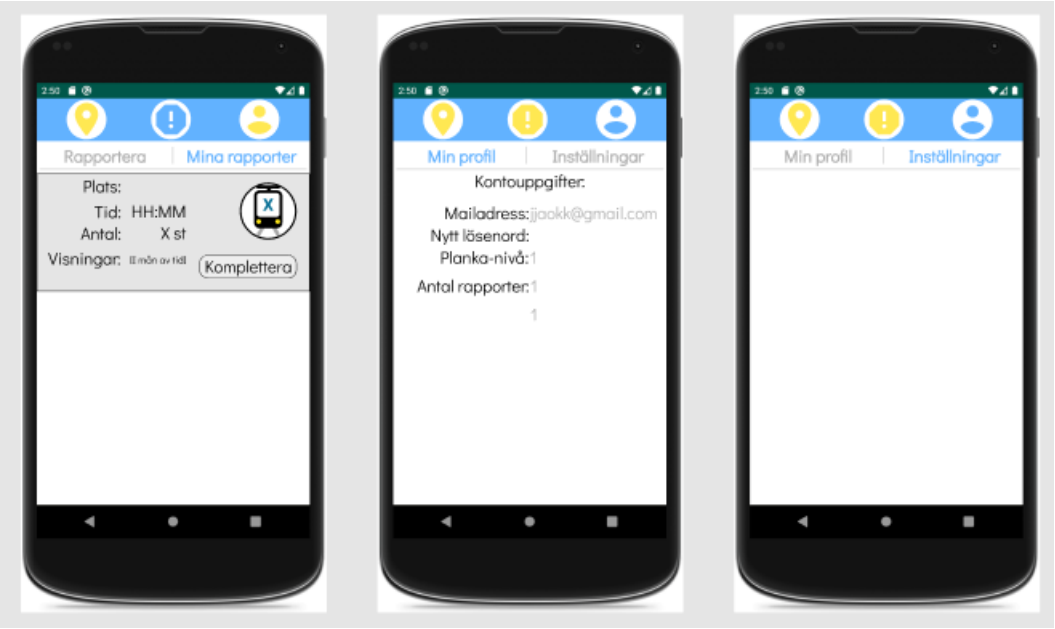

Figure 2: Preliminary GUI

2.4 Class responsibilities

3 Domain model

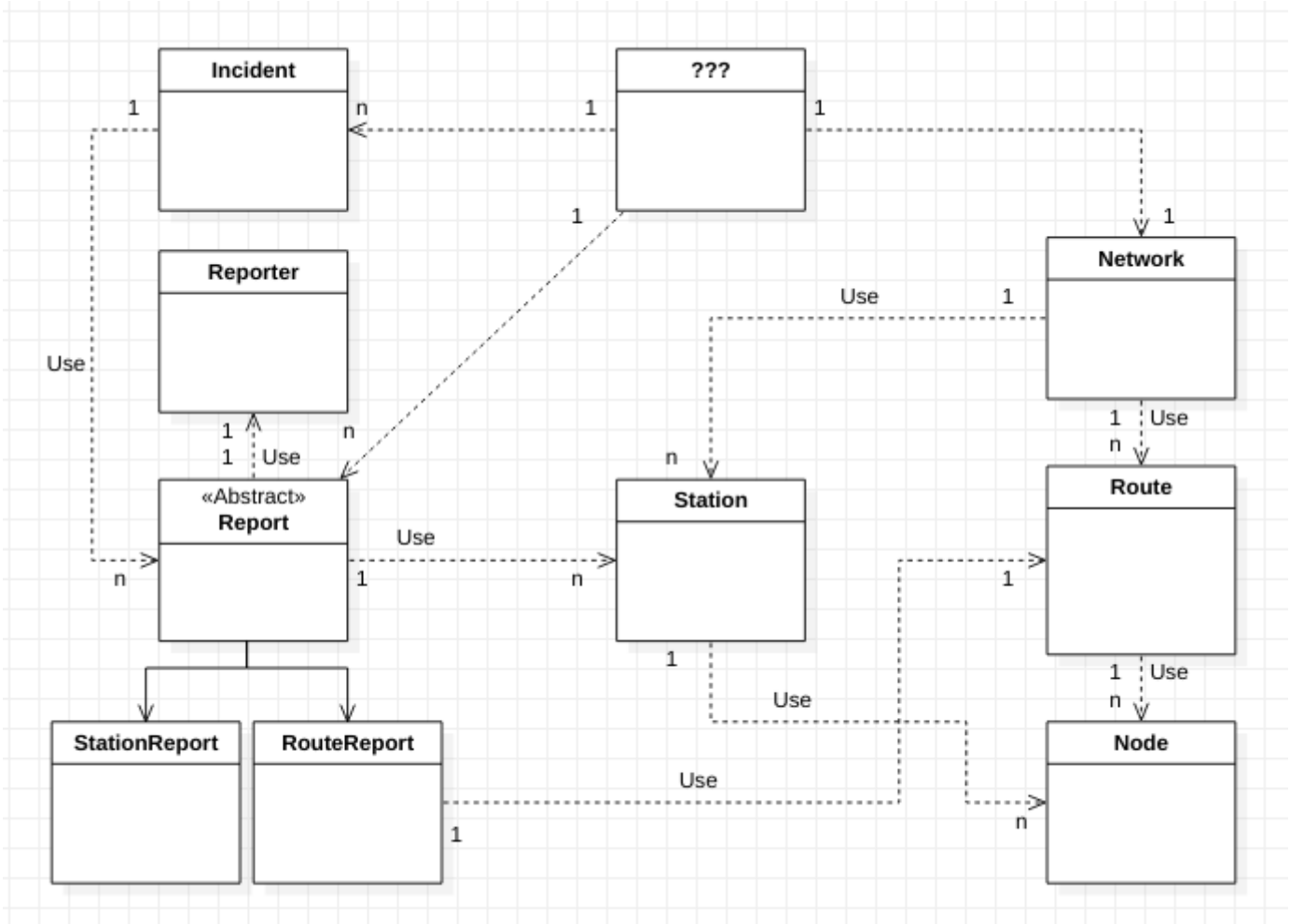

Figure 3: Domain Model

3.1 Class responsibilities

Explanation of responsibilities of classes in diagram.

1. Node
    - Class responsible for each tram station in the network.
    - Contains information such as name and alarm state.

2. Route
    - Class responsible for each tram line in the network.
    - Takes a List<Node>, one Node for each station the tram passes by.
3. Network
    - Class responsible for the entire tram network and its connections. Built up by adding every route to the graph data structure.
4. Station
    - Class responsible for representing the stations in our tram network, a station contains a list of individual stop positions for a station (position A,B,C...).
5. Reporter
    - Class responsible for representing the user in a report.
    - Protects the users integrity and only displays a email and a factor for how trustworthy the user is.
6. AbstractReport
    - Abstracts content from the RouteReport class and StationReport class and represents the reports for the incident class.
7. RouteReport
    - Class responsible for reports that involve both a station and a route
8. StationReport
    - Class responsible for report that involve a station
9. Incident
    - Class responsible to gather reports that all involve the same station and/or route.
10. Station
    - Something something
11. ???
    - Something something

4 References

List all references to external tools, platforms, libraries, papers, etc.