Requirements and Analysis Document for PlankaGBG

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

2020-10-23

1 Introduction

The purpose of the application is to report the current positions of ticket controllers in the Gothenburg public transport network and display this information to users in near real-time. The reports are created by users of the application, after which other users may choose to vouch for the credibility of the report if they deem it to be correct. Users may also choose to receive a notification when controllers are reported along a route of their choice.

1.1 Definitions, acronyms, and abbreviations

1. **GUI**: The graphical interface, how the functionality is displayed to the user.
2. **Model**: The package of the program that will manage all functionality of the application.
3. **View**: The package of the program that that contains the GUI.
4. **Controller**:
   a) (**Code**) The package of the program that connects the *View* package to the *Model* package.
   b) (**Model**) An individual responsible for checking whether or not a public transit passenger has a valid ticket.
5. **Incident**: Multiple reports that takes place on the same route or station.
6. **Report**: When the user sees controllers the application lets the user compile the information and submit it. A report.

7. **Reporter**: The user.
8. **Non-functional requirement**: Requirement that defines how a system should be.
9. **Functional requirement**: Requirement that defines what a system should do.
10. **Java-doc**: Java-doc is a documentation generator that generates HTML documents from java source code.

2 Requirements

2.1 User Stories

Story Identifier *US1* - **Make a Report - Implemented**

1. Description:
   - As a User, I want to be able to report when I see controllers so that other users can see where the controllers are.
2. Acceptance Criteria:
   - Users can create Reports.
   - Reports are always connected to an Incident when created.
     - If no matching Incident exists, a new one is created alongside the Report.
   - The Incident is updated and displayed in the GUI.
3. Functional Requirements:
   - Can I click a button to make a Report?
   - Can I make different kinds of Reports?
4. Non-functional Requirements:
   - **Reliability:** Application should not crash when creating a new Report.
   - **Usability:** It should be easy for the User to create a new Report.

Story Identifier *US2* - **Endorse Incident - Implemented**

1. Description:
   - As a User, I want to be able to endorse an Incident so that other Users can see that it is trustworthy.
2. Acceptance Criteria:

- An Incident can be endorsed by users.
- Said Incident is deemed more trustworthy if a User reports it as such.
- The incident is updated and displayed in the GUI.

3. Functional Requirements:
   - Can I confirm that an Incident is correct?
4. Non-functional Requirements:
   - **Reliability:** Application should not crash when endorsing an incident.
   - **Usability:** It should be easy for the User endorse an incident.

Story Identifier *US3* - **Search and Browse the Network - Implemented**

1. Description:
   - As a User, I want to be able to see where the controllers are so that I can be more careful in my travels.
2. Acceptance Criteria:
   - Recent incidents should be visible in the GUI.
   - A User should be able to search for specific stations and routes to see if they are affected by recent incidents.
3. Functional Requirements:
   - Can I click on a button to search for Incidents related to a tram line?
   - Can I see recent Incidents in a the GUI?
4. Non-functional Requirements:
   - **Scalability:** It should be easy for Developers to scale the functionality.
   - **Usability:** It should be easy for the User browse the controllers in the Network.

Story Identifier *US4* - **Identifying Image - Not Implemented**

1. Description:
   - As a User, I want to be able to attach an image to my report to further strengthen its credibility.
2. Acceptance Criteria:
   - If an image is uploaded, it is displayed alongside the incident it is attached to.
   - The credibility of the incident and report is strengthened if an image is attached.

3. Functional Requirements:
    - If I create an Report, can I add an image?
    - Can I change the image afterwards?
4. Non-functional Requirements:
    - **Usability:** It should be easy for the User attach an image to a report.

Story Identifier *US5* - **Trust Factor - Implemented**

1. Description:
    - As a User, I want to have a trust factor that impacts the credibility of my Reports.
2. Acceptance Criteria:
    - The trust factor of a user is increased when a Report they've made is backed up by other users or reports.
    - When a Report is created, an auto generated trust factor is assigned to that Report based on the trust factor of the user.
3. Functional Requirements:
    - Can the trust factor of a user be altered?
    - Can a user impact the trust factor of other users?
4. Non-functional Requirements:
    - **Security:** One User should not have direct access to the private information of another User.
    - **Reliability:** Trust factor should not be changed randomly for each user.

Story Identifier *US6* - **Modify Report - Implemented**

1. Description:
    - As a User, I want to be able to change or modify a Report after it has been made.
2. Acceptance Criteria:
    - After a Report has been made, the reporter should be able to modify it.
    - The Report is updated when a modification is made.
    - The updated Report is displayed in the GUI.
3. Functional Requirements:
    - Can I add or remove something from a Report I have created?

4. Non-functional Requirements:
    - **Security:** One User should not have direct access to another User's report, meaning they should not be able to change or modify it.
    - **Reliability:** A report should not be altered unless the User modifies it.

Story Identifier *US7* - **Notification - Implemented**

1. Description:
    - As a User, I want to be notified if a controller is on my route and/or near me.
2. Acceptance Criteria:
    - Users should have the option to receive notifications.
    - Users should be notified about controllers on their route, or near them, if they have said option enabled.
3. Functional Requirements:
    - Can I enable a notification function?
    - Is the notification function affected by Reports?
4. Non-functional Requirements:
    - **Manageability:** The User should easily be able to turn the Notification on and off.
    - **Reliability:** The application should not crash when sending the User notifications.

Story Identifier *US8* - **Map - Not Implemented**

1. Description:
    - As a User, I want a graphical representation of where reports has been made.
2. Acceptance Criteria:
    - Users should be able to access a map that displays currently active Reports.
    - The Map should correctly display where and when Reports were made.
3. Functional Requirements:
    - Can I see where Reports has been made on a map?
    - Can I see when the Reports were made?
4. Non-functional Requirements:
    - **Manageability:** The User should easily be able to switch to the Map view.
    - **Reliability:** The Map should not force the User to close the app to switch view.

2.2 Definition of Done

The entire code should be commented using Java-doc 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. Any visual elements that are part of the code should be implemented in the GUI of the application. After every feature branch has been approved they should be merged to master to allow for continuous development and integration.

2.3 User Interface

Screenshots from our GUI can be seen in Figures 1 through 5 below. The design of the application has been purposefully kept simple to make it easy to use under stress, as we reckon that people tend to be heavily stressed while using the public transport network.The GUI consists of three primary ”tabs”. Navigation is handled using a toolbar at the top of the screen.

2.3.1 Incidents

The Incident List View [Figure 1] displays all currently active incidents in Gothenburg. Each incident features information regarding its location, the time at which the incident occurred as well as the total number of reports. This view also allows the user to search for specific routes and stations, to see if any incidents currently affect them.

2.3.2 Reporting

The Report View [Figure 2] allows users to fill in and report information regarding incidents they witness, after which the Live Reports View is updated. This view also lists all reports that the user has made and allows recently made reports to be modified [Figure 3 and Figure 4, respectively].

2.3.3 Personal Information

The Profile View [Figure 5] allows the user to change which email address is associated with their account and lets the user opt into receiving a push-notification when an Incident is reported on a tram line of their choice.

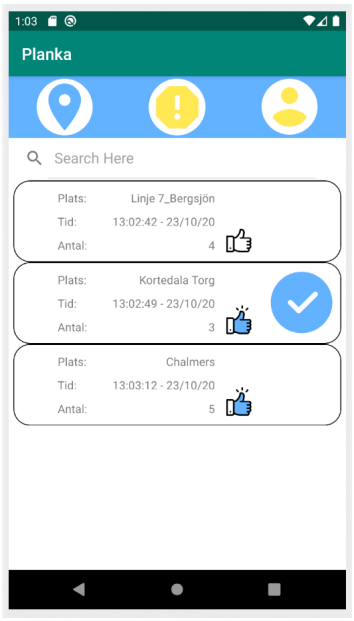

Figure 1: Incident List

Figure 2: Report Creation

Figure 3: User Reports

Figure 4: Report Editing

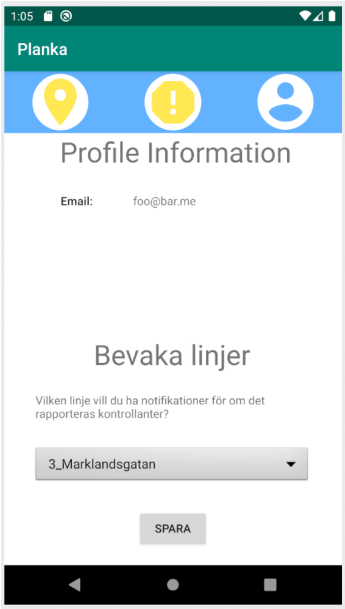

Figure 5: Profile

3 Domain model

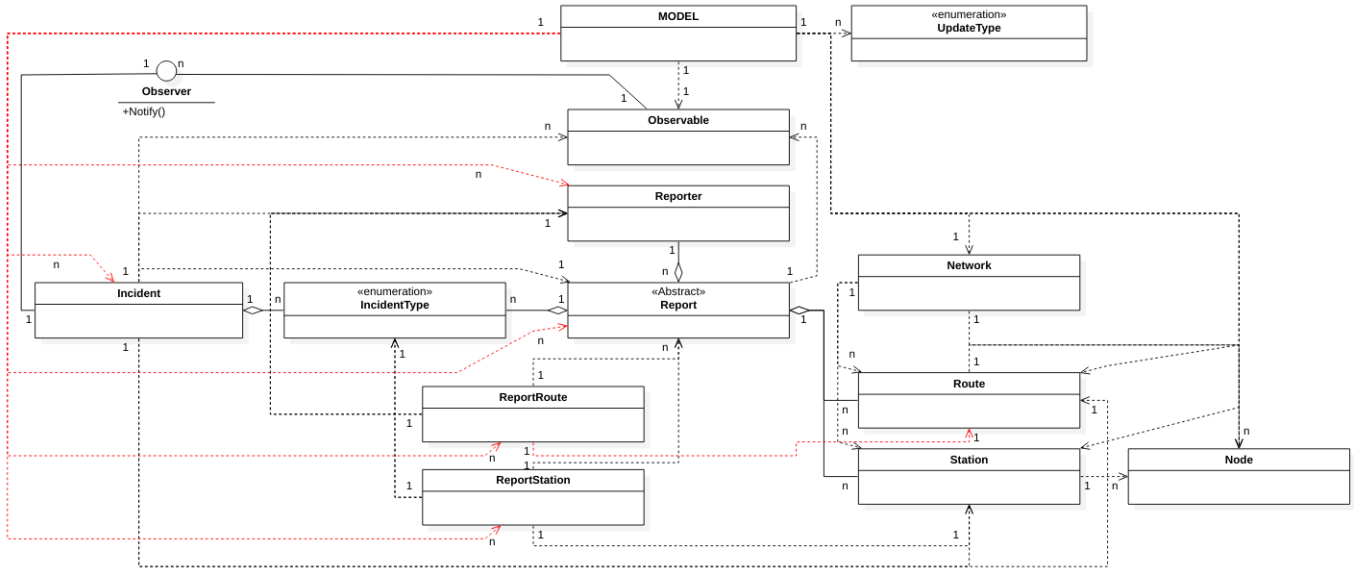

Figure 6: Domain Model

3.1 Class responsibilities

Explanation of responsibilities for the classes in the domain model above:

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 of Nodes, 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 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.

- For example, Station Centralstationen contains Centralstationen A, Centralstationen B etc.

5. Reporter
   - Class responsible for representing the User in a report.
   - Protects the Users integrity and only contains an email address 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 a Route.
8. StationReport
   - Class responsible for Reports that involve a Station.
9. Incident
   - Class responsible for gathering Reports that all involve the same Station or Route.

4 References

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