Ingegneria del software e progettazione web

Progetto A.A. 2024/2025

R.A.I.L

0327406

Alessio Fior

Table of contents

1. Introduction

Aim of the documentation

The aim of the documentation is to provide a full description of the software system. The system has been developed by following a well-defined approach, based on practices of software engineering, toward the satisfaction of the project goals.

Overview of the defined system

R.A.I.L. (*Report Anomalies in Infrastructure and Lines*) is a system designed to simplify and expedite the reporting of anomalies occurring within the railway environment.
It enables users to report issues such as **track defects**, **malfunctioning level crossings**, or other infrastructure-related problems.
The main goal of the application is to provide a simple and intuitive solution that allows users—such as railway personnel or authorized staff—to report these problems quickly and efficiently.

Users are able to monitor the status of each individual report, ensuring full transparency and traceability throughout the entire resolution process.
With R.A.I.L., tracking a report from submission to resolution is easy and accessible, improving overall maintenance workflows and enhancing the safety and efficiency of railway operations.

Interfaces Available in the Application:

- **Graphical User Interface (GUI)**: Users can access the system through a minimalist and intuitive GUI. Through the GUI, users can report anomalies by selecting the type of railway asset (e.g., track or level crossing), entering relevant details, and submitting their report with just a few clicks. The interface also allows users to monitor the status of their submitted reports in real time
- **Command Line Interface (CLI)**: For more advanced users, the system also offers a structured and easy-to-use command-line interface, allowing users to perform all essential tasks through text commands.

  The system is a desktop application that provides the following functionalities:

  -User Registration and Login:
  Users can create a new account or log into an existing one to access the system's features.

-Report Railway Infrastructure Issues:

Users can report key problems affecting the safety and functionality of the railway infrastructure, such as level crossing malfunctions or track anomalies.

-Submit Feedback:

The system allows users to suggest new functionalities for the application, helping to evaluate the overall effectiveness and efficiency of the reporting process.

-Real-Time Monitoring of Reports:

Users can submit an unlimited number of reports and track the status of each one in real time. This ensures full transparency throughout the resolution process.

HW ad SW requirements

Software and hardware requirements:

- RAM: 128 MB
- CPU: Pentium 2 266 MHz processor
- Disk space: 124 MB
- Operating system: Windows 7 or later, macOS 10.8.3 or later, any Linux distributions that supports Gnome, KDE or Unity DE

Related system, pros and cons

R.A.I.L can be compared to other infrastructure reporting systems, particularly those used for urban or public works reporting, such as **FixMyStreet** for road maintenance or **Waze** for real-time road hazard reporting. While such systems are generally focused on city environments, RAIL distinguishes itself by being specifically designed for the railway sector, focusing on the reporting of anomalies related to tracks, level crossing, and other railway infrastructure.

Pros of RAIL compared to FixMyStreet and Waze:

- **Domain-specific design**:
  Unlike general-purpose reporting apps like FixMyStreet, which focus on urban issues such as road maintenance, or Waze, which reports real-time traffic hazards, RAIL is tailored specifically to railway infrastructure.
- **Dual interface availability**:
  RAIL offers both a Graphical User Interface (GUI) for user-friendly reporting and a Command Line Interface (CLI) for more advanced users. While Waze and FixMyStreet only offer user-friendly interfaces, RAIL's flexibility caters to both casual users and professionals who may prefer a more technical approach.
- **Real-time report monitoring**:
  In RAIL, users can track the status of their submitted reports in real-time, ensuring transparency and user trust throughout the resolution process. While FixMyStreet allows tracking of reports, and Waze provides live updates on traffic conditions, RAIL specifically focuses on monitoring the resolution process of railway infrastructure issues.

However, some limitations include:

- **No automated anomaly detection**:
  Unlike systems like Waze, which automatically reports real-time road hazards based on user data and sensors, RAIL relies entirely on user-submitted reports. There is currently no integration with sensor data or automated monitoring systems that could detect and report issues, such as malfunctioning signals or damaged tracks, without human intervention.
- **No built-in collaboration tools**:
  While FixMyStreet allows users to comment on and discuss reports within the community, RAIL currently lacks collaborative reporting features. Users can track their own reports, but there is no community feedback, collaborative issue reporting, or the ability to comment on or verify other reports. This could limit the app's engagement and the opportunity for cross-verification of issues that could improve the resolution process.

- **Limited scope of issues**:
  Apps like FixMyStreet and Waze cover a much broader range of infrastructure problems, such as potholes, traffic lights, and public services, which have a wider user base. In contrast, RAIL is more niche, focusing only on railway infrastructure, which may not appeal to as large a general public as apps like FixMyStreet or Waze

2. User Stories

1. As a user, I want to report malfunctions at a level crossing, so that the responsible authorities can take immediate action to restore proper railway operations and ensure safety.

2. As a user, I want to suggest additional functionalities for the application, so that I can contribute to improving the system based on real user needs and experiences.

3. As an administrator, I want to send a notification* to the user, so that they are informed when the issue they reported has been resolved.

notification* = an in app text message
Note: the second and third user stories are not implemented.

3. Functional Requirements

1. The system shall send an email to the user confirming the acknowledgment of the report they submitted.

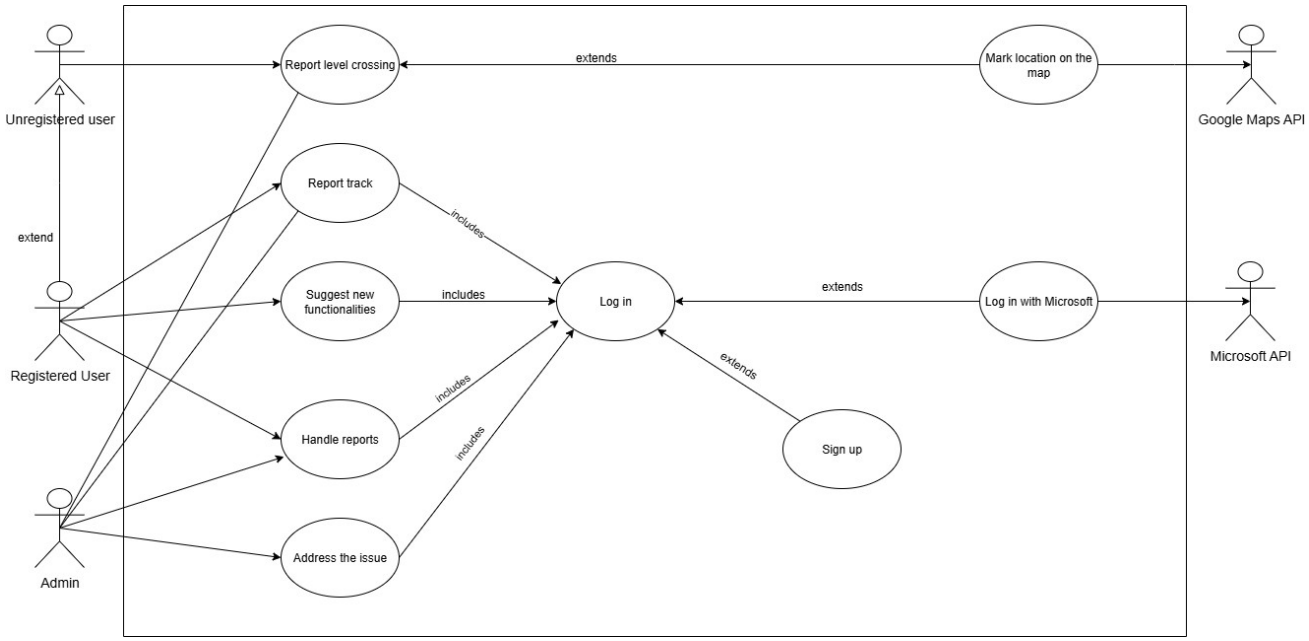
2. The system shall provide a list of reports, including the status* of each report and a description of the reported issue.

3. The system shall provide a blank form to enter the information** about a track report.

*status = reported or fixed
**information: location, track number, description of the issue.

Note: the first functional requirement is not implemented.

4.Use cases Diagram



Note: *Mark location on the map*, *Log in with Microsoft*, *Suggest new functionalities* and all admin use cases are not implemented.

Internal steps

Use case: Report track

1.The user requests to report a railway asset issue.

2.The system shows the different assets that can be reported.

3.The user chooses to report a track.

4.The system provides the fields for the user to fill in.

5.The user fills in the fields.

6.The user requests to submit the data.

7.The system saves the report.

8.The system sends a notification to the administrator regarding the reported issue.

Extensions:

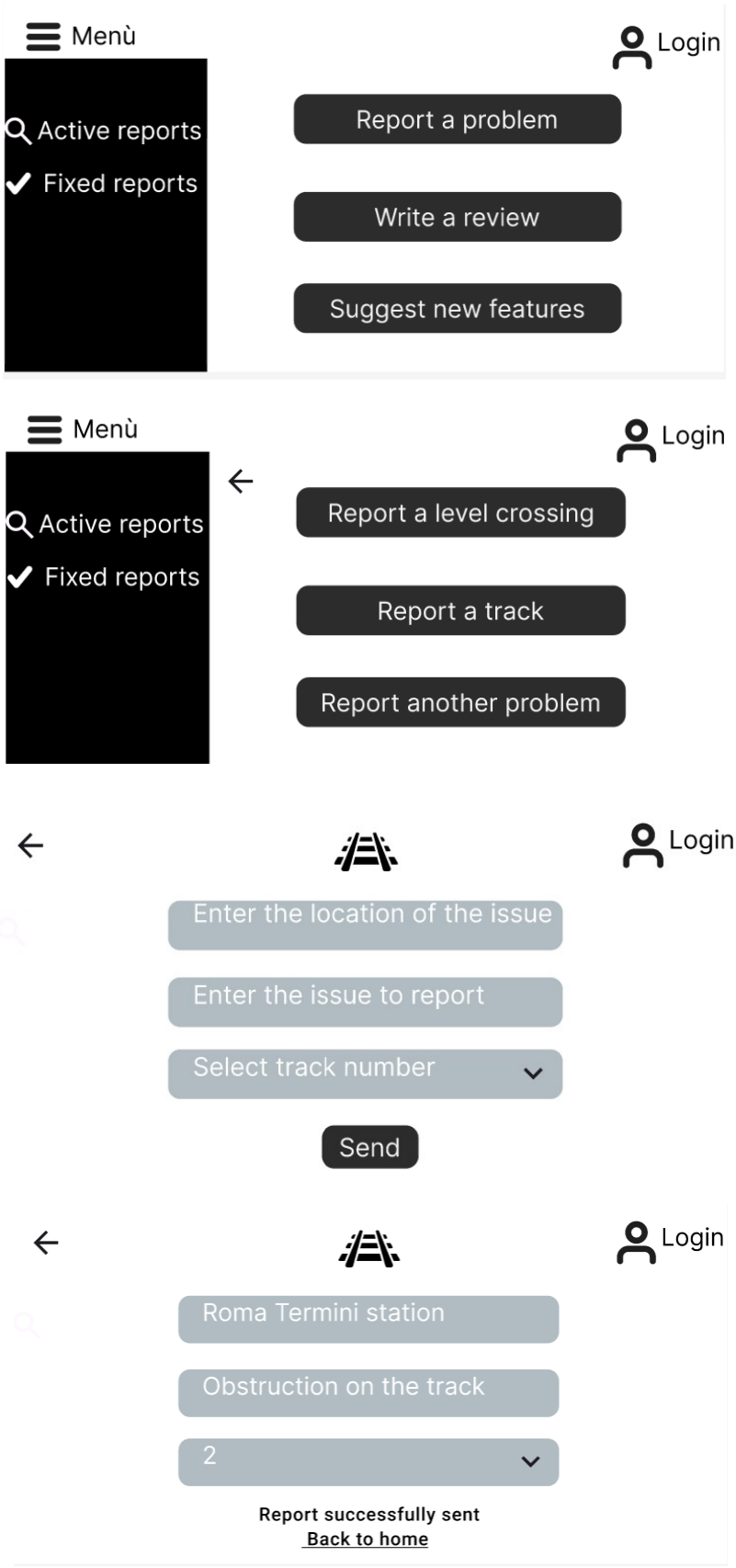
6 a. Invalid data entered: The system notifies the user and asks them to try again.

6 b: User not logged in: the system notifies the user and ends the use case.

7 a. Report already exists in the system: The system notifies the user and ends the use case.

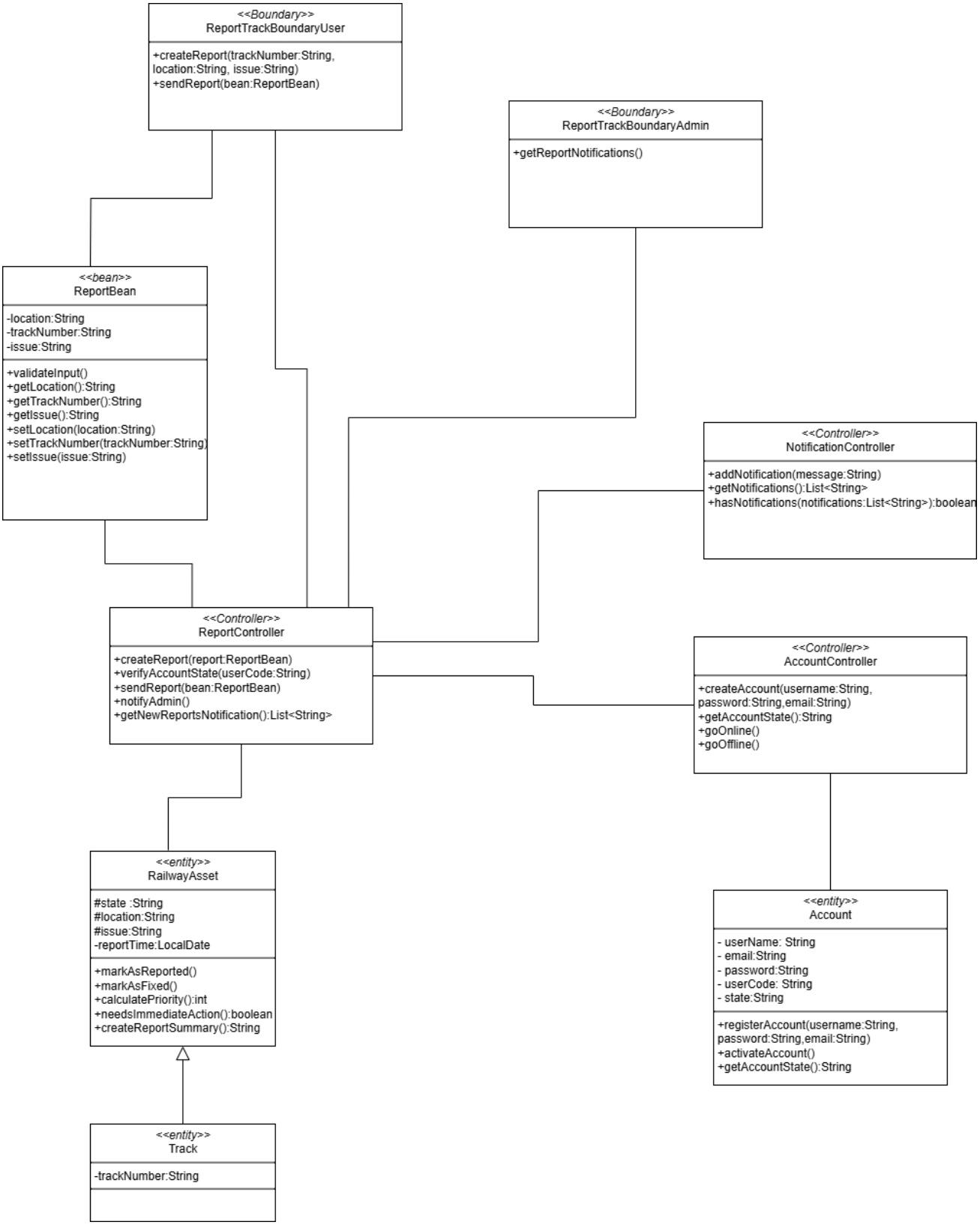
5.Storyboards

The following storyboards illustrate the Home Page, the asset issue Selection page, and the Report Track page, with the last one showing a successfully submitted report.

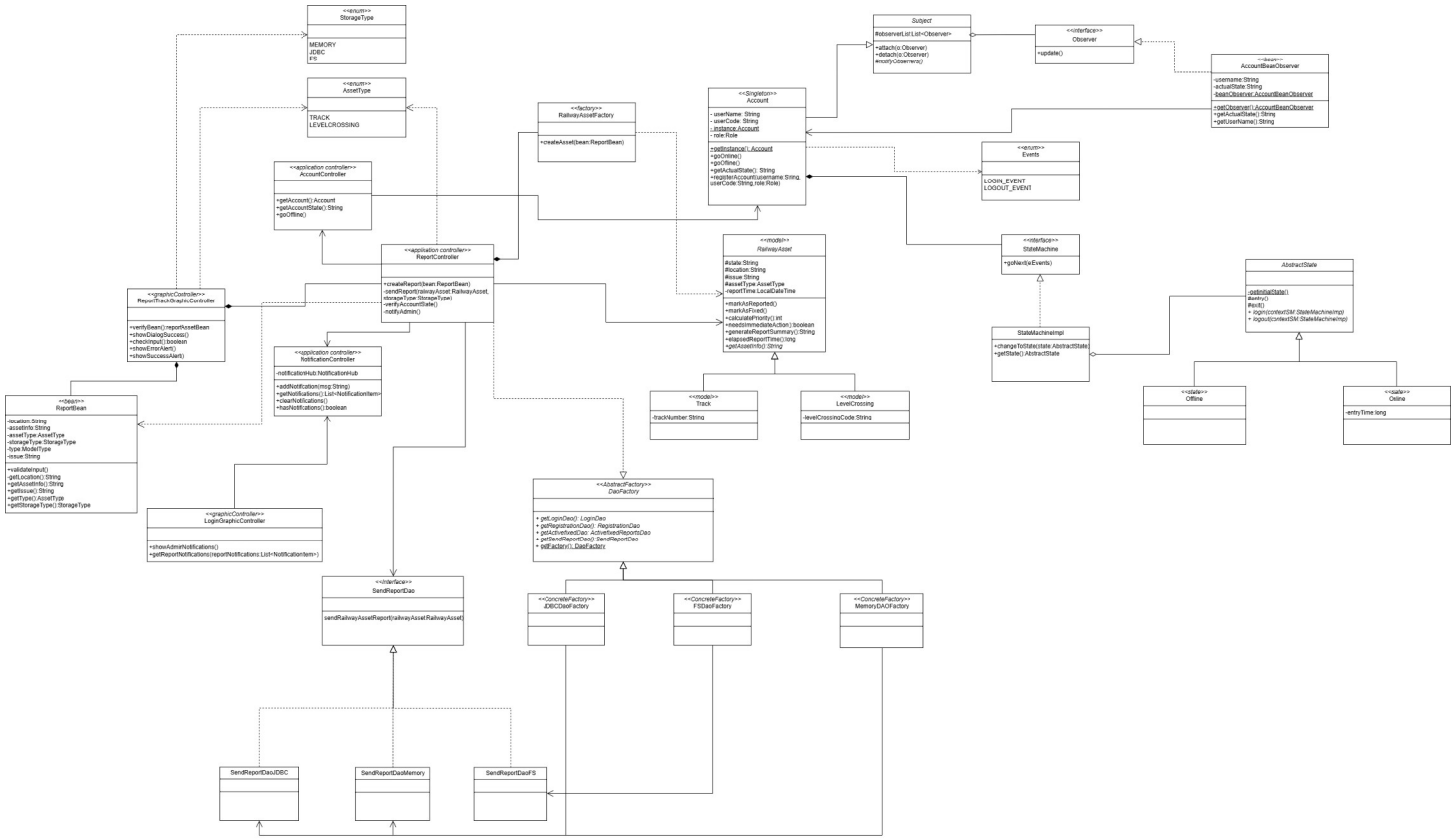


6.Class diagram

BCE - Use case: Report track

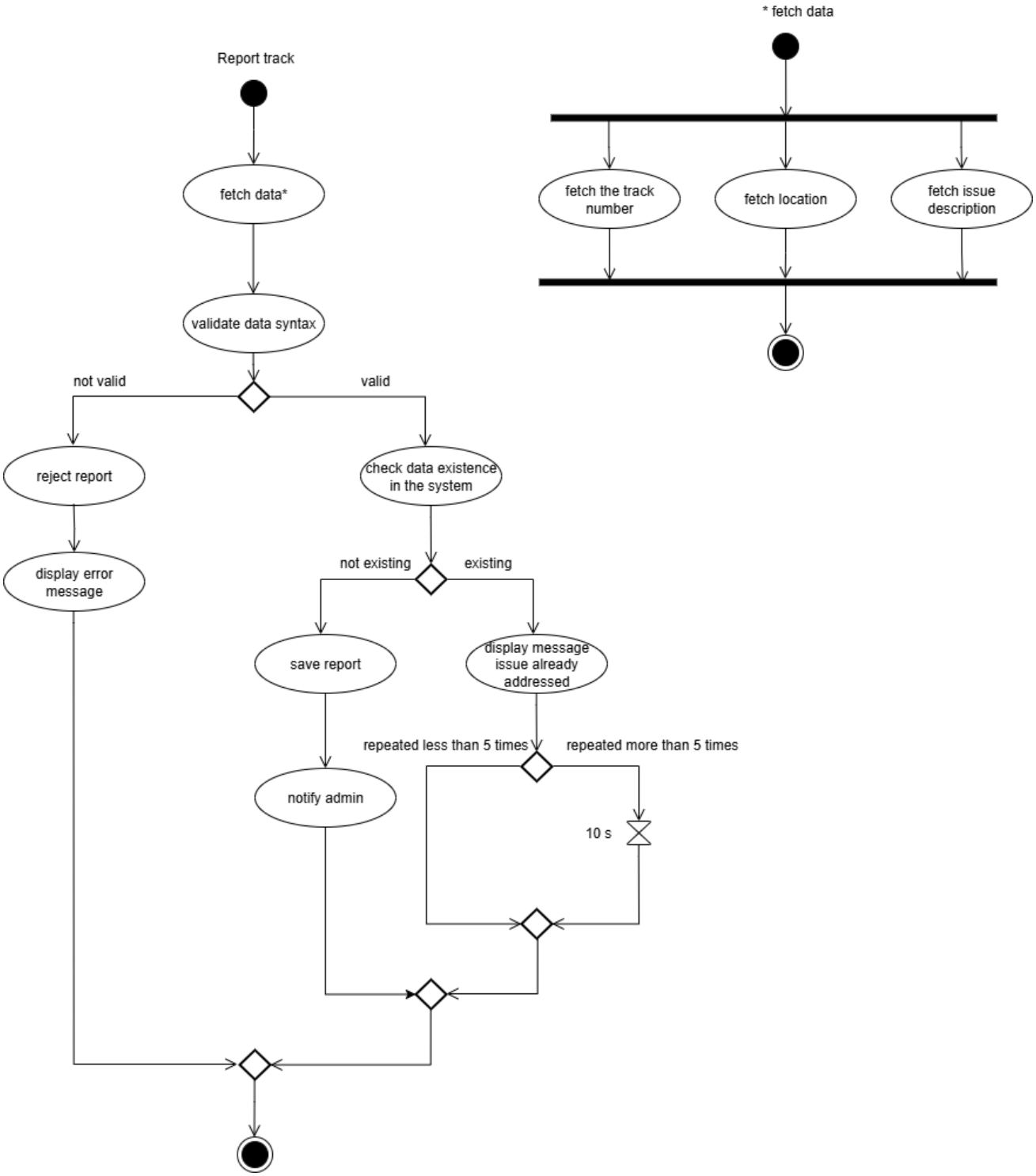


Design-level diagram - use case: Report track



7. Activity diagram

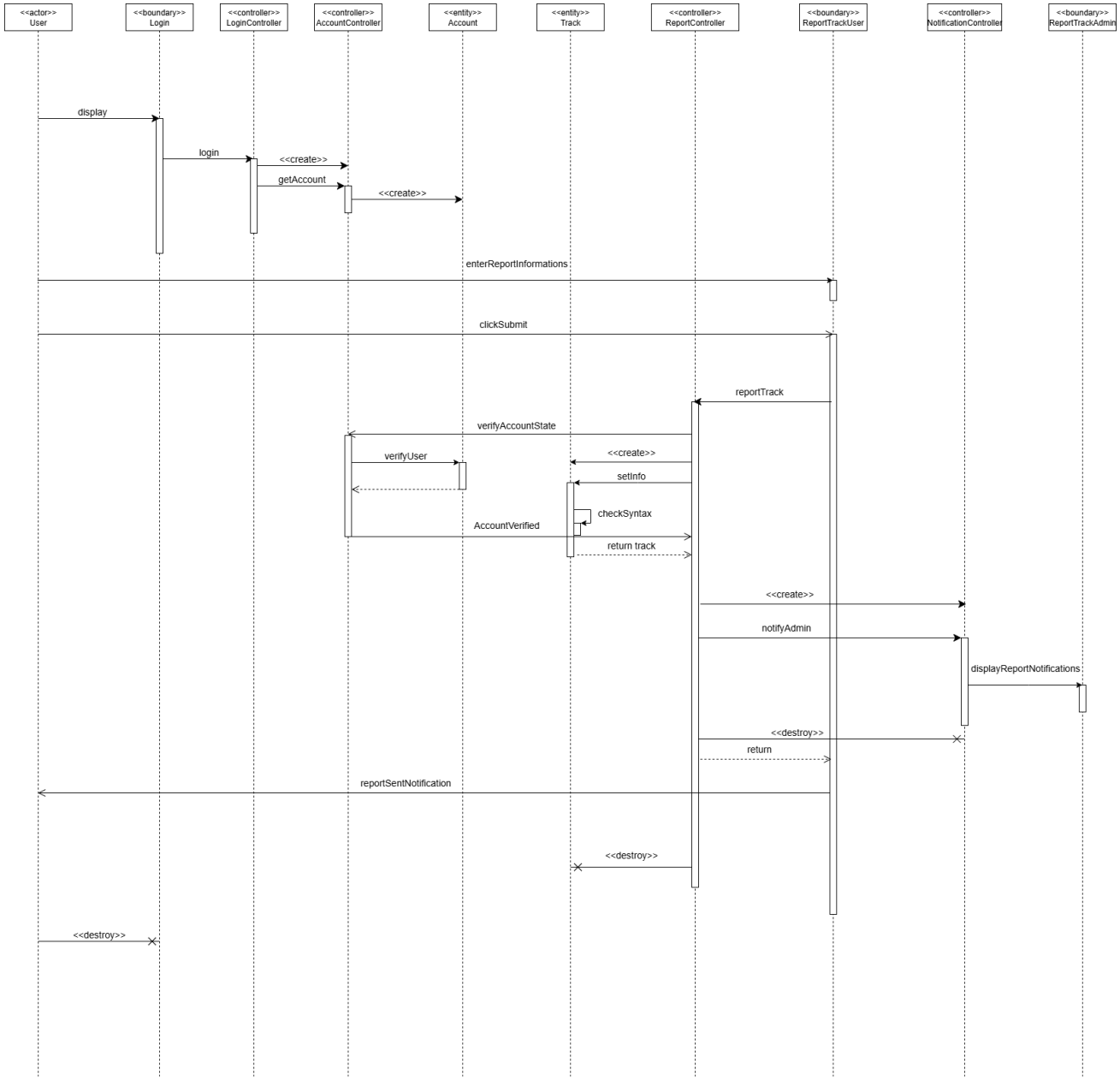
Use case: Report track



Note: in the final implementation the user do not have to wait 10 seconds if the same issue is reported more than 5 times.

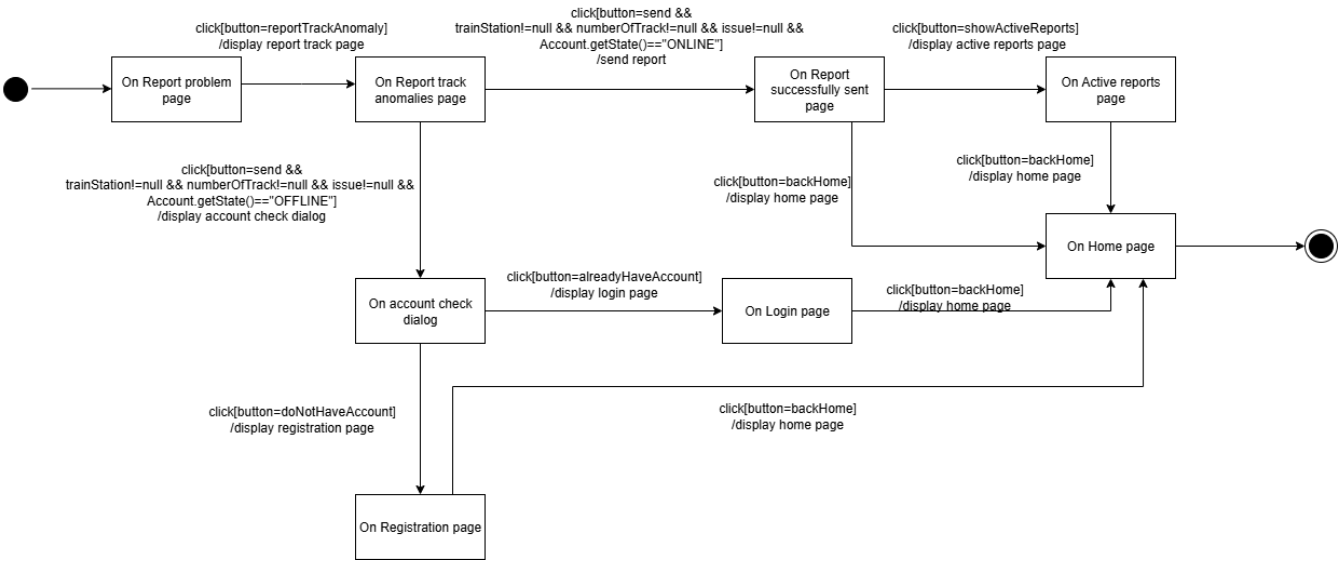
8. Sequence diagram

Use case: Report track



9. State diagram

Use case: Report track



10. Testing

Here is the link: LinkTesting

Six tests have been implemented:

- **LoginDaoTest – verifyExistance**: it checks that the method verifyAccount returns true when an email and password corresponding to an existing account are provided.
- **LoginDaoTest – verifyNotExistInSystem**: it verifies that the method verifyAccount returns false when an email not present in the system is used.
- **RegistrationTest – registerUser**: it checks that the method registrateUser returns true when a new user is registered, while it should return false if the user already exists in the system.
- **RegistrationTest – verifyUserExistance**: it verifies that the method verifyUserExistance returns true if the provided credentials are not yet in use, and false otherwise.
- **SendTrackDaoTest – saveTrackNotInDB**: it tests that a new railway track not yet present in the database can be successfully saved, and that the outcome of the operation is 0.
- **SendTrackDaoTest – saveTrackAlreadyInDB**: it checks that the system prevents inserting a duplicate railway track, raising an exception and returning an outcome of -1.

11. Code

Here is the link to the code: AleFLower/R.A.I.L: ISPW project

12. Video

Here is the link to the video: R.A.I.L-Deliverables/video at main · AleFLower/R.A.I.L-Deliverables

13.Link deliverables

If some of the diagrams are not clearly readable, please find below the link to access each diagram individually in PDF format: AleFLower/R.A.I.L-Deliverables

14. SonarCloud

Here is the link to SonarCloud: link SonarCloud