Digitalization of Individual Railway Transport with a Cross-Platform App

Jannis Gehrt

Bachelor's thesis
September 27, 2023

Prof. Dr. Reinhard von Hanxleden
Real-Time and Embedded Systems Group
Department of Computer Science
Kiel University

Advised by
Niklas Rentz
Alexander Schulz-Rosengarten

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Weiterhin erkläre ich, dass die digitale Fassung dieser Arbeit, die dem Prüfungsamt per E-Mail zugegangen ist, der vorliegenden schriftlichen Fassung entspricht.

Kiel,

Abstract

Reactivating old railway tracks has the potential to enhance transport in rural areas and reduce transport related emissions. To increase the attractiveness of those railway tracks, digitalization has the potential to improve competitiveness and enhance safety.

Since the widespread adoption of smartphones, apps have proven as a great solution to digitize direct to consumer contact. In the realm of individual railway transport, apps can provide a detailed map for better orientation, usage statistics for a better user experience and dynamic warnings to improve safety.

This thesis will develop such an app and give insights into the development process and underlying concepts and of this application, such as use cases, user flow, and communication with a bigger system that includes hardware on the vehicles and a server.

The evaluation of the app showed that it has the potential to improve safety and can enhance the user experience by providing additional information.

Acknowledgements

I would first like to thank Prof. Dr. Reinhard von Hanxleden as the head of the working group. Thanks to him, I was able to participate in this project and write my bachelor's thesis which marks the end of my studies at this university.

Furthermore, I would like to thank Niklas Rentz and Alexander Schulz-Rosengarten who gave me guidance and ideas during the development of the app, as well as assisting me in the writing of this thesis.

I would also like to thank Sven Ratjens for granting us access to the railway track and for his support during on-site testing.

Finally, I would like to thank Liam Boddin, Daniel Mäckelmann, Nico Biernat, Julian Grabitzky, Darlin Stücker, and Kevin Ebsen for the good company and joint development of the RailTrail System.

Contents

Chapter 1

Introduction

In Germany, as well as in numerous other countries, there are many small railway tracks that are not included in the network of freight and passenger transport and have been abandoned over the years due to various reasons, such as high costs and logistical challenges. Reactivating such tracks can improve the transportation network in rural areas, contribute to the fight against climate change, and be more cost-effective than building new railway tracks.

One example of such a track is the railway between Malente and Lütjenburg, which currently serves as a testing and development environment for universities and companies in the context of the REAKT initiative[1]. By revitalizing this track the local transportation infrastructure could be improved in a climate-friendly manner, while serving as a positive example for similar railway tracks.

The railway track is currently used as a tourist attraction, providing draisines for rental to tourists to let them explore the track. Draisines, shown in Figure 1.1, can be described as rail-bound bicycles, representing the most basic form of individual railway transport. In the future, beyond the scope of this thesis, introducing other vehicles such as solar trams or conventional trains is also possible.

Figure 1.1. Draisine

In an increasingly digitized world, the significance of providing digital solutions to meet customer needs has never been more important. With the widespread adoption of smartphones, apps have become widely accessible, offer an intuitive user experience and have greater functionality in terms of sensor data when compared to traditional websites. Furthermore, the rise of cross-platform UI frameworks like React Native, enables smaller development teams to efficiently release applications on all relevant platforms, ensuring a fast and cost-effective development process.

1.1 Problem Statement

To assist the REAKT initiative in reactivating the railway track, the work of this thesis contributes to a project that includes six other students. This project aims to address the following problems:

[1] https://www.schiene-m-l.de/

- Enhancing the safety of track drivers.
- Improving the overall user experience for drivers.
- Increasing the attractiveness of individual railway transportation.
- Collecting data to support the track's future development.

To address these issues, a decision was made to develop a mobile application. This smartphone app is designed for passengers and drivers of rail-bound vehicles and should contribute to improved safety and enjoyment of the users. It should be well-integrated into a larger system that is being developed by the rest of the project, and includes hardware on the vehicles and a server.

The following chapters will provide a detailed view of the app's underlying concepts, use cases, and implementation details.

1.2 Related Work

The academic research in this particular field remains rather limited, however, there have been papers released in the broader field of digitalization in railway transport and reactivation of older railway tracks.

Hertel et al. [HPK23] emphasized the importance of reactivating old railway tracks to reduce green house gas emissions and improve transport in rural areas, but also discussed the challenges associated with reactivating these tracks, such as high operational costs and worn-out infrastructure. To address these issues, they proposed digital solutions, including the use of driverless vehicles.

In a briefing for the European Parliament, Scordamaglia [Sco19] evaluated the advantages and challenges of digitalization in the railway sector. He highlighted that digitalization could improve the competitiveness and efficiency of the railway sector, but also pointed out that investment cost and cybersecurity could prove as a challenge. Furthermore, it was noted that the European Union aims to further enhance the digitalization of railway transport.

In the commercial sector, companies have introduced applications that share certain similarities with the app of this thesis. While most of the available apps in the railway domain have primarily focused on topics such as route planning and ticket booking, there are applications beyond the railway context such as Komoot[2] or Runtastic[3] that show resemblances to this application.

Komoot is a hiking and trekking app designed to display trip statistics and route tracking for users. It shares basic features with this application, such as the presentation of usage metrics, position tracking, and marking the user's current location on a map. However, it is important to highlight a few distinctive factors that set the applications apart. These include

[2] https://www.komoot.com/

[3] https://www.runtastic.com/

the fundamental differences between static railway tracks and dynamic trekking routes. Additionally, the app of this thesis includes functionality in the realm of safety, providing warnings before level crossings and oncoming vehicles, that are not present in Komoot or similar applications.

Runtastic is an app that enables users to record their workouts and aims at providing an improved user experience during running sessions. The app tracks the path and speed of the user and provides metrics to measure progress. The similarities and differences between Runtastic and the app of this thesis are comparable to those with Komoot.

1.3 Outline

Chapter 2 introduces the key technologies used in the development of this app. The information of this chapter can help to better understand the subsequent chapters. Chapter 3 explains the core concepts of the application, such as use cases and user flow and provides insights into the integration with the larger system. Chapter 4 will follow up with the evaluation of the app. The chapter discusses the findings of testers of the app and will look back on which features that were discussed earlier ended up implemented. Chapter 5 serves as the thesis conclusion, summarizing key findings and identifying areas for future app improvement.

Chapter 2

Technologies

This chapter introduces the key technologies used in the development of the app. These technologies were selected to simplify the implementation and enhance the functionality of the app. Subsequent sections will describe each technology, explain the reason behind its selection and list its advantages.

2.1 React Native

React Native[1] is an open-source UI framework that forms the foundation of this app. It enables developers to create cross-platform apps with a single JavaScript codebase, significantly reducing development time, maintenance efforts, and project costs. React Native uses *components* as the building blocks of the user interface. These components are similar to web components, but map directly to native UI elements. This integration delivers a native-like user experience.

Another notable advantage of React Native is its big community, offering a large amount of libraries to effortlessly enhance app functionality, Section 2.4 will provide a deeper look into the libraries that were used. Additionally, the framework includes a feature known as "hot reloading", allowing instant visualization of changes within the app and accelerate the development process.

2.2 TypeScript

TypeScript[2] is a strongly typed programming language that compiles to JavaScript. Being a strict superset of JavaScript, TypeScript offers numerous advantages without losing compatibility. Its strong type system enhances the capabilities of IDEs by providing more extensive auto-completion and highlighting errors when accessing undefined fields. This elevated level of type safety reduces the potential for errors in code, making TypeScript the preferred choice for most developers when starting new React Native projects.

[1] https://reactnative.dev

[2] https://www.typescriptlang.org/

2.3 Expo

Expo[3] is an open-source framework for React Native to simplify many steps of the development process. The tools provided by Expo increase development speed, simplify the testing on devices and make React Native more accessible to beginners.

2.4 Libraries

This application leverages several libraries to enhance its functionality and provide a more comprehensive user experience. The most impactful of these libraries are detailed below.

2.4.1 Redux

Redux[4] is an open-source state management library. States are an important part of React Native, as they save data and trigger re-renderings when their data changes. This library centralizes state management, resulting in cleaner and easier understandable code, improved testability, enhanced flexibility, and reduced errors.

2.4.2 react-native-maps

react-native-maps[5] is an open-source library designed for React Native, enabling the integration of maps in iOS and Android applications. Developers can choose between Apple Maps and Google Maps as map provides, but since Apple Maps can only be used on iOS, this application will use Google Maps for uniformity. This library offers an array of features, including smooth animations for position changes, multiple viewing options, including a satellite view, and support for adding markers on the map. The capability to add markers was extensively utilized in this app to highlight points of interest and vehicles on the map.

This library was chosen over the Open Street Maps alternative react-native-maps-osmdroid[6] for a few different reasons:

- Osmdroid is still in the experimental stage.
- Osmdroid has not received updates in four years and lacks active maintenance.
- Compiling with react-native-maps-osmdroid triggers an array of warnings.
- Osmdroid only has 0.01% of the weekly downloads.

[3] https://docs.expo.dev/

[4] https://redux.js.org/

[5] https://github.com/react-native-maps/react-native-maps

[6] https://github.com/fqborges/react-native-maps-osmdroid

2.4.3 expo-location

expo-location[7] is an open-source library for React Native apps, focusing on handling location-related events. This library simplifies tasks such as checking permission status, requesting system permissions for location access, and accessing location data. The library provides multiple methods for developers to access the device's location information:

- Polling: Offers one-time access to the current location.
- Foreground listener: Monitors location changes while the app is open and sends location data to a callback function.
- Background listener: Continuously tracks location changes, even when the app is running in the background, the location data is forwarded to a specified callback function.

2.4.4 i18n-js

i18n-js[8] is an open-source library for managing translations and text entries. This library simplifies the process of selecting a given language within the code and extracting string resources from the code into a separate file, resulting in cleaner code and enhanced maintainability.

2.4.5 Axios

Axios[9] is an open-source library designed for making HTTP requests. It offers automatic JSON transformation, improved error handling, and enhanced code readability compared to the native fetch API.

[7] https://docs.expo.dev/versions/latest/sdk/location/

[8] https://github.com/fnando/i18n-js

[9] https://github.com/axios/axios/

Chapter 3

Concepts

This chapter presents the concepts, ideas and decision process behind the development of the app to fulfill the goals stated in Section 1.1.

3.1 Integration of the App in the Whole System

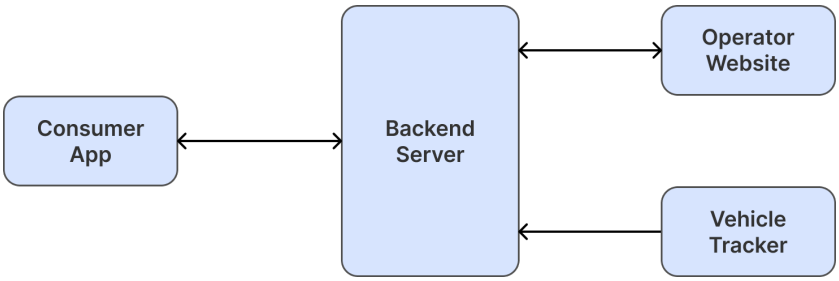


Figure 3.1. Communication Diagram of the System

The objective of this thesis is the development of the RailTrail app, which is an integral component of the larger RailTrail System. The system contains the app itself, a server as backend, a website for the track operators, and vehicle trackers, as illustrated in Figure 3.1. The arrows in the figure indicate the flow of data within the system.

The website is designed to satisfy the needs of railway operators, providing them with comprehensive tools for managing tracks, vehicles, trackers, and points of interest.

Vehicle trackers are mounted to the rail vehicles and regularly transmit important data, including position and speed information. Combined with the position data supplied by the app, these trackers form the fundamental basis for the backend's position calculations.

The server acts as the central hub, enabling communication between all the components of the RailTrail System, and offers vital data to both the app and website.

3.2 Use Cases

Conceptualizing the use cases of an app before development is important, as they guide the identification of essential features. These cases stem from a thorough analysis of user expectations and desires within the app's context.

The primary users for this app are passengers and drivers of rail-bound vehicles, while railway operators rely on the website for track and vehicle management.

Map Information

A big part of the user base of this application consists of tourists who may not be familiar with the railway track. These users have a desire for detailed map information, such as the railway track's path, the locations of other vehicles, and points of interest such as level crossings.

In summary, the application should include features to:

- Mark the railway track.
- Display other vehicles.
- Highlight points of interest such as level crossings, picnic places, track ends, and turning points.
- Offer a satellite view for enhanced orientation and additional information.

Current Position

Users would like to know their current location to enhance their orientation and track their progress. The application should include a distinct icon to mark the user's location and provide a feature that centers the map around the user's position.

Real-time Trip Statistics

Users are interested in real-time statistics about their trip, such as the distance traveled and current speed. Providing these statistics enhances the user experience, lets them see their progress, and fulfills their curiosity.

Safety

Safety is an important factor for both drivers and passengers of the vehicles. They desire warnings in safety-critical situations to enable focused and timely responses. Therefore, the application should issue warnings in two key situations:

- When the user approaches a level crossing.
- When the user and an oncoming vehicle are approaching each other.

These warnings should include the remaining estimated time or distance to help users make an informed decision in these situations.

Safety Instructions and Vehicle Functionality

Many users of the application are tourists who may be unfamiliar with operating these vehicles. Railway operators prefer correct, gentle, and safe handling of their vehicles, while the users want easy access to the information.

To meet these needs, the application should provide instructional videos explaining vehicle functionality and safety guidelines to users.

Privacy

Privacy-conscious users may prefer not to grant location permissions. To accommodate these users while maintaining functionality, the application should offer nearly identical features even when location services are disabled. However, users should be informed that enabling location services enhances accuracy.

Languages

Not all tourists visiting railway tracks are fluent in the native language. To ensure a broader reach, the application should offer both the native language, German, and English as a fallback language for tourists. Language selection should be based on the system language for an improved user experience.

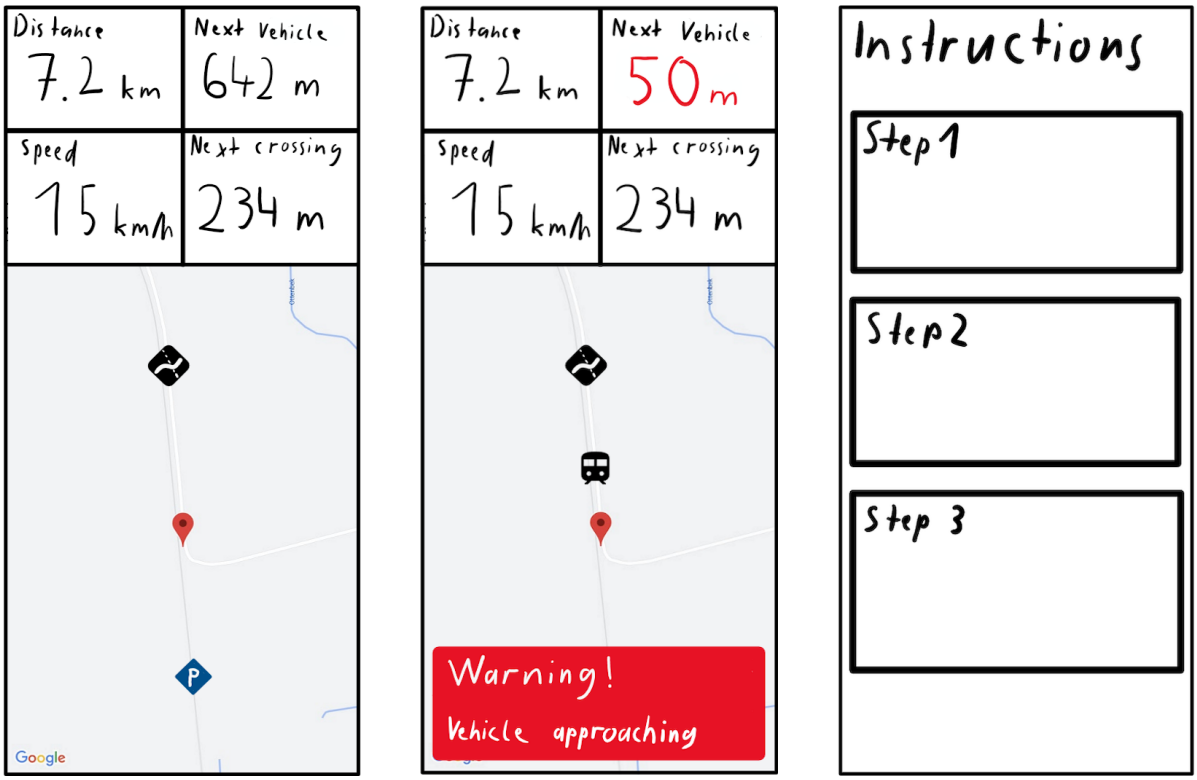
3.3 Mockups

The mockups, shown in Figure 3.2, display the most essential features of the app. They are used to guide the app's design process and show how use cases could be implemented in the app.

Figure 3.2a shows the initial concept for the map screen. It has a header that displays important information, such as distance driven, current speed, and distances to the next vehicle and level crossing. Since draisines are similar to a rail-bound bicycles, the first two statistics were selected to provide similar functionality to a bike computer. The other two statistics are shown to improve safety, by providing additional information on the distances to the two biggest potential safety hazards. Below the header is a map that shows the surrounding area and has additional icons, in this case for a level crossing and picnic place, as well as a red marker for the users current location. The mockup was designed to fulfill potential use cases for the app, including map information, current position, and real-time trip statistics.

Next, Figure 3.2b shows a potential implementation for the safety use case. The mockup shows what should happen if another vehicle is approaching. The other vehicle is marked on the map and a red warning at the bottom of the screen informs the user about the potential danger of an oncoming vehicle and attracts the user's attention.

Finally, Figure 3.2c focuses on the safety instructions use case. Displayed is a potential step for step guide that could be used for different topics, such as *how to secure a level crossing*.



(a) Map Screen **(b)** Warning **(c)** Info Screen

Figure 3.2. Mockups of the App's UI Design

3.4 User Flow

The user flow of the app is shown in Figure 3.3.

When the user opens the application, a welcome page (Figure 3.4a) is shown and the users can see basic information about the app, a checkbox to accept the data protection agreement, and are presented with a choice, whether they want to use the app with or without location services. This page provides a soft starting point into the app and enables users that are concerned about their location data to use the app with less precision but more data protection.

The app offers the option to be used for different railway tracks, this allows for easy adoption of additional railways in the future. To load all relevant data, the app needs to know which track the user would like to travel on. If the user agrees to use location services, their location is used to determine the nearest railway track and the map screen is displayed. If the user continues without location access, they are presented with a list of railway tracks

(Figure 3.4b) where they can select the one they want to travel on. They can then continue to the map screen. This user flow aims to provide the optimal user experience for different types of users. Most users can enjoy reduced complexity by skipping a step, while users who care about their privacy can use the app with nearly the same functionality.

Next, the user sees the map screen (Figure 3.4c), which is the heart of the app. It highlights the path of the railway track and shows the positions of all points of interest. The user can navigate the screen by swiping and zooming, just like they are used to with other maps on smartphones. The map screen offers two options for the user to continue. If the user wants to get additional information, they can click on the *Info* tab in the navigation bar at the bottom. This screen (Figure 3.4f) provides videos on safety guidelines and vehicle instructions to answer any questions that might come up when handling the vehicle.

The other option for the user to continue from the map screen is to start a trip. To do this, the user can click on a prominent banner at the bottom of the screen. A bottom sheet (Figure 3.4d) then pops up that informs the user that they need to enter the vehicle number to continue and where to find it. If the entered vehicle number exists, the trip starts. If not, a dialog pops up informing the user that they entered a wrong number and should try again. In case the user has malicious intend and intentionally enters a wrong vehicle number, the position data from the user is ignored, since the distance to the vehicle tracker is above a certain threshold.

After the trip is started, the user can see an extended version of the map screen (Figure 3.4e). It now also includes other vehicles that are near to the user and a header at the top that provides additional information. To further assist the user, the vehicle icons also show the travel direction, so the user can plan ahead and make space for oncoming vehicles. The header displays useful information such as the current speed, distance driven, and distances to level crossings and vehicles. The vehicles shown to the user only consist of vehicles that are within a certain radius, this adds protection to the location data of other users and saves resources on the server. If the user approaches a safety-critical situation, such as a level crossing, a warning banner is displayed to the user, so they can take appropriate action. All of these features were implemented to assist the user in driving safely and to provide information that the user is curious about.

If the user changes the vehicle during a trip, they can update the vehicle number by clicking on an icon next to the current vehicle number and entering the new number in a bottom sheet similar to the bottom sheet they started the trip with. Reusing the bottom sheet component makes the app more intuitive and easier to understand for the user.

To end the trip, the user simply clicks on the red stop button. To avoid accidental clicks, an additional dialog for confirmation pops up. After the confirmation, the user is directed to the normal map screen.

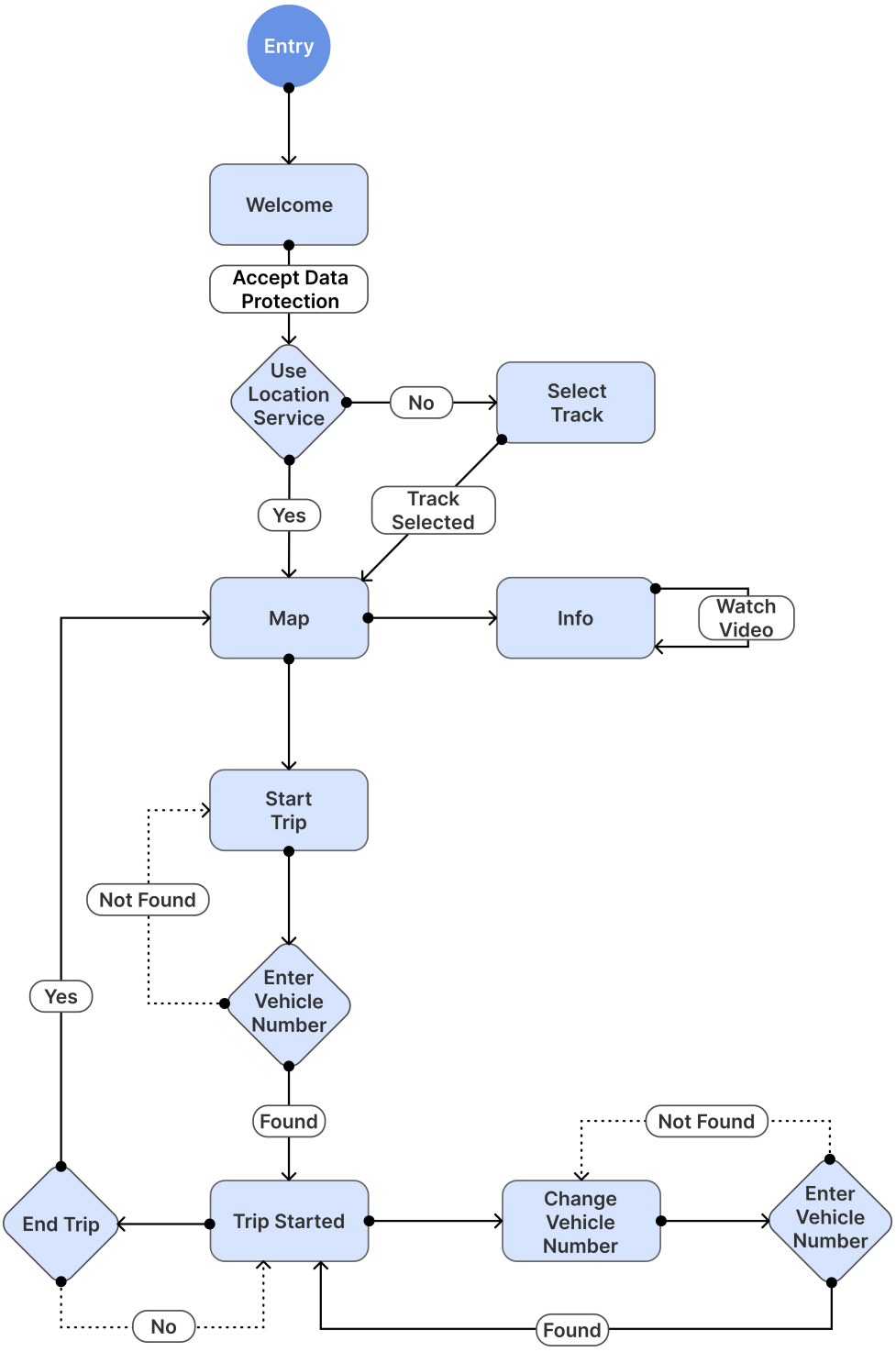
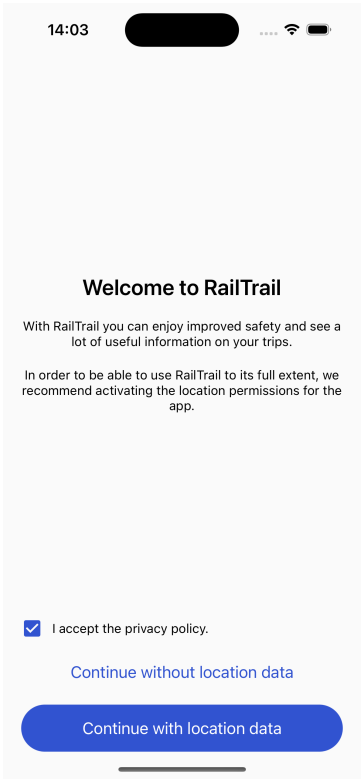
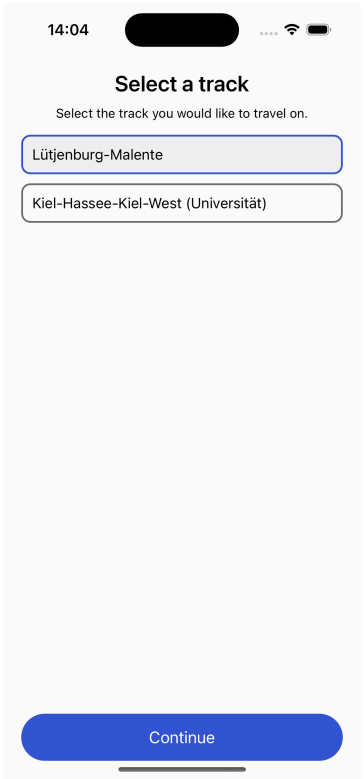


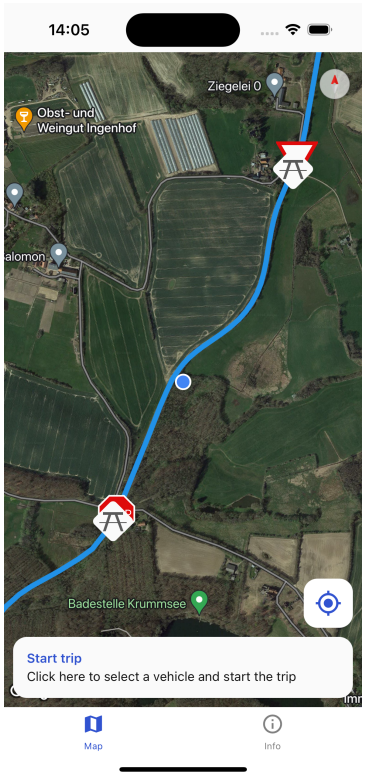
Figure 3.3. User Flow Diagram



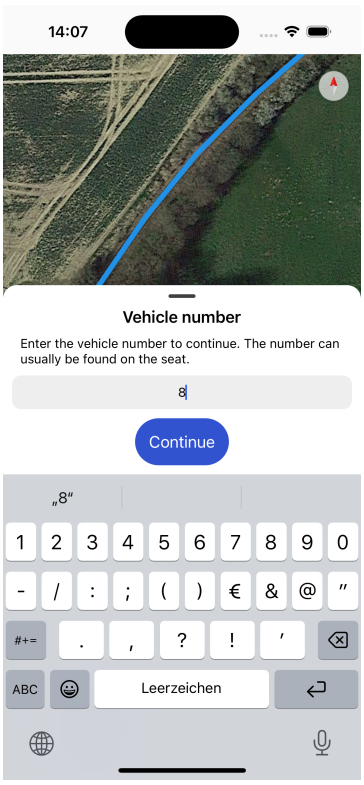
(a) Welcome Screen



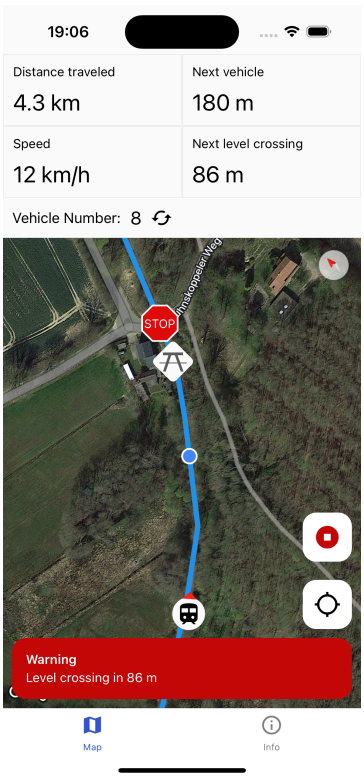
(b) Track Selection



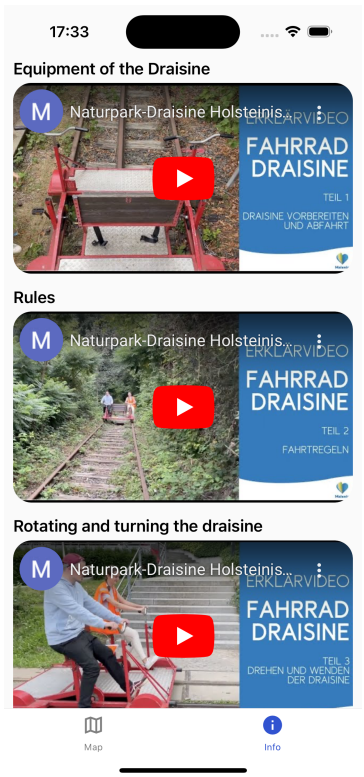
(c) Map Screen



(d) Start Trip



(e) Trip Started



(f) Info Screen

Figure 3.4. Screenshots of the App

3.5 Communication between App and Server

The app relies heavily on data from the server to provide the user with up-to-date information. Evaluating and implementing the use cases resulted in the need for the following API interfaces.

3.5.1 Retrieve Tracks

Users are able to choose from a number of tracks they want to travel on. If they have location data enabled, the track is automatically selected, if not they need to choose the track manually. This API provides this list of tracks from the server to the app.

3.5.2 Initialization

To fulfill the use case of providing extended map data to the user, the app needs information about the track path and points of interest from the server. Additionally, this API provides the track length, which is needed to calculate the distance driven by the user.

3.5.3 Retrieve Vehicle ID from Vehicle Number

To start a trip the app needs the vehicle ID of the vehicle the user wants to drive. This vehicle ID is different from the number that is written onto the vehicle, since the vehicle ID needs to be unique and different tracks could write the same number onto a vehicle. This API converts the vehicle number together with the track ID to the vehicle ID that is used by the server. Additionally, this API checks if the user entered a valid number, so the app can show an error message if the entered number did not match a vehicle.

3.5.4 Update

This API route handles all dynamic data. If the user has granted location permissions, the app uses this API to update the server of the current position of the user. The server uses this information to calculate more accurate position data of vehicles. In both cases the app additionally sends the current vehicle ID, so the server can assign the data to the correct vehicle.

The server's response of this request holds the following data:

- Position of the user: This position has improved accuracy compared to the users position data and is mapped to the track.
- Vehicle speed: The current speed of the user is shown in the header.
- Heading: The heading is used to rotate the map, so the user is driving from the bottom to the top of the screen. This allows for better orientation.

- A percentage-based indication of the user's position along the railway track, spanning from one end to the other. This data in combination with the track length is used to calculate all distances.

- Vehicles: The position and heading of all vehicles that are near the user. This enables the app to show markers of other vehicles on the map and display safety warning if an oncoming vehicle is near the user.

The server provides this data, so the app is able to fulfill the use cases of map information, current position, trip statistics, and safety.

The APIs have been designed with a strong emphasis on ensuring high standards of privacy. The following decisions were made to improve protection of user data:

- User location data is only transmitted to the server when necessary. Before a trip is started, location information is transmitted only once, specifically for the purpose of track selection.

- To anonymize user data, no device fingerprints or user identifiers are transmitted.

- Users are only presented with vehicles in proximity to them, thereby preventing any spying or tracking of other users' location data. The cut-off-point is set by the server and should be in the range of a few kilometers.

The refresh rate of the update route is set to five seconds. This value was set as a compromise to reduce the load on the server while retaining a responsive app, that is able to warn of potential hazards in a timely manner. After further performance optimizations on the server, the aim is to reduce this time to one second for a smoother app experience.

3.6 Calculations

To provide warnings and distance metrics, the app employs different calculations.

3.6.1 Distance between two Points

The application uses the *percentage positions* and track length provided by the server to calculate the distance between two points on the track. Percentage positions express where the position is relative to the start and end point of a track. 0% means that a given point is at the start of the track and 100% that the point is at the end. To calculate the distance between the percentage positions x_1 and x_2 with the track length l, we use the following equation:

$$Distance = |x_1 - x_2| * l$$

Using this formula has the advantage that the actual track path is used for the distance calculation instead of just the air-line distance.

3.6.2 Distance driven

We extend the last formula to sum up all previous positions of the user, to calculate the distance driven by the user. To calculate the distance driven at the time n, we add the distance since the last update y, that we calculated with the last formula, to the last calculated distance driven d_{n-1}:

$$d_n = d_{n-1} + y$$

Summing up all changes in position has the advantage that it ensures that the function is monotonic increasing and reflects all movement of the user. Both properties are important to avoid confusion of the user. The disadvantages include that this formula is accumulating errors such as GPS drift. To enhance this formula, a threshold could be added, that needs to be surpassed before changes in the opposite direction would be added.

3.6.3 Warnings

Warnings are shown to the user if the remaining distance to a potential hazard is less than the specified threshold. The thresholds are set to 300 meters for oncoming vehicles, 150 meters for level crossings and 10 meters for vehicle that are in front of the user and drive in the same direction. Since both vehicles are moving, the threshold for oncoming vehicles is double of the on for the level crossings. Both thresholds give the user 27 seconds of time at a speed of 20 km/h. This ensures that the user has enough time to react.

The user is warned that another user is driving in front of him is he is within 10 meters. This reminds the user that he is not keeping a save distance to the user in front of him. Possible improvements include variable, speed dependent thresholds.

3.7 Permission Handling

To be able to access location data on smartphones, applications need certain permissions. Since this app relies heavily on location data, it is important to handle these permissions in a sensible manner. Figure 3.5 has a visual representation of the different choices and outcomes that relate to the handling of location permissions in the app.

When users start the app, they are presented with the benefits of enabling location data, such as greater functionality and improved accuracy, and have an initial choice whether they want to use location data or not. If the user decides to not use the device's location data, the app uses the location of the vehicle trackers as a fallback option. Since these trackers only update their location every few minutes compared to the five-second interval of the app, this option is less desirable. Other disadvantages include that the user position can only be shown after a trip has been started and vehicle was selected, and that fewer data can be collected to assist the track's future development.

If the user decides on the other hand that they want to enable location data, they are

presented with a system dialog that is required by the operating system. [1] [2] In this dialog the user is presented with three different choices:

- "Don't allow" location access, in this case the behavior is identical as if the location access was denied in the first place.
- "Allow once", which means that the location access is granted for this session.
- "Allow while using the app", that gives the app permission to access the location in the current and future sessions, as long as the app is in the foreground.

If the user allows the app only once to access the location, the foreground location listener is used, because the operating system disables the possibility to ask for access to the location while the app is opened in the background if this option is selected.

In case the user selects the remaining least restrictive option, the app will start using a foreground location listener since the following benefits of the background location listener are only present during an active trip:

- Other users will continue to have accurate position information about the user's vehicle, even if they leave the app.
- Usage data can still be collected even if the user has the app only open in the background, this data could later be evaluated to assist the railway tracks future development.

After the user starts a trip, they get informed that it is recommended to enable background location permissions. Such explanatory dialogs are a best practice before permission dialogs to inform the user why the permission is necessary. In this case the dialog just provides generic information, because the background location access does not benefit the users themselves, although it would still be the preferred option for the railway operator.

In the next step the system dialog is shown and depending on the user's input, either the foreground listener remains the used option or the app switches to the background location listener.

[1] Android: `https://developer.android.com/training/location/permissions`

[2] iOS: `https://developer.apple.com/documentation/corelocation/requesting_authorization_to_use_location_services`

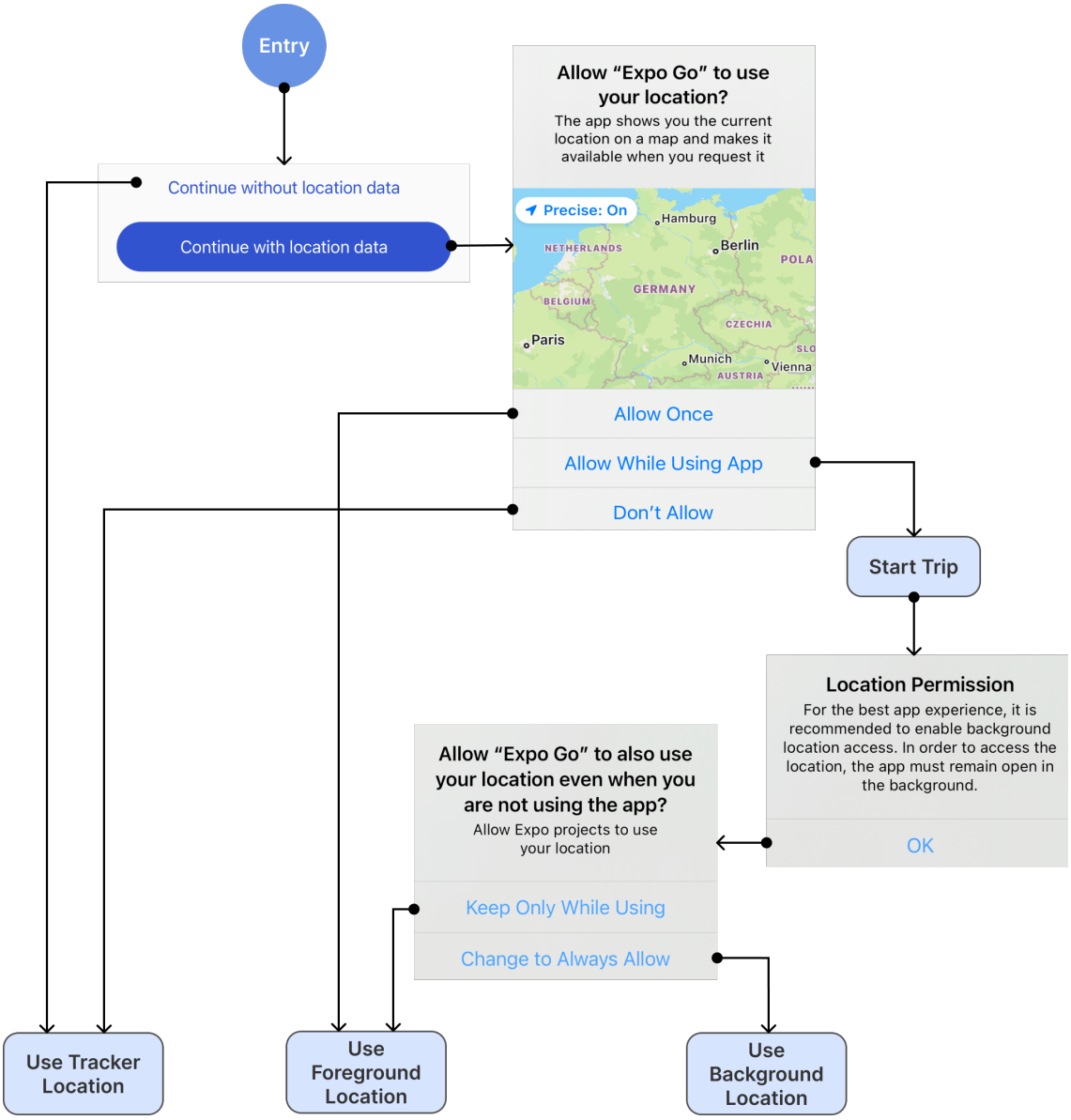


Figure 3.5. Permission Flow Diagram

3.8 Translation Handling

```
export const useTranslation = (): I18n => {
   const i18n = new I18n(translations)
   i18n.locale = Localization.locale
   i18n.enableFallback = true
   i18n.defaultLocale = "en"

   return i18n
}
```

Listing 3.1. Definition of Translation Hook

One use case of the app included accessibility to tourists who do not speak the native language, in this case German. To those tourists it is vital to offer a second language and since English is considered a world language, it was the best choice as a fallback language.

Offering a great user experience, the language should automatically be selected based on the device's system language. To get the current system language the library expo-localization[3] was used.

Maintaining a clean code base while integrating the translation feature was important, since the readability and maintainability of the code should not be impacted by the addition of the feature. To achieve this, the translation logic and translation strings where each extracted into separate files.

The translation logic was extracted into a *React Hook* seen in Listing 3.1. It enables easy usage of the translation logic in the whole app. The hook includes the *I18n* library that was introduced in Section 2.4.4 and initializes all important properties such as translation strings, device language, and fallback language.

The translation strings are extracted into an object which has a property for each language. Each of the language properties have one property each for every string that is used in the app. The extraction of the strings into this object allows for enhanced maintainability, since all strings are bundled at one location, as well as the possibility to easily extend more languages in the future.

[3] https://github.com/expo/expo/tree/sdk-49/packages/expo-localization

Chapter 4

Evaluation

This chapter evaluates the feedback that I could gather by users of the app. Additionally, this chapter looks back at the proposed requirements and if and how they were fulfilled.

4.1 Evaluation of Feedback

One month before the end of this thesis, a test drive was conducted on the railway track between Malente and Lütjenburg. Participants in the test drive included the professor of the working group, one of the supervisors of this project, and four doctoral students of the same working group, but with no direct involvement into the project. The process of gathering feedback was carried out in two stages: Direct interaction with participants during the test drive and providing feedback forms to the participants to fill out. The feedback form focused on four key aspects:

- "Did the use of the RailTrail app improve your perceived level of safety?"
- "How useful was the displayed information in the app?"
- "How intuitive is the app? Were there parts of the app that you needed an explanation for?"
- "Which features did you miss in the app?"

These questions were asked to evaluate whether the app effectively fulfilled its tasks in terms of safety and added value to the user. Furthermore, they aimed at finding areas where improvements could be made in terms of user interface and functionality.

4.1.1 Threats to Validity

During the test drive, the backend server experienced performance issues, resulting in prolonged response times and inaccurate position data. Since this thesis primarily focuses on the app, testers were instructed to focus on the app's performance. Although this situation introduces a potential threat to validity, it is noteworthy that testers were able to imagine how the app would perform under normal conditions.

Another potential threat to the validity of the results comes from the composition of the test group. All test users are members of the same working group where this thesis is written. This may result in biased feedback and a deeper understanding of the app compared to an

average user. While this circumstance does not make the evaluation invalid, a potential bias should be taken into account while evaluating the feedback.

In summary, the feedback provided can be evaluated within the context of this thesis. However, for a more comprehensive evaluation the test should be repeated at a later stage with a neutral user base.

4.1.2 Perceived Level of Safety

The testers of the app overwhelmingly described, that the delayed position information resulted in no improvement of the perceived level of safety. However, they pointed out the belief that accurate position information would enhance safety. Some testers also noted that using the app could be distracting, leading to reduced focus on the railway track and decreased safety.

In conclusion, under normal conditions the app has the potential to improve safety, but further testing would be necessary for confirmation.

4.1.3 Usefulness of Displayed Information

The information displayed in the app was generally regarded as valuable. Specifically, points of interest and vehicle locations on the map, as well as distance traveled, were found to be helpful by testers.

4.1.4 Intuitiveness of the App

Testers described the app as “intuitive” and “self-explanatory”. Nevertheless, there is room for improvement, particularly in the design of certain app components. Examples include the button to travel back to the own location and the icons on the map, which could be made clickable to provide additional information.

4.1.5 Missing Features

Testers provided valuable suggestions for enhancing the app:

- Implementing the ability to click on points of interest and vehicles to access more information.
- Incorporating oral warnings, similar to those found in navigation apps.
- Including a history of traveled path and speed.
- Providing additional textual information alongside videos.
- Creating a legend that explains the meaning of all icons on the map.
- Adding a button to quickly zoom in and out to view the entire track.

4.2 Completion of Proposed Goals

At the very beginning of writing of this thesis I wrote a proposal that included certain requirements the app should have. These requirements were divided in hard requirements, that are a core part of the thesis, and soft requirements that would be nice additions to the app, but not necessary.

4.2.1 Hard Requirements

The first proposed hard requirement was to show the current position of the user on a map. This goal was realized by marking the user's position with a blue dot on the map.

Another requirement was to inform users about upcoming level crossings, this feature was implemented by displaying the remaining distance to the next level crossing at the top of the screen and using a warning banner at the bottom of the screen when the user is within a certain distance of the crossing.

Furthermore, to warn users of imminent head-on collisions with other vehicles was set as a goal and realized by showing a prominent, red warning at the bottom of the screen, if a vehicle is within a certain distance in front of the user's vehicle, moving in the opposite direction.

Displaying the current vehicle speed was also an important goal, which was accomplished by showing the speed in the top left corner during a trip. The precision of the speed shown to the user is depending on how fast the user is driving. Below the speed of 1 km/h the shown speed is displayed as zero, to prevent displaying erroneous stationary speeds due to GPS drift. From 1 to 10 km/h the speed is rounded to one decimal place, while speeds above 10 km/h are displayed without decimal places.

The next required feature was to send the current position of the user to the server. The goal of this feature was to gather additional information about the track usage and improve the position accuracy of vehicles by using the app as another tracker. If the user has location permissions enabled, the app updates the backend every time it gets new location data from the system's location API.

One of the most important hard requirements was to show points of interest, such as level crossings, on the map. This requirement was later extended to also include the path of the track. The track path is marked on the map by a blue line, while icons are used for the points of interest. The icons were made in Figma[1] and closely resemble the signs at the track for improved recognition. If the user zooms out, at a certain threshold the icons get smaller in an attempt to preserve readability of the icons.

The last hard requirement stated that data such as points of interest should be retrieved from the backend server. In the end 5 RESTful APIs were implemented and nearly all data presented to the user is retrieved from the server.

In summary, all the initial hard requirements proposed at the beginning of this project

[1] https://www.figma.com/

were successfully met, highlighting the alignment between the project's initial vision and its completion.

4.2.2 Soft Requirements

The first of the proposed soft requirements was to show optimal positions for users to pass each other. This feature was implemented in the code by showing an icon at the proposed passing position, that was sent by the server. Since the server is not yet sending these positions, the feature remains non-functional despite implemented.

Another optional goal was to allow users to change their vehicle during a trip. It was achieved by displaying the current vehicle number at the top of the screen, along with a small icon for changing the vehicle number. Future improvements might include automatic or assisted detection of the current vehicle.

Furthermore, showing users safety instructions like how to secure a level crossing was set as a goal and implemented by a second tab on the screen called *Info*. The tab lists helpful videos from the railway operators on safety related topics. The videos seem to be a good way to convey the information, although being track specific. Currently, the user needs to actively go into the tab, so a more seamless integration into the user flow could be considered for future enhancements.

The next soft requirement was to show usage stats to the user. The app displays one usage statistic which is the driven distance by the user, it is calculated by summing up the difference in position between the individual points in time. This method works well if the position provided by the backend is correct, which is not always the case. Testing revealed that if multiple users connect to the same vehicle, the vehicle can jump between multiple positions, resulting in a flawed usage statistic. This issue can be solved by improving the backend's position calculation algorithm. Additional statistics like average speed or time driven would be good additions for the future.

Next, informing users about the distance to the next/previous vehicle on track was an optional feature. Then final app displays the distance to the next vehicle, while the distance to the previous vehicle is left out, as it was deemed not useful enough.

To add a second language and choose the best one based on the system language was another important optional feature which was implemented to its full extent. The app offers German and English as language options and automatically selects German for German-speaking countries and English for all other countries, due to it being set as the fallback language.

The last soft requirement proposed was to add gamification to the app. This requirement was not implemented to any significance and forwarded to future work.

In conclusion, the majority of the proposed optional goals were successfully implemented, but there is still ongoing work to be done to enhance the application further.

Chapter 5

Conclusion

This thesis has shown the conceptualization and development of a real-time mobile cross-platform application with the aim of enhancing the competitiveness of railway transportation.

The evaluation has highlighted the potential benefits of digitizing railway transport through the use of a mobile application. Increasing the robustness and performance of the system would lead to an elevated perception of safety while providing users with valuable information to enhance their travel experience.

Furthermore, the application has laid the foundation for data collection, which can be important in guiding future developments of the railway track. The analysis of these datasets can support future decisions on how to use and improve the track.

In the future the app could be used in production and increase the attractiveness of individual railway transport. To launch the app, a few minor adjustments are needed. Specifically, an actual data protection policy should be used instead of the current placeholder. Additionally, the vehicles on the track would have to be fitted with vehicle trackers, that provide frequent updates. This would ensure that the app consistently displays the locations of other vehicles on the map and fully delivers all its functionalities.

Moreover, the app could easily be used on other railways than the one tested in this thesis. Currently, the *Info* Screen is the only part of the app, that is not dynamic to the selected track. If other railway tracks have different safety guidelines or vehicles, adjustments would have to be made.

5.1 Future Work

While the app fulfills fundamental requirements, there is still a lot of room for future improvements. The following list includes new and left over ideas during the implementation as well as feedback that was gathered by testers of the app.

5.1.1 Better Implementation of SVG Support

The app uses many SVG images for markers on the map, currently these SVGs need to be converted to *JSX Components* to be used in the app. Even though there are websites[1] that make the conversion fast and easy, native SVG support would improve the maintainability of

[1] https://react-svgr.com/playground/?native=true&typescript=true

the application. Additional research would be necessity to decide if the current technology constraints allow for a simpler solution of this problem.

5.1.2 Showing GPS Accuracy on the Map

Handling GPS data always comes with some uncertainty regarding the accuracy of the information. Testing showed that these uncertainties can accumulate if multiple devices connect to the same vehicle. Visually showing this uncertainty radius with a blue circle could reduce confusion for the user in these scenarios.

5.1.3 Add Acoustic Warnings

Since warnings are only useful if the user actually notices the warning, the chance of it happening should be maximized. Acoustic warnings could be a sensible addition to the visual warnings currently implemented in the app. Since draisines are very loud vehicles, further testing would need to evaluate if the phone speakers are loud enough for the user to hear the warning.

5.1.4 Prompting Users when Location Matches other Vehicle

During a trip it is quite possible that the user switches vehicles. If this happens the user currently also needs to switch the vehicle in the app. If a user forgets to inform the app about the vehicle switch, it can result in flawed data and less accuracy. When the backend notices a bigger deviation between the position of the user's phone and the vehicle tracker and additionally the phones position matches another vehicle, the app could ask the users if they switched to that vehicle.

5.1.5 Using Internal Position when Server does not Respond

Testing showed that the app has a very high dependence on the backend server with no implemented fallback. While a connection to the server remains as a necessity for most functionality, the app could use its own position data of the user as a fallback option if the server does not respond within a certain time frame.

5.1.6 Icons on the Map

Feedback by testers highlighted that the meaning of some icons was not intuitive enough and that a legend for the map could help. Additionally, the icons on the map could be made clickable to explain the meaning of the icon and give additional information, such as a vehicle number for other vehicles.

5.1.7 Show Trip Statistics when Trip is Finished

When a user finishes a trip, the app could show a screen with the statistics of the trip taken. These statistics could include distance, average speed and time driven.

5.1.8 Show History of Trips

Currently, the app forgets all information about a trip once it was finished. A history of the taken trips would be a nice addition for the user to better remember trips and see personal progress.

5.1.9 Gamification

Gamification could be a good addition to the app to improve the enjoyment of the user. Potential ideas for gamification in the app include:

- Letting the user search and find certain points along the track.
- Compare and show statistics between users, such as "You have driven further than 80% of users".

Bibliography

[HPK23] Benedikt Hertel, Johannes Pagenkopf, and Jens König. "Challenges in the (re-) connection of peripheral areas to the rail network from a rolling stock perspective: the case of germany". In: *Vehicles* 5.3 (2023), pp. 1138–1148.

[Sco19] Damiano Scordamaglia. *Digitalisation in railway transport: a lever to improve rail competitiveness*. 2019.

List of Figures

Listings