TelaSol

A coach cockpit application

L.V. Gerlach
B.F. Janssen
M. Kroon
S. Vijlbrief

TU Delft

TelaSol
A coach cockpit application

by

L.V. Gerlach
B.F. Janssen
M. Kroon
S. Vijlbrief

to obtain the degree of Bachelor of Science
at the Delft University of Technology,
to be defended publicly on Tuesday July 2, 2019 at 12:00.

Project duration: April 22, 2019 – July 5, 2019
TU Delft Coach: M.A. Migut
TU SEI Liason: R.R. Dukalski
Team Sunweb Liason: A.H.M. van Erp
Bachelor End Project Coordinators: Ir. O.W. Visser & Dr. H. Wang

Summary

Team Sunweb, a professional cycling team and our client, is constantly looking for innovations to help them win races. They tasked us with creating an application which could assist coaches with determining the strategy during a race. This application, which we dubbed TelaSol, is supposed to run on a tablet that is mounted on the dashboard inside the coach car. For this project we developed an application that allows races to be prepared on a desktop computer and tracked during a race on a tablet-optimized interactive dashboard. On this dashboard, there will be information on the riders, the route and comments that can be added before the race.

During development we have considered existing solutions, relevant literature and useful technologies to get an idea of what was possible and how we could achieve our goal. We used this knowledge to create our initial set of requirements. We then proceeded development of application using an agile approach, which involves regular feedback moments from our client to update the requirements and adjust our focus accordingly. To verify the quality of our product we relied on a combination of automated tests, user testing and validation through the client.

Initially the application was supposed to integrate live data coming from the riders during the race, but due to a regulation change we had to change our focus. Instead, we focused primarily on creating the application for playback purposes, while still keeping it adaptable to live data. The application performs the main tasks that were initially defined properly. After further development on live data and extensive situational testing, the app can be used to its full potential. Using TelaSol, Team Sunweb will improve their ability to analyze races and increase their chances of winning.

Preface

Throughout this project, we have had excellent guidance by a few people. Gosia Migut, as our coach, helped us continuously with improving our process, product and the report you are currently reading. Our client Rado Dukalski provided us with much feedback on our application and with his expertise, he had a large share in making the application as functional as it currently is. Lastly, we would like to thank our liason at Team Sunweb, Teun van Erp, for the insightful videocalls and for inviting us to Team Sunweb's headquarters in Deventer to show off our product.

L.V. Gerlach
B.F. Janssen
M. Kroon
S. Vijlbrief
Delft, July 2019

Contents

1 Introduction

Professional cycling is a sport with a lot of history: This year's Tour de France will be the 106th edition. Since that first race in 1903, the sport has seen many changes and innovations. One of the more recent innovations is the usage of science and data. Riders have little boxes on their bikes that measure all their vitals such as heart-rate, speed and power. Using such data, riders themselves but also their team coaches can carefully plan their races. In addition to this, coaches can know more precisely what their riders are capable of. These new technologies have had a large share in Team Sky's (now Team INEOS) many victories. To close this gap to Team INEOS, other cycling teams have sought out their own technological solutions.

There is a variety of scientific areas that contribute to innovations in the area of professional cycling. Some examples of important studies in the field are: the physiology of riders [24], the ideal cadence to cycle at [22] and of course motion physics of bicycles [23]. But also other scientific areas such as nutrition [20] [25].

The field of computer science enhances innovations in professional cycling through data analysis and visualizations. Currently, coaches rarely have an idea of what their riders are up to during a race, besides through infrequent contact via earpieces. To improve this, professional teams have been looking at applications that are connected to the data of their riders, to get a more clear overview of the race. An example of such an application, is VeloViewer [13]. This application can be used for static analysis of past races and trainings. It also features a live version that can be used during races. The current issue with applications such as VeloViewer however, is that they are very expensive and do not provide teams with much freedom to input their own features.

To address these issues, the professional cycling team ”Team Sunweb” has tasked us to build a coach cockpit application. Our project addresses the cost- and freedom-related issues mentioned in the previous paragraph and provides the user with a great deal of integrated features, both on live and past races. In addition to that, the app is built lightweight and adaptable and as such can be adjusted to the user's wishes. In this way, our application serves us an attractive alternative over the currently available applications.

In this report we describe the process of building the application along with the scientific background and evaluation of the product. In chapter 2 we line out the research we have performed leading up to the application. In chapter 3 we describe the different phases of the application and the various implementation choices we made. Our interaction with the client and the way we validated the functioning of our software, are described in chapter 4 and chapter 5. Our conclusion about the project can be found in chapter 6. The appendices features a variety of supporting documents such as meeting notes, SIG feedback, the original project description, a user guide, technical guide and the info sheet.

2 Research Phase

Before we start building the application, research should be performed. We have split this research up into four different parts:

1. **List of Requirements**: We record which requirements we have for the project and how we prioritize them.
2. **Literature Review**: We evaluate existing literature on subjects relevant to our application.
3. **Existing Tools**: We consider existing applications that perform similar tasks and evaluate which of their features we consider valuable for our own application.
4. **Technology Options**: We note down our different options for various technologies for the application.

2.1. List of Requirements

In collaboration with Team Sunweb, we propose the following list of requirements. Our liason at the team, Teun van Erp, has cooperated with us in composing this list. For these requirements, we will consider two different attributes: Feasibility and Importance. The explanation of the attributes can be found below this paragraph. The aggregation of these two attributes will be used to divide the requirements over the different categories of the MoSCoW method [16]. As an example, if a requirement has both a high feasibility and a high importance, it will most likely be considered as a ”Must Have” from the MoSCoW method.

The initial list of requirements will be defined at the beginning of the project (Stage 1). We will also redefine this list another time, after the midterm meeting with Team Sunweb (Stage 2). This midterm redefinition will allow us to clearly lay out the issues for the latter half of the project. There will also be two evaluation moments to see how many of these requirements were completed, which can be found in section 5.2. These evaluation moments will be halfway through the project for Stage 1 (during/after the midterm meeting with Team Sunweb) and at the end for Stage 2 (during/after the final meeting with Team Sunweb).

Classification

- *Feasibility*: The level to which we believe we will be able to create this feature within the given time and financial constraints (values can be between 0 and 3).
- *Importance*: The degree to which we believe this feature to be important. This is based on our meetings with the client and our own insights into their wishes (values also between 0 and 3).

2.1.1. Stage 1

1. **Live data from the cyclists (e.g. Heart Rate, Cadence)**
   All the riders from the team in race should have their main details displayed on the screen. As there will be 6-8 riders, we intend to use the right half of the screen to display all of these at the same time. When clicked on a specific rider, the screen (or half) will be covered with more detailed information. A potential extra feature would be all the riders currently in course (thus also those of other teams). This adds the extra difficulty of obtaining their data which will most likely be much more limited.
   Feasibility: 3/3, Importance: 3/3

2. **Live position of the riders on a map**
   The left half of the screen should be covered by a map of the race. On this map the riders should be moving through the course as they are in the race. More details about the rider can be found when clicked on.
   Feasibility: 3/3, Importance: 3/3

3. **Add notes from team leader in advance with Google Streetview**
   Team Leaders want to be able to explore the race beforehand. They want to use streetview to make annotations about dangerous points on the race course. This could be integrated into the app but it’s undecided whether this will receive an extra screen or not.
   Feasibility: 2/3, Importance: 3/3

4. **Tablet based design**
   The app should be functioning for tablets. As this is the main platform the team leaders will use the application on, it is essential that the app is adjusted to perform on this platform. The power of the tablet is fairly unimportant at this point in time.
   Feasibility: 3/3, Importance: 3/3

5. **Ideally working with unreliable connection**
   Our application should accommodate for a loss of internet connection, as the coach cars will be without a connection in the mountains throughout a large portion of the race. Fixing this is also not a major focus of the app at this point as we will first focus

on making an app based on a reliable connection. After the app has been established with a reliable connection, we can look at connectionless expansion.
Feasibility: 1/3, Importance: 2/3

6. **Extensible for data sources other than csv file**
   Currently we will be provided with a .csv file containing a clear overview of the riders in the race. This might not always be the case, especially when concerning data from other teams (this is thus directly related to point 1). After having built the app using the available data, we can take a look at expansion options.
   Feasibility: 2/3, Importance: 1/3

7. **First on recorded race, then also on live data**
   At first we will built the app with data of an already finished race. We intend to add a scrollbar at the bottom of the screen such that moments in the race can be manually selected. We will also add a play button that allows the user to play the race as if it were occurring in real time. In this way, a live race is simulated already. As such, the difficulty of adding a live race will be limited to the data inflow only.
   Feasibility: 2/3, Importance: 3/3

8. **Integration with video images**
   There are many videos to be found of races. These can be very useful for teams to evaluate their decisions and results. As an extra feature we could integrate these video images into our application such that they can be evaluated alongside the data in our application.
   Feasibility: 2/3, Importance: 1/3

Below the four categories from the MoSCoW method can be found along with the aforementioned requirements, sorted by category. Every requirement will feature a brief explanation of why we categorize it as such.

- **Must Have** *1, 2, 3*: These are core features of the Veloviewer and Veloviewer live app that need to be combined into our application. These features form the core of our application.

- **Should Have** *6*: A very relevant issue for this stage. The application should already support as much as possible to prevent problems later on. Not having it now however, will make the product still function albeit less.

- **Could Have** *4*: While this feature is quite important for the end product, it is not of vital importance to have this finished at this stage of the product.
  7: This feature could potentially already be added at this stage. For this feature to be incorporated however, Team Sunweb needs to provide us with their live connection which is very unlikely at this stage.
  8: While it is technically possible to add this, it is not of great importance at this stage and will thus be added only if time allows it.

- **Won't Have (at this stage)** *5*: While potentially important, this is something for much later in the application, if even still relevant at that point.

2.1.2. Stage 2

In this subsection we will highlight the most important requirements from the second phase of the project, in the same way that we did for the first phase. Some from the previous stage that were unfinished will logically be carried over.

1. **Tablet based design**
   The app should be functioning for tablets. As this is the main platform the team leaders will use the application on, it is essential that the app is adjusted to perform on this platform. The power of the tablet is fairly unimportant at this point in time.
   Feasibility: 3/3, Importance: 3/3

2. **First on recorded race, then also on live data**
   At first we will built the app with data of an already finished race. We intend to add a scrollbar at the bottom of the screen such that moments in the race can be manually selected. We will also add a play button that allows the user to play the race as if it were occurring in real time. In this way, a live race is simulated already. As such, the difficulty of adding a live race will be limited to the data inflow only.
   Feasibility: 2/3, Importance: 3/3

3. **Integration with video images**
   There are many videos to be found of races. These can be very useful for teams to evaluate their decisions and results. As an extra feature we could integrate these video images into our application such that they can be evaluated alongside the data in our application.
   Feasibility: 2/3, Importance: 2/3

4. **Ideally working with unreliable connection**
   As the cars will be without an internet connection in the mountains throughout a large portion of the race, our app should accommodate for this. This is also not a major focus of the app at this point as we will first focus on making an app based on a reliable connection. After this has been established, we can look at connectionless expansion.
   Feasibility: 1/3, Importance: 1/3

5. **Race and Training versions**
   As the Interational Cycling Union (UCI) has forbidden the use of live rider data during a race (more details on this in subsection 4.2.2), there needs to be a version of the app that is permissible during live races. All the other features that we have built which are currently not allowed under UCI regulations, still need to exist in the training version of the app.
   Feasibility 3/3, Importance: 3/3

6. **More information about climbs**
   As climbing is one of the important characteristics of many cycling races, it is important that climbs are displayed well in our application. We plan to achieve this by having specific comments for climbs, altitude profiles accompanying those comments and a more elaborate height profile.
   Feasibility 2/3, Importance: 3/3

Below the four categories from the MoSCoW method can be found along with the aforementioned requirements, sorted by category. Every requirement will feature a brief explanation of why we categorize it as such.

- **Must Have** *1, 2, 5, 6*: These features are very important to meet Team Sunweb's wishes of a functional app that is a better alternative over Veloviewer (live). These are absolute core features of the second stage of the project.
- **Should Have** *3*: While this is a requested feature that could really improve the app, the technical feasibility as well as the actual usefulness of it, remains to be determined. As such, this feature is of less importance, yet still quite important.
- **Could Have** *4*: This issue did not come up specifically during the midterm meeting with Team Sunweb. We will however maintain it as a ”Could Have” since it could still come up during the implementation of the live version.
- **Won't Have (at this stage)**

2.2. Literature Review

In this section we evaluate existing literature in order to gain understanding about the field in which we are building this application. We consider the core aspects of our application and look for relevant literature on those. The relevant questions we ask ourselves are:

- What data is available in professional cycling and which parts are important?
- What are the state-of-the-art solutions in sport visualization?
- What are the design guidelines for creating a tablet-based application?

We will now discuss our findings and point out what we think is relevant knowledge from each of these fields ahead of developing our application.

2.2.1. Data in Professional Cycling

Professional cycling has been quickly moving into the digital age and the reliability on data has become very significant. Teams are analyzing everything they can get to know about their riders and an increasing number of teams are taking a highly scientific approach to training and racing. This approach has been paying off, as teams that are leading in this scientific approach like Team INEOS and Team Sunweb have been getting great results.

The data that is most essential for our application is the live tracking data of the riders during a race. This data is a combination of bicycle data (e.g. speed, cadence), physiological measurements (e.g. heartrate) and GPS coordinates. All these types of data are collected from a small sensor fitted under the saddle, which sends it to a data collection hub, from which it can be distributed to all interested parties [17]. These include the broadcasters, as spectators are interested to know where their favorite riders are and how they are faring in the race, but they also include the team coaches driving alongside the riders. The coaches use the data from their riders to decide on the tactics of the team.

As such, the aim of our application is to use this variety of collected rider data and visualize it in an intuitive way. The app should provide coaches with the ability to make decisions in the race, based on the displayed data. Displaying the right data and displaying the data in the right way, is the challenge of the application. These issues will be discussed in the following sections.

2.2.2. Visualization in Sports

With the quick development of technology, the access to all kinds of live data and statistics is increasing rapidly, and with it are the options for visualization. In their paper about the state of the art of sports data visualization, Perin et al. [27] specify three categories of sports data: box-score data, tracking data and metadata. For our application we are mostly interested in visualizing tracking data, since that is the category under which the live race data falls. The description given by Perin et al. for “tracking data” is that it is often a series of spatio-temporal events that consist of multiple dimensions. This is indeed the case for the data which we use, since the data consists of the location of a rider at a certain point in time combined with the speed, power, heart rate etc. as the other dimensions of the data. We will consider how we can apply methods of data mining and smart algorithm methods to aid us in the creation of our application in accordance with the instructions of the client.

Identifying the type of data that we will be displaying is merely the initial step. The crucial issue in our application is how to visualize it properly. Proper visualization is especially important as the team coach needs to be able to quickly analyze the position of the riders as well as their vital data signs. In their paper about visualizing measurements to improve performance, Nieuwenhuizen [26] describes different visual analysis tools based on the given data. One of these tools is a trajectory plot for positional data. This is something we also need in our application to display the route, riders and comments on a map.

2.2.3. Effective Data Visualization on Tablets

The application that we build , will mostly be used on tablets. This means that it must be compatible with a screen that is smaller than a PC screen. In addition to this, the app should be functional with touch controls. There are some control elements such as dragging and double clicking that are very hard to use on touchscreens. Therefore, avoiding features such as these should be an important base of our development process. Another challenge we face is the balance between showing a lot of data concurrently while keeping the interface organized. These problems are the reason that some research on data visualization on tablets is needed.

In their paper about effective data visualization on tablets, Games & Joshi [19] point out a handful of pitfalls and give some recommendations. We will briefly highlight the ones most relevant for our application:

- **Avoid overloading on widgets**: A large amount of interactive data elements will create confusion and will make the user have to tap too much to use the application to its full potential.
- **Avoid adding interactive functionality to numeric quantities**: Users do not see numbers as interactive elements. Since our application will definitely display a multitude of numeric values we should take this advice into account.
- **Use tables and charts**: This has been proven to be effective, since users find them the easiest form of visualization to understand.

Due to the chaotic nature of cycling races, it is important for the application to minimize errors made by the coaches during the races. Coaches must be able to access the most relevant information quickly and effortlessly. One of the guidelines we will use to ensure this, is Fitts's Law. This law dictates that the time required for a human to accurately move to a specific control (e.g. a button) depends on the distance to the target control, and the size of the given control. The law is described in detail in the paper by Fitts [18], and is a very useful and relevant model to adhere to in modern interface design. We will incorporate this into our application by ensuring buttons that are likely to be pressed together are close together, to minimize travel time between the control elements. In addition, we will ensure all controls and buttons are large and easy to click. An example of this is the action of switching between riders; instead of having a small button on the rider to move them into focus, the entire rider card can be clicked, which thus provides a very large control element.

2.3. Existing Solutions

In order to develop a tablet-based coach cockpit application, it is important to research existing tools that provide similar solutions. By familiarizing ourselves with these tools, we can determine whether a certain feature is useful for our application and in which areas these other tools are lacking. The tools that will be described in this chapter can be divided into two categories: Live Data and Evaluation Data. Live data applications, such as Veloviewer Live [13], are used to display the data of the cyclists during a race. Evaluation data applications, such as Strava [12] and Veloviewer, are used to visualize the data after the race. In the next sections, these applications will be described in more detail and we will consider which of their features are useful for our application.

2.3.1. Strava

Strava is an application for runners and cyclists to track their data using GPS. During trainings and races, each cyclist has a Quarq Qollector [10], which is a small box behind the seatpost, which captures data such as power, heart rate, gear selection, footpod stride rate, location and speed. After the session, this data can be uploaded to Strava which then displays a very clear overview of the rider's activity. There are many more features, such as making training plans, connecting with friends and sharing data amongst them, but those are not really used by the cycling teams. Strava is primarily used to load its data into Veloviewer, which is described in the next paragraph.

2.3.2. Veloviewer

Veloviewer is an alternative for viewing your Strava data and since it is especially designed for cycling, it is a better option for the cycling teams. Besides the general data like total distance, average and maximum speed, cadence, heart rate, energy, etc., there are also some more advanced features which are described below.

Map This is a very simple but certainly not unimportant feature. It shows not only the route of the race, but also a 3D-route map of the stage allowing the user to anticipate the next climb.

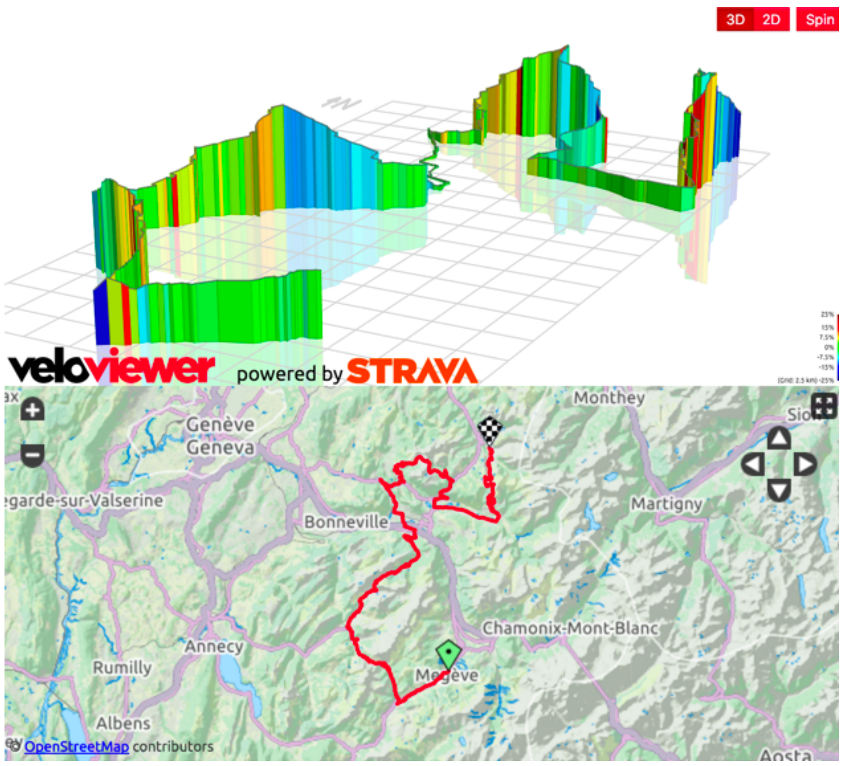


Figure 2.1: The race course in Veloviewer

Data Using the data tab, the user can select any point during the race on a 2D-profile of the stage, and view his data at that specific point like speed, power, cadence, heart rate, etc.

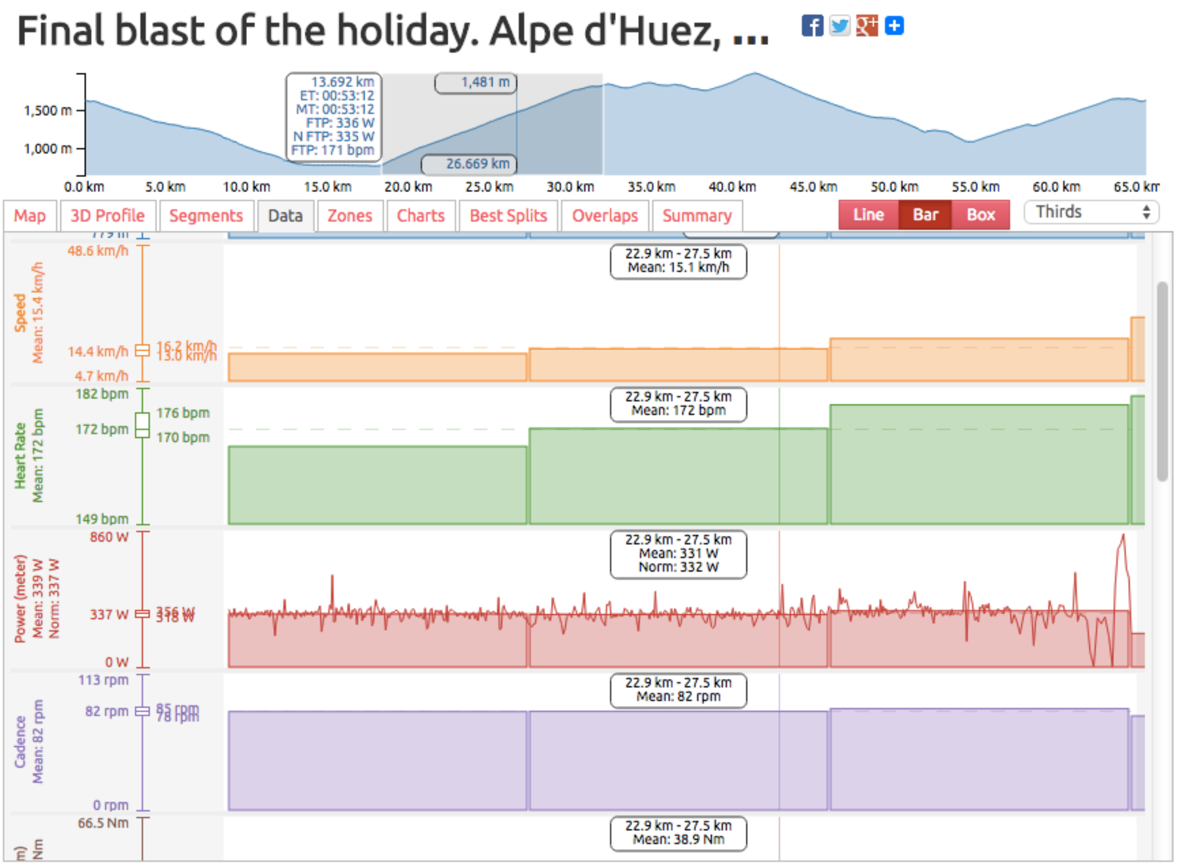
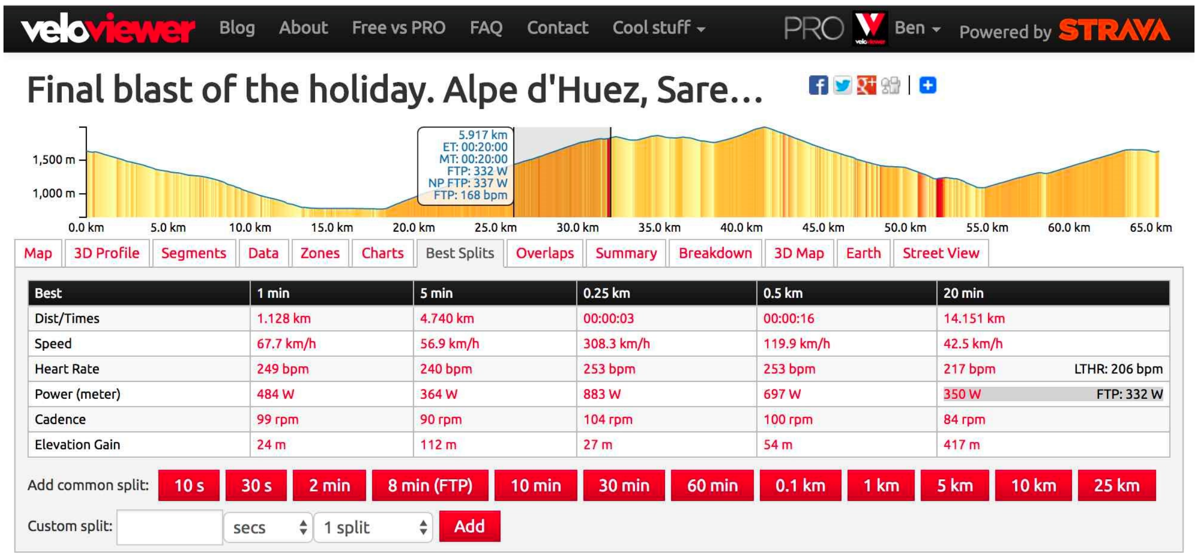

Final blast of the holiday. Alpe d'Huez, ...
13.692 km
ET: 00:53:12
MT: 00:53:12
FTP: 336 W
N FTP: 335 W
FTP: 171 bpm
1,481 m
26.669 km
Map
3D Profile
Segments
Data
Zones
Charts
Best Splits
Overlaps
Summary
Line
Bar
Box
Thirds
Speed Mean: 15.4 km/h
22.9 km - 27.5 km Mean: 15.1 km/h
Heart Rate Mean: 172 bpm
22.9 km - 27.5 km Mean: 172 bpm
Power (meter) Mean: 339 W Norm: 337 W
22.9 km - 27.5 km Mean: 331 W Norm: 332 W
Cadence Mean: 82 rpm
22.9 km - 27.5 km Mean: 82 rpm
22.9 km - 27.5 km Mean: 38.9 Nm

Figure 2.2: The data of a rider in Veloviewer

Best Splits The best splits tab can be used to discover the user's best splits over any distance or time period. The user can specify a time or distance and it will display their best splits. When clicking on a specific split (dist/times, speed, heart rate, power, cadence or elevation gained), it will highlight that part of the race.



Best	1 min	5 min	0.25 km	0.5 km	20 min
Dist/Times	1.128 km	4.740 km	00:00:03	00:00:16	14.151 km
Speed	67.7 km/h	56.9 km/h	308.3 km/h	119.9 km/h	42.5 km/h
Heart Rate	249 bpm	240 bpm	253 bpm	253 bpm	217 bpm LTHR: 206 bpm
Power (meter)	484 W	364 W	883 W	697 W	350 W FTP: 332 W
Cadence	99 rpm	90 rpm	104 rpm	100 rpm	84 rpm
Elevation Gain	24 m	112 m	27 m	54 m	417 m

Add common split: 10 s | 30 s | 2 min | 8 min (FTP) | 10 min | 30 min | 60 min | 0.1 km | 1 km | 5 km | 10 km | 25 km

Custom split: secs | 1 split | Add

Figure 2.3: The best splits in Veloviewer

2.3.3. Veloviewer Live

Veloviewer Live is the application that is used during the races. Before the race, way markers can be added to the route in the team's race hub. This can be feed zones, hazards, sprint points or anything else of interest. These markers are listed on the right of the screen as a countdown which updates constantly based on the location of the device. When tapping on a way marker, more information will be shown. Once the driver is within 5 kilometers, the way marker goes orange and it will go red when within 2 kilometers. Above the list of way markers, it displays the distance travelled from the start and the distance to the finish as well as a coloured profile of the next station of the route based on the gradient of the road.

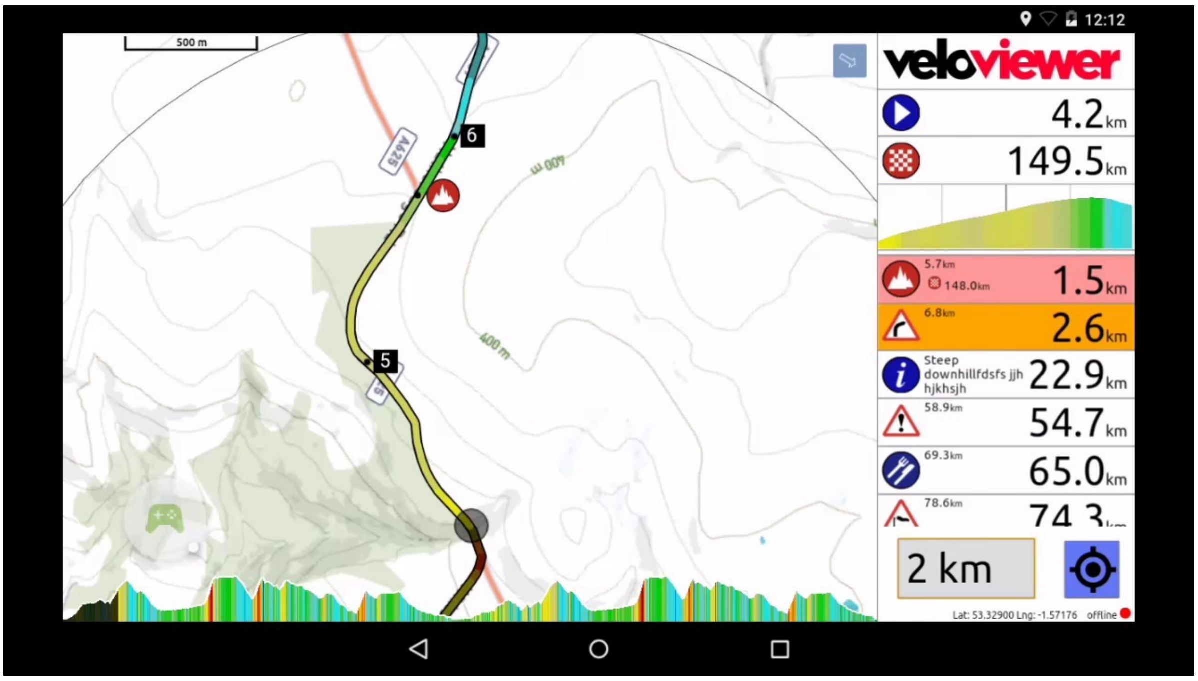


Figure 2.4: The user interface of Veloviewer Live

Shortcomings Unfortunately, VeloViewer Live does not give the users of the application a lot of control over exactly what markers they want to use. The selection of markers is dictated by VeloViewer which limits the teams in their ability to use markers suitable for them. In addition, VeloViewer Live requires substantial payments for usage which is problematic for teams on limited resources.

2.4. Technology Options

In this chapter we will discuss the most relevant technological aspects for our application. These aspects are:

- Data Visualization
- Mapping
- CSS Framework
- Front-end framework
- Back-end framework

2.4.1. Data Visualization

Data visualization is an important part of our application, and picking a suitable library for this is quite challenging. There are many different options that are all a trade off between complexity and customizability, but unfortunately some of the best choices are not free.

One of the options is Chart.js [4], a very simple but free library. Unfortunately we believe this will be too limiting for our purposes and, additionally, does not handle large datasets well at all. While most of the data per graph will be relatively small, our application will also be running a map and updating both the map and charts in real-time. This means that optimal performance is key to getting our application to run well.

Highcharts [9] is a much more powerful and efficient library that we have experience with from previous projects. This would likely be our choice for this project also, except for one major downside: Highcharts requires a considerable amount of money to use for commercial purposes, which we would like to avoid if possible.

Google [7] offers a set of good-looking mobile friendly visualization options that are easy to use. Unfortunately they are very limited in customization options and require a connection to Google servers to use. As we already are dependent on Google this may not be a major problem, but the lack of customizability may be as we need a small number of more complicated components.

Lastly we have D3 [6], which is a very powerful but complex library. Additionally it is entirely free and offers many different examples and documentation. We may use this for at least some of our components, but perhaps we could consider using another library in addition for some of the less complex components that would benefit more from simplicity.

2.4.2. Mapping

The most common choice for mapping libraries is Google Maps [8]. However, unfortunately Google Maps is not free and as such we would like to consider different options before settling for this. Additionally it may not be the most privacy-sensitive option.

Cesium [3] is a powerful library for displaying maps, specifically 3D maps and globes. This is an excellent choice that will give us a huge amount of customizability, as well as being fully open source which can be helpful in some cases. Unfortunately, Cesium requires payment for larger companies, and does not offer Street View, which we may need for our application.

2.4.3. CSS Framework

To alleviate some of the CSS-related work, we should find a suitable front-end CSS library for use in our project. While there are numerous options available, we believe Bootstrap [2] to be the best choice because of their well-designed grid system. In addition, we have experience in using Bootstrap, which means we will save ourselves valuable time by using this. Overall the choice for this library is not the most important as it will only help us set up the layout and give us a consistent style, we can always customize or replace it if we feel it does not need our requirements later on in the project.

2.4.4. Front-end Framework

For our front-end framework there are a number of competing choices to consider. For our choice, we will considering the following points:

- Experience with the options
- Speed of loading the page
- Development velocity
- Simplicity

One option is to use simple HTML with jQuery. While this will work just fine and provide us with great speed, as well as being experienced in using this, it will be extremely slow to develop as we have to write a lot of code that other options will automate for us. On top of this, the code will grow to be hard to read and make matters even worse. As such we should consider a different option.

Angular [1] is an option developed by Google that is unfortunately somewhat outdated already. Most Angular developers seem to dislike the framework a lot and as such, we think it would be wise for us to stay away from it. It is simply too complicated and slow to develop.

React [11] is developed by Facebook and is very popular. However, React is quite complicated and cannot be easily mixed with regular HTML. This means that our application be needlessly complex. In addition, React is a very large code base and requires time consuming compilation to work during development, as well as being slow for loading pages.

Lastly we have Vue [14], which is a small and lightweight framework that will still offer us numerous benefits by automating tasks and providing a good structure for our code. Page loads may be slower than plain HTML, but it will help us in improving the simplicity of our application and, more importantly, significantly speeding up development. Vue can also be mixed quite well with plain HTML which will help improve page load times and keep simple parts of the application simple.

2.4.5. Back-end Framework

For our back-end, the main choices available are PHP and Node.js. Both of these have their advantages and disadvantages. However, we know that our application will be sending live updates to clients. This is best done using websockets [15], a technology to allow continuous connections between the client and server. These websockets are unfortunately not supported at all on PHP, which means that we cannot build our application optionally using PHP. Therefore we will have to use Node.js to host our back-end. This has some other advantages too, such as ensuring that our back-end can maintain its own connection to a cloud provider (such as Strava) to collect the data. With Node.js, we will use the well-known Express library as it is ubiquitous and easy to use. We are also experienced in using this which will save us some development time.

3 Method

This chapter will outline the way in which we built our application. It is divided into several parts:

1. **Application Outline**: A general description of what our application offers.
2. **Development Principles**: The basis for development process.
3. **Technology Options**: A brief description of what technologies and libraries we made use of.
4. **Core Features**: This section features a description of five of the most important features in our application. The subsections explain both how the features work technically and what their function is.
5. **Development of User Interface**: This section outlines several stages of the user interface and describes why and how it changes throughout the development cycle.
6. **Ethical Considerations**: During this project, we had to handle some sensitive data. In this section we will shortly describe the ethical choices we made.

3.1. Application Outline

Our own solution, TelaSol, takes the most desired features of the applications mentioned in chapter 2, such as VeloViewer (Live) and Strava, while eliminating the most critical flaws. Our application supports adding comments and viewing them and their distances during a live race, similar to VeloViewer Live. However, our application allows the user to have full control over which markers they want to use. Unlike VeloViewer Live, our application will not come pre-loaded with all of the upcoming races. Instead this is left up to the teams themselves, giving them greater control and allowing them to add training races.

Additionally, TelaSol builds on the live view by showing the positions of the riders in real-time. This is a huge improvement over existing solutions which are limited to the position of the current user, which is typically a coach car. This way, our application can offer greater insight into the status of the riders and provide the coaches with more useful and accurate information. There is also a height profile of the race present, which is also synced with the live display of riders.

Syncing is an important feature of our application. When the application is opened on two different systems, their playback is fully synced. This means two coaches can follow a race at the same time.

3.2. Development Principles

For the development of our application, it is important to be responsive to user feedback as the needs and requirements of our client are constantly changing due to new regulations, new ideas and innovations in technology. As such we have worked according to the principles outlined in the Agile Manifesto [21]. We have chosen not to implement the well-known SCRUM method directly as we feel some of the aspects of this, such as the clearly defined roles and added process, may hinder our development.

Primarily, we implement the following points in our process:

- Conduct regular meetings between our project group to collaborate, discuss progress and (re-)assign tasks.
- Conduct regular meetings with our coach and customer to adjust our requirements and ensure our project is meeting the requirements.
- Maintain a board with issues and tasks that must be completed either in the long term or in the short term to keep track of our progress. This board is regularly updated to reflect changing goals and requirements.
- Ensure our master branch is always a fully functional version of our application so that we are able to deliver new versions of our application at any moment.

In addition to this process, we have decided to use a pull-based development method that involves thorough code reviews to ensure our project remains maintainable, consistent and functional at all times.

3.3. Technology Choices

3.3.1. Data Visualisation

Due to the shifting focus of our project, we never needed to make use of a data visualisation library. The only aspect for which we required the use of data visualisation was the height profile, however, we determined our needs for this to be too specific to use a general purpose library. We therefore decided to forego the usage of a library here and to implement our own solution.

3.3.2. Mapping

For the creation of maps, we found that our application needs Street View or an equivalent to function well. The only offering for such a technology is by Google. We decided that it would make sense for us to use a full Google technology stack for the mapping aspects of our project rather than to piece together multiple ones. This lead us to go for the Google Maps API for our project.

3.3.3. Front-end Framework

Due to the development velocity advantages of Vue, as well as maintaining sufficient simplicity, we found Vue to be the most suitable choice for our project. Therefore we have chosen to implement our user interface entirely in Vue, with plain HTML mixed in for non-interactive and static sections.

3.3.4. Back-end Framework

Due to the need for websockets in our projects, PHP was unfortunately not a realistic choice for our project. Therefore, we went with the most common alternative option of Node.js. Overall we were quite satisfied with this choice and Node.js was very capable of running our server on a variety of different platforms and operating systems without much trouble.

3.4. Core features

The five features that will be described in this section are:

1. **Distance Calculation**: A description of the way we do the distance calculation in the application.
2. **Client & Server Connection**: A description of how the client and server interact in the application.
3. **Creation of Comments**: A description of how the comment feature works.
4. **Height Profile**: A description of how we implemented the height profile, our main data visualization aspect.
5. **Data Input & Handling**: A description of how the comment feature works.

3.4.1. Distance Calculation

In the application, there are several points at which a distance between two points needs to be calculated. Our method will calculate the distance of a rider or the coach car to the finish or a certain location along the route. The first action performed, is the analysis of the race route, when it is loaded in. The route of a race is a list of points on the map represented by coordinates, with the actual route being the line segments that you can draw between those points.

The first step for our distance calculation method is to calculate the distance to the finish or last point for every one of the points that make up the race route, this gets saved and we use it later. This is shown in Figure 3.1. The finish point has a distance to the end of 0. For each other point, the lengths of each line segment from the finish to that point are summed up and the distance value is stored in the point.

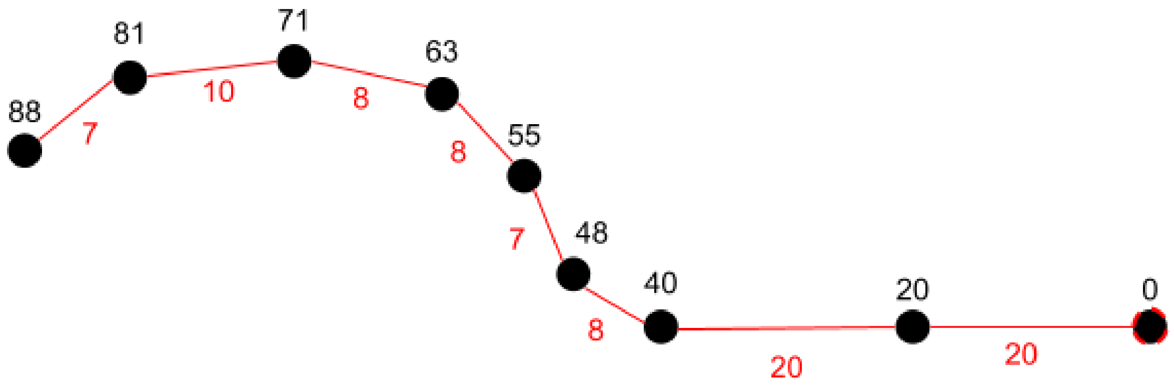


Figure 3.1: Distance calculation values, line segment lengths are shown under the line segments in red whereas the computed distance value for each point is shown in black above the point. The rightmost point with red dashed line is the finish.

These distances are used to calculate the distance to the end for the tracked location, but also when creating the comments.

Since the distance to the tracked location has to be done every second, we originally decided this had to be done as efficiently as possible. Therefore, we implemented an algorithm similar to the logarithmic time binary search algorithm, in which we would constantly check whether the point was closer to the first or last point in the race. Unfortunately, this proved to be unsuitable for curvy routes. Consequently, we decided to switch to an iterative approach to find the closest point in linear time. While somewhat less efficient, this method provides greater accuracy while being performant enough to run at the required speed within the constraints of our application.

Once the closest point has been found, we look at the next point in the order of the race route. We do this in order to determine whether the location for which the location is being calculated, lies before or past the closest point. If the distance to the next point is smaller than the distance between the closest point and the next point, the location will be past the closest point. This means the the distance to the closest point needs to be subtracted from the already calculated distance to the finish that belongs to the closest point, otherwise you will have to add the distance to it. Since the distance to finish for every comment has been determined when creating them before starting a race, it is easy to obtain the distance to the comments by subtracting this value from the calculated distance to the finish of the tracking location. This process is shown in Figure 3.2, where the grey point is closest to the second point from the right.

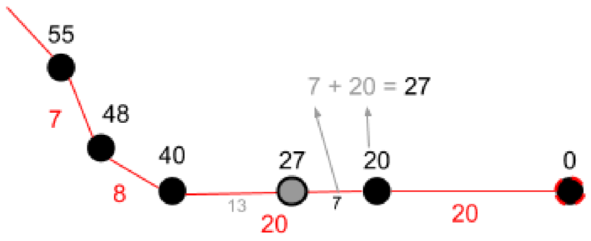


Figure 3.2: The rider is the grey point, the computed distance is the one of the closest point plus the distance to that point.

3.4.2. Client & Server Connection

An important part of our application is how the client and server communicate together. We have a few requirements that this part of the application needs to meet:

- Communication must be relatively low-latency to ensure data is given as close to live as possible.
- Data must be be 'live', meaning the server needs to be able to send data to the client when it wants and the client should not have to ask for data periodically.
- Different clients opening pages at the same time must be synchronized to prevent inconsistencies in (e.g. to prevent two users adding the same race).

For this, we have designed a global data-flow sequence diagram as shown in Figure 3.3. This diagram shows the communication between a single client and the server, as well as the elevation API (using the Google Elevation API) used to retrieve the information for the height profile and the datastore used to store the race information. The diagram is read from top to bottom:

1. The server retrieves the race information from the datastore on startup.
2. The client connects and the server immediately sends a list with the basic race information.
3. The client creates a new race, which is then confirmed by the server and stored in the datastore.
4. The client retrieves the elevation map. This is a slow process, during which the client can input the rest of the race information.
5. The client uploads .fit files for each of the riders, containing information on the route cycled by the rider.
6. The client saves the race, which is then stored in the datastore by the server.
7. The client has opened the comments page and requested the list of comments, which are retrieved from the datastore and sent to the client. For a new race, there will not be any comments initially.
8. The client adds new comments which are immediately stored in the datastore.

We have an additional diagram for the data-flow of the dashboard page. This is described in Figure 3.4. Note that this diagram describes non-live race data only. One detail not made clear in the diagram is that the access to the weather API is only done through the server due to a restriction of the API preventing client-side access. In this case, the server only functions as a proxy and therefore we have decided to leave this as a separate item in the sequence diagram.

1. When a client connects to the server, it is sent the initial race information including the route, comments and list of riders. The client then starts by loading the race and retrieving the weather information from the weather API.

2. When one of the clients clicks on play, the server starts sending rider data packets to all of the clients. These are sent once per second and include the position and other data (e.g. heart-rate, cadence, power) for each of the riders.

3. When a client changes the playback time, the server starts sending packets starting at the new playback time. This is changed for all clients.

4. When a client changes the playback speed, the server informs all clients of the changed playback speed and starts sending data packets at a higher speed.

5. Periodically, the client will retrieve additional weather information from the server.

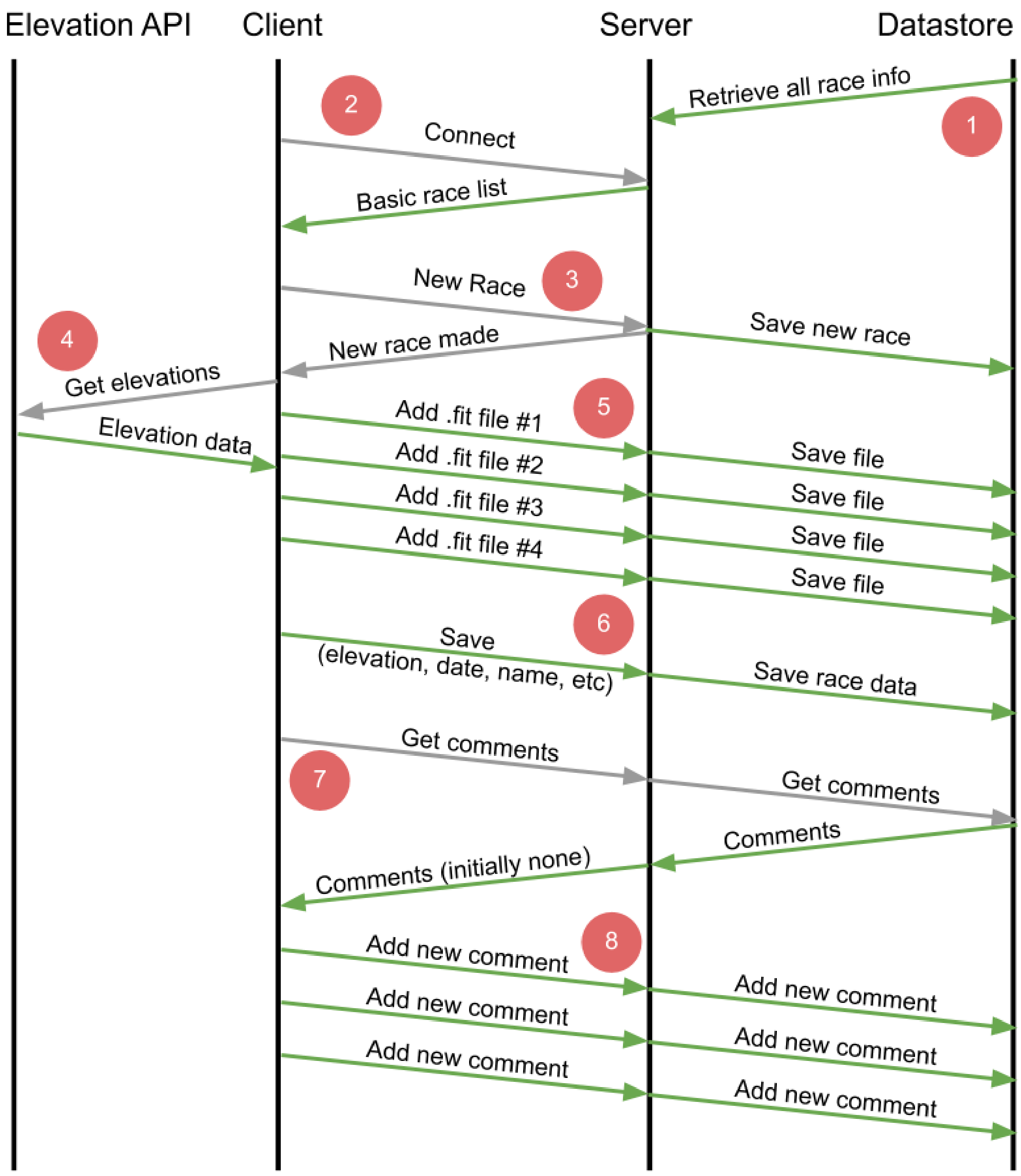


Figure 3.3: Data flow diagram of the entire application workflow.

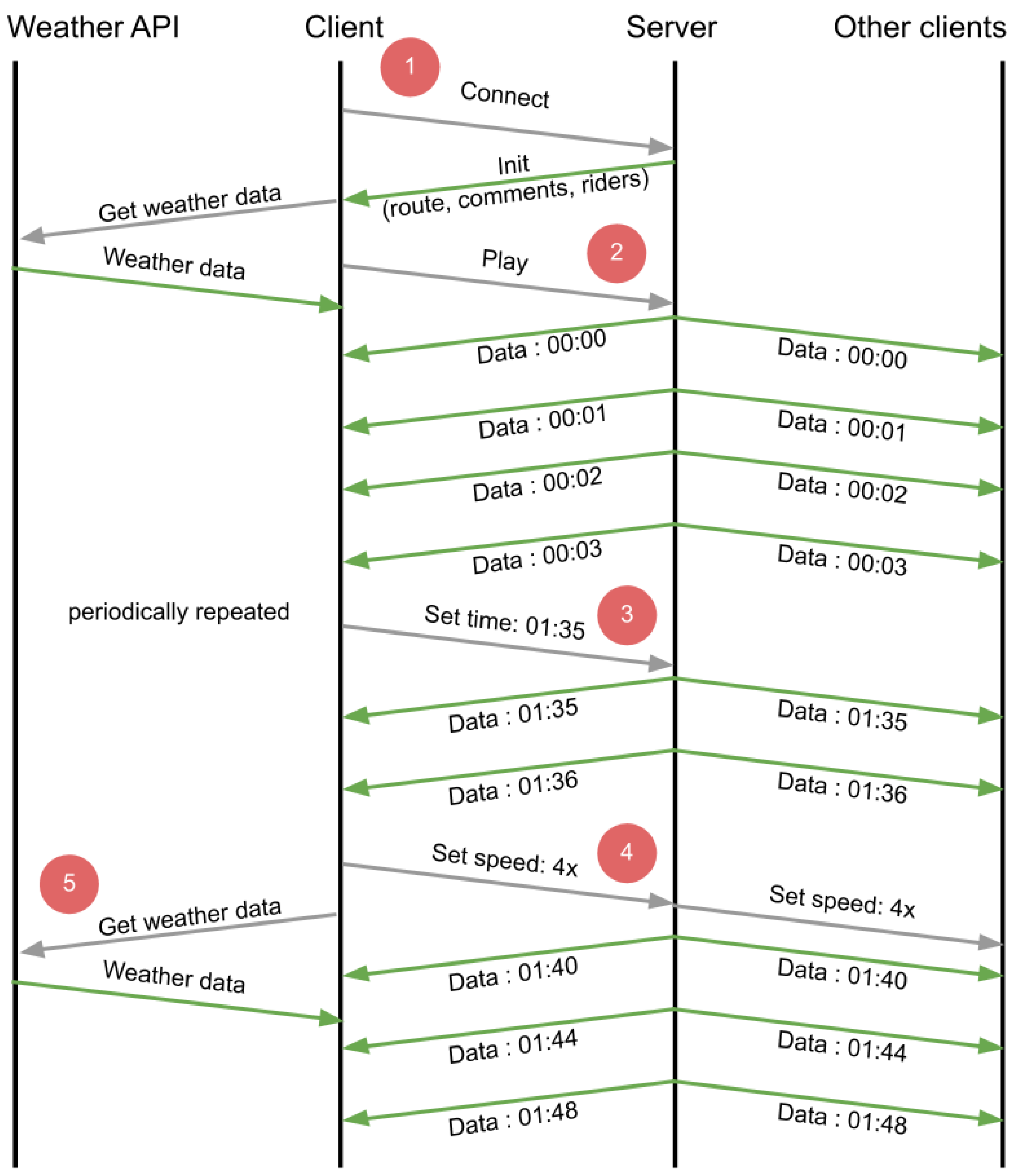


Figure 3.4: Data flow diagram of the dashboard page.

3.4.3. Creation of Comments

To allow coaches to create useful comments about the route of a race we use a combination of Google Maps and Google Street View. The route of a race is displayed on a map with a container next to it for Google Street View. This way the user can look at the race route on the map to identify tricky parts of a race and then visualize them using Google Street View. Once a user has identified a location of the course that requires special attention, they give the comment a type, possibly add some extra info as text and then save it. The location of the comment will then get annotated on the map, both on the comment creation page as well as on the actual race dashboard screen. When saving a comment the distance to the finish of that comment is also calculated and stored, by looking at the closest point of the route and the distance to it. The final design of the comment screen is shown in Figure 3.5

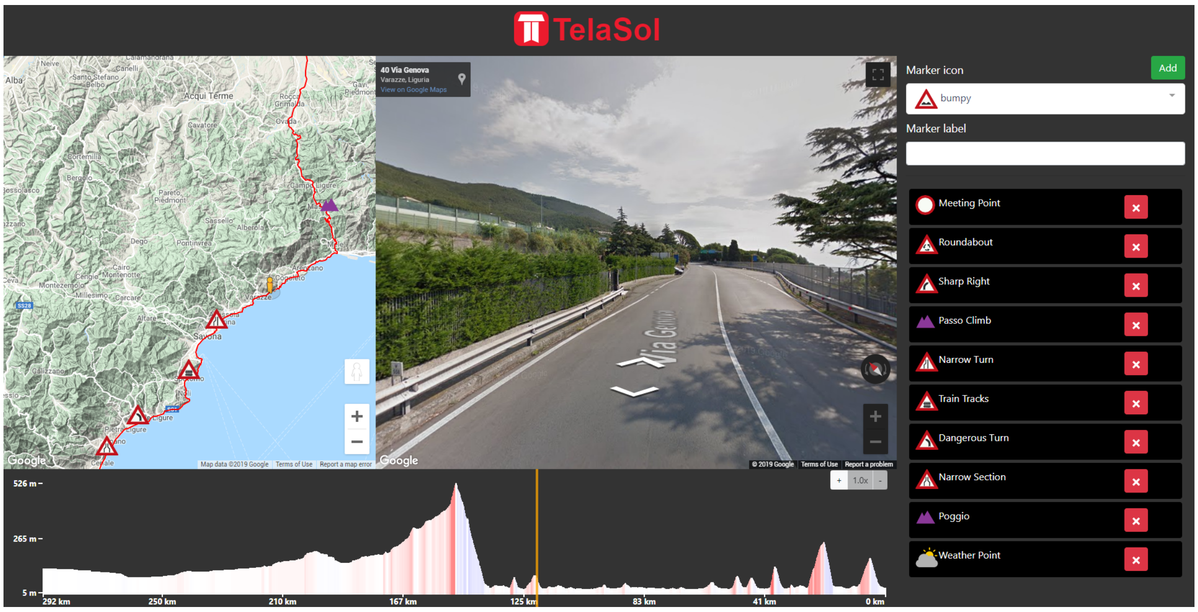


Figure 3.5: Final version of the comment screen.

Some comment types have unique properties, for example a wind comment provides you the up-to-date wind speed and direction for the annotated location during a race. The comments that require some extra work when creating are the climb comments, since they are made up of two parts, a start and a summit. When creating a comment for a climb, the user will start by selecting the start location of the climb and confirming it. Once the user confirms the start location of the climb, a prediction algorithm will try to predict where the summit of the climb is and move the selected location on the map and of the Street View window to the predicted top of the climb. This prediction algorithm loops through the route of the race in order, starting from the point closest to the selected start, and it looks for the highest point coming up. Naturally, this point might be the summit of the climb that the user wishes to highlight. To make sure this will not be the top of the next climb, a check has been put in place. This checks that if it is unable to find a higher point than the current highest point encountered within a reasonable distance of the current highest point, the current highest point will be the summit. This ensures that the top of the next climb will not be indicated as the top of this climb. While the top of the next climb might be higher, it still allows for relatively short, flat or descending sections of a climb. The predicted summit can be verified by looking at the height profile or by checking with the course information. The summit location can always be changed if it is not completely accurate or satisfactory to the user.

3.4.4. Height Profile

To visualize the height profile of a race, we start by retrieving and storing the elevation data for each point of our race. Typically this is several thousands of points. Loading the elevation data for these is limited by the rate limits of the Google Maps Elevation API. As such, loading these may take about a minute, depending on the number of points in the race route.

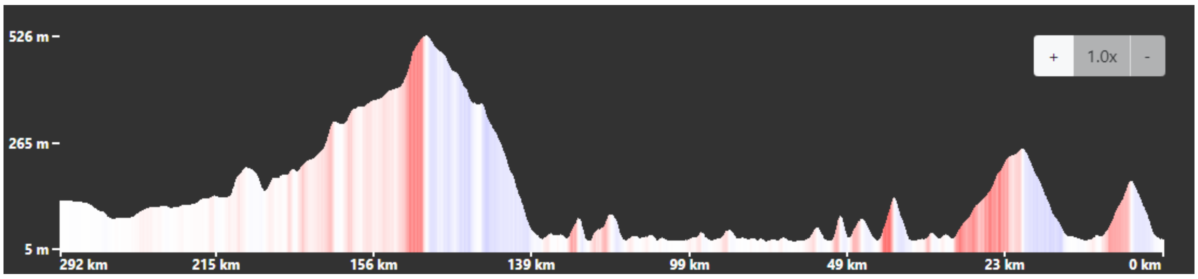


Figure 3.6: Final version of the height profile without any zoom.

When the heigh profile is drawn, the number of bars depends on the size of the window, where each bar is 1 pixel wide. In addition, we apply smoothing by averaging the height of each bar over the neighbouring bar. The exact number of bars used for smoothing varies based on the number of bars in the race and the zoom level. For a race of 6000 points (with the default zoom), this is around 30. After smoothing, the slopes are computed for each of the bars by looking at a few bars ahead and behind. The bars are then colored based on the computed slope.

To allow users to focus on the most relevant part of the race, we have zooming functionality built in that will zoom in on the part of the race where the rider currently is. The zoomed in area will automatically follow the current active rider.

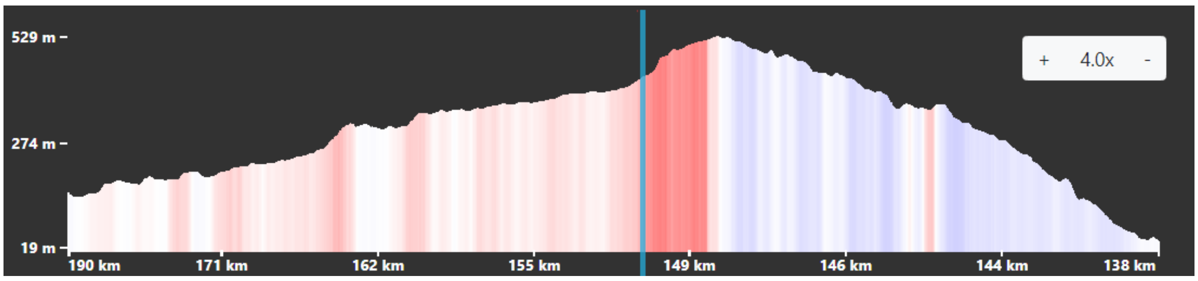


Figure 3.7: Final version of the height profile zoomed in to 4 times.

Lastly we have labelled axes to make it clear where in the race we are and what kind of distance the height profile is zoomed to. Note that the distance reads from the highest number on the left to the lowest on the right, as it indicates the distance left and not the distance from the start of the race.

3.4.5. Data Input & Handling

In the application we make use of two key pieces of input, the race course stored in a KML/KMZ file and the rider data, which is kept in separate FIT files. The FIT files contain rider data of past races. At the start of the project, there was also the intention of having live rider data as input, but due to complications with supply of this data from Team Sunweb this was not possible to implement during the duration of the project.

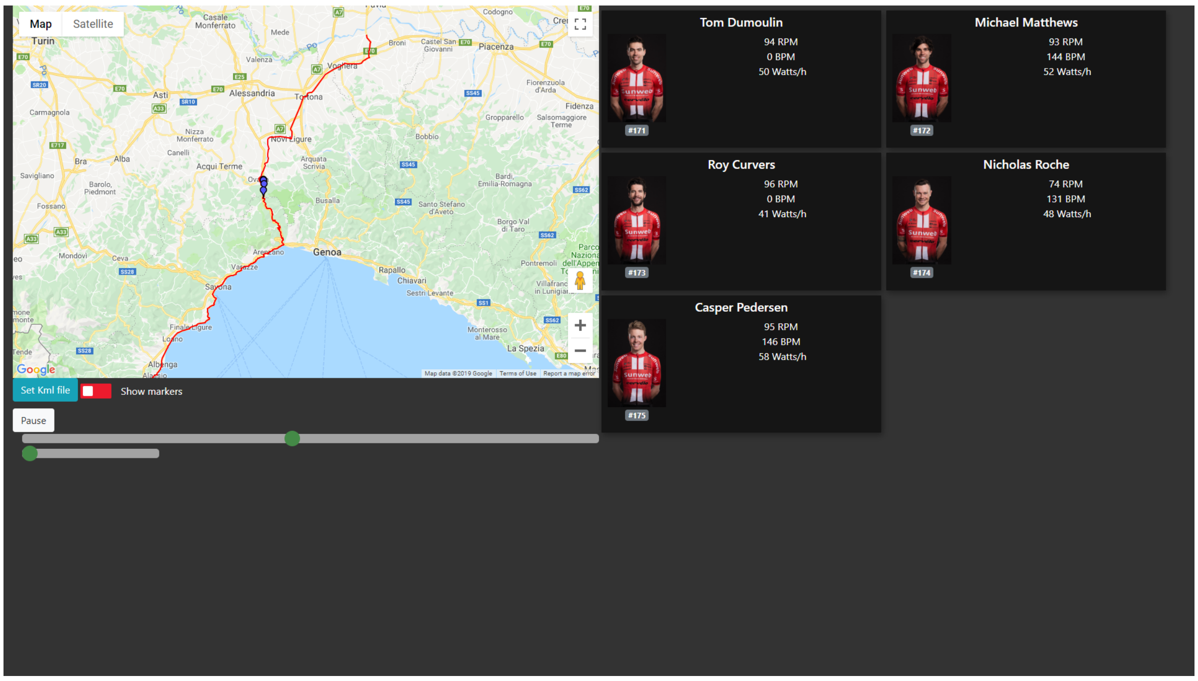
The race course (KML/KMZ file) has to be entered immediately at the start when you creating a new race. At this point the file gets read, the relevant data gets extracted, sent to the server in JSON and the server then stores it in a JSON file. The relevant data for each point of the entered race course are the coordinates in latitude and longitude, the distance to the finish and the altitude. The coordinates can directly be read from the KML/KMZ file, the distance to finish gets calculated point by point using Google's geometry library and the altitude gets requested for each point using Google Maps's Elevation API.

The rider data that is collected during a race is stored in FIT files, which are used for the playback function of our application. These files contain data about the rider for each second, including the rider's location, data from bike sensors and possibly data from a physiological sensor. Initially we opened the files using the program Golden Cheetah [5], in which we could export it to other file formats. We started by exporting the data to csv files to build the first prototype of our application, because this was the simplest format available. We quickly ran into problems using this format, because it lacked important metadata such as the exact start time of the rider's activity. To fix this issue we exported the data to JSON files created by Golden Cheetah and used this format instead. Finally when we implemented the functionality to send the FIT files to the server we decided that it was better to save the FIT files on the server and parse them using a dependency.

3.5. Development of the User Interface

3.5.1. Initial Design

The initial user interface is displayed in Figure 3.8. Every rider and his corresponding data is displayed on the right as well as on the map. There are two sliders, big and small, respectively for the time scale and the speed-up.



Tom Dumoulin
94 RPM
0 BPM
50 Watts/h
#171

Michael Matthews
93 RPM
144 BPM
52 Watts/h
#172

Roy Curvers
96 RPM
0 BPM
41 Watts/h
#173

Nicholas Roche
74 RPM
131 BPM
48 Watts/h
#174

Casper Pedersen
95 RPM
146 BPM
58 Watts/h
#175

Figure 3.8: Initial design of the user interface.

3.5.2. Intermediate Design

A first big change in the design of user interface, was the addition of the height profile of the course below the map upon client's request (section A.2). Another requested change was the display of the riders. Instead of showing all riders on the right at the same time, only one rider can be selected whose data is then shown. Each rider gets a unique color. Each rider can also be found on the map, indicated by their bike in the corresponding color. The center function on the map and the position on the height profile are also based on this rider. The last requested addition was the comment section between the map and the riders. A screenshot of this design is displayed in Figure 3.9.

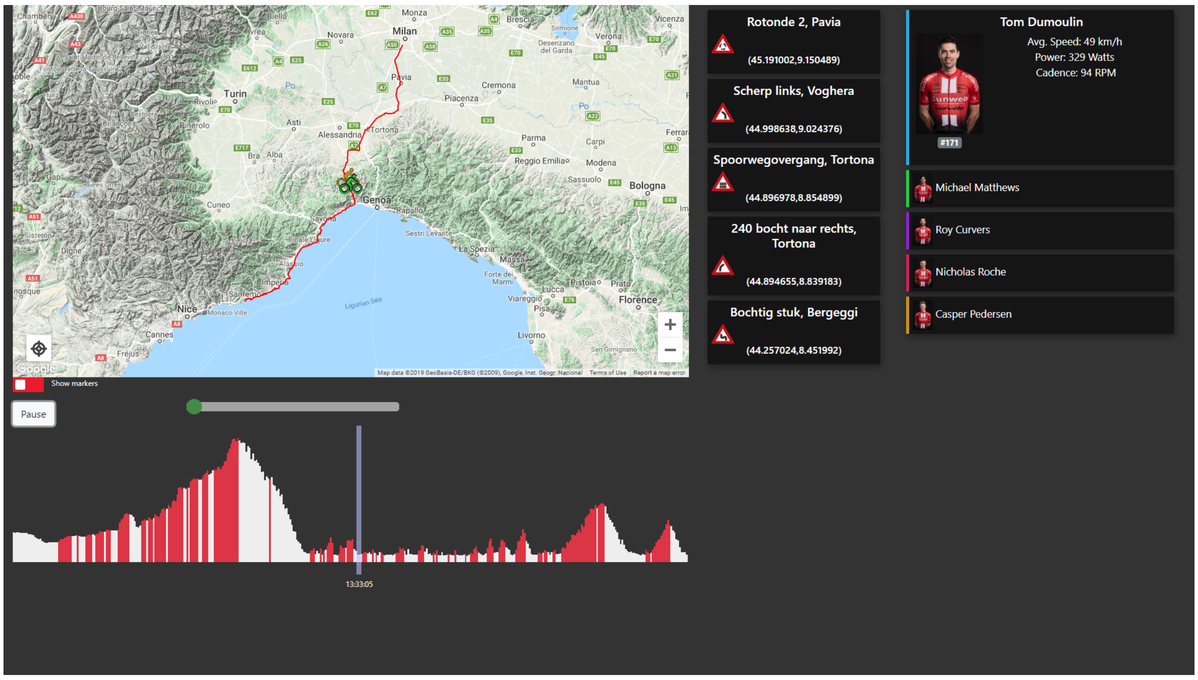
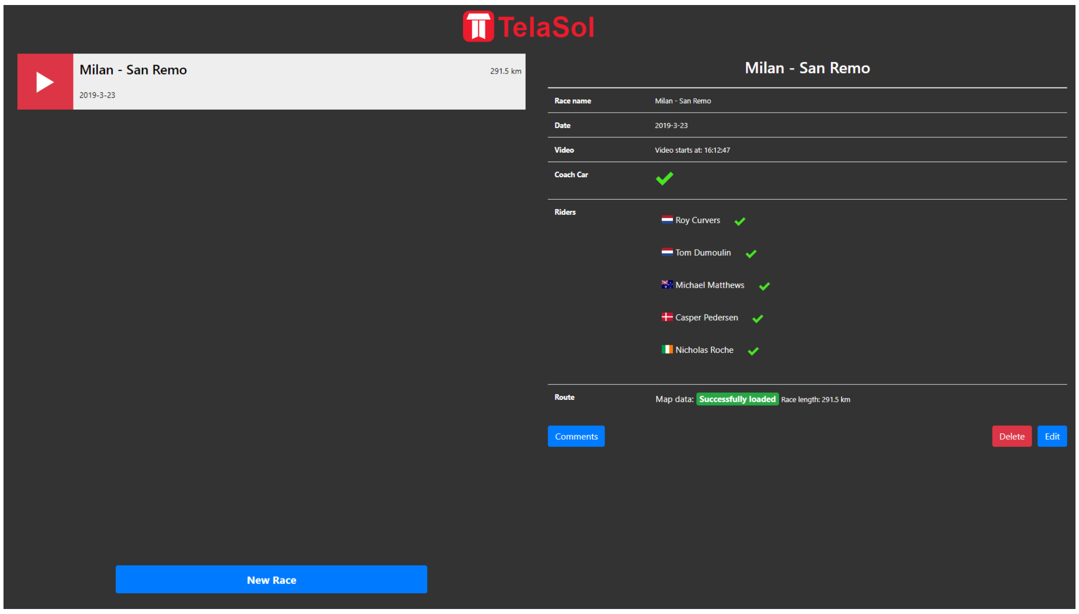


Figure 3.9: Intermediate design of the user interface.

3.5.3. Final Design

The final design of the user interface is displayed in Figure 3.10, 3.11 and 3.12 as the application contains multiple important screens. Compared with the previous version of the user interface, many things have changed. First of all, a set-up screen is added to make, delete and edit races. This is shown in Figure 3.10. Another screen that is added, is the comment editing screen. This allows the user to mark a specific point on the course. This is shown in Figure 3.11. The screen of the race itself, is displayed in Figure 3.12. This is the race dashboard, which was also available in the previous two versions, but now with many improvements, among which are weather information, height profile zooming, the display of the comments and the synchronized video.



Milan - San Remo 291.5 km
2019-3-23

Milan - San Remo

Race name	Milan - San Remo
Date	2019-3-23
Video	Video starts at: 16:12:47
Coach Car	✓
Riders	Roy Curvers ✓ Tom Dumoulin ✓ Michael Matthews ✓ Casper Pedersen ✓ Nicholas Roche ✓
Route	Map data: Successfully loaded Race length: 291.5 km

Figure 3.10: Final design of the set-up page.

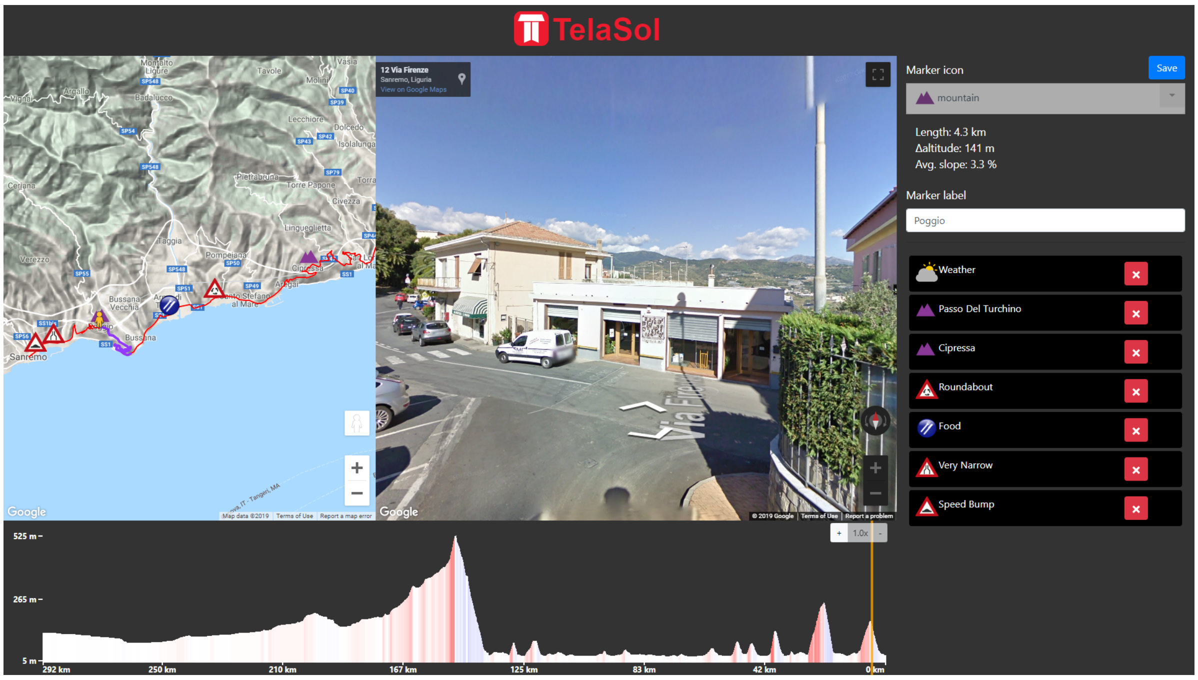


Figure 3.11: Final design of the comment editing page.

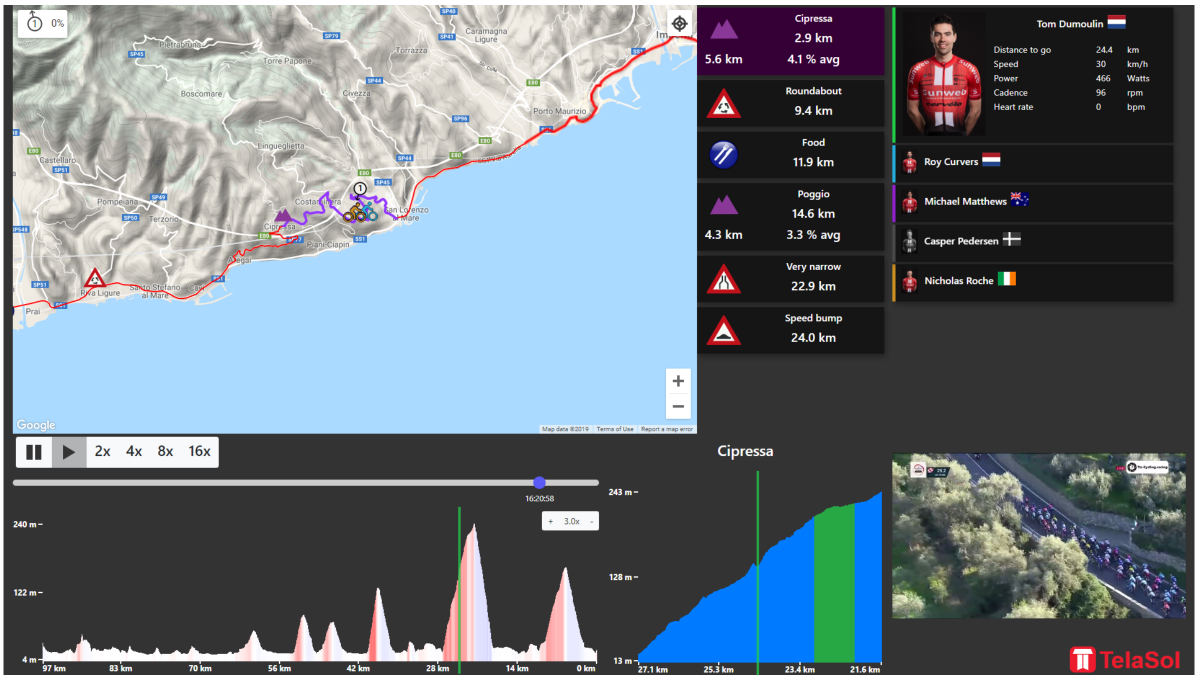


Figure 3.12: Final design of the dashboard page.

3.6. Ethical Considerations

In this section, we will shortly lay out some of the ethical issues which we have taken into account while building the application.

As our application does not handle a lot of data, the risk of sensitive data falling in the wrong hands is limited. The most important piece of data in our application is also the most sensitive piece: rider locations and vitals during a race. These data may contain pieces of information that would prove very useful for the opposing teams. If say, Rider X, can be seen to have a very low power output during races, this information could potentially be used by other teams to take advantage of during races.

We have addressed this problem through authentication. Our app is available online at https://TelaSol.nl but cannot be publicly used by everybody. When accessing the website, you will be presented with a login screen which is protected with an authenticator. Only the proper people at Team Sunweb and the developers can therefore actually access the website and data.

Quite a different ethical issue that also majorly influenced our project was the decision by the UCI to forbid the use of live rider data during a race. The proposed reason this was done, was to keep the sport more authentic and to level the playing field. Allowing the usage of so much data, would give more affluent teams even more advantage over the other teams.

Regardless of the UCI's decision, we decided in collaboration with Team Sunweb to continue developing the app as initially planned. This meant that we maintained the feature of tracking riders live data, even if it meant that it would not be allowed to be used during races. Team Sunweb hoped and expected this decision to be overturned in time, at which the app could be employed immediately. At the moment of writing, the UCI's decision has not been overturned.

4 Interaction with the client

As this project was commissioned by Team Sunweb, our interaction with them throughout the project was of vital importance. There were several people within the company that we had interactions with and these meetings provided us with valuable insight regarding the requirements for the application. In this chapter we will shortly describe these people and their roles. Most importantly, we will describe the meetings and the feedback we took from that. Our two most important sources of feedback were:

1. **Rado Dukalski** PhD at Sports Engineering Institute and client-role
2. **Teun van Erp** Scientific Expert at Team Sunweb and our liason at the team

4.1. Rado Dukalski

Rado was our coach from the Sports Research institute at TU Delft. He was mostly responsible for providing us with (bi)-weekly feedback on our application and the process of it. He also took care of arranging meetings with Teun, our liaison at Sunweb, to gather more valuable feedback from the client. The notes we took of these meetings can be found in Appendix A.

4.2. Teun van Erp

The meetings with Teun were vital to the development of our app. Throughout the project we had three meetings with him, each at a different stage of the project. We will describe the meetings shortly below. Extensive notes from the meetings can be found in Appendix A.

4.2.1. Meeting 1: April 26th

This was a very short Skype meeting with Teun at the early stage of the project. This meeting was meant to get to know the needs and the wants of the client. These were explained in broad strokes to get the development started.

4.2.2. Meeting 2: May 10th

Two weeks after the first Skype meeting with Teun, we arranged a second one. This meeting was meant to show the client the initial version of the user interface and the currently implemented features. The feedback we received during this meeting from Teun was useful in order to improve the application for the Midterm Meeting on the 28th of May in Deventer. At this meeting we also got to hear the crucial detail that the UCI had forbidden the use of live rider data during races. This had no direct effect on our development of the application but it did limit the current usability and usefulness of our application.

4.2.3. Midterm Meeting: May 28th

This was a very important meeting in which we visited the Team Sunweb Headquarters in Deventer with the team. This meeting was meant as an evaluation point for us and Team

Sunweb to determine the current state of the application and our projections for the second phase of the project. A short description of those two events will be given, for more detailed meeting notes, please refer to Appendix A.

During this meeting, we presented our application to two scientists at Team Sunweb: Teun, our liaison, and Jorn, another employee at Team Sunweb. Our presentation was a walk-through of the features of our application, in which we provided room for Teun and Jorn to ask us about the application. Overall, they were very satisfied with the results we provided as most of the features they requested were already present. Most of the conversation was about the flushing out of some features such as altitude profiles and the display of comments.

Afterwards we took a look at the features that still needed to be implemented for the future. Most of these were the aforementioned quality of life fixes. The most important requirement for the second phase however, was the implementation of live races.

About a week after the meeting, Teun contacted us with the news that they wanted to leave the live rider data for now as they deemed it too complicated. Therefore, in collaboration with Teun, we decided to focus on playback features and using the location of the coach car for live races.

5 Software Validation

In this chapter we will explain what actions we performed to validate our software. These actions are:

1. **Use Cases**: In this section we describe some of the use cases we had for the application.
2. **Requirements Evaluation**: The requirements defined in Chapter SECTION MISSCHIEN chapter 2 are evaluated in this section.
3. **User Interface Questionnaire**: Here we describe the questionnaire we created to test the user's satisfaction with the interface.
4. **Software Improvement Group**: A brief description of the feedback we received from SIG and how we handled it.
5. **Unit Testing**: A description of how we handled unit testing in our application.
6. **System Testing**: This section features a description of the system tests we performed.

5.1. Use Cases

In the following section we will be describing the various use cases of the application. We will explain in which circumstances and why a user wants to perform a certain action while using the application. Since the application can be used in three different ways, namely during a live race, during a training session and as tool to analyze a past race, we will indicate to which purpose each use case applies.

5.1.1. Add comments to a route

As a user you want to put in comments about important points in a race.
When you have loaded in the race data, there is the possibility to put in comments about the route of the race. The user can add these comments to the race data, by looking at the route on a map and then selecting the locations for where comments might be necessary. With the help of Google street view the user can get a full picture of a specific point of interest in the race. There are different kinds of comments that a user can attach to a location and these vary from dangerous parts of the race like sharp turns and narrow sections to location where the riders can be supplied with food and drinks. The user selects a type for the comment from the presented options and he can also add a custom description to the comment. Most of the comment types behave as regular point in the race that represents a certain piece of information, but there are some special comment types, which are weather comments and annotations for climbs. The weather comments make sure that during the race you get up to date information about the weather for that specific location, while for the climb annotations you will have to indicate the start and end of the climb and this will grant you the ability of more detailed information about the climb.

This use case mostly applies to when you want to use application during a live race, but the comments could also be helpful during training and then especially recon of upcoming races.

5.1.2. Comment notification

As a user you want to get notified about when your prepared comment locations are approaching.
When you start a live race or training session and you have a set of comments prepared you will get notified accordingly when you encounter them along the route. The next comments that you will encounter, shall be displayed next to the map on the dashboard and sorted on their proximity. What is displayed, is the content of the comment and the distance that is left until you pass the location of the comment, which is updated every time the current tracking location is updated. During a race this location is the current location of the device, which will be in the team vehicle. During a training session this can be the location of one of the riders. To make it more clear that you are approaching the location of one of the comments, the color of the comments will change color to indicate their proximity. When the location of a comment is passed, the comment gets greyed out and a few moments later it will disappear, making room for the other comments to move up and a new comment to appear.

5.1.3. Analyze using the height profile

As a user you want to be able analyze the race route using a map and its height profile.
On the race dashboard, but also on the comment creation screen the route of the race will be displayed on a map. On the dashboard the location of the device will also be displayed on the map, which allows you to see where you are on the route. During training or a playback of a race the location of each rider will also be displayed on the map. The riders will be color coded such that they are easily identified. The height profile will be displayed underneath the map and will show the change in elevation of the race. Sections that go significantly uphill will be colored in red, to indicate the hard parts of the race. The tracking location, car or rider, will also be displayed on the height profile and there is the ability to zoom in to see the profile in more detail. When climbs are annotated in the comments sections you can select them to view their profile in even more detail, this view will show you the difference in steepness of all the sections of the climb.

5.1.4. Live weather information

During a race, a user wants to be constantly informed about live weather updates.
Live weather data will be available during races and training sessions. The wind speed and its direction will be displayed on the map for a preset amount of points along the race route. It will also be displayed for locations that the user has selected during preparation on the comment creation screen. These locations could for example be open sections that allow for the possibility of echelons. The probability of rain will also be displayed on the dashboard, such that the coaches know ahead of time if, when and where they should supply their riders with rain jackets.

5.1.5. Analyze using playback option

As a user you want to analyze previous races using playback capabilities.
In the playback mode of the application you are able to view the dashboard as if the race was live, but now you can also see the rider data since it is allowed for playback purposes. The playback mode allows the user to select a specific point in the race that they want to analyze and it also allows for speeding up the flow of the race data. A video stream option will also be available in the playback mode, which provides the ability to compare what happens on camera with what happens in the data. The stream will be synced to the data flow, meaning you can properly see how certain race situations play out both visually and numerically.

5.2. Requirements Evaluation

In section 2.1 we define our requirements for the two stages of the project. In this section, we will evaluate whether those requirements have been met. This is based both on our idea of what the requirement entails as well as the opinion of the client and Team Sunweb.

5.2.1. Stage 1

In chapter 4 we summarize the meetings with Team Sunweb. These notes show that Team Sunweb deemed our progress more than sufficient for this stage of the project. A large portion of the requested features was implemented according to their wishes. In this part we will shortly evaluate for each requirement whether it has been completed and how it relates to the category we put it in.

1. **Live data from the cyclists (e.g. Heart Rate, Cadence)** *Must have*
   Completed and present in the application. The exact implementation can still be changed according to Team Sunweb's wishes.

2. **Live position of the riders on a map** *Must have*
   Fully completed and present in the application. Implemented according to Team Sunweb's wishes and thus is not of importance for the future.

3. **Add notes from team leader in advance with Google Streetview** *Must have*
   Completed and present in the application. While present, it was one of the latter features we implemented. As such, it was not fully flushed out yet, leading to suggested improvements by Team Sunweb. These will be implemented in the second stage of the project.

4. **Tablet based design** *Could have*
   Not completed. This was an issue we had not paid much attention to thus far. While we have designed our application with tablet functionality in mind, we have not yet tested it properly on tablets yet. This will definitely be more important in the second stage of the project.

5. **Ideally working with unreliable connection** *Won't have (at this stage)*
   Not attempted. This requirement simply was not in the scope of the first half of the project.

6. **Extensible for data sources other than csv file** *Should have*
   Fully completed. Throughout the development, we needed the ability to read different file formats, thus this has been implemented.

7. **First on recorded race, then also on live data** *Could have*
   Not completed. As we had not received any information from Team Sunweb to this point in time about how to implement this exactly, it was not part of the first half of the project.

8. **Integration with video images** *Could have*
   Not completed. This is something we deemed not relevant enough for the first half of the project.

5.2.2. Stage 2

The evaluation of this stage of the project proved a bit more difficult. Since we did not have a final meeting with the client before finishing this report, exact details of that meeting are not present. We do however have a good indication of our progress from both our intuition as well as short contact with both Teun and Rado in the final weeks of the project.

1. **Tablet based design** *Must have*
   Fully completed and present in the application. The app functions both on tablets as well as regular PCs. As the team coaches use tablets in their car during a race, it is a very important feature.

2. **First on recorded race, then also on live data** *Must have*
   On the 5th of June, Teun mailed us that it does not make any sense to let us work on the live data and that we should shift our focus to the front end and the visualization of the tool. Some features of the live version are still implemented, such as selecting the coach car as a driver and live weather information, but the connection between the data of the rider and the application is missing.

3. **Integration with video images** *Should have*
   Fully completed and present in the application. When a video is provided and the start time of the broadcast, the video is synchronized with the active rider.

4. **Ideally working with unreliable connection** *Could have*
   Not completed. It was decided quite early on that this requirement would not be a focus of the application. This has not changed since then, and thus we have not implemented it.

5. **Race and Training versions** *Must have*
   Partially completed. As we shifted our focus away from the live version according to Team Sunweb's wishes, we did not put much attention into separate race and training versions. The features for both versions are present (such as tracking the coach car for the live version), but a clear distinction between the two is left as future work.

6. **More information about climbs** *Must have*
   Fully completed. The height profile itself has been improved slightly by adding scales. In addition we have created separate climb comments that can be used to indicate specific climbs. These climbs have their own height profiles that can be used for further information.

5.3. User Interface Questionnaire

In order to assess the users' satisfaction about our user interface, we used a personalized version of the QUIS tool (Questionnaire For User Interaction Satisfaction). Using this approach, we tried to measure the satisfaction of five different aspects of the user interface, and optionally extra remarks. The questions of each subject asked to the client are listed below. Each question is answered with a number from 1 to 5.

1. Screen
   - Reading characters on the screen (very hard-very easy)
   - Highlighting simplifies task (not at all-very much)
   - Organization of information (very confusing-very clear)
   - Sequence of screens (very confusing-very clear)
2. Terminology and system information
   - Use of terms throughout system (inconsistent-consistent)
   - Terminology related to task (never-always)
   - Position of messages on screen (inconsistent-consistent)
   - Prompts for input (very confusing-very clear)
   - Computer informs about its progress (never-always)
   - Error messages (unhelpful-helpful)
3. Learning
   - Learning to operate the system (very difficult-very easy)
   - Exploring new features by trial and error (very difficult-very easy)
   - Remembering names and use of commands (very difficult-very easy)

- Performing tasks is straightforward (never-always)
- Help messages on the screen (unhelpful-helpful)
- Supplemental reference materials (very confusing-very clear)

4. System capabilities
   - System speed (too slow-fast enough)
   - System reliability (unreliable-reliable)
   - Systems tends to be noisy/quiet (noisy-quiet)
   - Correcting your mistakes (difficult-easy)
   - Designed for all levels of users (never-always)
5. Specific questions related to TelaSol
   - Adding a race is easy (disagree strongly-agree strongly)
   - The program is secure (disagree strongly-agree strongly)
   - Adding/Editing comments is easy (disagree strongly-agree strongly)
   - The interface is intuitive (disagree strongly-agree strongly)
   - The information of the riders (location and measurements) is displayed clearly (disagree strongly-agree strongly)
   - The information of the circuit is displayed clearly (disagree strongly-agree strongly)

5.4. Software Improvement Group

In order to improve our software quality, we had our code evaluated by the Software Improvement Group (SIG). We received our first feedback moment a little over halfway through the project, which is included in Appendix B. Overall, our code maintainability was assessed as being more than sufficient. Following this feedback moment, we attempted to continue the line of our well modularized pieces of code and improve current ones where possible. Another point of improvement was the number of unit tests. Since then, we have increased this amount significantly.

Unfortunately, we did not receive the second piece of feedback from SIG before the submission of this thesis.

5.5. Unit testing

In order to verify if all features of our application work properly, we have unit tests present. These tests test a variety of different functions, such as calculations, integrated functions as well as UI elements. They were put in place both to ensure that functionality was correctly implemented when added, and to ensure that no new faults occurred through regression when adding new functionality. Overall these unit tests proved very useful to us by highlighting several implementation errors that would not have been spotted through manual testing alone.

5.6. System testing

During the final phase of the project, we performed a system test. One of the team members was cycling while the others were following him in the car testing the application on a tablet. At the beginning of this test, we faced two problems that we had to fix. Firstly the watcher of the coach car was not set on high accuracy so it only updated on big changes of the location. Secondly, the coach car did not work properly when there were no riders added to the race. After some small code changes done on the laptop in the car, the second part of the test went flawlessly. Even on a very sunny day and in a car with a panoramic sunroof, the tablet was perfectly readable. The full test set-up is displayed in Figure 5.1 and 5.2.

Figure 5.1: System testing of the application.

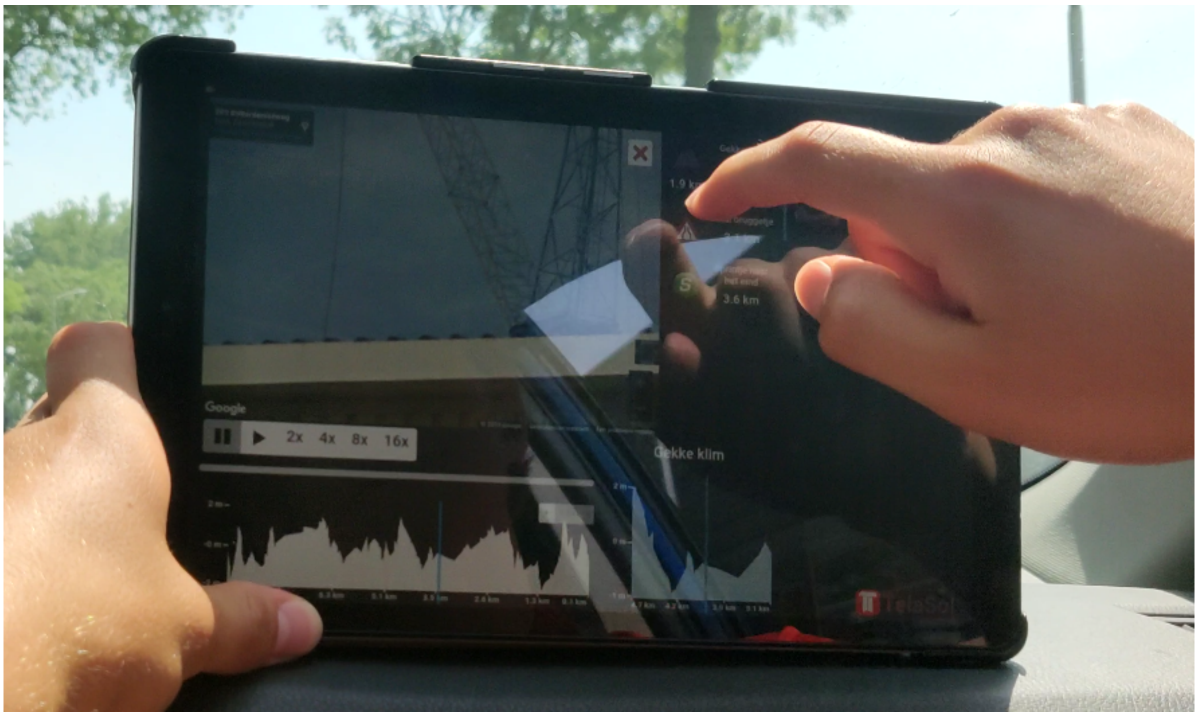


Figure 5.2: System testing of the application.

6 Conclusion

In conclusion, our product has met nearly all of the originally defined requirements, as can be read in section 5.2. Team Sunweb has shown throughout the various meetings that they were content with our progress and the product at that stage. Evaluating the quality of our final product was made difficult by the limited contact we had with Team Sunweb. We only had a scarce amount of meetings with Team Sunweb themselves, primarily in the early stages of the project. In addition, the data used in the project was under strict NDA by the supervising company which made it unfeasibile for us to conduct user testing on users not familiar with the project. This means that the only functional evaluation method we were able to apply was through our supervisor, Rado Dukalski.

In the process of building our application, we encountered numerous challenges that hindered our development somewhat. One of these was the decision by UCI to restrict usage of data in cars during the race, which meant that the focus of our project was shifted from live data to a tool more suitable for training. The live aspects had to be restricted to focus on the GPS coordinates of the device instead of the position of the riders. The training aspect met further difficulties when we were unable to receive assistance from Team Sunweb in implementing the live aspects using their rider data solution. As such, we were unable to fully implement the live rider data functionality in our final application, which is one of the major shortcomings of our final product.

Future expansions could focus primarily on the implementation of live functionality to fully support both training and real races. In addition, the user experience of the application could potentially be improved through extensive user testing and further analysis of the design. While we invested resources in these aspects during our project, due to time constraints and limited access to users we were not able to conduct in these activities to the fullest extent.

Our application meets the primary requirements and has shown throughout the development period to be a useful tool for Team Sunweb. With the addition of the aformentioned full live functionality, combined with extensive user testing in the proper environment, the app should become fully usable for its intended purpose.

A Meeting Notes

A.1. Rado

A.1.1. 04-06-2019

Meeting Rado and Gosia: 04/06/2019

- Think aloud protocol: Observe people using the app and take notes / ask them questions
- Missing in report: Sketches of interface (initial version and evolution) and pipeline of the application
- Show screenshots of different interfaces to coaches and ask their preferences (io: knippen en plakken)
- In report: focus not on visualization of data
- Deliver report on June 20th to get feedback from Gosia (Or send method section earlier)
- Make a poster
- Demo on tablet?! (Airserver for streaming)
- Try to avoid the passive voice (”we have been tasked”), don’t be afraid of we
- Be consistent with the tense
- Very little references in literature review chapter (Look at the references of used papers)
- More about data visualization on tables in the literature review chapter
- Each chapter should stand on its own
- contrast of the interface (Write in report the bad situation (direct sunlight and tv) of the demo)

Golden tips from Rado on writing:

- Don’t leave it till the last moment
- Don’t be afraid to put things in the appendix
- Don’t state how much effort it costed

Next meetings: - June 13th with Rado - June 21st with Gosia: 12:30

A.2. Teun

A.2.1. 26-04-2019

Skype meeting with Teun

Features to implement:

- Live data from the cyclists (Hear Ratio, cadence ...)
- Live position of the riders on a map
- Add notes from team leader in advance with Google Streetview
- Ideally working with unreliable connection
- Extensible for data sources other than csv file
- Usable on tablet

A.2.2. 10-05-2019

Skype meeting with Teun

Feedback:

- Comments about climbs (start, end & gradient of sections)
- Selecting the proper markers (only useful info should stay)
- Height profile to include gradient (Might be available from a website)
- Color coding riders and their markers
- Selecting the riders to show data of only a single one
- Alerts for crashes and problems with the bike
- Connection loss should not alert the coach, but handled differently (maybe an option to retire a rider)
- Selecting a rider who will be the one who determines how far it is to the finish, comments etc. (The leading rider)
- Playback might be useful if it is synched with a video stream
- Options for how data is displayed dependent on the situation (Race, time trial, training)
- Map should zoom properly and track the 'leading'/showing rider
- Possibility to use street-view during the race to see the annotated locations
- You should be able to prepare a race fully and then quickly load the prepared race when the race starts
- Multiple races should be possible and also multiple instances of the same race
- The location of the car (device) should be visible during a live race
- Physical data might be useful to have an indication for how riders are doing compared to their known values
- The map should have the ability to be oriented according to the race direction (Probably the default when tracking a rider)
- Making the weather data available would be very useful

A.3. Team Sunweb

A.3.1. 28-05-2019

Notes meeting Deventer

- Comments over samenkomen, bidons ...
- Comments tijdens toevoegen ook sorteren op afstand?
- Begin van de klim en eind van de klim invullen => zelf procent berekenen en afstand
- Comments per race: 50-60, also days with 3-4 comments
- Ploegleider kan windplaatsen selecteren en aantal punten vooraf (en weer constant op scherm)
- Regenvoorspelling ook toevoegen Of weervoorspelling over x aantal km
- Meerdere renners tegelijk zien
- Inzoomen op het hoogteprofiel
- Kleuren van comments veranderen naar km: groen, geel, rood
- Volgorde van de renners: Maakt niet veel uit Als eerste training versie met videobeelden en eerstvolgende klim ingezoomd, kleuren van comments en meer kleurtjes op hoogteprofiel Of eerst auto kunnen toevoegen om te kunnen testen bij dames/onder-23

  Prioriteiten en tijdschema mailen naar Teun

B SIG Feedback

B.1. Initial Feedback

De code van het systeem scoort 3.9 sterren op ons onderhoudbaarheidsmodel, wat betekent dat de code boven marktgemiddeld onderhoudbaar is. We zien Unit Size vanwege de lagere deelscore als mogelijke verbeterpunten.

Bij Unit Size wordt er gekeken naar het percentage code dat bovengemiddeld lang is. Dit kan verschillende redenen hebben, maar de meest voorkomende is dat een methode te veel functionaliteit bevat. Vaak was de methode oorspronkelijk kleiner, maar is deze in de loop van tijd steeds verder uitgebreid. De aanwezigheid van commentaar die stukken code van elkaar scheiden is meestal een indicator dat de methode meerdere verantwoordelijkheden bevat. Het opsplitsen van dit soort methodes zorgt er voor dat elke methode een duidelijke en specifieke functionele scope heeft. Daarnaast wordt de functionaliteit op deze manier vanzelf gedocumenteerd via methodenamen.

Voorbeelden in jullie project:

- commentConnection.js:TelaSol/modules/commentConnection.js
- RaceManager.js:RaceInformation.constructor
- RaceManager.js:RaceManager.readFromFile

Als laatste nog de opmerking dat er geen (unit)test-code is gevonden in de code-upload. Het is sterk aan te raden om in ieder geval voor de belangrijkste delen van de functionaliteit automatische tests gedefinieerd te hebben om ervoor te zorgen dat eventuele aanpassingen niet voor ongewenst gedrag zorgen. Op lange termijn maakt de aanwezigheid van unit tests je code ook flexibeler, omdat aanpassingen kunnen worden doorgevoerd zonder de stabiliteit in gevaar te brengen.

Over het algemeen is er dus nog wat verbetering mogelijk, hopelijk lukt het om dit tijdens de rest van de ontwikkelfase te realiseren.

C Original Project Description

The coach car of the professional road cycling team Team Sunweb needs your help. Professional cycling is a sport of both brain and brawn, where teams attempt coordinated efforts to efficiently use their collective physical capabilities in unique and ever-changing circumstances.

Team Sunweb is looking to end its reliance on proprietary 3rd-party solutions such as their current go-to VeloViewer to display the data, and hopes to develop their own tablet-based Coach Cockpit, allowing it full control over how data is processed and visualized, with future expansion possibilities (e.g. predictive algorithms, wind). Tour de France is in July, are you up to the challenge?

Technology is being developed and applied with the aim of collecting the performance data (power, cadence, heart rate) from the cyclists in both race and training conditions. Unfortunately, being reliant on closed systems, at the current moment the coach following them in the coach-car remains largely in the dark of their real-time performance, and their execution of the strategy.

Pre-made advanced pacing models for races give the team an edge, however without live-monitoring of their execution, and if needed adapting to the dynamic race conditions, the team loses precious seconds and ranking points.

The riders may be aware of their own data and surroundings but remain unaware of their surroundings, fellow teammates or their remaining capabilities. The role of the coach is to absorb and comprehend the information as it is collected from various sources, fill in the blanks, and communicate the advice back down to the individual riders – currently the equivalent of conducting an orchestra, on a conference call, from a car, going through a tunnel.

Requirements

- On-the-fly synchronisation of data from multiple riders
- Visualise data in an appropriate and informative manner for a team
- performance overview
- Mapping integration
- Tablet based, touch controls
- Preferably web-tech based (for cross-platform compatibility)
- Mock-data for now, live streams from web APIs time-permitting
- Replay capabilities are nice to have – for post-race analysis, or training of coaches

References
https://veloviewer.com/
https://teamsunweb.com/

Other information Work to be conducted in Delft. Meetings through the Team's TU liaison. Depending on the progress, live-testing/demo to be conducted at the team's training grounds in Sittard. (travel expenses covered).

D

Technical Guide

D.1. Running the server

To run the server, Node and NPM are required. Any recent version with ES6 support should work. First run `npm install` to set up all depencies of the project. Then, run `webpack` to compile and transpile all Vue files and JavaScript files to runnable ES5 code. For development, using `webpack --watch` will set up automatic recompiling of JavaScript on change.

Running the server can be done with `node bin/www`. This will run the server in secure mode, meaning authentication and HTTPS are enabled. `node bin/www --insecure` will disable these features which makes development and local servers possible. To run the server in secure mode, an ssl certificate is needed (such as one generated by Let's Encrypt), as well as an authkey. The full list of required files for secure mode are:

- `/.ssl/authkey`: A file containing the key used for authentication. When the server starts in secure mode, it will print a URL in the console that can be converted to a QR code to be read by applications such as Google Authenticator. These will then generate codes that are used to login to the application.
- `/.ssl/ca_bundle.crt`: One of the files used for HTTPS
- `/.ssl/certificate.crt`: One of the files used for HTTPS
- `/.ssl/private.key`: One of the files used for HTTPS

To change the server port (3000 by default) the `/bin/www` file can be modified.

D.1.1. Development or Production mode

In `webpack.config.js`, changing `mode: 'development'` to `mode: 'production'` will improve the efficiency of the compiled JavaScript and reduce the filesize further. This is recommended for production environments. For development, `mode: 'development'` will significantly improve compile times.

To run Vue in production mode, the `js/lib/vue.js` file must be replaced with the `js/lib/vue.min.js` file which can be downloaded from vuejs.org. The file imports in each of the .html files must also be updated.

D.2. Vue elements

D.2.1. Race overview page

- **v-race-form.vue**: The race form is the basic input form for editing the race information.
- **v-race-info.vue**: Race info is an element in the list of existing races on the left side of the overview page.
- **v-race-info-container.vue**: The list of v-race-info objects.
- **v-toggler.vue**: A simple checkbox-like element that looks better.

D.2.2. Dashboard

- **v-comments.vue**: The list of user-defined comments in the race dashboard page. The colours of the elements are based on the computed distance. The thresholds of where the colours change can be controlled by the values in the data object, as well as the exact colours used in the style section below.
- **v-google-map.vue**: Controls the map on the race page. This includes rendering the route, markers and computing the closest point to the active rider. The used map layer (terrain, earth or roads) can be modified here, as well as which default markers (restaurants, train stations, etc) are visible. By default, most of these are hidden.
- **v-heightprofile.vue**: Responsible for drawing the heightprofile, and the position of the rider on the profile. Also used in D.2.3
- **v-heightprofile-bars.vue**: Draws the bars in the heightprofile. Zoom levels can be modified in the data object, as well as the colours and several other settings. The number of bars drawn is equal to the number of pixels in the container.
- **v-riders.vue**: List of riders shown on the right side of the race dashboard. This class takes care of managing the active rider.
- **v-rider.vue**: The rider details per rider, can take both the form of an inactive rider and an active rider.
- **v-timeline.vue**: This object handles the timeline elements, including communication with the server to retrieve data, the time control bar, and rendering the height profile. This is the largest vue element on the dashboard page.

D.2.3. Comments page

- **v-google-streetview.vue**: This vue takes care of the comment screen, and includes all major components of the page.

E
User Guide

This is the user guide for www.telasol.nl.

E.1. Log in

- Go to www.telasol.nl
- Use an authenticator app, for instance Google Authenticator (available on Android and Apple store) and scan the QR-code (should be in your possession if you have rights to the application).
- Fill in the 6 digit-code which updates every minute and your are logged in.

E.2. Add a race

- Press the blue button “New Race”.
- Load the kml or kmz file of the race.
- Fill in the other race information fields. The race can be saved once the name of the race is filled in and at least one rider is selected.

E.3. Delete or edit a race

- Press on the right side on the race you want to edit or delete.
- Then press on the “edit” or “delete” button the the right.

E.4. Adding comments to a race

- Select an existing race on the left and press the blue “Comments” button.
- Set the streetview marker on the desired location or drag the orange bar on the height profile to the desired location to add a comment there.
- Fill in the marker icon and the marker label on the right and press save.
- To edit a comment, press on the comment on the right and the streetview and map center are set to this location. Then you can change its name or its location.
- To delete a comment, press the white cross in the red box.

E.4.1. Adding a climb comment

- Select “Mountain” as marker comment.
- Set the streetview marker to the start of a climb and press “Set start of climb”.
- The streetview marker will be moved to the predicted peak.
- Change this location if necessary and press add.

E.4.2. Adding a weather comment

- Adding a weather comment is done the same way as other comments (except for the climb comment).
- When playing the race, the weather comment is changed to live wind speed and direction information on the location of where the weather comment was.

E.5. Play a race

- Press the play button on the left next to the race you want to see.
- The race can be played and paused with the buttons next to the speed-up options. These options are 2x, 4x, 8x and 16x.
- A rider can be selected on the right to display his data. Also the distances of the comments, the weather information on the top left and the center of the map and the position on the height profile are based on this rider.
- The selected rider can be followed activating the button on the top right of the map.
- When pressing on a comment, you can see the streetview of the location of that comment. Pressing the red cross on the top right will get you back to normal view.

E.6. Weather information

- On the top left of the map, the chance of precipitation and the wind speed and direction are shown. This updates every 5 minutes and it is based on the location of the active rider.
- Divided over the course, 7 weather points are displayed which give information about the wind speed and direction on that location. These points also update every 5 minutes.

*Adding a **climb comment**:

- Select “Mountain” as marker comment.
- Set the streetview marker to the start of a climb and press “Set start of climb”.
- The streetview marker will be moved to the predicted peak.
- Change this location if necessary and press add.

Adding a **weather comment:

- Adding a weather comment is done the same way as other comments (except for the climb comment).
- When playing the race, the weather comment is changed to live wind speed and direction information on the location of where the weather comment was.

F

Infosheet

TelaSol

A Coach Cockpit Application

Team Sunweb - 2019-07-02 Final report at http://repository.tudelft.nl

To gain an edge over the competition, Team Sunweb has tasked us with the building of a coach cockpit application. This tablet-based application can be used by the coaches of Team Sunweb to track their riders during races and trainings. While similar tools do exist, they are expensive and limited in functionality. Turning the available data into a responsive and intuitive interface was the great challenge of this project. The app provides the team with the ability to evaluate and learn from past races, as well as guide their riders to victory during a live race. On the dashboard screen (in the image) the riders can be tracked on the map and corresponding height profile of the race. On the right side, the rider's vitals and upcoming points of interest can be used for analysis. Below that are the video recording (on a past race or training) and the next upcoming climb.

During our research, we mostly focused on the display of data. As our app concerns sports data visualized on a tablet, sports and tablet-based visualization were two main areas of focus in the research. While designing the user interface, we ensured to keep buttons large and the amount of clicks minimal to ensure the most appropriate design for a tablet. We used Github for version control the issue board and keeping track of all the features we had to implement for the project.

We had frequent meetings with Team Sunweb to evaluate our progress, and overall they were very happy with our continuous progression. The intention was that the app would focus on following live races, but a rule change by the International Cycling Union restricted the use of live rider data during races. In addition to this, from discussions with the client company we learned that implementing a true live view of the riders would be difficult to do within the scope of our project. As such, we shifted the focus of our application towards enhanced playback features, such as the video, allowing for future expansion of the live functionality when possible.

There is a good chance Team Sunweb will continue to develop and expand the app in collaboration with the Delft University of Technology to work towards realising its full potential. We are very excited to see where Team Sunweb can take our project in the future and hope that they will be able to make extensive use of it during their training sessions and races.

Laurens Gerlach
During this project, I served as the minutes secretary. Every meeting with the TU supervisors or the client, I took useful notes. As regards to coding, I was responsible for the input reader, from csv reader to json reader to fit reader, and the weather information.

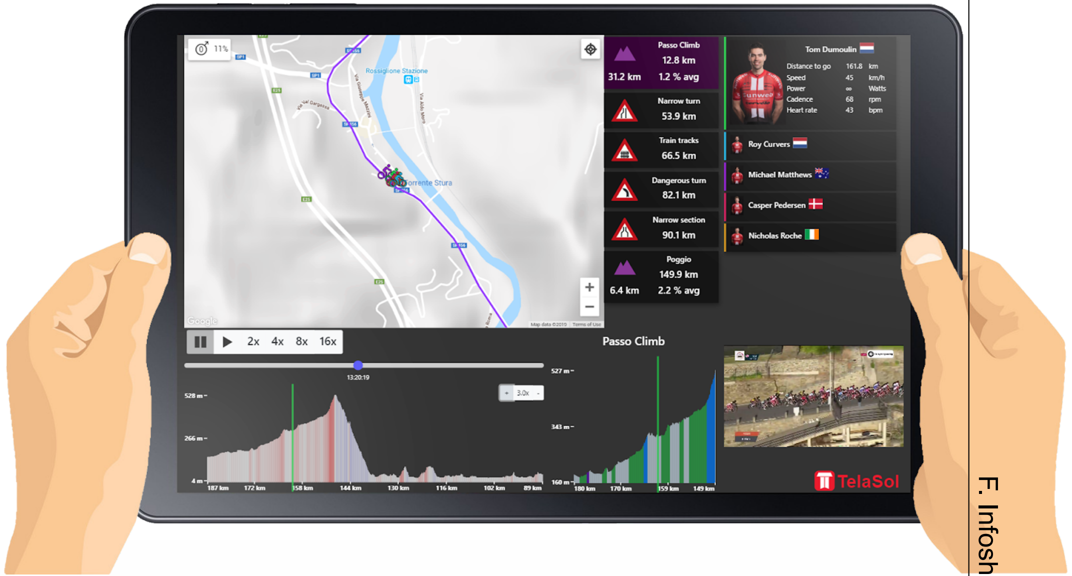
Boris Janssen
My interests include travelling, cycling and watching sports, especially cycling. During the project I served as the team's cycling expert, because I have been a devoted follower of the sport since I was very young. With regards to coding I was responsible for all features that require distance calculation and the form that allows you to create races.

Mirco Kroon
Outside of computer science, I have interests in photography, videography and visual design. These were of use during the project through the design of logos and other material. My role in the project was to lead the software development aspects and to ensure that our application remained consistent in quality and written with maintainability and extensibility in mind.

Sam Vijlbrief
As an avid viewer of professional cycling and sports in general, this project was an amazing opportunity for me. The sports analysis aspect combined with the science behind designing the application, was an entertaining and useful experience for me. During this project I mostly served as team leader, doing most of the planning of the project and discussing with the client and coach. Coding-wise I mostly focused on User Interface and some other technical aspects.

Client: Rado Dukalski, TU Delft Sports Engineering Institute
Team Sunweb Liason: Teun van Erp, Scientific Expert at Team Sunweb
TU Delft Coach: Gosia Migut, Department of Intelligent Systems

Contact Info Team TelaSol:
Laurens Gerlach, student, laurensgerlach@gmail.com
Boris Janssen, student, boris.fabian@hotmail.com
Mirco Kroon, student, mirco.kroon@gmail.com
Sam Vijlbrief, student, samvijlb@gmail.com



Bibliography

[1] Angular. URL https://angular.io/.

[2] Bootstrap. URL https://getbootstrap.com/.

[3] Cesium. URL https://cesiumjs.org/.

[4] Chart.js. URL https://www.chartjs.org/.

[5] Golden cheetah. URL https://www.goldencheetah.org/.

[6] D3. URL https://d3js.org/.

[7] Google charts, . URL https://developers.google.com/chart/.

[8] Google maps, . URL https://www.google.com/maps.

[9] Highcharts. URL https://www.highcharts.com/.

[10] Quarq qollector. URL https://www.quarq.com/product/quarq-qollector/.

[11] React. URL https://reactjs.org/.

[12] Strava. URL https://www.strava.com/.

[13] (live) veloviewer. URL https://veloviewer.com.

[14] Vue. URL https://vuejs.org/.

[15] Websockets. URL https://developer.mozilla.org/nl/docs/WebSockets.

[16] Agile Business Consortium. The dsdm agile project framework (2014 onwards), 2014. URL https://www.agilebusiness.org/content/moscow-prioritisation.

[17] John E Dunn. Cycling 2.0 – how the tour de france is reinventing itself using big data, Jul 2016. URL https://www.techworld.com/data/cycling-20-how-tour-de-france-is-reinventing-itself-using-big-data-3642667/.

[18] P.M. Fitts. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, 47(6):381–391, 1954. doi: 10.1037/h0055392. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-46649087024&doi=10.1037%2fh0055392&partnerID=40&md5=50d73f664b95513b645bc27c7eaa222b. cited By 4207.

[19] Peter S. Games and Joshi Alark. An evaluation-guided approach for effective data visualization on tablets, 2015. URL https://doi.org/10.1117/12.2076523.

[20] Asker E. Jeukendrup. Nutrition for endurance sports: Marathon, triathlon, and road cycling. *Journal of Sports Sciences*, 29(sup1):S91–S99, 2011. doi: 10.1080/02640414.2011.610348. PMID: 21916794.

[21] Mike Beedle Kent Beck et al. Manifesto for agile software development, 2001. URL https://agilemanifesto.org/principles.html.

[22] Alejandro Lucia, JESÚS Hoyos, JosÉ L Chicharro, et al. Preferred pedalling cadence in professional cycling. *Medicine and science in sports and exercise*, 33(8):1361–1366, 2001.

[23] R. Lukes, J. Hart, and S. Haake. An analytical model for track cycling. *Proceedings of the Institution of Mechanical Engineers, Part P: Journal of Sports Engineering and Technology*, 226(2):143–151, 2012. doi: 10.1177/1754337111433242. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-84871872287&doi=10.1177%2f1754337111433242&partnerID=40&md5=479456f3b5fcf76f9930a55550fc3820. cited By 8.

[24] Iñigo Mujika and Sabino Padilla. Physiological and performance characteristics of male professional road cyclists. *Sports Medicine*, 31(7):479–487, Jun 2001. ISSN 1179-2035. doi: 10.2165/00007256-200131070-00003. URL https://doi.org/10.2165/00007256-200131070-00003.

[25] J.J. Muros, C. Sánchez-Muñoz, J. Hoyos, and M. Zabala. Nutritional intake and body composition changes in a uci world tour cycling team during the tour of spain. *European Journal of Sport Science*, 19(1):86–94, 2019. doi: 10.1080/17461391.2018.1497088. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-85049933447&doi=10.1080%2f17461391.2018.1497088&partnerID=40&md5=a36b94cb4521d890d7cac8818a99b1c6. cited By 1.

[26] R. Nieuwenhuizen. Visual analysis for sports sensor data. pages 11–15, 2017.

[27] C. Perin, R. Vuillemot, C.D. Stolper, J.T. Stasko, J. Wood, and S. Carpendale. State of the art of sports data visualization. *Computer Graphics Forum*, 37(3):663–686, 2018. doi: 10.1111/cgf.13447. cited By 1.