Mälardalen University
School of Innovation, Design and Engineering
Västerås, Sweden

Software Engineering 2 : Project Teamwork - 7.5 hp - DVA313

BLACK RIVER RUN TIME KEEPING

Group 5 - Project Report

Bastien Delbouys
bds18002@student.mdh.se

Zacharias Claesson
zcn16001@student.mdh.se

Sebastian Oveland
sod16003@student.mdh.se

Cécile Cayèré
cce18001@student.mdh.se

Mohammed Abuayyash
mah18005@student.mdh.se

Rikard Gestlöf
rgf16001@student.mdh.se

Johannes Sörman
jsn16009@student.mdh.se

January 15, 2019

Contents

1 Introduction

The purpose of this document is to provide an in-depth analysis of the Black River Run project conducted this year.

Firstly, the document explains the project itself. It presents the client, explaines the context and the initial goal. This part ends with the changes made during the development process to the client's requests. Then, the front-end and the back-end parts of the final product is presented. For each parts, acceptance tests were conducted. Next, a work analysis is performed. Finally, the document shows feedback from members of the group towards this project.

2 Project

2.1 Client

The representative of the client is Christoffer Holmstedt from the Organisation Västerås Running Club. Some important information about the client are shown below:

- **Name:** Christoffer Holmstedt
- **Organization:** Västerås Running Club
- **Email:** christoffer.holmstedt@gmail.com
- **Phone number:** +46 (0)73 7816126

2.2 Context

Västerås Running Club has an annual race called Black River Run which is held in September, and consists of several running categories for men and women:

- 20 miles race.
- 50 miles race.
- 100 miles race.

The race is held in Västerås and the runners have to follow a determined path around the Svartån. The race track has four checkpoints (stations) stationed around the whole lap. In order to complete the race correctly, the runners have to pass these stations. Each runner has an SI-unit that works as an identifier for the runner. The stations are sending out signals to the SI-units in an approximate area of each station. When a SI-unit receives a signal, it connects to the station that sent the signal. The station forwards a timestamp related to the runner to an online server (Olresult). The information, called punch, is later fetched by a PHP script from that server to the database.

2.3 Initial Goal

The goal of this project is to create a new web application that displays the results of the active races as well history of previous races. Through this interface, an administrator should be able to manage data of the system. In order to do this, a new database was implemented. To connect the database with the webpage and the database to the server, the group used PHP.

2.4 Changes

There were no major changes during this project.
However, some features to add to the website were asked by the client :

- A map for locating and showing an active individual runner's estimated position. This feature was only discussed on the first meeting, but the client representative and the group agreed on the second meeting, that the group had time to implement this in the product.
- Several views that differs in style from the website in order to be display on the race site TV screens.
- Different pages of results with different informations.

Some minor changes were also asked by the client like style change (color, etc).

3 Final Product

3.1 Front-End

One of this project deliverables was website with server-side PHP script that allows database interaction. The client had two intended users for this website:

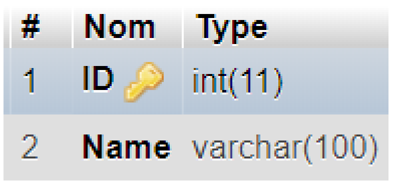
- The guest user, that can consult some data.
- The administrator, that can have direct interaction with the database.

To go deeper in the subject of the website the features will be listed with regards to which user it is intended. Then a parenthesis is done to show some of the changes the team had to deal with. Finally, the acceptance test for the front-end is explained.

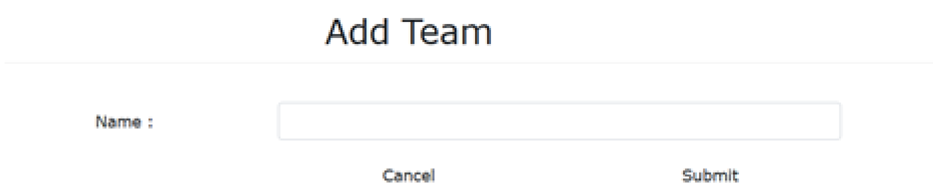
3.1.1 Implemented Features

At first the website features could be divided into five categories:

- The database interactions means that a logged in user (or administrator), can INSERT, DELETE, UPDATE elements in the database such as Runner, Race, Team, SI-units and Categories directly through the web-interface.
- The data visualization is performed, whenever data is displayed on the web page by fetching data directly from the database this includes displaying, searching and sorting functions, which is useful for every user.
- The account system, allowing log in, log out, and a settings tab which allows an administrator to change its password.
- Responsive UI.
- Additional functionalities, this part is detailed further in the report.
- **Database Interactions**
  One of the most common database interactions consists of filling out fields in a form to match the database:



#	Nom	Type
1	ID	int(11)
2	Name	varchar(100)



Add Team

Name :

Cancel Submit

  The two other ones are deletion and update, these are also performed from the website.
  On the website another form exist which allows the creation of a race. In detail that means when a race is created the administrator can add its categories, runners, and SI-units used in it.

- **Data Visualization**

  The client asked for many different things to be show on the website, here is the list of the elements viewable on the website by any user:

  - The visualization of a race results which can be performed by visiting any race.
  - The list of every runner, teams, races can be consulted by anyone, these lists can be searched and sorted to ease the task of searching for something/someone.
  - A profile for every runner is accessible from the runners list or a race to know more about his personal records.
  - Other views are available on the website for the administrator, as the client wanted to have the results display in direct on television around the track two kinds of results view are available from any race category when logged in.
  - The last things consultable on the website is the final results of a race listing every participant of it and their time with their place.

- **Account System**
  It is not possible for a user to access the account system or the administrator login page through the main website. In order to access it, he/she must know the URL for it. Once logged in, the user has access to the administrator features as well as the guest features.

- **Responsive UI**
  The website is supported by most devices. In terms of screen size, the website will automatically adjust its content to fit the size of the device used to view it.

- **Additional Functionalities**
  In this last category, non-classable features are listed and explained.
  The first of these features is the import-export feature, which is only available for administrators. It allows the possibility to download data about the race participants of a specific race (bib, SI unit, runner name, runner birth date, category, team) in the form of an excel file. It is also possible to upload excel files to import data into the database. This is useful because it allows the administrator to insert initial data about multiple runners at once.
  Another feature of the website is a dynamically updated map that shows the estimated position on the track of a specific runner during an active race. This map can be accessed by anyone when viewing the data of an individual runner that is registered to a current race.

Some acceptance test was done by comparing the final features looks and behaviors to the original idea formulated by the client as user stories. Whenever a user-stories was not please changes were done on the website to please them. In the following table the list of user stories is shown in comparison to the actual features of the website;

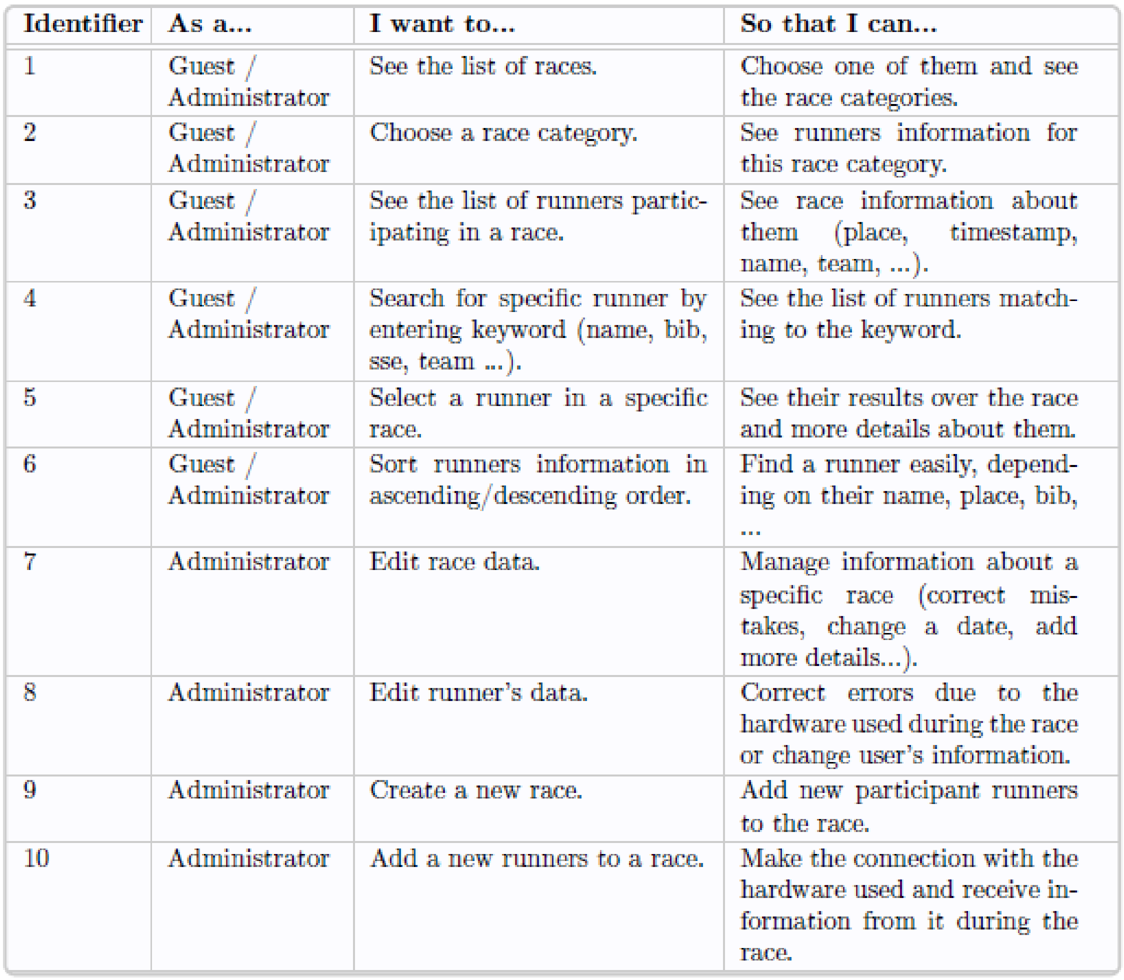
As a reminder here is the original table listing the user stories we had to please:

3.1.2 Non-Implemented Features

Every required feature that was specified by the client was implemented. Some of his non-required ideas was also added to the website (dynamic map, different views, etc).
However, one of the mentioned optional features have not been implemented for lack of time. This feature is the possibility of changing the language of the website.

3.1.3 Acceptance Tests

As a reminder, table 1 lists the user stories we had to please:



Identifier	As a...	I want to...	So that I can...
1	Guest / Administrator	See the list of races.	Choose one of them and see the race categories.
2	Guest / Administrator	Choose a race category.	See runners information for this race category.
3	Guest / Administrator	See the list of runners participating in a race.	See race information about them (place, timestamp, name, team, ...).
4	Guest / Administrator	Search for specific runner by entering keyword (name, bib, sse, team ...).	See the list of runners matching to the keyword.
5	Guest / Administrator	Select a runner in a specific race.	See their results over the race and more details about them.
6	Guest / Administrator	Sort runners information in ascending/descending order.	Find a runner easily, depending on their name, place, bib, ...
7	Administrator	Edit race data.	Manage information about a specific race (correct mistakes, change a date, add more details...).
8	Administrator	Edit runner's data.	Correct errors due to the hardware used during the race or change user's information.
9	Administrator	Create a new race.	Add new participant runners to the race.
10	Administrator	Add a new runners to a race.	Make the connection with the hardware used and receive information from it during the race.

Figure 1: User-Stories

Acceptance tests were also conducted. This was done by using different user-stories as starting points when comparing the features and behavior of the final product to the original idea formulated by the client. Whenever a user-story was not pleased by the current version of the website, changes were done in order to try to please them. In the table the list of user stories is shown in comparison to the actual features of the website;

Identifier	Location (page)	How to do it?
1	Races tab	Click on a specific race in the list, then more details about it will be displayed on screen, on the left side menu the race categories are displayed.
2	On a selected Race	Click on the desired category of the left side menu to see more details about it.
3	On a selected Race	This is displayed on as the Race summary on every race when accessing its page.
4	Runners tab, on a selected Race	The search bar can be filled to search for a specific runner easily.
5	On a selected Race	Runner line are clickable to lead to their personal profile page where many data related to them are available.
6	Runners tab	The name of the column can be clicked to sort in an ascending or descending way the list of the runners to find a runner easily.
7	Dashboard	On the dashboard in the side menu Current and Planned Races can have their complete data edited, such as changing name, dates, participants.
8	On a selected Runner	When on a specific runner pages by default his last race is displayed, it is possible to see other races on the history tab in the left side menu. When the desired race is displayed the runner timestamps can be edited, deleted, and even added. The system will calculate base on the entries some data such as Status (running, finish, do not finish, do not start), Place, time behind, total time, to display and store some of it.
9	Dashboard, Race tab	The initial creation of a race can be done from both pages, I will require some data such as Name, Starting Date, Ending Date, Finishing hour of the event. Then the race data such as runner list, categories is doable as specified in 7.
10	Dashboard	First the Race where you want to do action has to be selected in the dashboard. From here, runner data related to a race can be added directly from the form mentioned in 7, or by uploading an excel file filled with some data following the client specifications.

This table shows that all the client requirements were validated and tested to acts as intended. The other functionalities not mentioned in the initial requirements were tested by using back-ended generated data to ensure that every behavior is working as we describe them.

This end this part of the report, where all of the final website features were listed and explained.

3.2 Back-End

3.2.1 Implemented Features

- **Database**
  The group have built a database to store information regarding the Black River Run. The

database holds all the race information, such as race dates, running classes, time tracking units and stations along the route. The Database also store everything the website needs to know about the runners, such as name, date of birth, gender and where the runner is from. This is used to register a runner to a race in the proper class and to register them to a time tracking unit (SI-unit).

When the runner passes a station, a timestamp is sent together with a unique SI-unit number to the database and is then connected to the correct runner by mapping the SI-unit in the database. This way the database stores all information needed to track the runners along the race.

- **Validation Script** (*getPunches* script)
  To make sure all information is fetched and stored correctly in the database, the group have developed a script that validates all information before it is saved in the database. This script makes sure the time tracking units are correctly registered and that the timestamps are within the accepted timeframe.
- **Race Simulator**
  To test the backend parts, the group have created a race simulator that creates a requested number of runners with random data that runs the race and sends timestamps to the script.

3.2.2 Non-Implemented Features

The only thing the group have not implemented is a complete testing environment where we quickly can see if a part of the system passes or fails.

3.2.3 Acceptance Tests

The implementation of a new back-end was an optional demand by the client representative and it had no requirements except that it should function as it previously did. Therefore, the client provided the rules of which the back-end script should fulfill. The client also provided the structure of the current database that client is using. The client representative had no direct requirements on the new database except that it should mimic the functionality of the old.

Since the client representative had no requirements on the back-end except that it should mimic the old back-ends functionality made it hard to perform an acceptance test. However, the group has performed other tests on the back-end to make sure that its functionality mimics the old back-ends functionality.

For the GetPunches script (the script that uploads data to the database) the group has created a test script which simulates the API during an active race. This script can be called with the same options as the real API and gives output in the same way. The only thing needed to change to run the tests is the requested URL in the GetPunches script which makes sure it's the same code running as if it was a real race. The test script outputs punches that can be invalid in some way that the back-end(GetPunches) script should be able to handle according to the rules the client provided.

Those kinds of invalid punches could be: a runner cheats by passing one lap too fast(one lap should not be able to be passed faster than 40 minutes), the punches from the stations are received in the wrong order based on time, if a station transmits multiple punches with a time with the interval of +-5seconds, and if an SI-unit breaks during the race it must be replaced. The result of this test had to be viewed manually in the database to see if it was correct. The group has executed the test script multiple times, and the *getPunches* script have been able to handle it.

The group would have preferred to do a real acceptance test on the back-end together with the client representative, using the real hardware(stations) and not only doing it virtual to see if it works properly. The group have not had the time or the opportunity to talk to the client representative about this.

Most of the database calls are hard coded. However, SQL-injections have been prevented when these kinds of dynamic calls occur.

The test script runs test statements to make sure the *getPunches* script acts and reacts the way it should. These test where done without the client representative being present. The statements are stated as follows:

1. Happy route
   - The runner finishes 100 miles without any errors at all. All punches come one by one at the correct station in the right order.
2. Delayed sending from SI-unit
   - The SI-unit missed to send a punch and sends it at a later stage. These punches can come as late as five stations later than expected, meaning that the runner can complete a whole lap without any punches getting sent. When these finally get sent, all the missing punches are sent at the same time in random order.
3. Broken and replaced SI-unit (including missed stations until start)
   - The SI-unit stops working during the race and gets replaced at the start of the next lap. Any stations passed after the SI-unit broke are missed until the runner reaches the starting station.
4. Finishing a lap in less than 40 minutes
   - A runner finishes a lap in less than 40 minutes. That runner is most likely cheating.
5. Broke race after 50 miles
   - Running the 100 miles race and breaks after 50 miles.
6. Broke race after 70 miles
   - Running the 100 miles race and breaks after 70 miles.
7. Broke race after 10 miles
   - Running the 100 miles race and breaks after 10 miles.
8. Did not start the race
   - A runner that signed up for the race, but never started running.
9. Multiple punches within a few seconds
   - The SI-unit sends more than one punch as the runner passes the station. The extra punches are within two seconds after the first punch at that station.
10. Random runners where any earlier error can occur
    - N number of random tests where any of the other listed errors can occur.

4 Work Analysis

During this part of the report, the working aspect of the project will be detailed.
First, some reminders of the organization and the routines followed during the project are detailed. Then, the changes that were applied are listed. Following that, the effort repartition within the project on different activities is detailed based on working hours, GitHub submission, amount of work, respect of the deadlines. With this detail, assigned working team member will be listed with their activities and roles. This part ends with the individual working hours of each team members.

4.1 Organization And Routines

The group was divided into two sub-groups; the Front-End Group and the Back-End Group. The group also decided early on to use Microsoft Teams and GitHub as tools to keep track on documents and code.

Each week started with a team meeting before the weekly meeting with the steering group. The purpose of this project group meeting was:

- Prepare the upcoming presentation.
- Report all working hours.
- Assign the tasks for the week.
- Sometimes, split task in two to assign work to several people.
- Sometimes, delay the deadline of the task that had not been completed in time. A small investigation of why the task was not finished in time would also be conducted.

During the week, the group members coordinated themselves to finish their tasks in time. An additional meeting or discussion was also organized towards the end of the week in order to see if things were going in the right direction.
At the start of a new week, a mail was sent to the client representative to keep him updated on what was done the previous week, as well as to ask him any questions that might have arisen.

While the routines throughout the project were kept the same, the organization of it changed a bit.
The first change was the re-definition of the working fields of the two original groups that compose the team, the Back-End and the Front-End Groups. The Back-End Group oversaw managing the database and working on back-end PHP script that communicate with the external systems of our system. The Front-End Group name was replace by Website Group, as their working area was focus on the website development in terms of both front-end with the UI design and implementation and back-end with the functions communicating with the database.
The second change was that the group switched their original Gantt chart for a Kanban Board. This choice was motivated by the purpose of more detailed activities, goals and milestones.
Finally, the last change in how the group communicated/worked, is the introduction of Discord to do online meetings as members of the group were not available to meet in person between the 19th of December to the 13th of January.

4.2 Effort

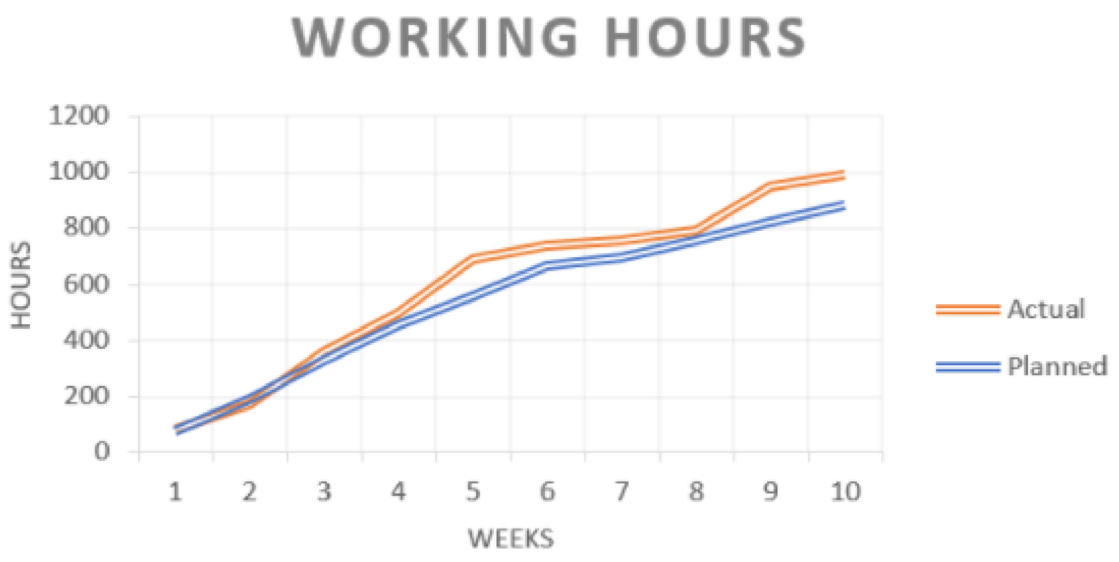


Figure 2: Working Hours

Graph 2 shows the total working hours, which helped to determine together with the data gathering of GitHub to see where and when the work was more intense than originally planned. We can see that two pikes present on the *Actual* line, this represents the two implementations phases.

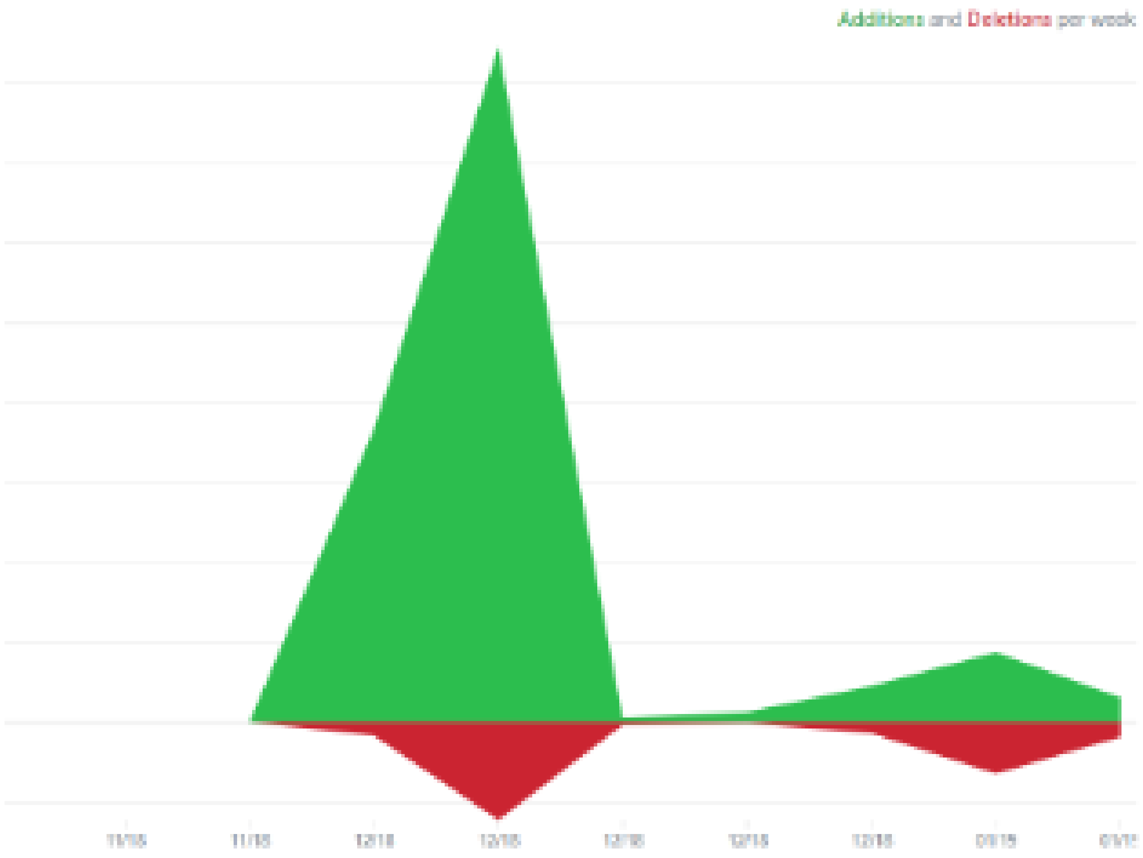


Figure 3: GitHub Repository Contribution

This effort is also confirmed by the graph 3 available on GitHub by clicking on the insight tab. This graph shows the contribution on the repository over time.

The following table gives the effort for each weeks of the project and the total project effort based on this data is 991 hours:

Week	46	47	48	49	50	51	52	1	2	3
Effort (hours)	83	88	189	137	192	50	20	36	156	40

During the first two week the work was focused on documentation and preparation, the project plan and the UML design were the main cause of effort. At week 48, the implementation phase of a prototype for UI design was started. On week 49, the detail design document and the development of the UI-design were the focus. On week 50-51 the prototype of the UI design came to an end and the first testing phase started in parallel. The further development of the prototype started slowly at the end of the year and hit its peak during the second testing phase from week 52 to week 2. And to end of the project on week 3 of 2019, the effort was distributed on finishing the project reports.

In order to detail the work distribution of the different activities, Graph 4 shows the original defined activities in the groups Gantt Chart.

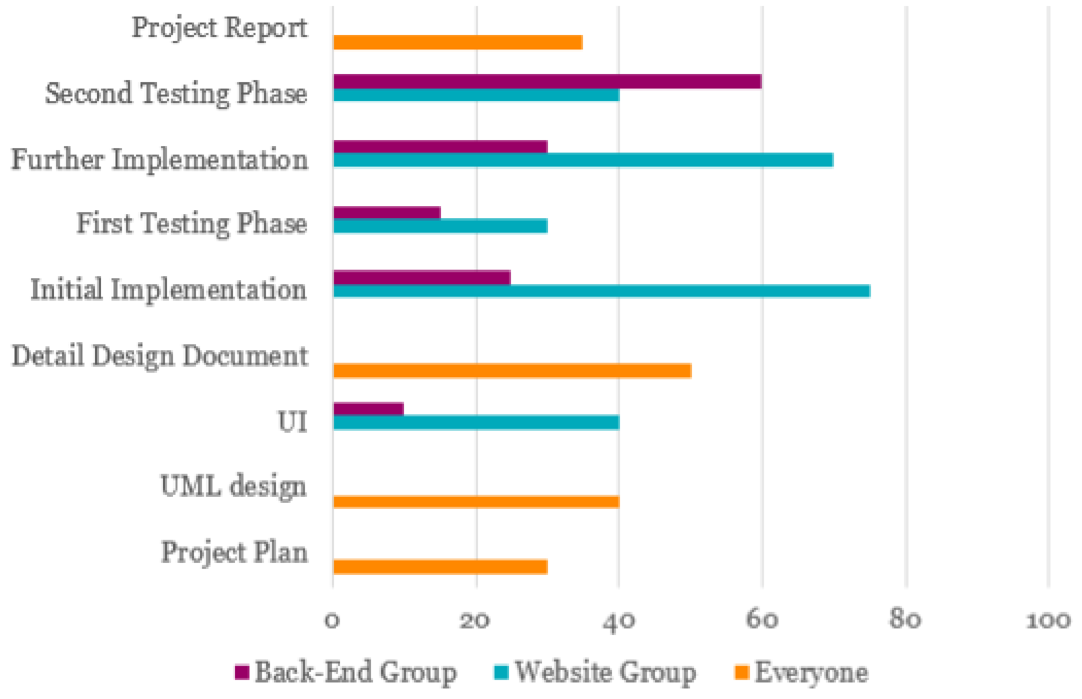


Figure 4: Work Repartition

The bars in Graph 4 show the time effort spent on each activity, as the hour reporting was done weekly, and the original activities continued during several weeks. The group had to consider that on this graph the team could not exceed a total of 100% of effort in a given period of time.

To complete the previous graph, Graph 5 shows the amount of working hours in the initial and second phases respectively of the project.

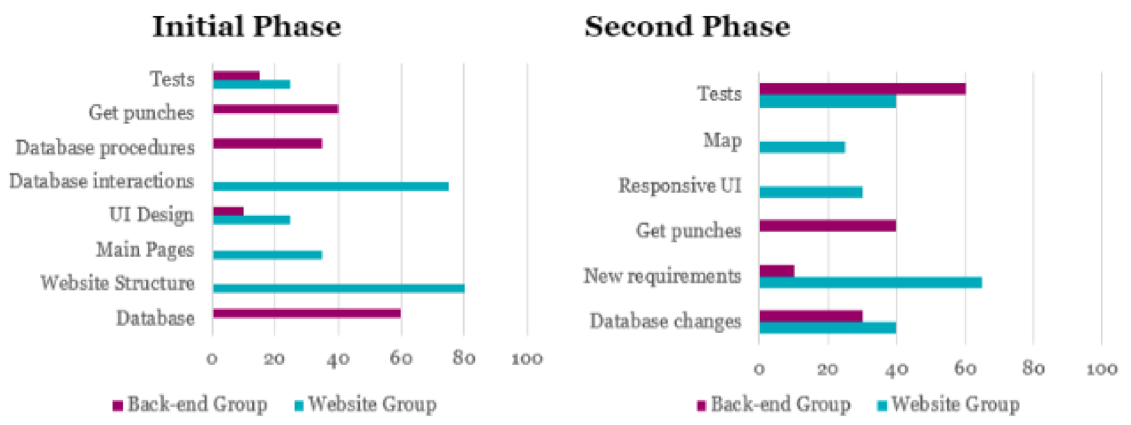


Figure 5: Initial And Second Phase Work Repartition

4.3 Work Load

The work load could be divided in two different parts, the document work and the application work. This part uses the task history of Teams and the data found on GitHub.

Concerning the document work every member had their own part to document before a deadline. Then all of the documentation was mainly reviewed by Rikard, Zacharias, Johannes, Cécile.

Once the documents were done it was copied into a Latex file by Cécile.

Regarding the actual development, as explained previously, members of the group were divided in two groups:

- Back-end Group:
  - Sebastian Oveland
  - Johannes Sörman
  - Rikard Gestlöf
  - Zacharias Claesson
  - Mohammed Abuayyash (until the end of the initial implementation, he then switched Website Group)
- Website Group:
  - Cécile Cayere
  - Bastien Delbouys

Each member had different responsibilities and activities along the project, primarily within their working area. If every member has worked equally on an activity only the group will be mentioned, if a member has worked with a member with responsibilities only the name of the member with responsibilities will be mentioned, if names only are mentioned it means that the activity was done by the people mentioned.

Persons with global responsibilities along the project were:

- Bastien Delbouys, was the project manager, in charge of the task repartition, preparation of the meetings, and the organization. .
- Zacharias Claesson was the client contact, in charge of contacting him and reporting interactions with him.
- Sebastian Oveland, was in charge of the maintenance of the database, assisted by the back-end group when needed.

Initially the Back-End Group decided on the structure of the database before starting any implementation. From this point on, the back-end and front-end of the system could interact with each other by using the database. The back-end fills the database with data from information fetched through an external API. The front-end fetches information from the database and displays it on the website.

During the first week of the project, the group member Cécile Cayèré oversaw the development of the first structure of the website, as she was most experienced in PHP in the Website Group. The rest of the Website Group used the first week to learn PHP and experiment with it.
The Back-End Group planned and started the implementation the database during the first week.

In the second week the website structure definition and main page development continued with Cécile as the responsive of the activity in the website Group. At this point the website allowed to view, add and delete certain data from the database with basic SQL queries.
The Back-End Group continued to work on the database, as well as doing some research about the communication with the external API on which the *getPunches* script had to use.

The following week an initial dashboard was deployed by Cécile. Bastien worked on the timestamp related functionalities and runner data when running (place, time behind, status).
The Back-End Group started working on the *getPunches* script and on some database procedures.

The dashboard ending was then the focus of the Website Group, with Bastien as the responsive developer. The Back-End Group continued working on the *getPunches* script.
Just before the first product delivery first tests were conducted on the website by the Website Group.

The development of new pages and view were done by Bastien such as Race summary, Race results, Race tracking and merge into the website by Cécile.
A first testable version of the *getPunches* script by the Back-end Group, starts a wave of testing on and with it.

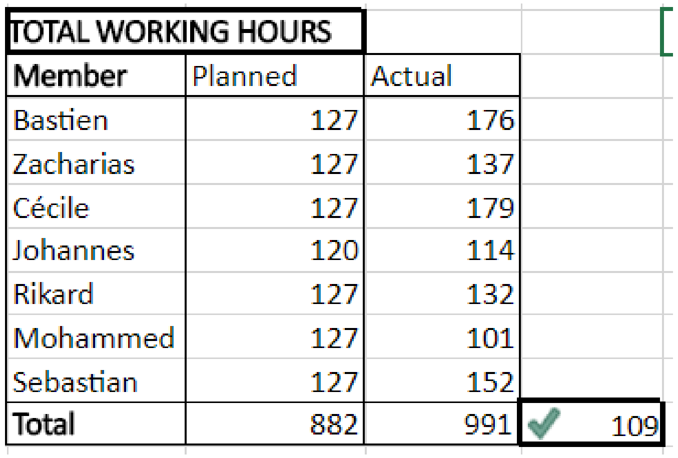
Johannes and Bastien conducted user-tests using the web interface to people outside the group members. Bastien then focused on importing and exporting excel files to the website and he also adjusted the website SQL queries to match changes made in the tables from the database. Cécile focused on bug fixing and changing the desktop website to be responsive. Mohammed started the development of a dynamic map to display a runner's position.
The Back-End Group started the development of a race simulator, a program that helped testing both side of the product by using the *getPunches* script and by testing if the website is presenting the data correctly.

During the final two weeks, the Website Group worked on the map and the final tests. Bastien was responsive on the last requirements formulated at the middle of December. While the Back-End Group continue its testing.

The member had to report their personal working hours each week throughout the whole project, furthermore the team manager had to report common working hours such as meetings. This choice was motivated to keep a track of the personal investment and work contribution.

While observing the results, the group could see that the number of working hours is proportional to the contribution to GitHub, that validate the reported hours for every individual in the group. The group could clearly see a gap in the working hours between the Website Group and the Back-end Group. This was due to a higher number of activities to work on for the Website Group, then was initially expected. Furthermore, the whole group had a high apprehension of the amount of work the Back-end Group had to put in from the beginning of the course. This led to that the Back-end Group was assigned most of the members from the start. The discrepancy between the front-end and back-end working hours was pointed out during a Weekly Meetings and addressed by assigning a third member to be a part of the Website Group. On top of this change, more of the report writing was also assigned to the Back-end Group such as text reviewing, to compensate for this problem.

4.4 Work Hours



TOTAL WORKING HOURS			
Member	Planned	Actual	
Bastien	127	176	
Zacharias	127	137	
Cécile	127	179	
Johannes	120	114	
Rikard	127	132	
Mohammed	127	101	
Sebastian	127	152	
Total	882	991	✓ 109

Figure 6: Personal Working Hours

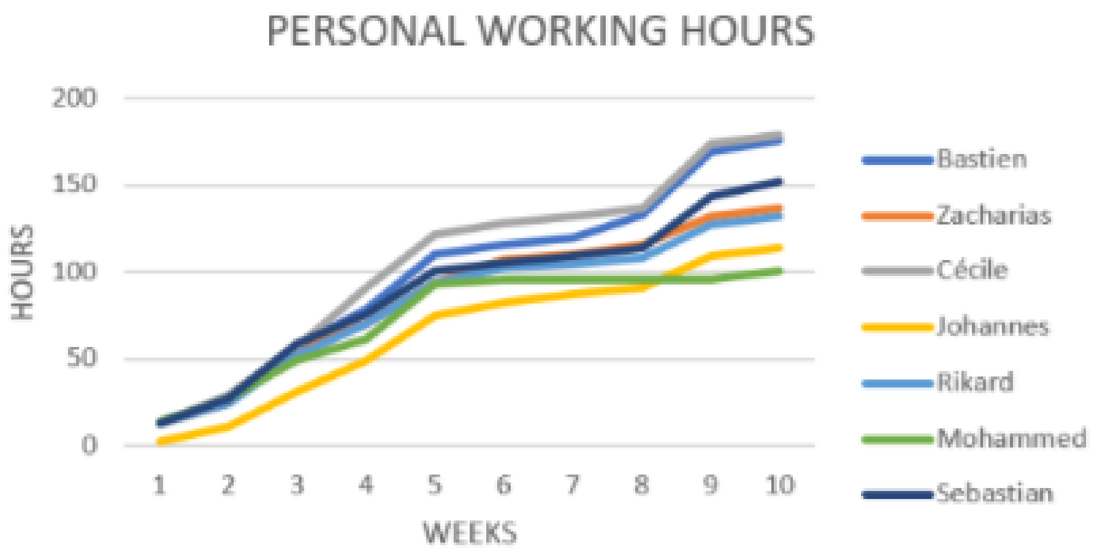


Figure 7: Personal Working Hours

Figure 6 shows the actual personal working hours compared to the planned personal working hours. Figure 7 shows a graph of the evolution of working hours over time by person.

5 Feedback

5.1 Experience Gained

During this project the group had to work a lot with PHP, SQL and HTML+CSS. This presented an excellent opportunity for the group members to either learn some of these languages from scratch, or sharpen up already possessed skills.
The group members who had special assigned roles (client contact, project manager, etc...) also got to experience how this works in practice and not just in theory.
Due to the diversity of the group members in terms of background and language, there were some

barriers that had to be overcome. This was particularly obvious when the group members were communicating with each other, as misunderstandings were prone to happen. This eventually led the group members to be extra clear when communicating with each other to avoid misunderstandings.
When working individually it is not enough to do a good work on what you were assigned, but also report back to the group that it is done and how it works (how it is intended to work).

5.2 Positive Experiences

Meeting and working with people from different backgrounds has been a positive experience shared among all the group members. This made it clear that there were some different approaches to how the members preferred to work in many different aspects. This provided the group members the opportunity to share and learn from each other. The group members have been opened to helping each other, when someone does not know what to do or do not have anything else to do.
Without repeating it, the things mentioned under "Experience Gained" are obviously positive experiences as well

5.3 Possible Improvements

To make sure everyone has the same view of the project, even if they themselves are not implementing that part. With everyone involved in the different parts the group gets many different inputs and angles that one member can't think of themselves. Another problem was the size of the product, which in this case was small and therefore hard to divide properly among the group members. The solution to this could have been by possibly talking even more, but the group feels that over all they did a decent job in communicating. If we talked more about everything, we would have been talking more than anything else and maybe not been able to finish the project instead. It is a fine line to walk on and is only solved by experience with these kinds of projects.
The group member should also have started testing the product earlier than what they did. This led them to some stressful moments towards the end of the project.

5.4 Advices For Future Students

Students that are taking the course next year should know that proper communication is extremely important, especially when working with people that are speaking different mother tongues. They should also make sure that every group member knows exactly what the client requirements are. If there are any uncertainties, bring them up as early as possible and sort them out to avoid stressful situations further along the project.
Do not under value the importance of the system structure. Try to figure it out as soon as possible, and as thoroughly as your group can.
Another advice would be for them to use tools and programming languages that all group members have at least some experience in using. Trying out new tools and languages are great, just do not overdo it. There should be a fine balance between what you know and what you want to learn. Remember that this is a real product that a client wants delivered within a specific amount of time and jumping onto completely new tools and programming languages can consume a lot of valuable time. Of course, sometimes the project itself requires you to learn completely new things, but if you do not have to, consider using known tools.