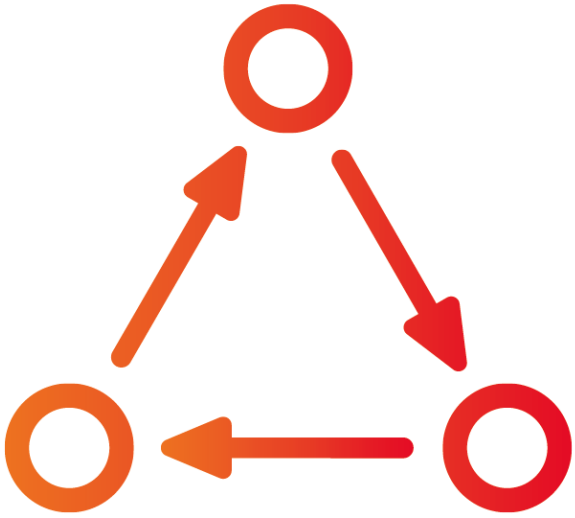
# Tactalyse-2

# Requirements Document



**Sangrok Lee (S3279480)**

**Bianca Raețchi (S4755138)**

**Matteo Gennaro Giordano (S5494974)**

**Mikko Brandon (S3781356)**

2023

RUG

# Contents

# Glossary

| Term | Definition |
|---|---|
| Tactalyse | The company that we have developed our project for |
| Football | The european term, not to be confused with American football. |
| Player position | Indicates football positions such as goalkeeper, winger, midfielder, etc. |
| (The) service | An abstraction of the generator application currently discussed (graph or PDF) |
| Graphs/Plots | Radar charts, line plots, or bar plots containing football player statistics. |
| Radar chart | A circular plot that showcases parameter values on axes of the circle. |
| Coaches | Football coaches working with Tactalyse. |
| Players | Football players. They're Tactalyse's clients, consulted by Tactalyse coaches and the recipients of the PDF reports. |
| (The) client | Depending on context either football players, or Tactalyse, as defined above. |
| Visual modules | Code modules that contain functionality for generating and outputting visual data. |
| (The) team | The student group that worked on the project, as listed on the title page. |
| Data files | The files that contain the data we need to generate the graphs (the league and player files, expanded on below). |
| Backend | The APIs we have developed, including endpoints and services. |
| Frontend | The web app developed by team Tactalyse-1 that will be used to send requests to our PDF generator backend. |

# 1. Introduction

Tactalyse is an organization that aims to provide data reports about football players. The data report contains statistics and graphs based on the football players' match data. To build those graphs and statistics, coaches who are users of Tactalyse upload the players' data in Excel file format on the company's website. Then, the Tactalyse frontend service receives the data and sends it to the Tactalyse backend service. The backend service generates a data report. These reports can be one of two types:

- Standard report: contains line plots and bar plots reflecting the target player's football match data.
- Comparison report: every generated plot has an additional line or bar for another player in the same position and same league. With this, clients can compare themselves to their fellow players.

The data report will be e-mailed to coaches by the frontend after being generated, for improving the players' performance. In this project, our main task was to implement reception of data from the frontend website, create related graphs and plots based on the received data, and create data reports from these graphs.

The company already had a completed mechanism that received data and created related graphs and reports. However, the original system had several bugs, and was written without good code logic and with no architecture in mind. Additionally, the used libraries were not optimized, and very slow. Since data report generation speed has an impact on the company's work performance, the owner wanted us to build their project from scratch, rather than extend the original project. In the below figure, we provide a simple workflow for this system. The first part of our project is about generating data reports and it is marked in yellow in the figure:


Figure 1: A simple process model diagram of the data report system

Additionally, Tactalyse is looking to increase the size of their customer base, and to that end, also brand awareness. For this, they want to have a Twitter bot that posts radar charts containing football statistics daily (check Appendix B for example). It operates automatically, and relies on local data files that are updated by Tactalyse employees. To account for user interaction with Tactalyse services, they also want to have a Discord bot. This bot allows users to make requests to generate a single graph. They have their own public discord server where this bot is used.

The Twitter bot mechanism already existed previous to our project. However, it did not generate graphs on its own; it took local image files, and uploaded them. This made the system quite rigid, and only allowed for as much variety in graphs as there were local image files. The Discord bot would ideally allow for users to customize their graphs to a greater extent than possible with local graph files. We were tasked with creating a graph generator that creates graphs on demand, with parameters specifying what should be in the graph. This graph generator could then be used by the Twitter and Discord bots. A diagram for the workflows of this system is provided below.

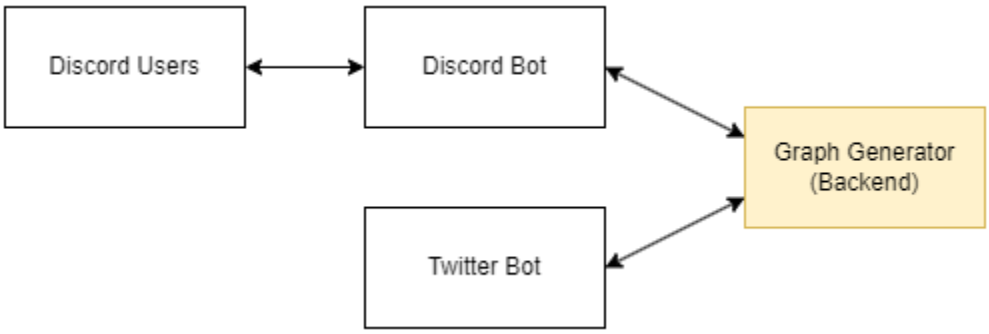



Figure 2: A simple process model diagram of the graph generator system

# 2. Stakeholders

For this project, there are multiple stakeholders who will be affected by our app. In order to make sure all of their requirements are satisfied, a description of their potential expectations is provided.

## Actors

- **Tactalyse employees**: they are football coaches employed by Tactalyse, and they will be the main users of our app, tasked with sending the clients the generated reports. Therefore, creating an *intuitive* and *efficient* app for the employees to work with is going to reduce the potential for errors in the future and should be kept at the forefront of our design choices.
- **Twitter bot**: since the Twitter bot posts daily graphs on Tactalyse's Twitter account, it will need to access our app. The bot is a system, so its main concern is that it gets proper output from our app, that it can process into a Twitter post. For this, we need to make sure that our output is *properly formatted*.
- **Discord users**: they can create requests to Tactalyse's Discord bot, which then accesses our app to generate the specified graph. Since end-users generally value low waiting times, the app needs to generate graphs as *quickly* as possible. Furthermore, the graphs should be *legible* and *useful*.

## General Stakeholders

- **Owner of Tactalyse**: this stakeholder owns the app. In order to keep them satisfied, *deadlines should be followed*, and their *desired features should be included* in the end product. The owner particularly values their time, so the app needs to be *fast* as well.
- **Employees**: they will generate the reports requested by the client. Since they work closely with the clients, they will also be involved in interpreting the document. It can be expected that they will favor data reports with a *clean layout*, so the information displayed will be easy and straightforward to understand.
- **Players**: the players are the main client of Tactalyse. They will make requests for PDFs to be generated. They will be impacted most by both their, and their coaches' interpretation of our data reports. In order to have a better understanding of their current athletic abilities, providing a *comprehensive selection* of such reports would be the most beneficial component.
- **Twitter users**: after the Twitter bot generates and posts a graph, the main audience for the graph will be Twitter users. Twitter users generally scroll through their timeline quite quickly, so the graphs need to be *simple*, and *readable*. Furthermore, Tactalyse's branding needs to be *recognizable*, to increase brand awareness.

# 3. Personas

For this project, we define three personas. Included below is a breakdown of how the employees are supposed to use the final products.

- **The employee**: a person who is granted access to the PDF generator app by Tactalyse. Their job is to create data reports for Tactalyse's clients, and deliver them. The employees have access to the API serving our PDF generator app through a frontend website user interface, which contains a form that has to be filled in detailing the player's required personal data, along with file inputs for providing datasheets. By providing this information and submitting it through the interface, the frontend will call the necessary endpoint and pass the information to the app. There, a report will be generated based on the information contained in the files, and the resulting PDF will be returned to the frontend. The employee will not be able to edit the report, or its general layout. The only possible type of interaction will be the generation of the data report.
- **The Discord user**: a person who interacts with Tactalyse's Discord bot. The main intended use for this bot is on Tactalyse's own Discord server. A user may detail one of three types of graphs they would like to generate with one request: a line plot, a radar chart, or a random graph between those two. For a line plot, the user must detail a football statistic they would like to graph, and the player they would like to create a graph for. Optionally, they may add a second player, to create a comparison graph. These players must exist within the graph generator's local player match data files. For a radar chart, the user must detail a football league, and one or two players within that league. The football league data file must exist in the graph generator's local league data files,

and the specified player(s) must exist within that file. For a random plot, the user may detail which plot they would like to randomize in terms of the players and statistics used. If no graph type is detailed, it is also randomly chosen, making it so that the request does not need to contain additional information at all. Once the request is sent, the Discord bot makes the request to our graph generator with the information provided by the user. After the graph is generated, the output is sent back to the bot, which then displays it on Tactalyse's Discord server.

- **The Twitter bot:** the application that periodically and automatically posts graphs on Tactalyse's Twitter account. The Twitter bot is a special persona, in that it is an application itself, and an intermediary. The user experience consists of the bot automatically making a request for a random graph. The random graph is generated as for the Discord bot, i.e. no parameters are required in the request. The graph generator outputs a graph based on local football data files relevant to the graph that is being generated (line or radar), and returns it to the Twitter bot, which posts it on Twitter. There is no direct user interaction between the Twitter bot and Twitter users. As such, Twitter users are not considered personas.

# 4. Division of Roles

To ensure that someone would take responsibility if a situation arose where the team got stuck and was waiting for someone to take the first step, the team assigned roles to each member. The roles were not meant to be a way of assigning one task to each member and making them solely responsible for that task. Rather, they were meant to be guidelines to ensure that no one would get stuck with work they were not comfortable with, or prepared for.

- **Communications officer:** the member with this role was tasked with setting up meetings with the client, keeping them informed about the work that was being done by the team, and other important communicative matters related to the project. We also decided to make them the main spokesperson in each client meeting. This role was taken by **Mikko**.
- **Chief architect:** the member with this role was tasked with researching the architectural needs of the project, and ensuring that these were adhered to throughout the coding process. We also decided to give them the tasks of having primary responsibility for the design document, and taking notes during client meetings regarding new requirements and design decisions. This role was taken by **Sangrok**.
- **Chief coder:** the member with this role was tasked with setting up a skeleton for the codebase, and ensuring that the identified features and requirements were present in the final product. We decided to extend the role's responsibility to taking care of visual (black box) testing of the code. This role was taken by **Bianca**.
- **Test officer:** the member with this role was tasked with coordinating unit tests, making sure that the requirements were all covered in the traceability matrix, and keeping the tests up to date with new code changes. This role was taken by **Matteo**.

As mentioned above, the roles were guidelines more so than strict boundaries. Each team member ended up working on tasks from multiple roles. We believe everyone contributed to the best of their ability, and we are satisfied with the division of labor.

# 5. Epics

In this section we are going to give a brief overview of the features that each of our services consists of.

## Epic 1: Data Report PDF Generator

Tactalyse's owner wants to provide a fast service that delivers PDF reports of football data. A coach of Tactalyse uploads the required data files to our backend through Tactalyse's frontend app. Our backend extracts data from the files, plots it into different types of graphs based on the extracted data, and generates a single PDF report (Appendix A), including the generated graphs and extracted data. The types of graphs that will be provided in this service are line plots and bar plots. This PDF report is returned to the employee. A diagram for this process can be seen below:

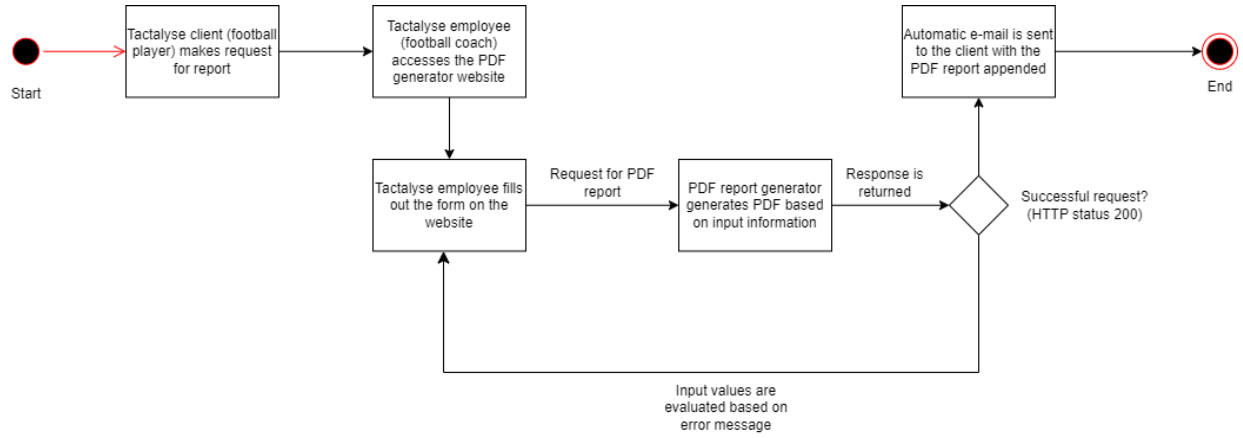



Figure 3: Workflow diagram of epic 1

## Epic 2: Graph Generator

Tactalyse has two bots: a Discord bot, and a Twitter bot. The Twitter bot posts radar charts (Appendix B) for randomly selected football players daily. The Discord bot allows users to generate their own graphs. These graphs are generated by a graph generator API, which contains separate functionality for creating line plots (Appendix C), radar charts, and a randomly chosen one of those two, with the data randomly chosen from local files. Since we have already developed the code for generating line plots for the data report pdf generator service, we will reuse it in this context as well.

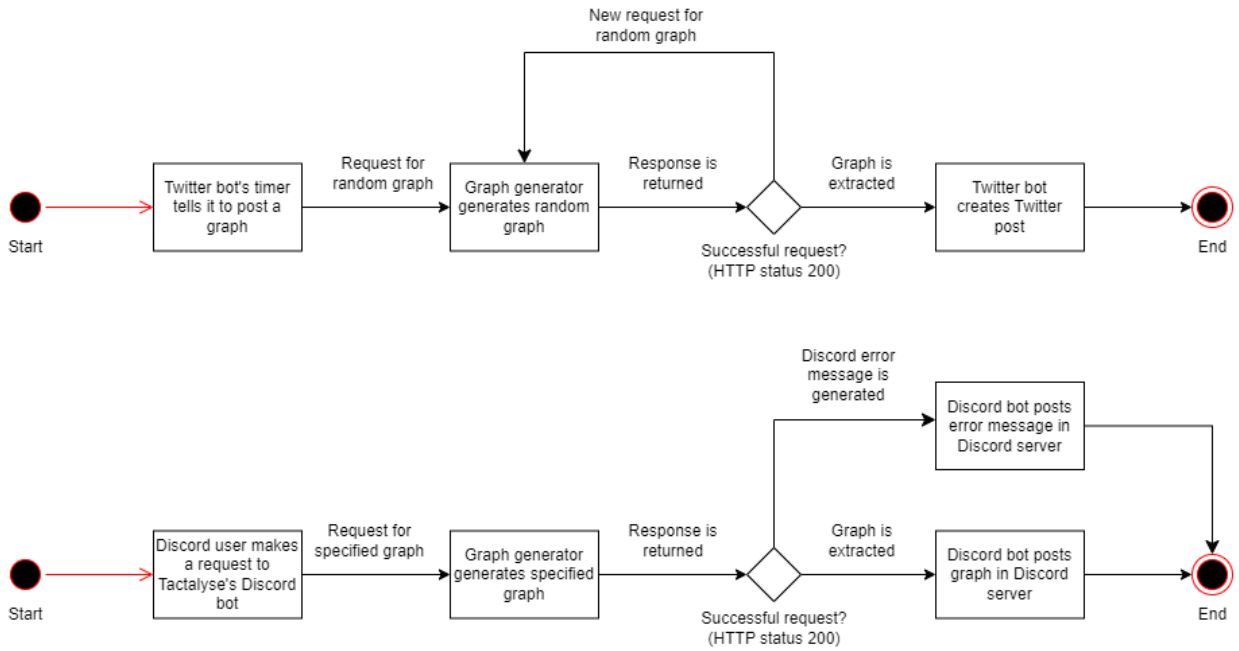



Figure 4: Workflow diagram of epic 2

# 6. Features

This section describes the features that can be extracted from the epics. Features can be seen as a higher-level grouping of user stories. As such, user stories are listed under the relevant feature.

## F1: Data Extraction and Refinement

Data extraction and refinement is a preprocessing step for the visual output of both the PDF generator and graph generator. The purpose of this feature is to extract and process data from Excel files, and store the data in formats usable by the visual modules. Although we were not required to create separate access to data processing functionality for use by the company, the functionality of this feature should be separately accessible and decoupled from the visual modules for future maintenance and expansion purposes.

- Convert Excel file to usable data: **US-M1**

## F2: Visual Data Generation

Graph generation is the core of both our projects. The PDF generator must be able to create data representations of football statistic graphs, and of football statistic PDF reports. The graph generator must be able to create data representations of football statistic graphs. Employees of Tactalyse directly use the features for generating graphs and data reports, and the generated

data reports are to be seen by the end-users (the players, Twitter users and Discord users), so the end product must be user-friendly.

- Create graphs out of the inputted data files: **US-M2, US-M3**
- Display the data reports: **US-M4**
- Generating multiple types of PDF documents with different graphs for different purposes: **US-M2, US-M5**
- Optimization of services: **US-C1**

# 7. User Stories

This section details user stories. The user stories are grouped using the MoSCoW method. This method divides fine-grained functionality into “**M**ust Have”, “**S**hould Have”, “**Co**uld Have”, and “**W**ill Not Have”, corresponding to the importance of the functionality to the final product. The “Must Have” user stories encompass the obligatory requirements that must be fulfilled. Below that are the “Should Have” user stories. These user stories have high priority and should be on the final project, but are not strictly necessary. The “Could Have” user stories are ones that could be done if there is spare time, but they are not necessary for proper functioning of the final product. The “Will Not Have” user stories are ones that are not significant to the final product, and are omitted from our project. By clearly defining what will not be done now, we prevent scope creep in our current project. These user stories may be implemented in the future. In the figure below, we have provided a brief description of the MoSCoW method.


Figure 5: The MoSCoW method description

In our project, each user story has a unique ID, formatted as **US-[M/S/C/W][number]**:

- **US:** user story key identifier
- **[M/S/C/W]:** Must/Should/Could/Will not
- **[number]**: identifier within its own class of user stories as classified by MoSCoW

Each user story is accompanied by associated requirements, formatted as **US-[M/S/C/W][number]-RQ-[F/NF][number]**:

- **US-[M/S/C/W][number]:** as described above
- **RQ:** requirement key identifier
- **[F/NF]:** describes whether the requirement is Functional or Non-Functional
- **[number]**: identifier within its own user story

## Must Have

**US-M1:** As a Tactalyse coach, I would like to be able to extract data from Excel files so that I can turn the data into related graphs or plots.

Requirements:

- **US-M1-RQ-F1** Frontend service must be able to pass Excel files to the PDF generator.
- **US-M1-RQ-F2** Target player's match data extracted from its file must be contained in a dataframe, which will be used as input data for the graph generator and PDF generator.

Acceptance criteria:

- **Given:** the backend has been provided with an Excel file.
- **When:** the backend receives a request with files attached from the frontend.
- **Then:** the data from the Excel file must be returned in dataframe format, with the original data from the file present.

**US-M2:** As a Tactalyse coach, I would like to be able to generate graphs and plots so that I can communicate football match data to clients.

Requirements:

- **US-M2-RQ-F1** Numeric Excel data must be turned into line plots, bar plots and radar charts which contain players' data.
- **US-M2-RQ-F2** The graphs must be in PNG format that can be included in the PDF.
- **US-M2-RQ-F3** The graphs must include only relevant stats for the passed player, based on the dictionary of relevant stats for each player position that Tactalyse provided us with.

Acceptance criteria:

- **Given:** the generator service has extracted players' football match data.
- **When:** the graph generator receives the players' match data.

- **Then:** the graph generator must output graphs and plots reflecting the data contained in the match data.

**US-M3:** As a Tactalyse coach, I would like to have an overview of two players' football match data in the same graphs so that I can compare two football players.

Requirements:
- **US-M3-RQ-F1** All graphs must contain two players' match data depending on if a comparison PDF was requested.
- **US-M3-RQ-NF1** The difference between the data of the two players within the graphs must be clearly visible.

Acceptance criteria:
- **Given:** the generator service has been provided with a second player's data.
- **When:** the graph generator receives two players' data.
- **Then:** two players' data must be represented in the generated plots for all comparable football statistics.

**US-M4:** As a football coach, I would like to have a basic text overview of relevant player information so that I can identify the correct player(s) in the data report.

Requirements:
- **US-M4-RQ-F1** General player information such as age, height, position, etc. contained in the league data file must be displayed at the start of the data report.

Acceptance criteria
- **Given:** the PDF service has been provided with players' data.
- **When:** the PDF service passes the data to the PDF generator.
- **Then:** the generated PDF must include each player's name, country, league, and club in string form in an introductory page.

**US-M5:** As a Tactalyse coach, I would like to be able to generate a PDF out of generated graphs and text so that I can provide the product to clients.

Requirements:
- **US-M5-RQ-F1** After employees send the league and player Excel files along with additional parameters to the PDF generator, the PDF service must output a complete PDF file to the employees.
- **US-M5-RQ-F2** There must be 2 different kinds of PDFs available based on input parameters: non-comparison and comparison reports.

Acceptance criteria:
- **Given:** graphs have been generated based on input data.
- **When:** the PDF generator receives the graphs and data.

- **Then:** the PDF generator must output the final PDF including the generated graphs and data.

**US-M6:** As a Twitter user, I would like to be able to generate a radar chart for two randomly selected players using the Twitter bot.

Requirements:
- **US-M6-RQ-F1** If only the graph type ‘radar’ is passed and no additional information, the data for the graph must be randomly selected for two players.
- **US-M6-RQ-NF1** The graph must include the names of the players that were randomly chosen for the data.

Acceptance criteria:
- **Given:** the graph generator gets no input for which players to use.
- **When:** the graph generator processes data to put in a radar graph.
- **Then:** the generated graph must have a line for two randomly selected players, along with their names.

**US-M7:** As a Discord user, I would like to be able to generate a radar chart for players that I select.

Requirements:
- **US-M7-RQ-F1** When the graph type ‘radar’ and target players are passed, the radar chart for the selected players must be generated.

Acceptance criteria:
- **Given**: the graph generator gets input for the type of the graph and target players.
- **When**: the graph generator processes data to put in a radar graph.
- **Then**: the generated graph must be shown in the discord bot.

**US-M8:** As a Discord user, I would like to be able to generate a line graph.

Requirements:
- **US-M8-RQ-F1:** When the graph type ‘line’ and one target player are passed, the line plot for the selected player must be generated.
- **US-M8-RQ-F2**: When the graph type ‘line’ and two target players are passed, the line plot for the selected players must be generated in the comparison version.

Acceptance criteria:
- **Given**: the graph generator gets input  for the type of the graph and target player/players.
- **When**: the graph generator processes data to put in a line plot.
- **Then**: the generated graph must be shown in the discord bot.

**US-M9:** As a Discord user, I would like to be able to generate a randomly selected graph type, with randomly selected data.

Requirements:

- **US-M9-RQ-F1:** The graph generator must be able to fully randomize data to generate a plot.
- **US-M9-RQ-F2** The random graph must either be a radar chart, or a line plot.
- **US-M9-RQ-F3:** When a line plot is randomly generated, it must contain data for only one player.
- **US-M9-RQ-F4:** When a radar chart is randomly generated, it must be a comparison graph containing two players' data.

Acceptance criteria:

- **Given:** the graph generator gets a request for a random graph.
- **When:** the graph generator starts its data processing.
- **Then:** it must randomize all parameters that were not defined.

## Should Have

As of the delivery of this document, no "Should Have" user stories have been identified.

## Could Have

**US-C1**: As the owner of Tactalyse, I want to have a fast-working PDF generator service so that I can deliver my product in a timely manner.

Requirements:

- **US-C1-RQ-F1** After sending a request to the PDF generator, the PDF should be generated within 20 seconds, as decided by Tactalyse.

Acceptance criteria:

- **Given:** the Tactalyse employee has been provided with an Excel file and additional information that needs to be turned into a PDF format.
- **When:** a PDF is requested from the Tactalyse frontend service.
- **Then:** the PDF should be returned within 20 seconds.

## Will Not Have

**US-W1:** As the owner of Tactalyse, I would like the created graph and PDF generator services to be deployed on the company's servers, so that the system can be used by Tactalyse.

Reason for omission: after discussion with the team's teaching assistant and the course organizer, it was determined that deployment of the apps on the company's servers fell outside of the scope of the course.

**US-W2:** As an employee of Tactalyse, I would like to be able to update the local files used for the graph generator app through a user interface.

Reason for omission: although the client did express that this functionality may be useful for future maintenance of the application, the team decided to purely focus on functionality of the graph generation process to avoid increasing the scope beyond what was possible in the given timeframe. Since the files are updated relatively infrequently, and it can be done manually, the team decided to leave out the implementation of file update functionality. However, the option has been kept open, i.e. it can be implemented in the future.

**US-W3:** As the owner of Tactalyse, I would like my APIs to be CSRF protected, so that only trusted and authorized applications can access and use them.

CSRF (Cross-Site Request Forgery) is a type of attack that tricks the victim into submitting a malicious request. A CSRF protected app would use a CSRF secure random token to prevent them. Reason for omission: this was something that did not come up until the final week of the project during Sonarqube testing. It was not considered earlier due to a lack of theoretical knowledge in the team regarding security. The client had not expressed a desire to have this feature, so technically it was not a requirement. However, it will be included in the future works section of both READMEs, as the team does understand the importance of system security.

# 8. System Use Cases

Use cases are used to formulate one or more grouped requirements in more detail. In this section, use cases of the project are depicted by describing the expected scenario. Each use case includes a context section that tells the basics of the use case, an evaluation of how much the feature will be used, and preconditions and postconditions for the feature. If there is a case of failure for the scenario, the exception scenario will be described.

## Use Case - 1 : Generate Football Data Report

### User Story

US - M1, US - M2, US - M3, US - M4, US - M5

### Context

This use case relates to full functionality of the PDF generator service. After a Tactalyse employee sends a request for a PDF report to the service, this use case starts when the request is received by our API.

### Frequency of occurrence

High. Tactalyse intends to make use of the PDF generator on a weekly basis.

### Precondition

A form has been filled out on the frontend website with a football league data Excel file, at least one football player match data Excel file, and at least one player name. The filled form is processed by the frontend, and the extracted data is sent to the PDF generator service in a request.

### Postcondition

A fully generated football PDF report is returned to the frontend.

### Main Success Scenario

1. The PDF generator service receives the football data Excel files
2. The service processes the file into a usable data format
3. The service uses the processed football data to generate graphs displaying the football data.
4. The service places the generated graphs in a PDF report, and generates it.
5. The generated PDF report is sent back to the frontend.

### Exception Scenario

1. The user inputs the wrong file (not an Excel file)
2. No player name is passed
3. Errors during reading of the input file

## Use Case - 2 : Generate Line Graphs

### User Story

US - M8

### Context

Given the requirements, this is basic functionality. The graph generator service should be able to make line graphs visualizing the player's statistics. These graphs will be displayed by the Discord bot on request.

### Frequency of occurrence

High. Ideally, Tactalyse's Discord bot is used on a frequent basis.

### Precondition

The graph service receives at least one football player name at the line graph endpoint.

### Postcondition

A graph is generated based on the input football player name, and potentially input optional football information.

### Main Success Scenario

1. The graph generator service receives a football player name
2. The service processes the player's local match data file into a usable data format
3. The service uses the processed match data to generate a line graph displaying the match data
4. The generated line graph is sent back to the frontend.

### Exception Scenario

1. The player match data file does not contain an input statistic to graph (e.g. goals stopped for a goalkeeper)
2. The player match data used in graphs is not properly formatted, i.e. may be missing entries
3. The input player name does not have a corresponding local match data file

## Use Case - 3 : Generate Radar Graphs

### User Story

US - M6, US - M7

### Context

Given the requirements, this is basic functionality. The graph generator service should be able to make radar graphs visualizing the player's statistics. These graphs will be displayed by the Discord bot on request.

### Frequency of occurrence

High. Ideally, Tactalyse's Discord bot is used on a frequent basis.

### Precondition

The graph service receives a football league name at the radar graph endpoint, and optionally at least one player name from this football league.

### Postcondition

A graph is generated based on the input football league and player names, and potentially input optional football information.

### Main Success Scenario

1. The graph generator service receives a football league and player name
2. The service processes the league's local football data file into a usable data format
3. The service uses the processed league data to generate a radar graph displaying the league data
4. The generated radar graph is sent back to the frontend.

### Exception Scenario

1. The football league data used in graphs is not properly formatted, i.e. may be missing entries
2. The input football league name does not have a corresponding local league data file
3. The input football player name does not have an entry in the selected league data file

## Use Case - 4 : Generate Random Graphs

### User Story

US - M9

### Context

Given the requirements, this is basic functionality. The graph generator service should be able to generate graphs visualizing randomly chosen players' statistics. These graphs will be posted on Twitter by the Twitter bot and displayed by the Discord bot on request.

### Frequency of occurrence

High. The Twitter bot requests a randomized radar plot daily. Ideally, Tactalyse's Discord bot is used on a frequent basis.

### Precondition

The graph service receives a request at the random graph endpoint, and optionally the type of graph to randomize.

### Postcondition

A graph is generated based on local football data files, optionally a graph of the type that was input.

## Main Success Scenario

1. The graph generator service receives a request for a random graph
2. If a radar graph is chosen, the service processes a random football league's local football data file into a usable data format
3. The service uses the processed league data to generate a radar graph displaying the league data
4. If a line graph is chosen, he service processes a random football player's local match data file into a usable data format
5. The service uses the processed match data to generate a line graph displaying the match data
6. The generated graph is sent back to the frontend

## Exception Scenario

1. The input optional graph type does not have a corresponding implementation in the graph generator service
2. The randomly chosen football data file is formatted poorly or missing entries

# Appendix

## A Excerpt of PDF report

TACTALYSE

Stats Report for T. Cleverley

**POSITION:** Defensive Midfielder
**CLUB:** Watford
**COUNTRY:** England
**LEAGUE:** Eredivisie

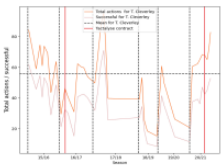


TACTALYSE

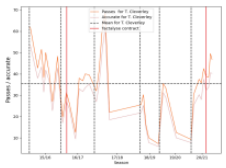
### Line Plots

These plots showcase player statistics over time.

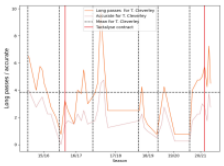
**Stat: Total actions / successful**



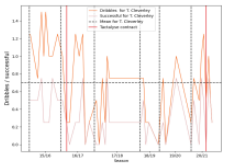
**Stat: Passes / accurate**



**Stat: Long passes / accurate**



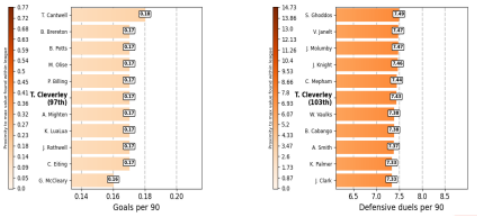
**Stat: Dribbles / successful**



TACTALYSE

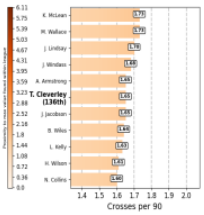
## Bar Plots

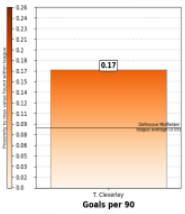
These plots showcase player statistics compared to those of players in the same position within the league.





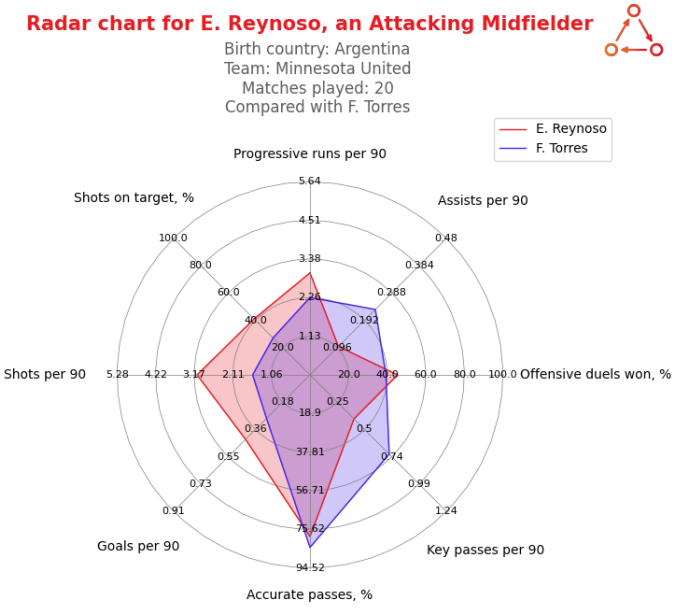




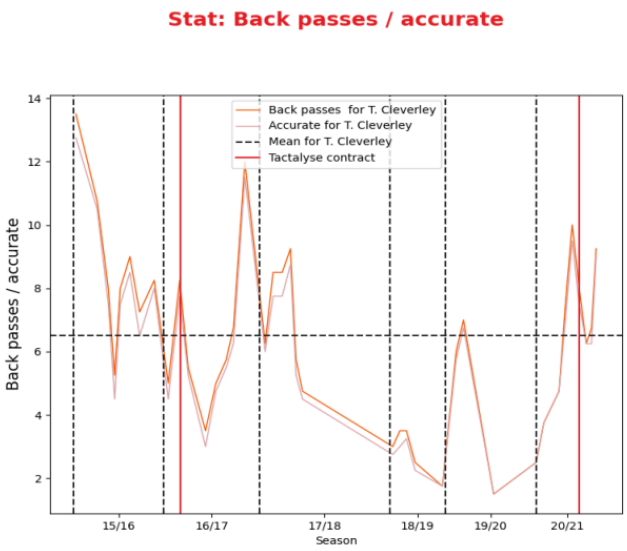






# B Radar Chart Example



# C Line Plot Example




# D Client Meetings

### A.1 17 February Fri 2023 First client meeting with Loran, Ayush

- Attendees: Sangrok, Mikko, Matteo, Bianca
- General client meeting : weekly, Monday, 3:30 - 4:00 pm

1. What is our project? What should our team do for the project?

→ Client has data(Excel file) as input and our program should generate a PDF report.

→ our team will basically work for the template of the PDF file, so make it more readable.

→ our team will not work for website tasks, there is another team to do that project.

→ There are some existing reports and code but needs to be rebuilt from scratch.

→ Data report needs to contain all the information needed.

→ Reports should be as simple as possible, and one template for all reports.

2. Framework freedom

→ It depends on our team, if we think another tool is easier and useful, we can use them. However, we need to explain why we chose another framework to work because the client wants to know for future work.

3. Put Deadlines in the team channel so that the client knows the schedule of our project.
4. Send Microsoft team mails of our team mail address  to communicate with them.

### A.2 20 February Mon 2023 3:30 - 4:00 pm Client meeting with Ayush

- Attendees: Sangrok, Mikko, Matteo, Bianca

Q. Do we have freedom in designing the layout of the PDF

→ Yes, use our imagination

Q. Who are the users?

→ Tactalyse employees create PDF and send it to coaches(so maybe employees and coaches)

Q. What is our task on top of the website

→ nothing yet, but as having consult with professor, we could take another team's task if we want

Q. Ask for clarification on different type of PDF

→ individual, comparison, we can imagine more

Q. What does “making code better” mean?

→ Time reduce for generating PDF, there are some unnecessary functions(code refactor)

### A.3 7 March Tue  2023 3:30 - 4:00 pm Client meeting with Ayush, Loran

- Attendees: Sangrok, Mikko, Matteo

Q. Twitter bot

→ Client wants us to focus on our main requirements(PDF generation) and then after this is done, we can work on extra features.

→ Access will be in shared dropbox

Q. Where is the Excel file(input data) from?

→ Excel file is from the website(https://wyscout.com/ ): coaches upload the Excel file for the PDF.

Request from the client.

→ There are two websites for the project, and the clients want them in one.

a. but this is not gonna be our work, this will be another team's work.

### A.4 13 March Mon 2023 3:00 - 3:30 pm Client meeting with Ayush, Loran and Tactalyse-1

- Attendees: Sangrok, Mikko, Matteo, Bianca

Only notes related to our project.

1. Twitter bot
    a. Graph generation for Twitter bot will be our responsibility
2. Among the plots in the example PDF, we should focus most on the line plot
3. Speeding up the PDF generation is also an important requirement, current source code generates simple PDF(3-4 pages) in 3-5 minutes which is too slow.

### A.5 21 March Mon 2023 3:30 - 4:00 pm Client meeting with Ayush, Loran

- Attendees: Sangrok, Mikko, Matteo, Bianca

Q. Do we need a database for saving Excel files from the past?

→ Since the column and features of the data will be the same, therefore it will be a good option to implement a database for the future. However, it depends on us.

Q. During the meeting with our TA, we discussed having one API for two endpoints since the PDF generation function contains graph generation. Is it fine to have one API for two of those endpoints?

→ It is fine to have one API for two endpoints. We explained our choice for this and the client agreed with our decision.

### A.6 27 March Mon 2023 3:30 - 4:00 pm Client meeting with Ayush, Loran

- Attendees: Sangrok, Matteo, Bianca

Q. There are some players who have more than one position in the Excel files. In this case, based on which position should the data report for that player be made?

→ Almost all players have their main position and sometimes play for other positions. We can choose the leftmost position in the Excel files as the main position and make a data report based on this position.

Q. There are statistics in the Excel files, do we need to follow the format of the Excel file in the data report?

→ yes.

Q. Can we add some more basic information for the target player which are not in the example data report?

→ of course yes.

### A.7 3 April Mon 2023 4:00 - 4:30 pm Client meeting with Ayush, Loran and Tactalyse -1

- Attendees: Sangrok, Matteo, Bianca, Mikko

- make it visually pleasing
- next meeting: make project plan
- no need to concern Agent this will not be used in further
- Front page(First player basic info page) can be in better layout

### A.8 27 April Mon 2023 12:00 - 1:00 pm Client meeting with Ayush

- Attendees: Sangrok, Matteo, Bianca, Mikko

Client meeting with Ayushi

Our team showed prototypes of our planning graphs

First prototype(graphs with image)

→ Feedback from Ayushi: no images on the left, need to figure out if it is able to be implemented in code.

Second prototype(Red cards/Yellow cards)

→ Yellow/Red cards stats graphs are not that important.

Q. Do we need personal information about comparison players in the first page?

→ if it is possible yes

Tips from client:

Matplotlib is more customizable, Seaborn has an easy function to implement simple graphs.

Top 5 stats graphs can be implemented in a scatter plot.

### A.9 15 May Mon 2023 4:00 - 5:00 pm Client meeting with Ayush, Loran and Tactalyse-1

- Attendees: Sangrok, Matteo, Bianca, Mikko

Get feedback our works from the client

front page:

- space things out over the page

- smaller title (stats report for x)

- same size image for comparison

line:

- minimum length for season

- do not stretch at the end

- more contrasting colors

bars:

- somehow explain the concept of darker color = closer to top

-> cbar

scatter:

- hue

- maybe not even required

- see if it is possible to easily do it with bar instead create sample PDF for frontend

### A.10 22 May Mon 2023 4:00 - 5:00 pm Client meeting with Ayush, Loran and Tactalyse-1

- Attendees: Sangrok, Matteo, Bianca, Mikko

Feedback from the client

line plot

- change colors
- season ticks (1st of july)
- discord bot: white background, Tactalyse logo top right

bar plot

- proximity bar on left
- remove y-axis ticks
- make scale of chart same as proximity bar
- remove title at bottom
- remove average bar

summary bar plot

- change colors to something similar (dark purple, purple, light purple for example)

ranking bar plot

- move to top of bar section
- make vertical

-> look into rotating player labels 45/90 degrees

- same changes as bar plot for layout

scatter plot

- we do not need this plot for our project, but the clients want to keep this in the code for the future use

**A.11 30 May Mon 2023 4:00 - 5:00 pm Client meeting with Ayush and Tactalyse-1**

line plots

- set season tick in middle of season

- change line colors

bar plots

- split last bar graph into offensive/defensive
- change degree sign in player ranking to something else (maybe uppercase th)
- ask Loran again which color scheme he prefers

Tactalyse-1 has announced to the client that they deployed our applications (against our will), potentially risking our offer for monetary compensation, where deployment was our only additional offer. We're discussing this with our TA.

# E Changelog

- 27 February 2023 **Sangrok** : Added first, second client meeting.
- 28 February 2023 **Sangrok, Mikko**: First draft of Epics, Stakeholders, Features, User stories based on MoSCoW with requirements (draft needs more opinions from other team members).
- 3 March 2023 **Sangrok, Mikko, Matteo**: Enhance the overall parts in requirements doc(Epics, Features, User Stories)
- 4 March 2023 **Bianca**: Only persona identified - employee
- 7 March 2023:
    - **Sangrok, Mikko, Matteo**: Add acceptance criteria for all user stories except US-S2 (US-S2 is now deleted because it is too obvious and basic)**.**
    - **Sangrok**: Add third client meeting.
- 11 March 2023 **Bianca, Sangrok, Mikko, Matteo**: Listed framework choices and added notes about architecture.
- 13 March 2023 **Sangrok**: Add 4th client meeting.
- 22 March 2023 **Sangrok**: Add 5th client meeting.
- 24 March 2023 **Sangrok**: Add Use cases section.
- 28 March 2023 **Sangrok**: Add 6th client meeting, table of content.
- 29 March 2023 **Sangrok:** Modify Introduction part based on the feedback.
- 30 March 2023 **Sangrok**: Add Glossary table, fixed most parts of the document based on the feedback. (User stories, Use cases)
- 30 March 2023 **Mikko:** Went over/updated Sangrok's changes, added title page.
- 3 April 2023:
    - **Sangrok**: Add 7th client meeting.
    - **Mikko:** Final fixes.
- 27 April 2023: **Sangrok**: Add 8th client meeting
- 1 May 2023: **Sangrok**: Add a simple process model diagram in the introduction.
- 2 May 2023:**Sangrok**: Add a diagram in the Epics section.
- 3 May 2023:**Sangrok**: Modify Epics section based on the Block 1 feedback, add description of MoSCoW method, add a notation on how to read our requirement marks, modify User story based on the Block 1 feedback.
- 18 May 2023:**Sangrok:** Add 9th client meeting.
- 23 May 2023: **Sangrok**: Add 10th client meeting, add captions for all figures.
- 31 May 2023: **Sangrok, Bianca, Matteo, Mikko**: Add new requirements in block 2, fix overall docs based on the feedback.
- 2 June 2023: **Mikko**: Further reworked the document according to block 1 feedback, added graph API to non-requirements sections, added 11th client meeting.
- 4 June 2023: **Mikko**: Added roles section.
- 6 June 2023: **Mikko**: Added user stories US-W2, US-W3.
- 8 June 2023: **Mikko**: Rewrote Use Case section to align with sequence diagrams in design document.

- 9 June 2023:
    - **Bianca**: Revised the document based on Charles' feedback
    - **Bianca, Sangrok, Mikko, Matteo**: Finished the document