Software Requirements Specification

For

APP #1 (Group 1-11)

Prepared by Group 11

Alvira Konovalov (40074264)
Hamzah Muhammad (40156621)
William Chittavong (40048632)
Dmytro Chychkov (40034351)
Samuel Chuang (40133237)

February 28, 2022

Table of Contents

Table of Contents **2**

Revision History **4**

Introduction **5**

- Purpose 5
- Document Conventions 5
- Intended Audience and Reading Suggestions 5
- Project Scope 5
  - The deliverable products are: 6
- References 6

Overall Description **6**

- Product Perspective 6
- Product Features 7
- Operating Environment 8
- Design and Implementation Constraints 9
- User documentation 9
- Assumptions and Dependencies 9
- Budget 10

System Features **11**

- Project Features 11
  - Project Schedule Plan 11
  - System Domain Model 12
  - Use Case Diagram 13
  - Class Model 13
- Use Cases 14
  - Use Case 1 - Activity Recap 14
    - Fulled Dressed Scenario 14
    - System Sequence Diagram 16
    - Sequence Diagram 17
    - Collaboration Diagram 17
  - Use Case 2 - Macros Chart 18
    - Fulled Dressed Scenario 18
    - System Sequence Diagram 19
    - Sequence Diagram 20
    - Collaboration Diagram 21
  - Use Case 3 - Distance Travelled 22

Fully Dressed Scenario 22
System Sequence Diagram 23
Sequence Diagram 24
Collaboration Diagram 25
Use Case 4 - Ride Intensity 26
Fully Dressed Scenario 26
System Sequence Diagram 27
Sequence Diagram 28
Collaboration Diagram 29
Use Case 5 - Moving Average Chart 30
Fully Dressed Scenario 30
System Sequence Diagram 32
Sequence Diagram 33
Collaboration Diagram 35

External Interface Requirements 36
User Interfaces 36
Hardware Interfaces 36
Software Interfaces 36
Communication Interfaces 36

Other Nonfunctional Requirements 36
Performance Requirements 36
Safety Requirements 36
Security Requirements 37
Software Quality Attributes 37

Other Requirements 37

Appendix A: COCOMO Adjustment factor (c) breakdown 38

Appendix B: Issues List 39

Revision History

Name	Date	Reason For Change	Version
Alvira Konovalov	Feb 28	Use Case 1 Domain Model Use Case Diagram Class Model Create and fill-in SRS	1.1
Hamzah Muhammad	Feb 28	Use Case 2 1.2 Document Conventions 1.4 Project Scope 3.1.4 Class Model	1.1
Samutl Chuang	Feb 28	Use Case 3 5.2 Safety requirement 5.4 Software Quality Attributes	1.1
Dmytro Chychkov	Feb 28	1.3 Intended Audience and Reading Suggestions 1.5 References 2.3 Operating Environment 2.5 User Documentation 2.6 Assumptions and Dependencies 2.7 Budget 3.0 System Features 3.1.2 System Domain Model 3.1.3 Use Case Diagram 3.2.4 Use Case 4 Appendix A	1.1
William Chittavong	Feb 28	Use case 5 2.4 Design and Implementation Constraints 3.1.2 System Domain Model 4 External Interface Requirements 5.3 Security Requirements	1.1

1. Introduction

1.1. Purpose

This software product is identified as **"APP #1 (Group 1-11)"** Version 1.0. This SRS will describe the part of the system that is concerned with UML statistics visualization. The document will describe new features related to the visualization of the statistics relating to the class diagram and the sequence diagram. The system and documentation are to be designed in terms of usability and user-friendliness.

1.2. Document Conventions

IEEE standard Times New Roman size 12 fonts will be used for this document. Important terms will be conveyed using bold/underlining. For each requirement statement there will be its priority.

1.3. Intended Audience and Reading Suggestions

This document is intended for all audience types, from the technical expert to the common everyday user. It is organized into 6 sections: 1) Introduction 2) Overall Description 3) System Features 4) External Interface Requirements 5) Other Nonfunctional Requirements 6) Other Requirements.
Developers may want to read sections 1.1 to 1.5, 2.1 to 2.5, 3, 3.1, 3.2, 4, 5 and 6
Project Managers may want to read sections 1.1 to 1.4, 2.1, 2.4, 2.5, 2.7, 3, 4, 5 and 6
Marketing staff will be interested in sections 1.1 to 1.4, 2.1 to 2.3, 2.7, 4.1 to 4.4, 5.2, 5.3 and 5.4
Everyday Users will want to read sections 1.1 to 1.4, 2.2, 2.3, 2.5, 4.1, 5.2 and 5.3.
Testers will want to read sections 1.1 to 1.4, 2.2, 2.3, 2.5, 4.1 to 4.4, 5.2 and 5.3.
Documentation writers may be interested in sections 1.1 to 1.4, 2.1, 2.4, 2.5, 2.7, 3, 4, 5 and 6

1.4. Project Scope

<u>App #1</u> is an initial release of an **open-source fitness tracker**. It is currently targeted towards cycling but is to be expanded to other physical activities in the future. The client, a cycling enthusiast, envisions an **Android Graphical User Interface** with a software that collects and analyzes data from the user's workout. In addition, a **Web Interface** to display enhanced statistics created from the collected data. Moreover, the Web Interface will allow the user to view these statistics using various constraints (activity type, time

frame, etc.). A login system for authentication will be implemented to allow users access their statistical data. The client expressed that applications that are currently in the market are not customizable enough to support all the desired functionalities. Desired features include auto-pause, data exchange, activity analysis, and device connectivity. Eleven groups are responsible for different aspects and features of the fitness tracker. The client would like to see a draft of the product by the end of the Winter semester.

The deliverable products are:

APP #1 will be implemented using one or more of the following programming languages: Java, Kotlin, JavaScript and Python.

1.5. References

[1] javaTpoint, “Software engineering: Cocomo Model - javatpoint,”

www.javatpoint.com.

[Online]. Available: https://www.javatpoint.com/cocomo-model. [Accessed:

01-Mar-2022].

2. Overall Description

2.1. Product Perspective

The product is being developed for our client Dr. Rilling, which would like to introduce a new **open-source fitness tracker application**. The features of the tracker will include an auto-stop option, data exchange, activity analysis, device connectivity, and more. A **Web Interface** will be implemented to display enhanced statistics such as maps and charts derived from the user's activity history.

2.2. Product Features

Detailed Functionalities

Group		**Description**	**Requirements**	**Comments**
1	12	GUI Main Screen (Android)	Start/Stop/Pause, Workout selection, Workout settings countdown start (setting) , split interval, voice prompt, type of summary information provided at the end of the workout	Currently only cycling but more types should be easy to add. Separate workout description file configuration file (XML) Provide unique workout – id which will be used for the workout.
2	13	User Profile and General settings	User info – age (DoB), Gender, weight, Name. General setting units (decimal vs. pounds/miles), language, GPS accuracy (less frequent GPS data polling), voice feedback options.	Separate user profile file (XML) Separate configuration file Separate general settings file (XML)
3	14	GPS data	Polling of GPS data, storing it in GPX format. Provide functions to save GPX data and to parse the GPX data.	Should support different GPX polling times (e.g., .5 second, 1(default), 2 seconds, 5 seconds (depending on battery saver mode see group #2 and sports type.

4	15	Other data	Weather conditions.	See if you can find free available weather data for the current location/day (e.g., temp, wind speed/direction) Add to the GPX or any other workout specific file ? Discuss with Group 3/14
5	16	Data Analytics	Altitude (corrected) Topspeed, Distance, Average speed, Speed corrected by auto pause (if enabled), moving average (e.g., 10 splits) Split: speed/average/distance/top speed,	Mainly service functions based on gpx information. Altitude correction – there should not be unrealistic fluctuation between data points Store split info in XML file? Speed (with and without auto pause)
6	17	Voice feedback	Voice feedback/text to speech (depending on options enable in the general settings file – group 2/14	At each split and at the end of the workout.
7	18	Map	At the end of the workout. Show workout route on a map.	Highlight splits
8	19	Summary info	Workout details for all the splits. Workout summary	
9	20	Statistics	All workouts – overall statistics across all workouts + being able to filter one category	Overall statistics (across all types of workouts) e.g., total hours, total kms filter a specific workout from a list of available workout types, filter date range range (e.g., year).
10	21	Maps	User should be able to browse through all workouts and select specific to view	
11	22	Charts	visualize statistics from 9/20	Bar charts, line charts…..

2.3. Operating Environment

Android App:
Supported on all Android devices with Android version 6.0 or above and the screen diagonal over 4 inches.

Web application is supported in the following browsers:

- Google chrome
- Mozilla Firefox
- Apple Safari
- Microsoft Edge
- Opera
- Apple Safari for iOS
- Google Chrome for Android

OS:

- **Windows:** Windows 8/8.1/10
- **Linux:** Linux Debian 9 or higher
- **iOS:** 12.0 or higher
- **Android:** 6.0 or higher

2.4. Design and Implementation Constraints

HTML, JavaScript, and CSS will be used to develop the Web Interface. Java will be used to implement the chart generator. The user will interact with the web interface by logging in with their user id. The user will be able to select the type of the chart they would like displayed. Depending on the type of chart, they will be required to select different inputs such as the period of days from which to display the data. Visualization will include charts for moving averages, distance travelled, elevation/riding intensity, macros chart, and activity recap. The statistics generator will provide the values for the charts which they have calculated by using the data from the database containing all the cyclists data. Errors will be displayed if the process fails to access the statistics generator or database.

2.5. User documentation

User documentation will be released along with the first complete prototype at the end of 2nd iteration.

2.6. Assumptions and Dependencies

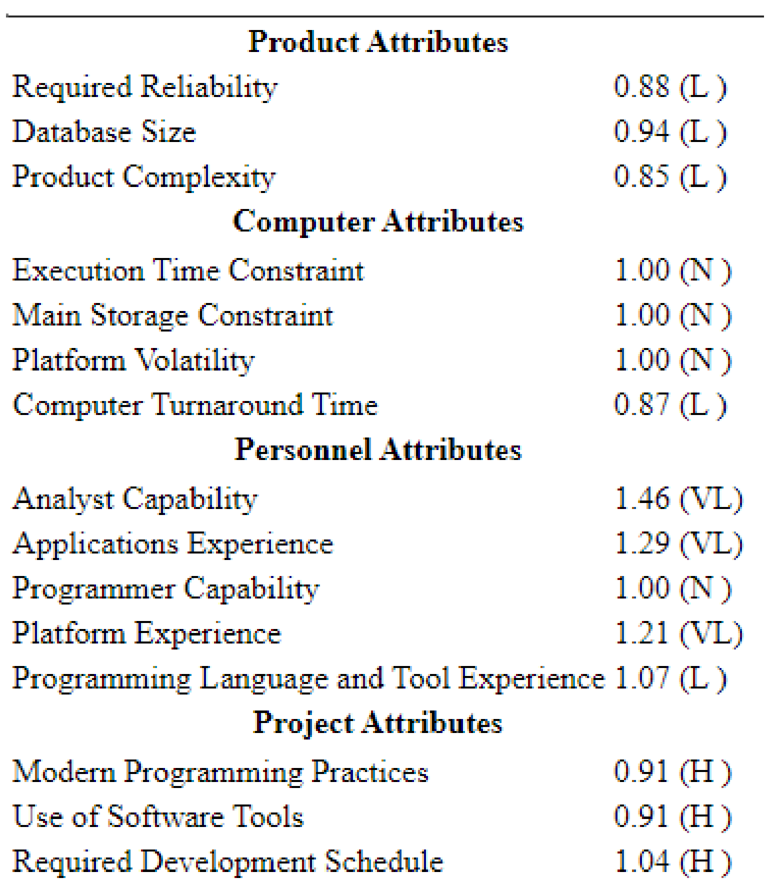
The external libraries (swing/awt) in use will remain supported for the foreseeable future. Access to all the modules and interfaces produced by the other teams.

Project requirements will not deviate too much from the initial requirements.
The scope of the project will not change.
Older Android versions will remain supported by the developer.

2.7. Budget

Based on the COCOMO model:

Adjustment factor summary table:



Product Attributes	
Required Reliability	0.88 (L)
Database Size	0.94 (L)
Product Complexity	0.85 (L)
Computer Attributes	
Execution Time Constraint	1.00 (N)
Main Storage Constraint	1.00 (N)
Platform Volatility	1.00 (N)
Computer Turnaround Time	0.87 (L)
Personnel Attributes	
Analyst Capability	1.46 (VL)
Applications Experience	1.29 (VL)
Programmer Capability	1.00 (N)
Platform Experience	1.21 (VL)
Programming Language and Tool Experience	1.07 (L)
Project Attributes	
Modern Programming Practices	0.91 (H)
Use of Software Tools	0.91 (H)
Required Development Schedule	1.04 (H)

Table 1: Detailed budget breakdown

**See Appendix A for the detailed breakdown*

a = 2.4 (organic project)
b = 1.05 (organic project)
KLOC = 1.2
$c = c_1 * c_2 * ... * c_{15} = 2.5$
Team size = 5
salary = 4200/month

Effort: a*KLOC^b*c = 2.4*1.2$^{1.05}$ *2.5 = 7.2659 person/months
Dev. time: 7.2659/5 = 1.45 months to develop
Dev. cost: 4200*5*1.45 = 29000 CAD

3. System Features

Cycling app tracks user’s movements by polling their device’s GPS data for each workout session. The workouts are saved on the user's device in GPX format and can be accessed at any time from the app interface.

Once a workout is complete, the Statistics Generator will analyze the data associated with the user's workout session and generate forensic data such as calories burned, altitude changes, distance traveled, average speed and moving average.

This statistics data can be visualised using the Chart Generator layer of the system that changes its content depending on the user’s chosen scenario(calories burned, distance traveled etc.) and the specified time frame (past workout/day/week/month/year or a specific time frame)

3.1. Project Features

3.1.1. Project Schedule Plan

Iteration #1:

We are planning to implement Use Case #1, Use Case #2, Use Case #3 in this iteration

Iteration #2:

Use Case #4 and Use Case #5 will be implemented and integration attempts with the statistics group 10 will be made.

3.1.2. System Domain Model

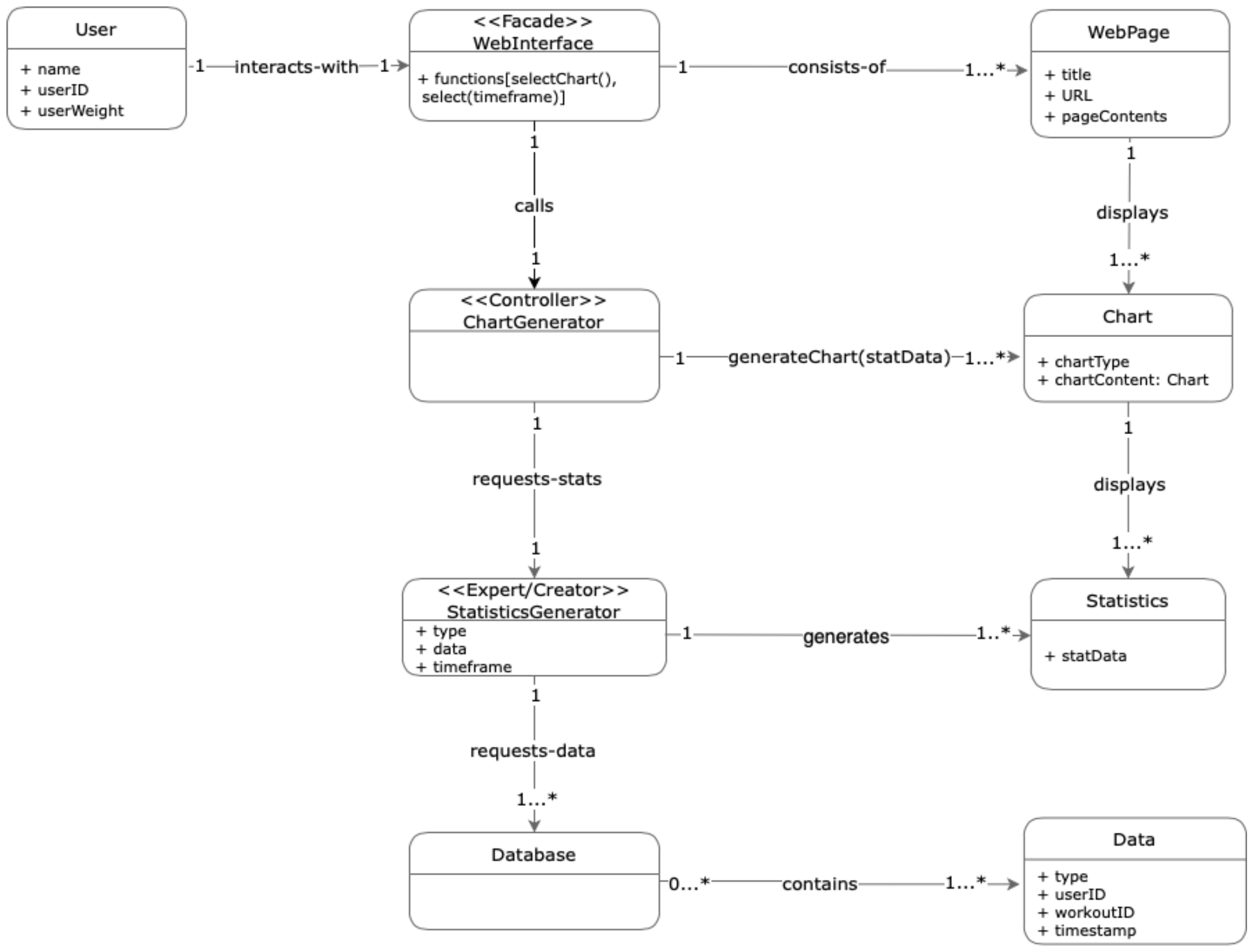


Figure 1: System Domain Model

3.1.3. Use Case Diagram

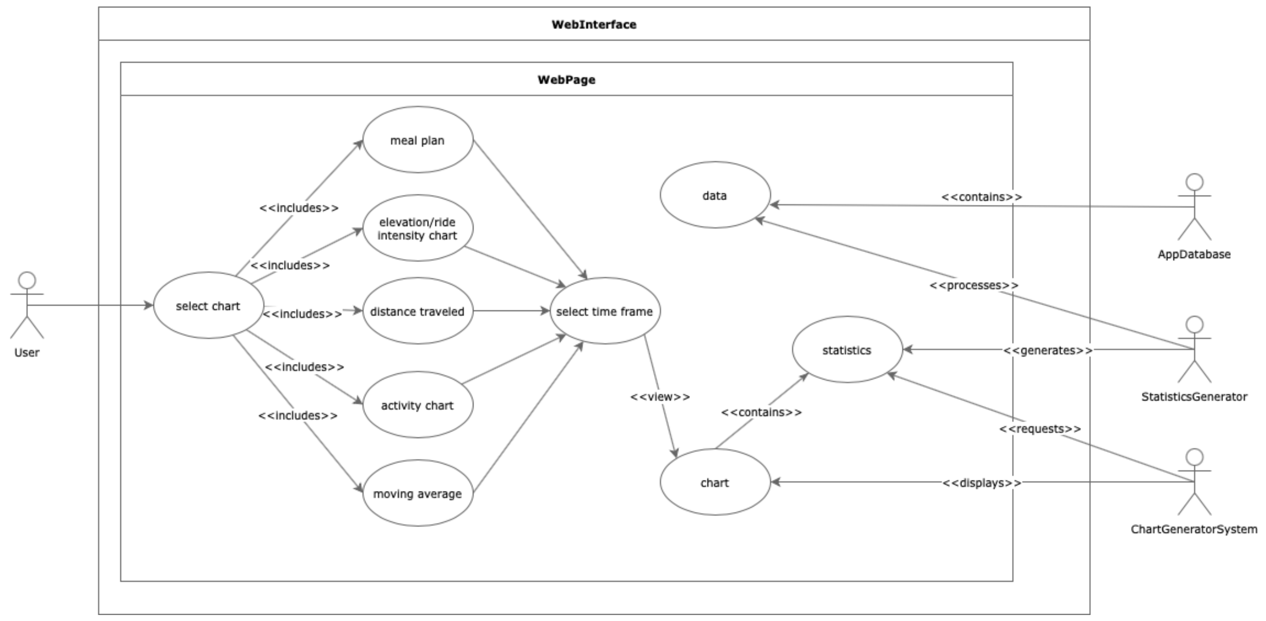


Figure 2: Use Case Diagram

3.1.4. Class Model

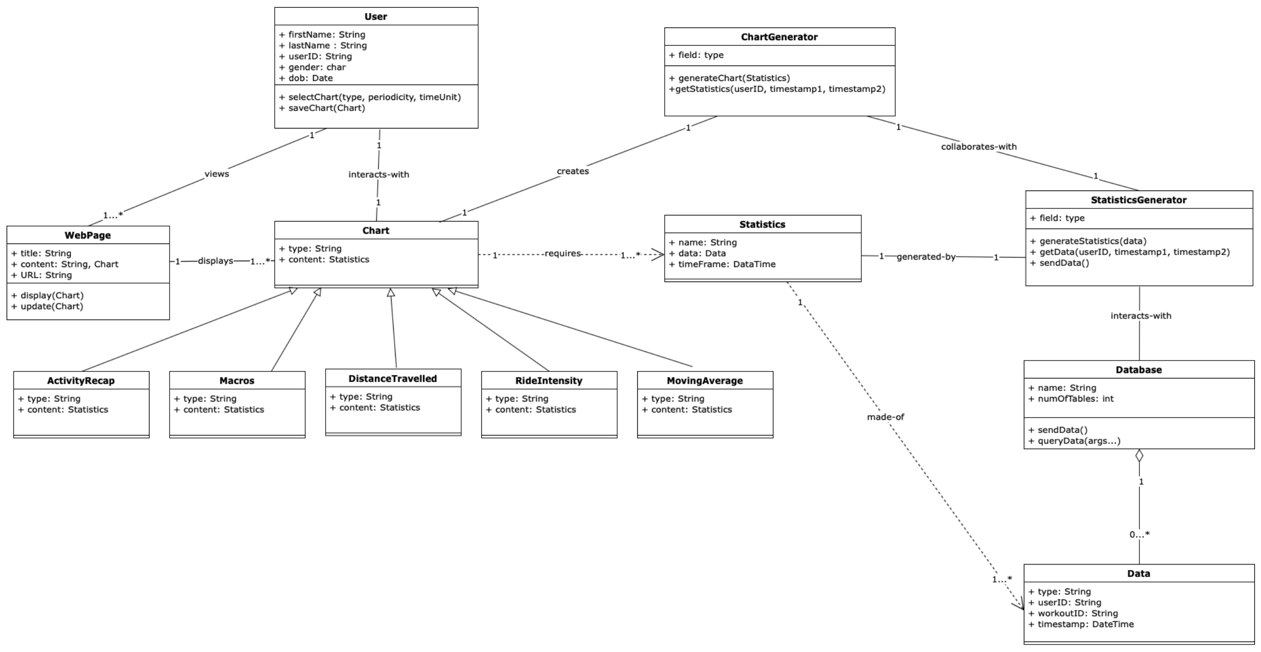


Figure 3: Class Model

3.2. Use Cases

3.2.1. Use Case 1 - Activity Recap

Use Case 1 describes the instance where the user desires to see a recap of its activities on the App. The activity recap chart summarizes all the physical activities accomplished throughout the year, month, week or day using the App.

3.2.1.1. Fulled Dressed Scenario

Use Case Name	Activity Recap Chart	
Summary	User selects activity recap chart and chart generator system displays the chart using the user's statistics	
Priority	2	
Pre-conditions	User goes on App's Web Interface	
Post-conditions	Chart is displayed with user's activity recap	
Primary Actor	App user	
Secondary Actor	Stats. Generator (Stats Group 10)	
Trigger	User opens Web Interface	
Main Scenario	**Step**	**Action**
	1	User selects the activity recap chart from the chart selection
	2	User selects type of activity recap (daily/weekly/monthly/yearly and hours/minutes/seconds)
	3	Chart generator system gets stats from Stats. generator
	4	Chart generator system generates
	5	Chart generator displays activity recap chart
	6	User views the chart
	7	User saves the chart
	8	User exits Web Interface
Extensions	**Step**	**Branching Action**

	3a	System is unable to access statistics – error message
	4a	System is unable to generate recap - error message
	5a	System is unable to display chart properly – error message
	7a	Unsuccessful save action – error message

Table 2: System Sequence Diagram for Use Case 1

3.2.1.2. System Sequence Diagram

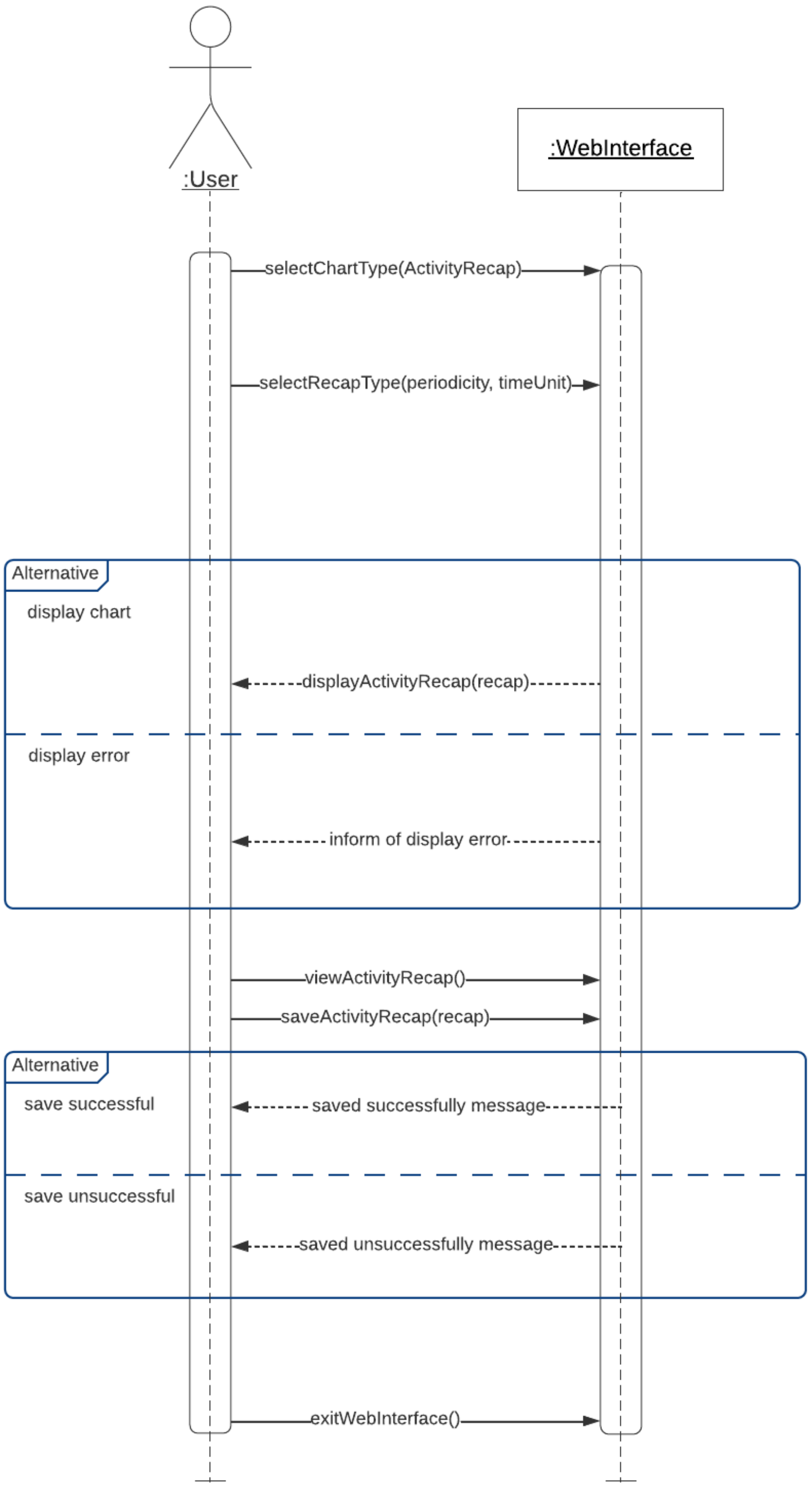
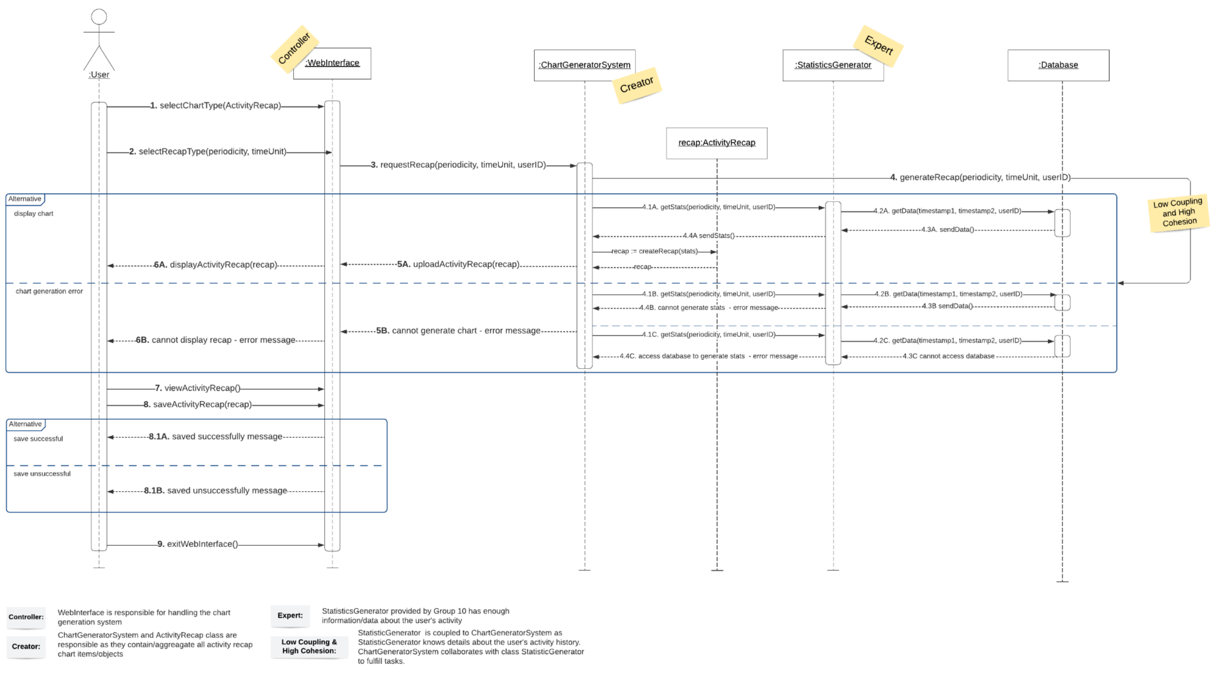


Figure 4: System Sequence Diagram for Use Case 1

3.2.1.3. Sequence Diagram


:User
Controller
:WebInterface
:ChartGeneratorSystem
Creator
:StatisticsGenerator
Expert
:Database
recap:ActivityRecap
Low Coupling and High Cohesion
1. selectChartType(ActivityRecap)
2. selectRecapType(periodicity, timeUnit)
3. requestRecap(periodicity, timeUnit, userID)
4. generateRecap(periodicity, timeUnit, userID)
Alternative
display chart
4.1A. getStats(periodicity, timeUnit, userID)
4.2A. getData(timestamp1, timestamp2, userID)
4.3A. sendData()
4.4A sendStats()
recap := createRecap(stats)
recap
5A. uploadActivityRecap(recap)
6A. displayActivityRecap(recap)
chart generation error
4.1B. getStats(periodicity, timeUnit, userID)
4.2B. getData(timestamp1, timestamp2, userID)
4.3B sendData()
4.4B. cannot generate stats - error message
5B. cannot generate chart - error message
6B. cannot display recap - error message
4.1C. getStats(periodicity, timeUnit, userID)
4.2C. getData(timestamp1, timestamp2, userID)
4.3C cannot access database
4.4C. access database to generate stats - error message
7. viewActivityRecap()
8. saveActivityRecap(recap)
Alternative
save successful
8.1A. saved successfully message
save unsuccessful
8.1B. saved unsuccessfully message
9. exitWebInterface()

Controller: WebInterface is responsible for handling the chart generation system

Creator: ChartGeneratorSystem and ActivityRecap class are responsible as they contain/aggreagate all activity recap chart items/objects

Expert: StatisticsGenerator provided by Group 10 has enough information/data about the user's activity

Low Coupling & High Cohesion: StatisticGenerator is coupled to ChartGeneratorSystem as StatisticGenerator knows details about the user's activity history. ChartGeneratorSystem collaborates with class StatisticGenerator to fulfill tasks.

Figure 5: Sequence Diagram for Use Case 1

3.2.1.4. Collaboration Diagram

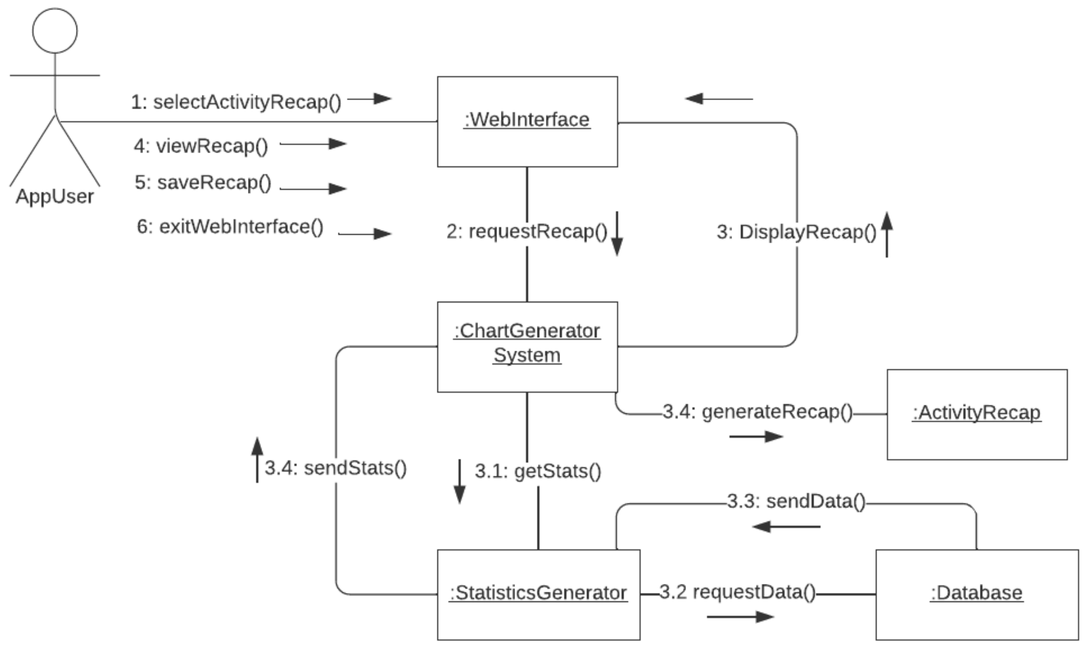


Figure 6: Collaboration Diagram for Use Case 1

3.2.2. Use Case 2 - Macros Chart

3.2.2.1. Fulled Dressed Scenario

Use Case Name	*Macros Chart*	
Summary	*User generates pie chart and views statistics visualization of their macros on the web interface*	
Priority	4	
Preconditions	*User has installed the app and has login. User has entered their meals planned/consumed for the day on the app. User has closed the app.*	
Postconditions	*Pie chart generated showing statistics on the web interface*	
Primary Actor(s)	*User*	
Secondary Actor(s)	*Web Interface, Statistics Database, Statistics Generator, Chart Generator System*	
Trigger	*User successfully logs into the website*	
Main Scenario	**Step**	**Action**
	1	*User selects to view charts*
	2	*App displays different categories of charts (pace, elevation, steps, macros, activity, etc.)*
	3	*User selects to view the macros of their meal plan for a certain day*
	4	*Statistics Generator uses meals entered into the app from the Statistics Database to generate stats regarding the macros consumed in the day: Carbohydrates (g, %), Fat (g, %), Protein (g, %)*
	5	*Chart Generator System uses the statistics calculated to generate a pie chart displaying the macros consumed in the day.*

	6	*Web Interface fetches the pie chart generated by the Graphics Engine and displays it*
	7	*User closes website*
Extensions	**Step**	**Branching Action**
	1a	*Web Interface displays error message and is unable to direct to charts section*
	3a	*Web Interface displays error message and is unable to display macro charts*

Table 3: Fully Dressed Scenario for Use Case 2

3.2.2.2. System Sequence Diagram

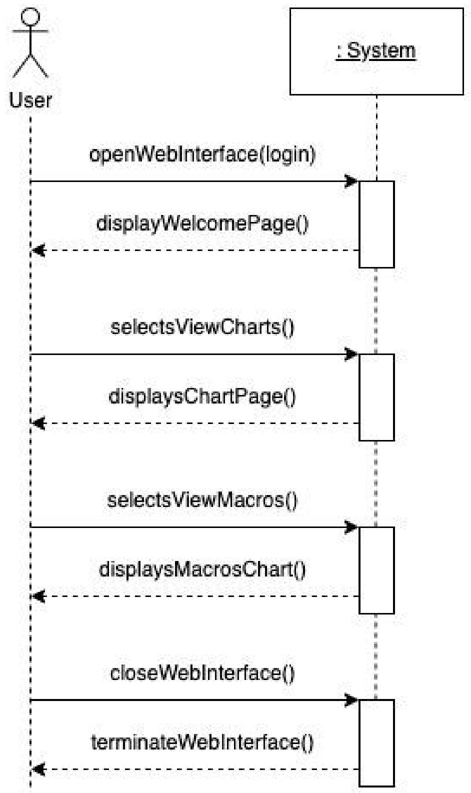


Figure 7: System Sequence Diagram for Use Case 2

3.2.2.3. Sequence Diagram

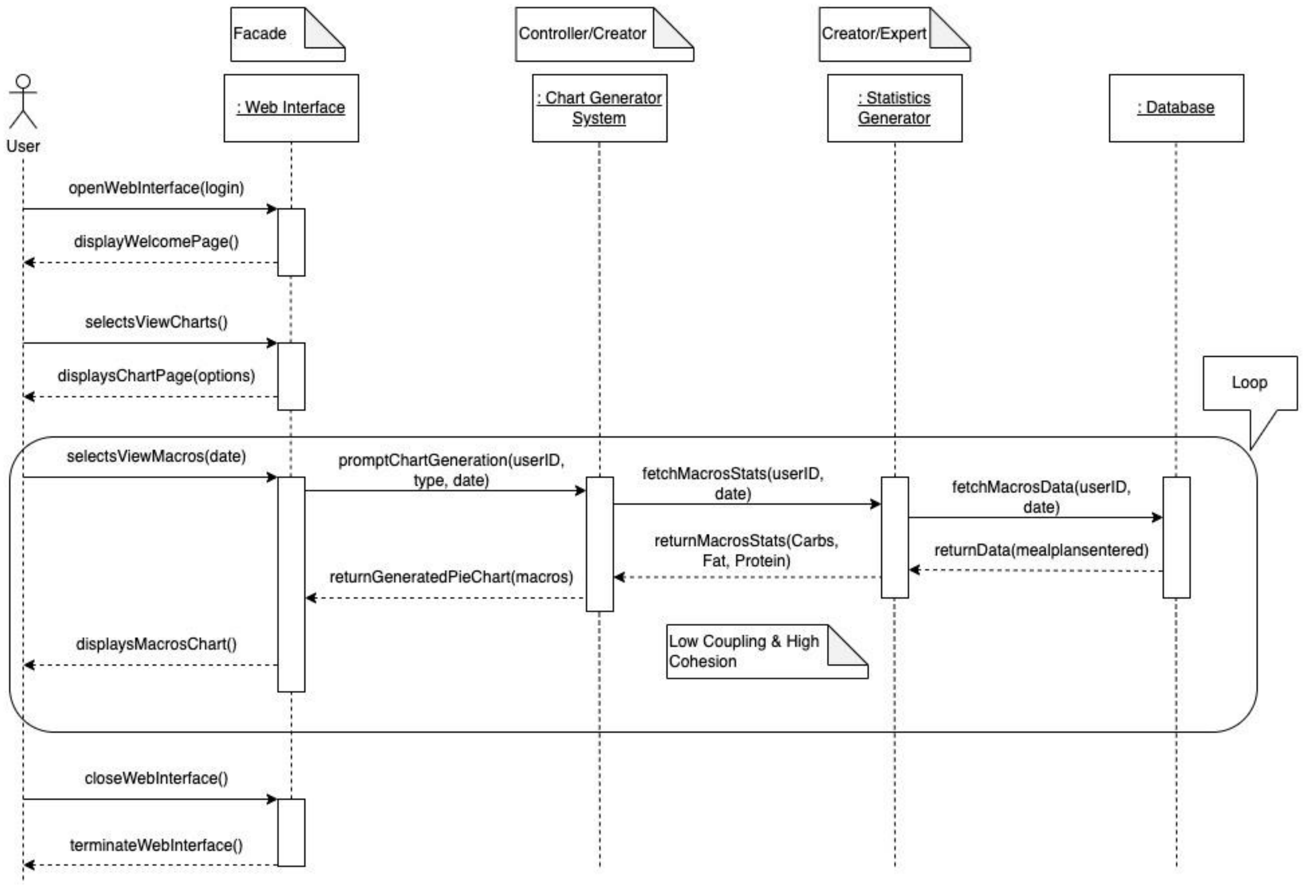


Figure 8: Sequence Diagram for Use Case 2

3.2.2.4. Collaboration Diagram

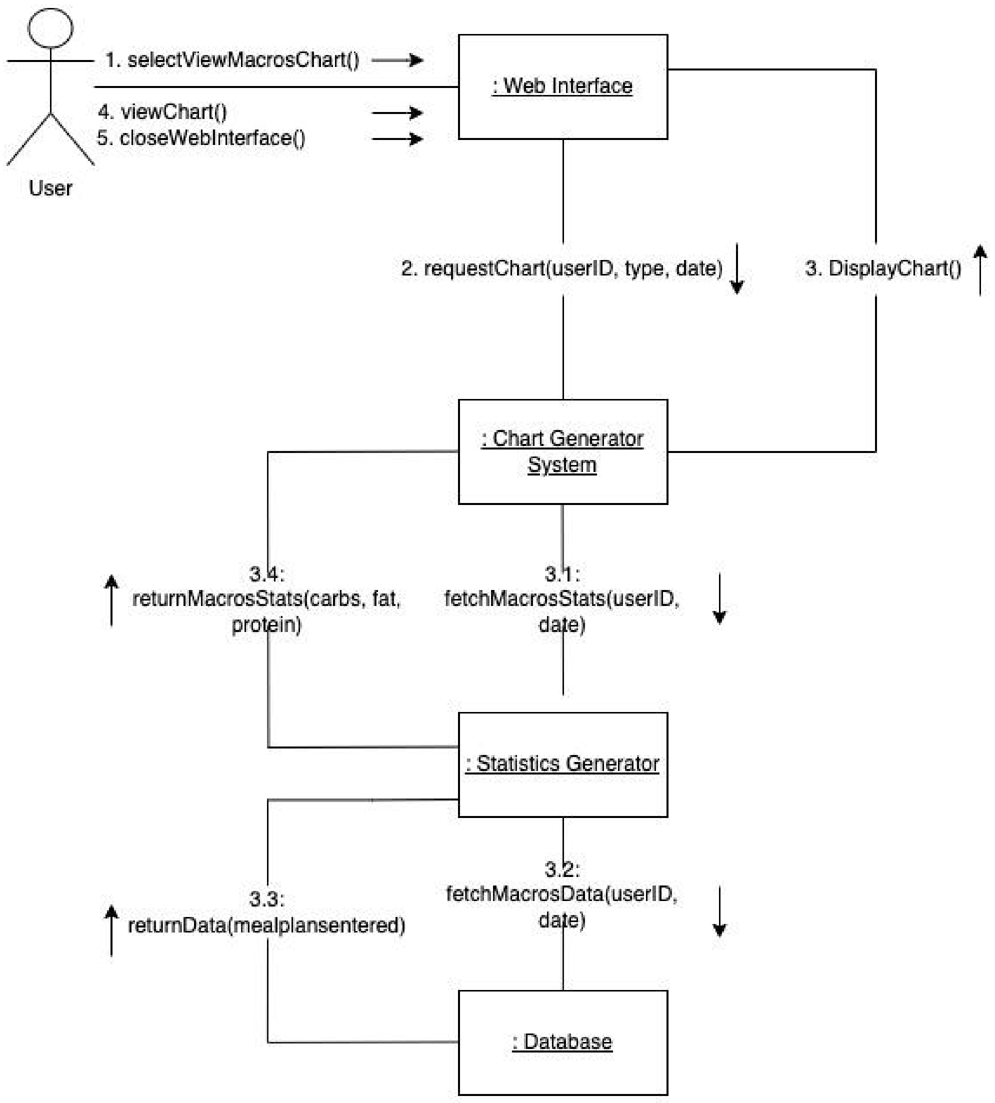

User
1. selectViewMacrosChart()
4. viewChart()
5. closeWebInterface()
: Web Interface
2. requestChart(userID, type, date)
3. DisplayChart()
: Chart Generator System
3.4: returnMacrosStats(carbs, fat, protein)
3.1: fetchMacrosStats(userID, date)
: Statistics Generator
3.3: returnData(mealplansentered)
3.2: fetchMacrosData(userID, date)
: Database

Figure 9: Collaboration Diagram for Use Case 2

3.2.3. Use Case 3 - Distance Travelled

3.2.3.1. Fully Dressed Scenario

Use Case Name	Distance travelled	
Summary	Counting the amount of distance travelled by the user	
Priority	2	
Preconditions	User logs into the web app	
Postconditions	Chart generated showing their distance per day, week, or month	
Primary Actors	App user	
Secondary Actors	Web Interface, chart generator, statistics generator, database	
Trigger	User successfully logs into the web app	
Scenario (Main)	**Steps**	**Scenario**
	1	User selects to view distance travelled
	2	User selects to view daily, weekly, or monthly steps
	3	Web interface requests chart from charts generator
	4	Charts generator requests stats from stats generator
	5	Stats generator fetches data from the database
	6	Stats generator sends stats to chart generator
	7	Chart generator generates a chart
	8	Chart generator sends chart to web interface
	9	Web interface displays the chart
	10	User views the chart
	11	User exits
Extensions	**Steps**	**Branching Action**
	5.1a	System unable to retrieve data from the database

Table 4: System Sequence Diagram for Use Case 3

3.2.3.2. System Sequence Diagram

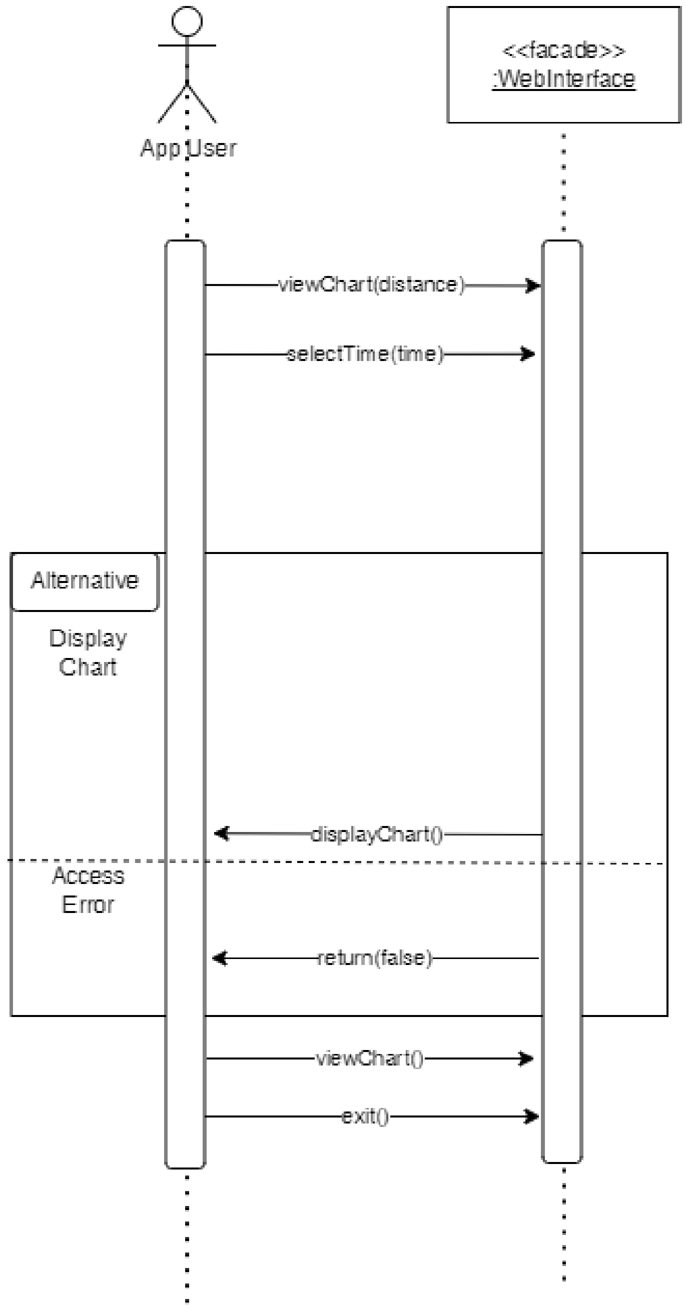


Figure 10: System Sequence Diagram for Use Case 3

3.2.3.3. Sequence Diagram

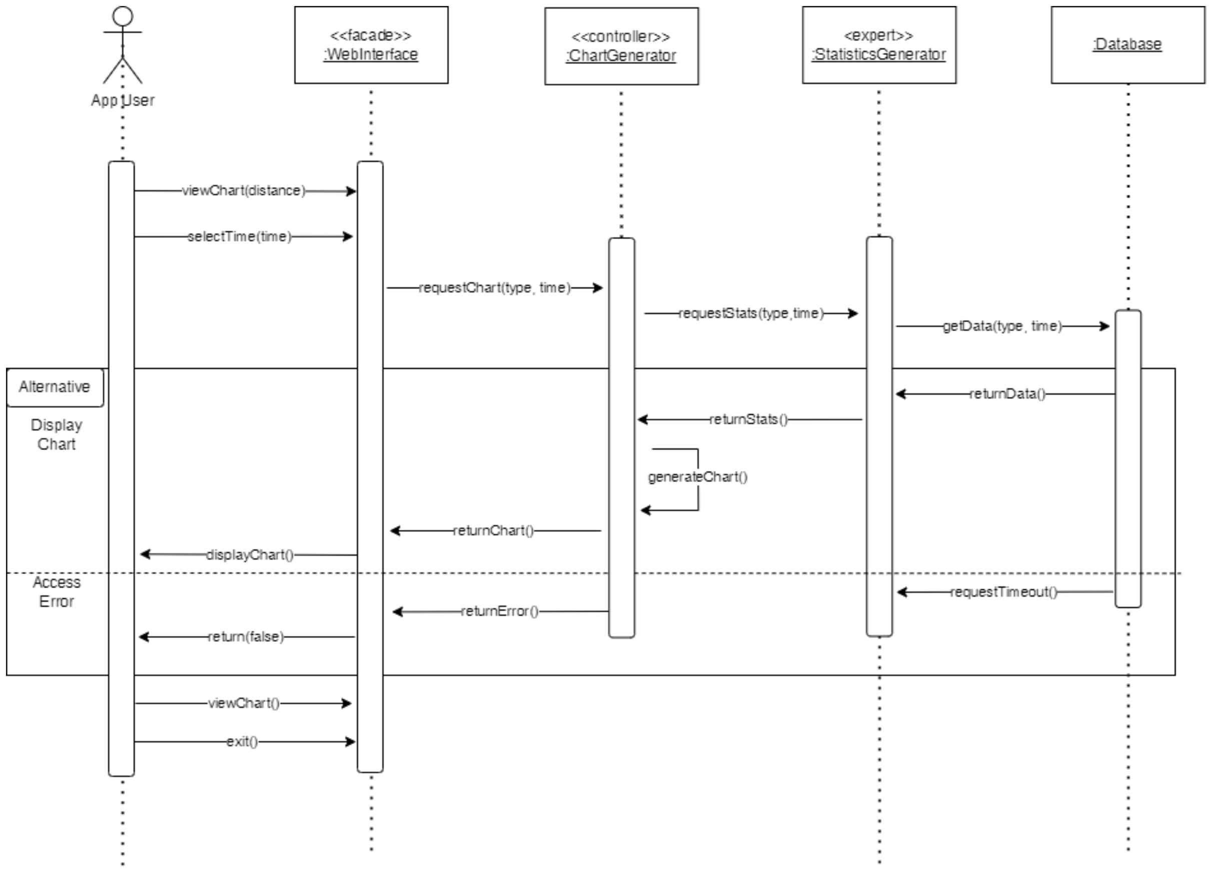


Figure 11: Sequence Diagram for Use Case 3

3.2.3.4. Collaboration Diagram

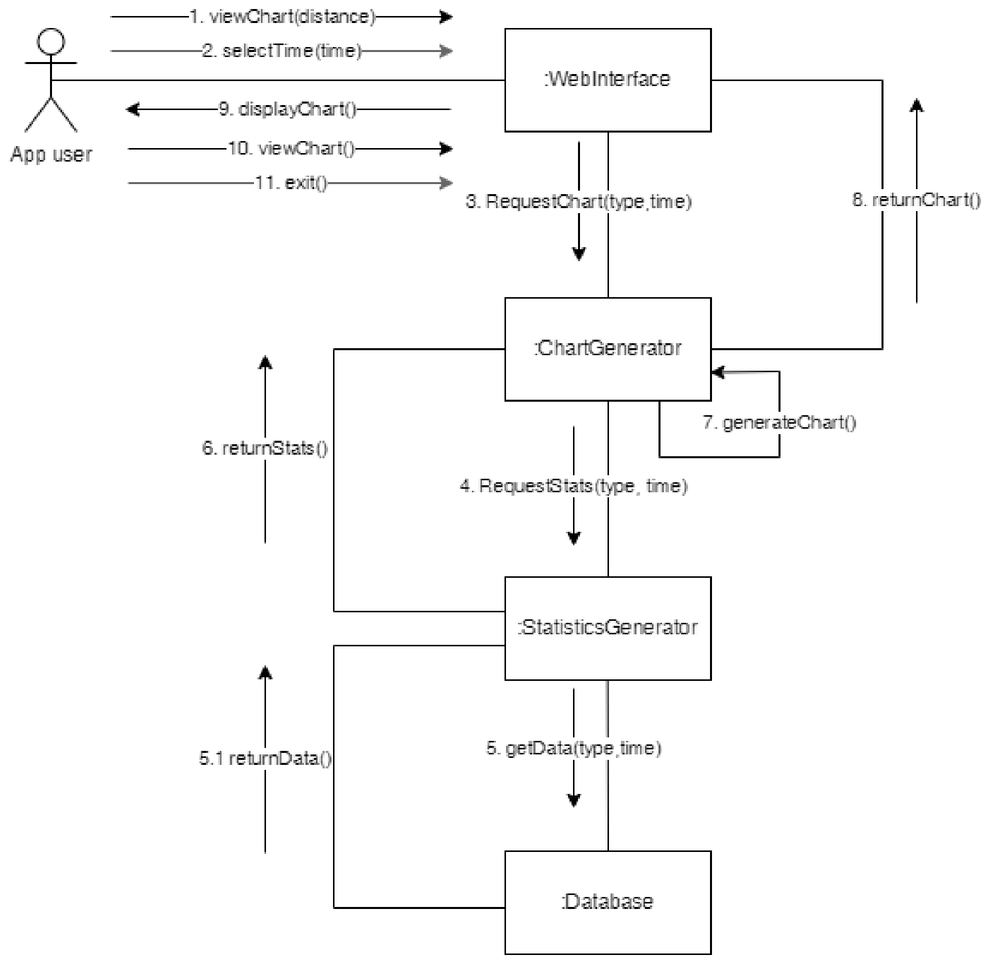


Figure 12: Collaboration Diagram for Use Case 3

3.2.4. Use Case 4 - Ride Intensity

3.2.4.1. Fully Dressed Scenario

Use Case Name	Elevation / Ride Intensity	
Summary	User selects Elevation / Ride Intensity. Chart generator system displays the chart using the user's statistics.	
Priority	4	
Pre-conditions	User logged in, Statistics page is open	
Post-conditions	Elevation/Ride Intensity graph is displayed	
Primary Actor	End User	
Secondary Actor	Statistics database (Stats Group 10)	
Trigger	“Elevation / Ride Intensity” chart is selected.	
Main Scenario	**Step**	**Action**
	1	User selects Elevation / Ride Intensity chart from the chart selection
	2	User selects time frame (Lifetime/Year/Month/Week/Day or s specific ride)
	3	Chart generator system gets data from the statistics database
	4	Chart generator system generates and displays chart
	5	User views the chart
	6	User exits
Extensions	**Step**	**Branching Action**
	3a	System is unable to access database – error message
	4a	System is unable to display the chart – not enough data
	5a.1	User selects a different time frame
	5a.2	Go back to Step 3

Table 5: Fully Dressed Scenario for Use Case 4

3.2.4.2. System Sequence Diagram

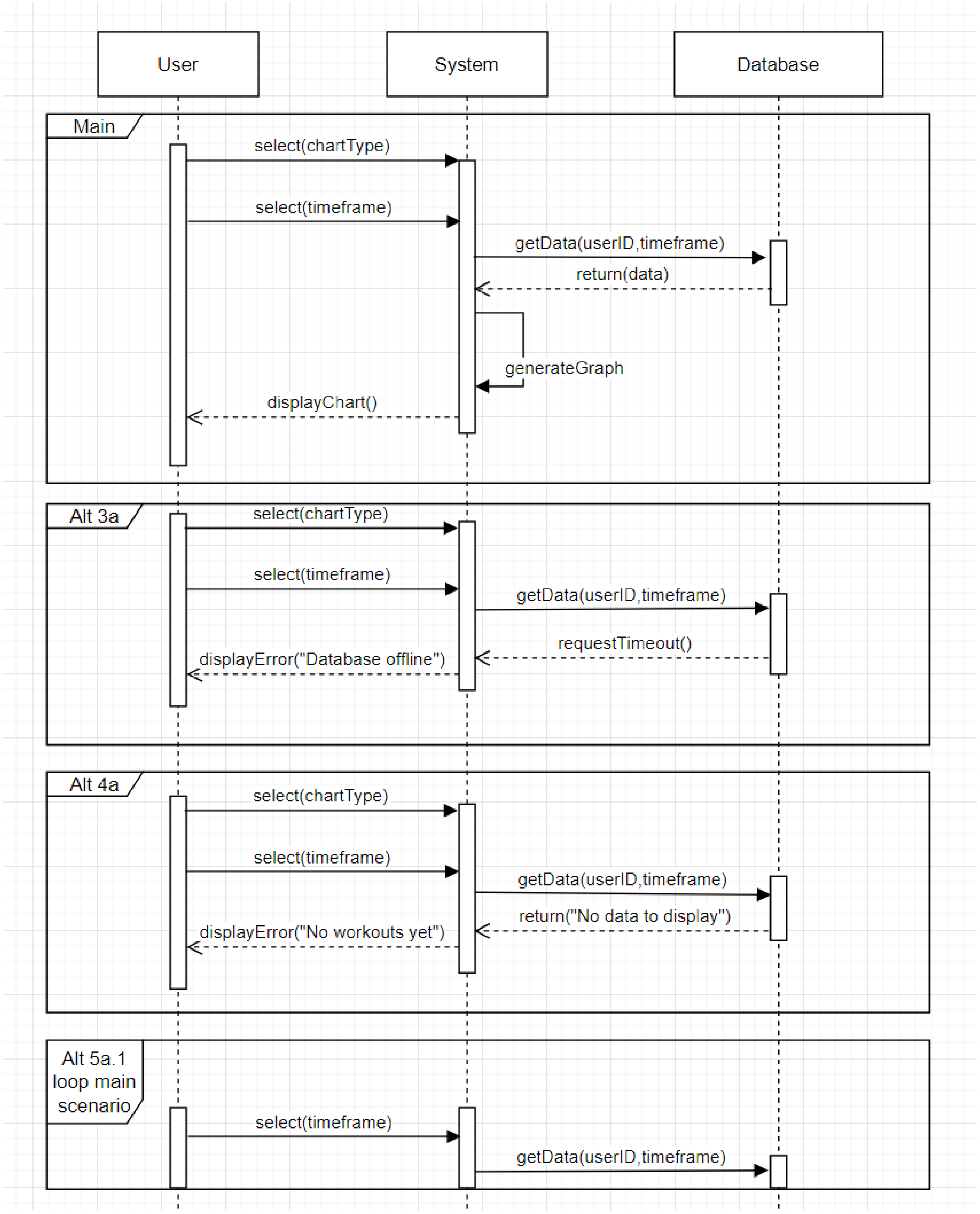

User
System
Database
Main
select(chartType)
select(timeframe)
getData(userID,timeframe)
return(data)
generateGraph
displayChart()
Alt 3a
select(chartType)
select(timeframe)
getData(userID,timeframe)
requestTimeout()
displayError("Database offline")
Alt 4a
select(chartType)
select(timeframe)
getData(userID,timeframe)
return("No data to display")
displayError("No workouts yet")
Alt 5a.1 loop main scenario
select(timeframe)
getData(userID,timeframe)

Figure 13: System Sequence Diagram for Use Case 4

3.2.4.3. Sequence Diagram

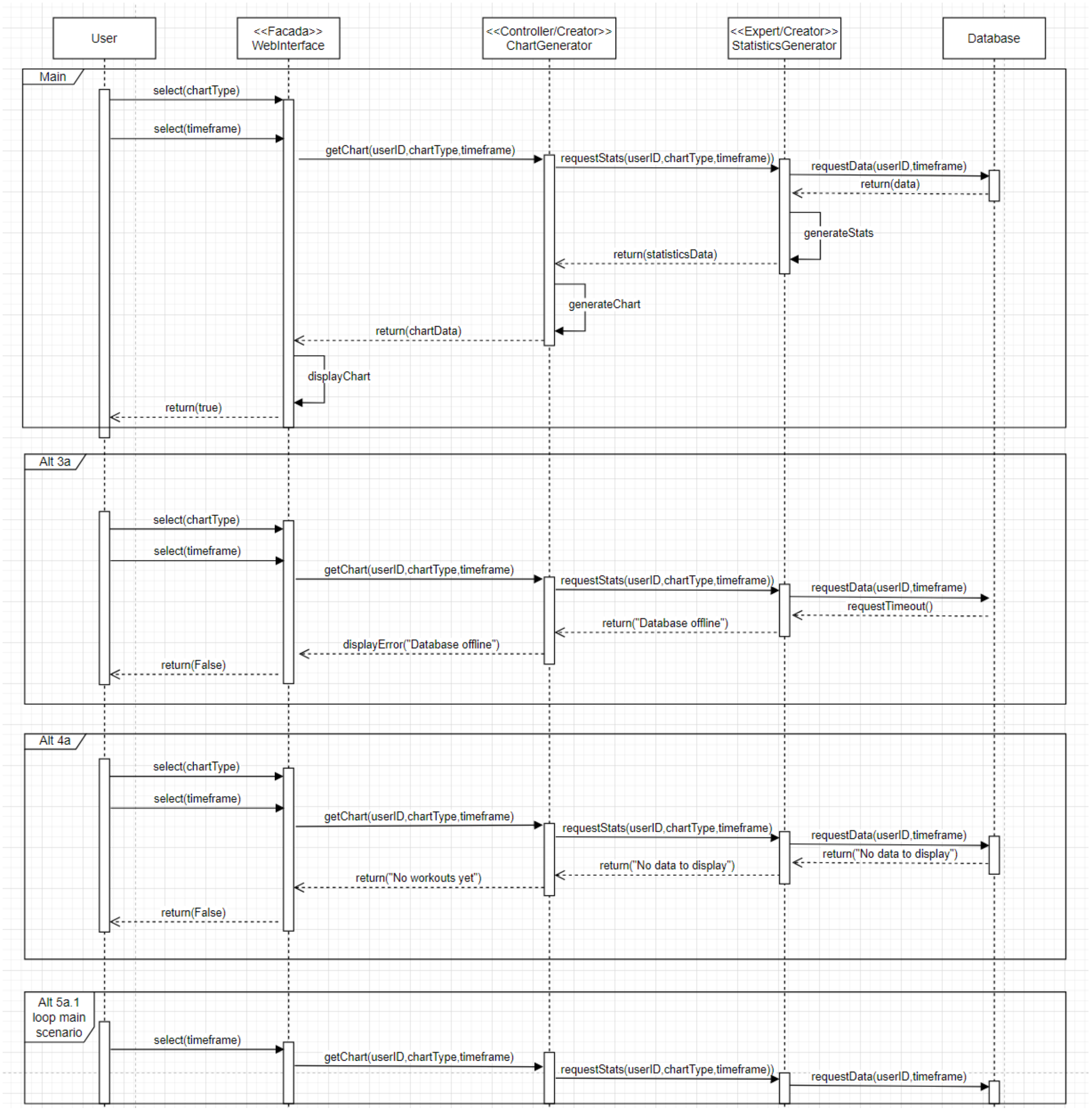

User
<<Facada>> WebInterface
<<Controller/Creator>> ChartGenerator
<<Expert/Creator>> StatisticsGenerator
Database
Main
select(chartType)
select(timeframe)
getChart(userID,chartType,timeframe)
requestStats(userID,chartType,timeframe))
requestData(userID,timeframe)
return(data)
generateStats
return(statisticsData)
generateChart
return(chartData)
displayChart
return(true)
Alt 3a
select(chartType)
select(timeframe)
getChart(userID,chartType,timeframe)
requestStats(userID,chartType,timeframe))
requestData(userID,timeframe)
requestTimeout()
return("Database offline")
displayError("Database offline")
return(False)
Alt 4a
select(chartType)
select(timeframe)
getChart(userID,chartType,timeframe)
requestStats(userID,chartType,timeframe)
requestData(userID,timeframe)
return("No data to display")
return("No data to display")
return("No workouts yet")
return(False)
Alt 5a.1 loop main scenario
select(timeframe)
getChart(userID,chartType,timeframe)
requestStats(userID,chartType,timeframe))
requestData(userID,timeframe)

Figure 14: Sequence Diagram for Use Case 4

3.2.4.4. Collaboration Diagram

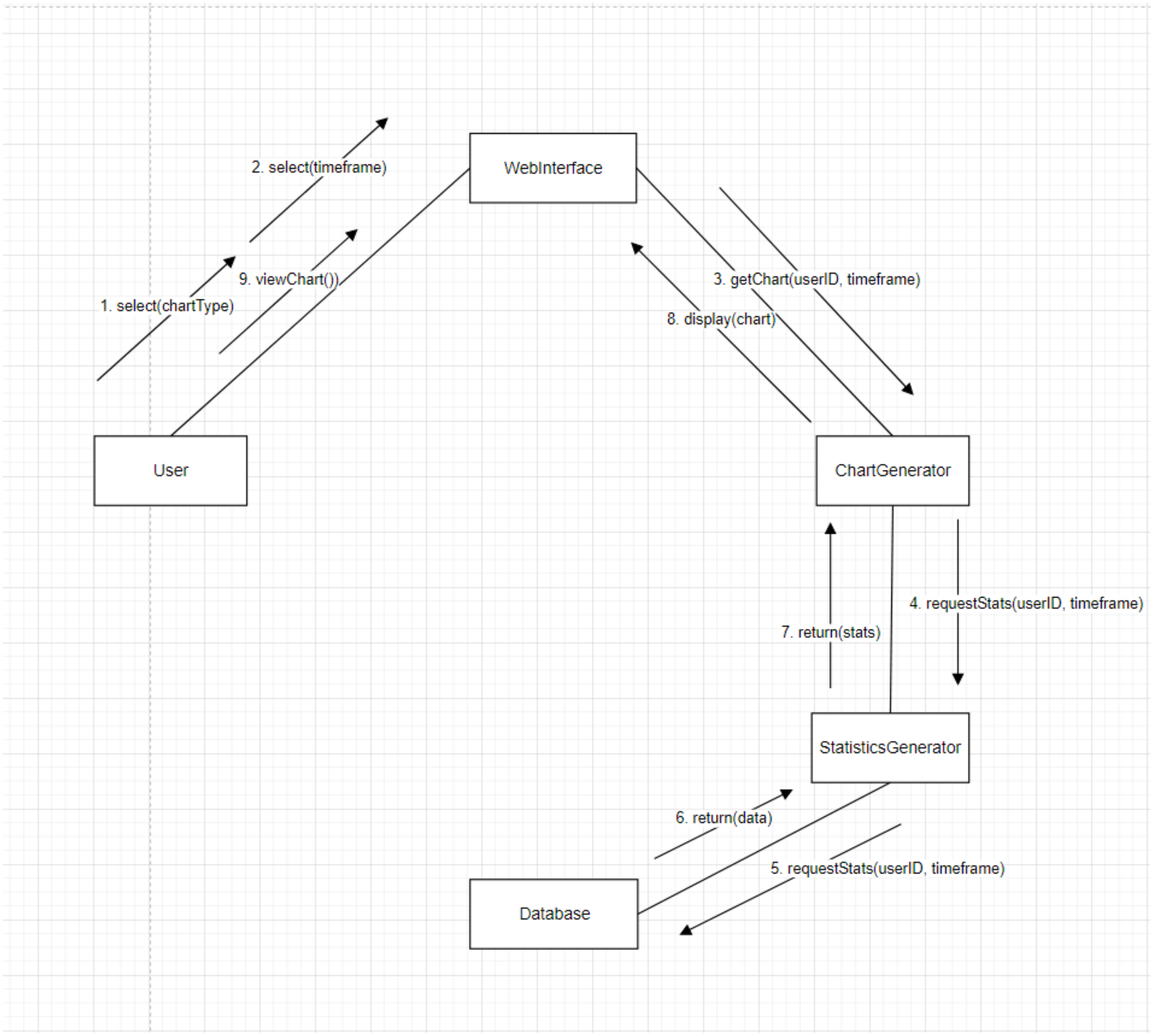


Figure 15: Collaboration Diagram for Use Case 4

3.2.5. Use Case 5 - Moving Average Chart

3.2.5.1. Fully Dressed Scenario

Use Case Name	Moving Average Chart	
Summary	Calculating the moving average in kilometers/hour given a time frame for the cyclist's workout.	
Priority	5	
Preconditions	User recorded their workout sessions on the app.	
Postconditions	Chart generated showing Moving Average in Kilometers/hour on a given time frame.	
Primary Actor(s)	User	
Secondary Actor(s)	Statistics Generator , Database	
Trigger	User opens the web interface of the fitness app and enters their ID.	
Main Scenario	Step	Action
	1	User selects the charts option of the web interface.
	2	User selects Moving Average Chart and the choice is given to Chart Generator System
	3	User selects the dates of the user's history to use in calculation of Moving Averages and the choice is given to the Chart Generator System
	4	Chart Generator System requests the Statistics Generator for Moving Averages for the given dates.
	5	Statistics Generator requests workout data of the user from a period of time from the database.
	6	Statistics Database obtains workout data of the user from a period of time from the database.
	7	Statistics Database generates Moving Averages from data obtained from Database.

Use Case Name	Moving Average Chart	
Summary	Calculating the moving average in kilometers/hour given a time frame for the cyclist's workout.	
	8	Statistics Database gives all Moving Averages calculated to Chart Generator System.
	9	Chart Generator System generates Moving Averages Chart.
	10	Web Interface receives Web Interface Moving Averages Chart.
	11	Web Interface displays Moving Averages Chart to User.
Extensions	Step	Branching Action
	11a	User changes dates selected. Continues from step 3.
	5a	Error Accessing Database. Error Message Displayed.
	4a	Error Accessing Statistics Generator. Error Message Displayed.

Table 6: Fully Dressed Scenario for Use Case 5

3.2.5.2. System Sequence Diagram

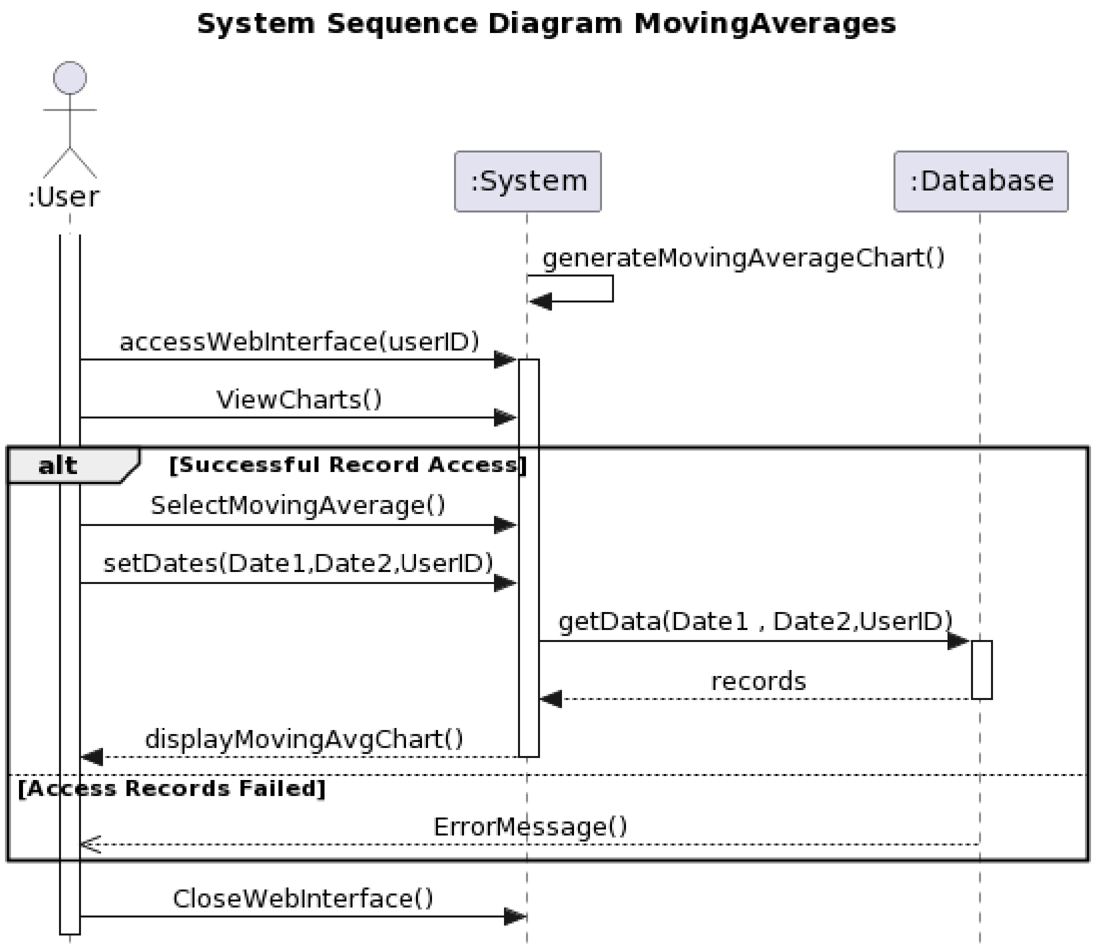
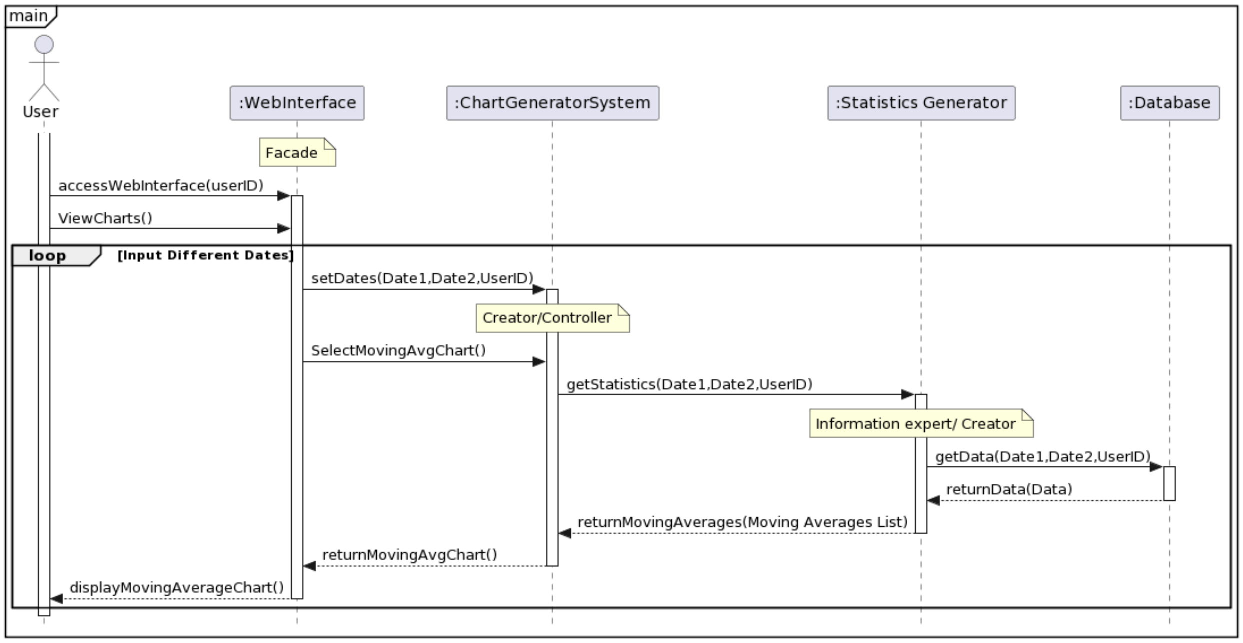
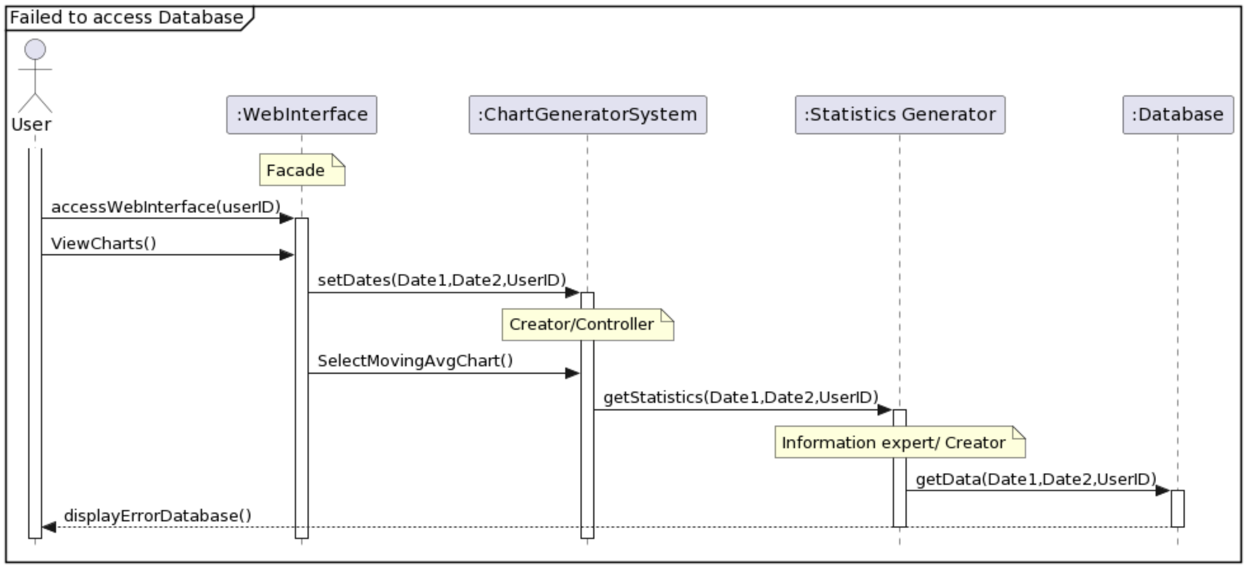

System Sequence Diagram MovingAverages
:User
:System
:Database
generateMovingAverageChart()
accessWebInterface(userID)
ViewCharts()
alt
[Successful Record Access]
SelectMovingAverage()
setDates(Date1,Date2,UserID)
getData(Date1 , Date2,UserID)
records
displayMovingAvgChart()
[Access Records Failed]
ErrorMessage()
CloseWebInterface()

Figure 16: System Sequence Diagram for Use Case 5

3.2.5.3. Sequence Diagram


main
User
:WebInterface
:ChartGeneratorSystem
:Statistics Generator
:Database
Facade
accessWebInterface(userID)
ViewCharts()
loop
[Input Different Dates]
setDates(Date1,Date2,UserID)
Creator/Controller
SelectMovingAvgChart()
getStatistics(Date1,Date2,UserID)
Information expert/ Creator
getData(Date1,Date2,UserID)
returnData(Data)
returnMovingAverages(Moving Averages List)
returnMovingAvgChart()
displayMovingAverageChart()


Failed to access Database
User
:WebInterface
:ChartGeneratorSystem
:Statistics Generator
:Database
Facade
accessWebInterface(userID)
ViewCharts()
setDates(Date1,Date2,UserID)
Creator/Controller
SelectMovingAvgChart()
getStatistics(Date1,Date2,UserID)
Information expert/ Creator
getData(Date1,Date2,UserID)
displayErrorDatabase()

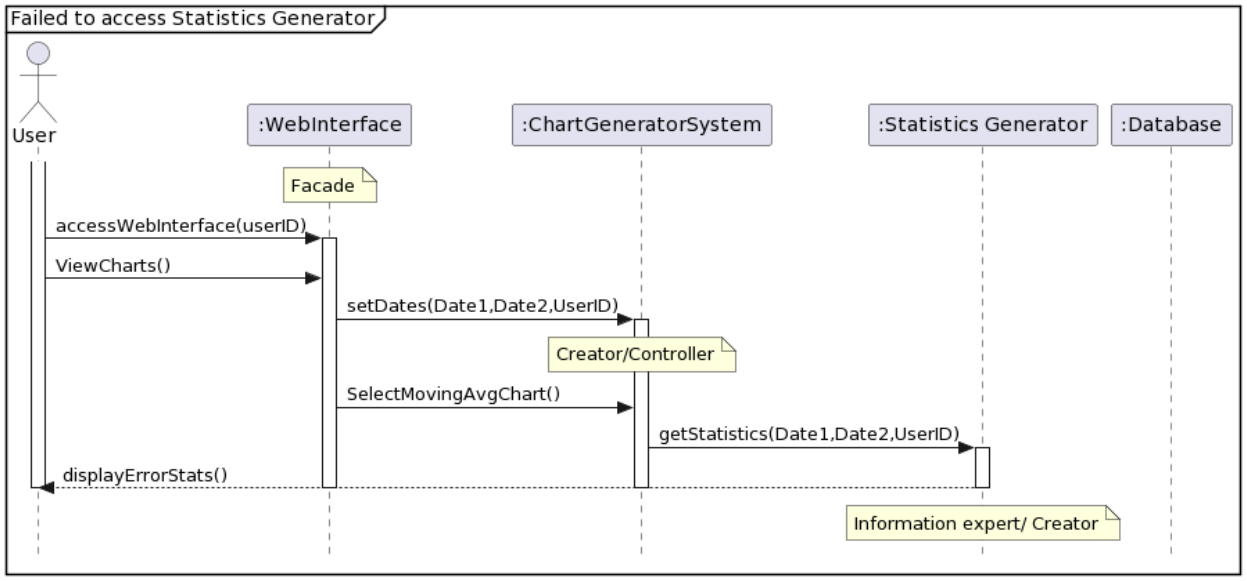

Failed to access Statistics Generator
User
:WebInterface
:ChartGeneratorSystem
:Statistics Generator
:Database
Facade
accessWebInterface(userID)
ViewCharts()
setDates(Date1,Date2,UserID)
Creator/Controller
SelectMovingAvgChart()
getStatistics(Date1,Date2,UserID)
displayErrorStats()
Information expert/ Creator

Figure 17: Sequence Diagram for Use Case 5

3.2.5.4. Collaboration Diagram

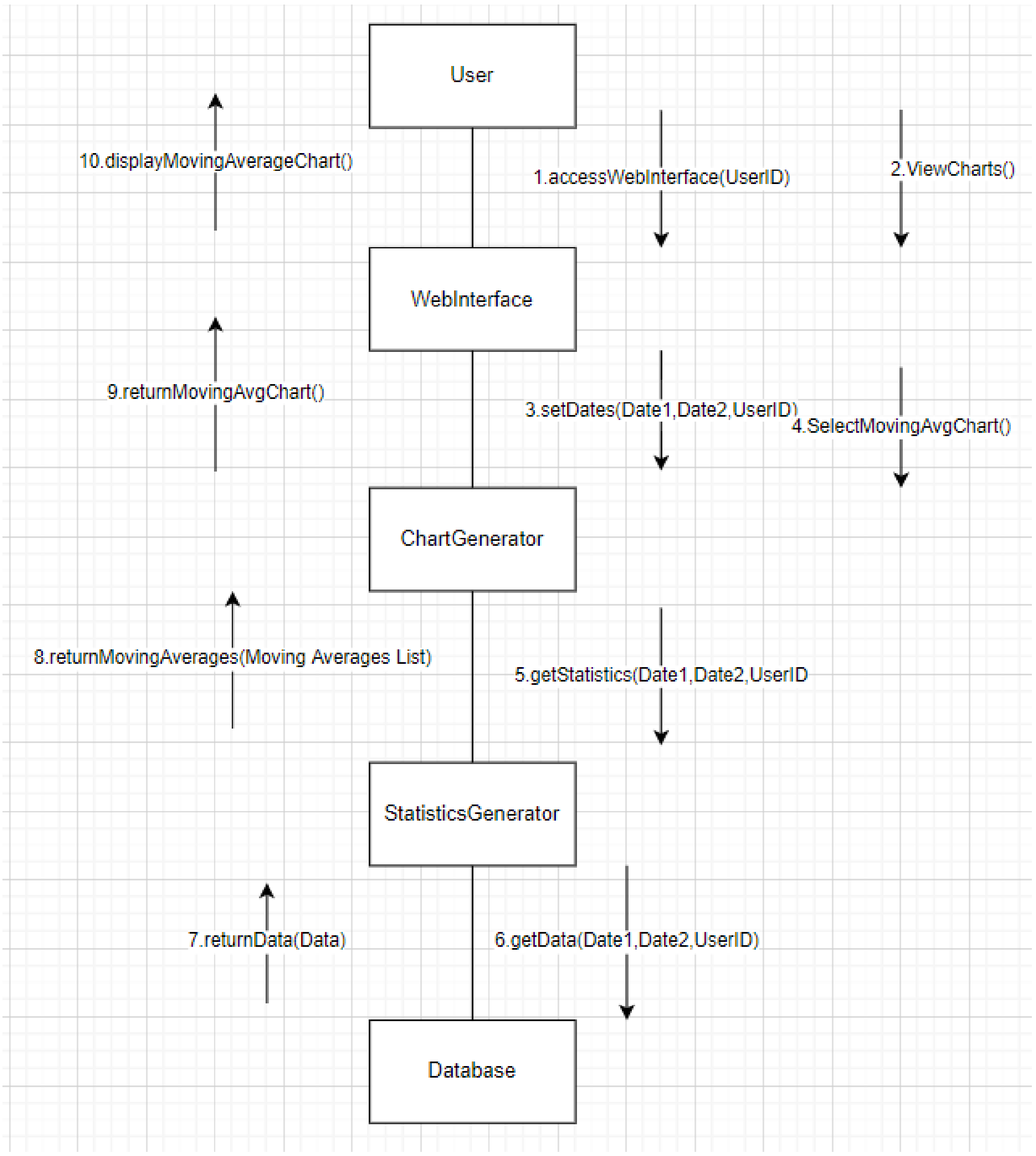

User
WebInterface
ChartGenerator
StatisticsGenerator
Database
1.accessWebInterface(UserID)
2.ViewCharts()
3.setDates(Date1,Date2,UserID)
4.SelectMovingAvgChart()
5.getStatistics(Date1,Date2,UserID
6.getData(Date1,Date2,UserID)
7.returnData(Data)
8.returnMovingAverages(Moving Averages List)
9.returnMovingAvgChart()
10.displayMovingAverageChart()

Figure 18: Collaboration Diagram for Use Case 5

4. External Interface Requirements

4.1. User Interfaces

The user can access their cycling profile using the web interface on their electronic device. The interface will be mobile responsive. Therefore, they can view it on their desktop, tablet and mobile devices.

4.2. Hardware Interfaces

The web interface created using Java, HTML5, CSS, PHP ,and javascript will be accessible through phone, laptop, and other devices that have access to the internet. It will also be using mySQL as a database to store all data.

4.3. Software Interfaces

The web interface will be developed using Java 8, HTML5, CSS, PHP, and Javascript. The chart of a statistic generated can be downloaded as a pdf or image file (.jped).

4.4. Communication Interfaces

There are no other communication interfaces. The statistics generator will communicate with the cyclist database. The web interface will interact with the chart generator system. The chart generator system interacts with the statistics generator.

5. Other Nonfunctional Requirements

5.1. Performance Requirements

The chart generator system must provide access to the statistics generator in order to display the chart selected. The speed of the statistics generator should be fast enough so that the chart generator system can generate the chart selected in a timely manner.

5.2. Safety Requirements

As a productivity software product, no special safety measure is necessary.
It is important that the user doesn't leave the fitness tracker exposed to the sun for too long, otherwise overheating may occur and will damage the product.

5.3. Security Requirements

A cyclist's history of cycling, where they cycled, dates of cycling, and the cyclist's physical attributes are part of their profile on the web interface. Therefore, the user must provide their username and password in order to protect this information.

5.4. Software Quality Attributes

Software development focus will primarily be on the correctness of the product and if time allows, the ease of testing. Regular checkups on the software will be necessary to uncover any unwanted bugs. Regular maintenance and updates will occur to provide users with the best quality product.

6. Other Requirements

No other requirements are to be noted for this project.

Appendix A: COCOMO Adjustment factor (c) breakdown

1. **Product Factors:**

   1.1) Required Software Reliability - Low

   Effects of a software failure will result in a slight inconvenience for the user since they can always come back to the graph visualisation later and try again

   1.2) Database Size - Low

   The visualisation feature will require minimal additional storage to contain user's pre-generated graphs for the duration of user session.

   1.3) Product Complexity - Low

   The visualisation feature will use data already processed by the statistics team, therefore it will only require evaluation of moderate level expressions

2. **Computer Attributes:**

   2.1) Execution Time Constraint - Nominal

   We expect <50% execution time resource consumption as the majority of CPU cycles will be used to generate graphs at the beginning of the user session.

   2.2) Main Storage Constraint - Nominal

   The visualisation feature will only require access to the data collected for the current user, therefore the constraint is set to *nominal*. This metric will change To high/very-high if the "compare scores to others" feature is implemented.

   2.3) Platform Volatility - Low / Nominal

   After the first release we aren't expecting any major changes or updates to the software.

   2.4) Computer Turnaround Time - Low

   CTT is set to low as the software is designed to be used in an interactive mode.

3. **Personal Attributes:**

   3.1) Analyst Capability - Very Low

   Analyst not available.

   3.2) Application Experience - Very Low

   The application experience of the project team is averaged to <4 months of experience.

   3.3) Programmer Capability - Nominal

   Programmer capability set to nominal. Main factors: low ability and efficiency, high thoroughness communication and cooperation.

   3.4) VM Experience - Very Low

The project team has less than 1 month experience with the platform (GUI, DB, networking etc.)

3.5) Programming Language Experience - Low

The project team has less than 4 months of experience with language and tools used.

4. **Project Attributes:**

4.1) Use of Modern Programming Practices - High

Use of modern programming practices such as requirements analysis and design (models, sequence diagrams), structured code, development schedules etc.

4.2) Use of Software Tools - High

Program design language, programming libraries, code compilers

4.3) Required Development Schedule - High

Strict deadlines with limited extension possibility.

Appendix B: Issues List

No issues to report as of Now.