Redback

Team Redback

Requirements	Development	Meeting Notes	Sprints
	Presentation	Quality Control	

Name	Preferred Name	Photo	Contact	Current Role
GUO YI LEE	Gant		Glee3@student.unimelb.edu.au	Product Owner
Yu Wu	Felix		ywwu13@student.unimelb.edu.au	Scrum Master
Qi Li	Leon		lql4@student.unimelb.edu.au	Backend Developer
Jiacheng Zhang	Gary		jiachzhang2@student.unimelb.edu.au	Frontend developer
Sirui Liang	Sirui		sirliang@student.unimelb.edu.au	Testing

Useful Links

Github: https://github.com/COMP90082SM12022/GA-Redback

Coach Ollie info: https://vic.tri-alliance.com/about/coaches/

Garmin API info: https://developer.garmin.com/fit/overview/

CoachingMate info: https://coachingmate.com/

Garmin Watch info: https://www.garmin.com/en-AU/c/wearables-smartwatches/

Requirements

This page is for project requirements.

Background - Redback

Project overview

This project is to integrate CoachingMate with the Garmin API and UI to allow athletes to be able to connect their Garmin devices with CoachingMate.

CoachingMate is a health and fitness application that transforms methods and practices of interaction between how coaches and athletes interact with one another. CoachingMate provides an array of services to coaches, allowing them to efficiently view and manage their athletes. Features such as visibility in income revenue, managing their fitness schedules, easy drag and drop workout builder system to allow coaches to create workout programs seamlessly.

To further elevate the CoachingMate capability, CoachingMate is focusing on tracking the fitness stats of their athletes, specifically their physical activity and details during their training program period. This would be achieved by having the athlete wear a Garmin watch, a fitness tracker watch, which will track the physical activity, in terms of activity levels, heart rates and other details. The Garmin watch data is being transferred over to the Garmin database servers.

CoachingMate will tap on this information through the Garmin API, which will provide the user/coach with the information required to analyse and track the athlete's performance, which would be displayed in a form of dashboard or visual analytics.

Vision Statement / Goal

In-scope :

The final product of this group is to achieve transferring data from Garmin watches to the CoachingMate backend by using the Garmin APIs, using the Garmin developer resources.

The key criteria of this project is for users to enable seamless data transmission from their Garmin watch to the CoachingMate dashboard.

Key functions to be achieved including:

- The connection is seamless and once data is synchronised the updates will automatically appear on the CoachingMate dashboard.
- The users can decide what data from Garmin devices they want to share with CoachingMate.
- Connection and disconnection will be instant with just a click.
- The user will be notified when data synchronisation with CoachingMate succeed or failed.
- Data transmitted will be stored in different databases for data access control purposes planned for later stages. (TBC)

Out-scope :

Once the API has been created successfully, the focus of the project would shift toward working on improving the user interface of the CoachingMate dashboard, including better visualisation of the data collected from the Garmin API, data access control and displaying of multiple users' data in one dashboard.

Business Case

Provides an in-depth analysis – currently coaches using CoachingMate do not have access to any of the athlete's performance data

Improves coaching advice – coaches can use the athlete's performance data to provide a more detailed and targeted training program depending on the performance

Easier accessibility – by integration into CoachingMate, the coaches do not require to individually access each athlete's Garmin data on their personal accounts just to view the information but can view easily on the CoachingMate platform.

Motivational Model - Redback

Versions

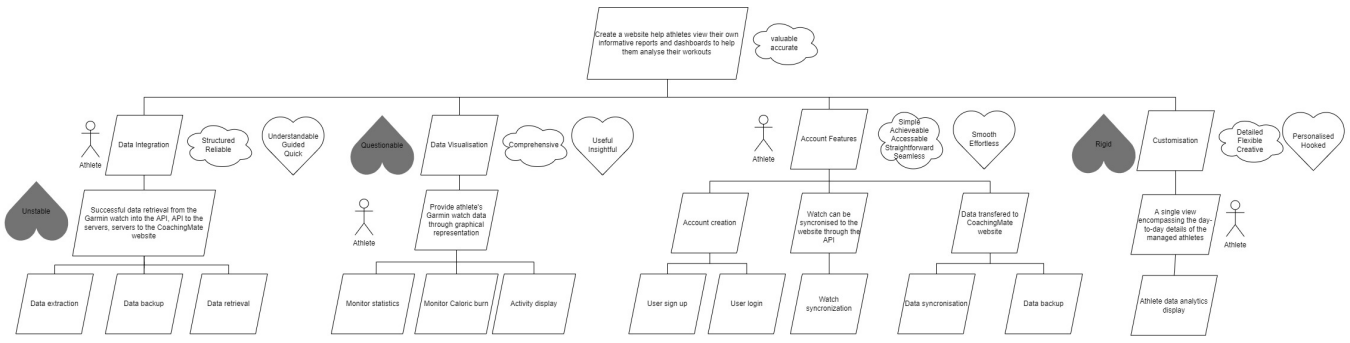
Version ID	Description	Date
2.1	Edited in accordance to what we target to achieve within this project, focusing on athletes as users	2022-04-28
2.0	The second version based on after consulting the supervisor, Hanna, and the deeper understanding of the project with Edwardo	2022-03-26
1.0	The first version is based on an initial understanding of the project and client meetings.	2022-03-23

Version 2.1

Do-Be-Feel-Who List - Version 2.1

Do			Be	Feel	Who
Data Integration	Successful data retrieval from the Garmin watch into the API, API to the servers, servers to the CoachingMate website	Data extraction	Structured Reliable	Understandable Guided Quick	Athlete
		Data backup			
		Data retrieval			
Data Visualisation	Provide athlete's Garmin watch data through graphical representation	Monitor statistics	Comprehensive	Useful Insightful	Athlete
		Monitor Caloric burn			Athlete
		Activity display			Athlete
Account Features	Account creation	User sign up	Simple Achieveable Accessable Straightforward Seamless	Smooth Effortless	Athlete
		User login			
	Watch can be syncronised to the website through the API	Watch syncronization			
	Data transfered to CoachingMate website	Data syncronization			
		Data backup			
Customisation	Able to see achievements and detailed targets on a personal level	Deep-dive data display	Detailed Flexible Creative	Personalised Hooked	Athlete

Goal Model - Version 2.1

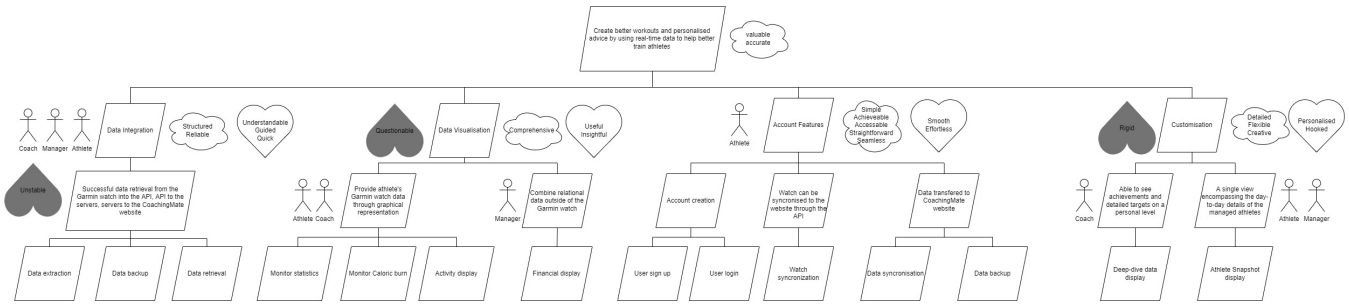


Version 2.0

Do-Be-Feel-Who List - Version 2.0

Do			Be	Feel	Who
Data Integration	Successful data retrieval from the Garmin watch into the API, API to the servers, servers to the CoachingMate website	Data extraction	Structured Reliable	Understandable Guided Quick	Athlete, Coach, Manager
		Data backup			
		Data retrieval			
Data Visualisation	Provide athlete's Garmin watch data through graphical representation	Monitor statistics	Comprehensive	Useful Insightful	Athlete, Coach
		Monitor Caloric burn			Athlete, Coach
		Activity display			Athlete, Coach
	Combine relational data outside of the Garmin watch	Financial display			Manager
Account Features	Account creation	User sign up	Simple Achieveable Accessable Straightforward Seamless	Smooth Effortless	Athlete
		User login			
	Watch can be syncronised to the website through the API	Watch syncronization			
	Data transfered to CoachingMate website	Data syncronization			
		Data backup			
Customisation	Able to see achievements and detailed targets on a personal level	Deep-dive data display	Detailed Flexible Creative	Personalised Hooked	Athlete
	A single view encompassing the day-to-day details of the managed athletes	Athlete Snapshot display			Coach, Manager

Goal Model - Version 2.0



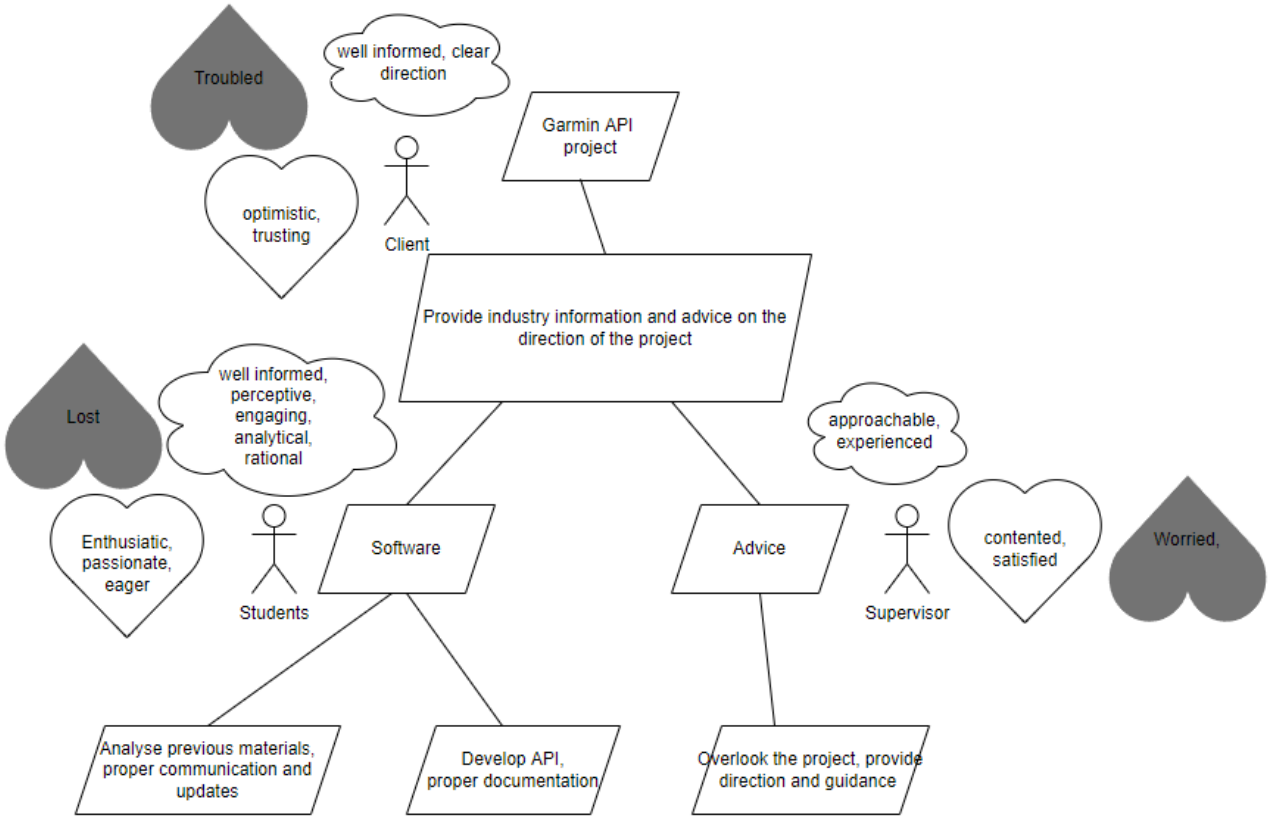
Version 1.0

Do-Be-Feel-Who List - Version 1.0

WHO	DO	BE	FEEL
Client	Provide industry information	well informed	optimistic
	advice on the direction of the project	clear direction	trusting

Students	Analyse previous materials proper communication and updates	well informed perceptive engaging analytical rational	enthusiastic passionate eager
Supervisor	Overlook the project progress provide direction and guidance	approachable experienced	contented satisfied

Goal Model - Version 1.0



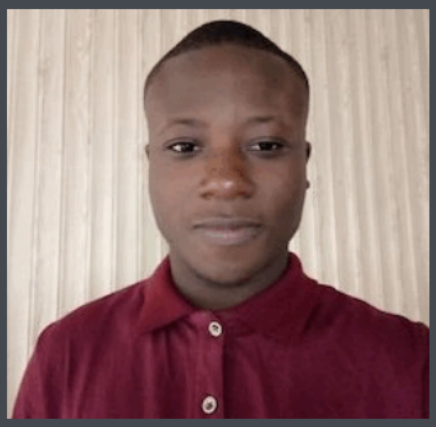
Personas - Redback

NAME

Jack Brown

TYPE

Athlete



Demographic

Male 26 years

Sydney

Single

Athlete

Quote

Only by constantly recording your sports performance can you make a better workout plan.

Motivations

As an athlete, Jack needs to plan his own training and keep track of the hours of training each day. Recently, he heard about a website called Garmin. The website can help athletes make training plans and record training time. The website offers its app, which can be used on a phone or smartwatch. Jack thinks Garmin's products are very convenient and allow him to focus more on training. He decided to sign up for its membership and download the app on his smartwatch.

Goals

- To efficiently manage his training program.
- To record the calories he burned during each exercise.
- To check his exercise history in a visualization way.
- After training, his performance will be digitized and can be viewed by him.

Frustrations

- The user interface is cluttered and he couldn't find the information he needed.
- The data could not be uploaded in time, and he could not immediately see his training results on the website.
- Unable to find the data or information he needs on the website (eg heat map, average heart rate).

Technology

Channels

UXPRESSIA

This persona was built in uxpressia.com

NAME

Scott Graham

Gym Manager

Demographic

Male 30 years

Vancouver, BC

Single

Gym Manager

Quote

I wish there is a tool that would allow me to assess whether the coaches are dutiful by monitoring the clients' training status.

I don't like analyzing many charts like a data scientist. It'd be better if the data visualization is as simple as possible

Motivations

As a gym manager, Scott is seeking a tool that can help him manage the gym. Although the gym itself has a management system, the system is unable to track the clients' training feedback and whether the coaches are dutiful.

In addition, Scott needs to use separate software in his work. For example, he needs to use financial software to manage the gym's finance and human resource software to manage the clients and coaches. Therefore, Scott wishes there is software that can combine these functions.

Goals

- To track the clients' and coaches' trainning status
- To schedule the trainning program
- To assign the clients to coaches
- To manage the gym's finance, such as clients' fee and coaches' income

Frustrations

- The system did not update the tranning and finance status in time
- The data visualization is not easy to understand
- The security of the account as the manager

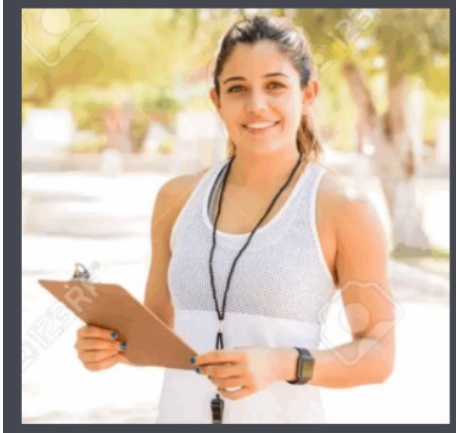
Technology

Channels

NAME

Angelina Johnson

Coach



Demographic

Female 26 years

Melbourne

Single

Quote

I always strive to help my athletes achieve their full potential and meanwhile enjoy a positive experience !

Motivations

Angelina is now working as a coach in a sports training club in Melbourne for 2 to 3 years with specializations in multiple sports including swimming, body-building, running, and so on. Angelina regularly coaches 5 to 8 people per week, her clients are mostly white-collar workers.

Angelina is looking for a system where she can see a comprehensive list of athletes' performance data to make better coaching plans. Besides, she also expects to monitor the performance of all of her athletes in a collective view.

Goals

- Track athletes' health and performance data transformed from smart devices.
- Develop and update training plans based on data collected to achieve more effective coaching outcomes.

Frustrations

- Lack of clear demonstration of the performance data from smart devices.
- Lack of interaction with data collected.
- Switching among different athletes' data is inconvenient.

Technology

Channels

User Stories - Redback

Versions

Version ID	Description	Date
3.0	Data Visualisation is labelled as out-of-scope, addtional of User Stories in Sprint 3 focusin on out-of-scope	2022-05-13
2.1	In-scope section is further branched out to further clarify the details of what was required to achieve from the user stories	2022-04-28
2.0	Redone the In-scope section in accordance to progress and better understanding of the current project situation	2022-04-14
1.0	The first version is based on an initial understanding of the project and client meetings.	2022-03-25

Version 3.0

In-scope

Epics	User	User Story ID	User Stories	MoSCoW Priority	Story Points(2 hour/pt)
E1- Account Administration Features	As an athlete, I am able to create an account, so that I can view and edit my profile and devices.				
	Athlete	#01	• The user can create the account, set the password and username. 1. Able to type into the username field 2. Able to type into the password field 3. User data is able to be stored into the backend database	Must have	3
	Athlete	#03	• The user account can bind with any sports watch. 1. The user account is able to have the option to sync with the Garmin server 2. Create the API required for the Garmin server data to sync with the CoachingMate server data 3. Create the API link from the Garmin server linking with the Backend (Java)	Must have	4
	Athlete	#04	• The user can unbind with any sports watch that they have currently binded with. Option screen of able to unbind the syncronisation to be seen displayed the Garmin Connect website	Must have	2
E2 - Data Synchronisation	As an athlete, I am able to sync the data from the watch to the website when I finish the workout so that I can govern my exercise data easily.				
	Athlete	#05	• Able to choose the option to sync and upload their data from Garmin server to the CoachingMate server. 1. Ensure that the API call from the Garmin server into the Backend (Java) 2. Option screen of the syncronisation to appear and for the user to accept to bind the watch 3. Able to choose which data files for the CoachingMate to retrieve 4. Ensure that the data files that can be read by the Backend (Java)	Must have	12
	Athlete	#06	• Able to upload the data consistently from the Garmin server on the CoachingMate website. 1. Create a MongoDB server in the cloud to store the data required 2. Ensure that the MongoDB server is able to Save data from Backend (Java) 3. Data retrieved in the Backend must be in proper format to ensure that data is properly stored in MongoDB server 4. Ensure that the Backend (Java) can retrieve data from the MongoDB server	Must have	10
E3 - Data Visualisation **(out of scope)**	As an athlete, I want to view my sports data on the website, so that it is easier for me to see what degree I achieved and dig into details to check my performance on different targets.				

Athlete	#09	• I want to view basic sports data on my activity, such as the duration of the activity, calories burn, distance, based on the specific activity transferred. 1. Ensure the User input is able to request data with the Frontend (React) 2. Backend (Java) is able to retrieve the data required from the MongoDB (Database) 3. Backend (Java) is able to produce output into the Frontend (React) 4. Frontend is able to display the necessary data from the Backend	Must have	4
Athlete	#10	• I want to be able to see a summary of my activities in a form of a dashboard to be able to see clearly what I have achieved overall 1. Create a statistic graph over time to show calories burnt over each day 2. Create a relational graph of two statistics to show relational data between them	Could have	2
As an athlete, I am able to see the specific details of each of my individual **Swim** activities				
<None>	#11	A backend retrieve controller (getSwimmingActivityByAccessToken) is to be created in the backend database for easier retrieval	Could have	3
Athlete	#12	I can see my **Time** taken for each of my Swim activities	Could have	0.5
Athlete	#13	I can see my **Distance** taken for my Swim activities	Could have	0.5
Athlete	#14	I can see my **Average Speed** for my Swim activities	Could have	0.5
Athlete	#15	I can see my **Calories** consumed for my Swim activities	Could have	0.5
Athlete	#16	I can see my **Pace** in time per 100m for my Swim activities	Could have	0.5
As an athlete, I am able to see the specific details of each of my individual **Cycling** activities				
<None>	#17	A backend retrieve controller (getCyclingActivityByAccessToken) is to be created in the backend database for easier data retrieval	Could have	3
Athlete	#18	I can see my **Time** taken for each of my Cycling activities	Could have	0.5
Athlete	#19	I can see my **Distance** taken for my Cycling activities	Could have	0.5
Athlete	#20	I can see my **Average Speed** for my Cycling activities	Could have	0.5
Athlete	#21	I can see my **Calories** consumed for my Cycling activities	Could have	0.5
Athlete	#22	I can see my **Average Cadence** for my Cycling activities	Could have	0.5
Athlete	#23	I can see my **Average Heart Rate** for my Cycling activities	Could have	0.5
Athlete	#24	I can see my **Elevation** for my Cycling activities	Could have	0.5
As an athlete, I am able to see the specific details of each of my individual **Run** activities				
<None>	#25	A backend retrieve controller (getRunningActivityByAccessToken) is to be created in the backend database for easier data retrieval	Could have	0.5
Athlete	#26	I can see my **Time** taken for each of my Run activities	Could have	0.5
Athlete	#27	I can see my **Distance** taken for each of my Run activities	Could have	0.5
Athlete	#28	I can see my **Average Speed** for my Run activities	Could have	0.5
Athlete	#29	I can see my **Calories** for my Run activities	Could have	0.5
Athlete	#30	I can see my **Pace in time per km** for my Run activities	Could have	0.5
Athlete	#31	I can see my **Average Cadence** for my Run activities	Could have	0.5
Athlete	#32	I can see my **Average Heart Rate** for my Run activities	Could have	0.5
Athlete	#33	I can see my **Elevation** for my Run activities	Could have	0.5

Out-scope

Epics	User	User Stories	MoSCoW Priority	Story Points(1 hour/pt)
A5	Athlete	As an athlete, I want to be able to know the intricate exercise statistics details of my exercise history that are displayed in a visualization way.	Must have	10
A6	Athlete	As an athlete, I want to know how many calories I used during each exercise so that I can know and control the exercise plan. • The website can automatically calculate the calories consumed, based on corresponding sports data.	Should have	5
A7	Athlete	As an athlete, I want to view my exercise history in a visualization way, so that I can get to know my status in an easily accessible way.	Should have	15

C2	Coach	As a coach, I want to be able to see the heat map of the athletes when they are running/cycling so that I can better manage their training.	Should have	8
C3	Coach	As a coach, I want to compare the performance of my athletes in the specific training period, so that I can track the performance of each athlete.	Should have	5
C4	Coach	As a coach, I want to be able to see the distance of athletes when they run/bike/swim so that I can better manage their training. • The website can extract the distance data from the database, which corresponds to the workout timezone.	Must have	10
C5	Coach	As a coach, I want to be able to see an athlete's heart rate, exercise time, and calorie burn as they work out so that I can better manage their training.	Should have	10
C6	Coach	As a coach, I want to be able to see all my athlete's performance data in a collective view so that it is easier to check the athletes' status.	Should have	5
M3	Manager	As a manager, I want to see the status of all the coaches and athletes so that I can track the athletes' feedback and assess whether the coaches are dutiful.	Must have	5
M4	Manager	As a manager, I want the product to be scalable and robust so that historical data can be inherited when users use different types of smart products.	Should have	10
M5	Manager	As a manager, I want the software to support financial and human resource management so that I only need to use other software.	Should have	10

Version 2.1

In-scope

Epics	User	User Story ID	User Stories	MoSCoW Priority	Story Points(2 hour/pt)
E1- Account Administration Features	As an athlete, I am able to create an account, so that I can view and edit my profile and devices.				
	Athlete	#01	• The user can create the account, set the password and username. 1. Able to type into the username field 2. Able to type into the password field 3. User data is able to be stored into the backend database	Must have	3
	Athlete	#02	• The user can reset the password, in case the password is forgotten. 1. Enable to change password link 2. Password is updated in the backend database	Must have	4
	Athlete	#03	• The user account can bind with any sports watch. 1. The user account is able to have the option to sync with the Garmin server 2. Create the API required for the Garmin server data to sync with the CoachingMate server data 3. Create the API link from the Garmin server linking with the Backend (Java)	Must have	4
	Athlete	#04	• The user can unbind with any sports watch that they have currently binded with. Option screen of able to unbind the syncronisation to be seen displayed the Garmin Connect website	Must have	2
E2 - Data Synchronisation	As an athlete, I am able to sync the data from the watch to the website when I finish the workout so that I can govern my exercise data easily.				
	Athlete	#05	• Able to choose the option to sync and upload their data from Garmin server to the CoachingMate server. 1. Ensure that the API call from the Garmin server into the Backend (Java) 2. Option screen of the syncronisation to appear and for the user to accept to bind the watch 3. Able to choose which data files for the CoachingMate to retrieve 4. Ensure that the data files that can be read by the Backend (Java)	Must have	12

	Athlete	#06	• Able to upload the data consistently from the Garmin server on the CoachingMate website. 1. Create a MongoDB server in the cloud to store the data required 2. Ensure that the MongoDB server is able to Save data from Backend (Java) 3. Data retrieved in the Backend must be in proper format to ensure that data is properly stored in MongoDB server 4. Ensure that the Backend (Java) can retrieve data from the MongoDB server	Must have	10
	As an athlete, I want to get a notification when the transition is completed.				
	Athlete	#07	• When the synchronization succeeds, it should show the latest date of the data that has been synchronised.	Should have	3
E3 - Data Visualisation	As an athlete, I want to view my sports data on the website, so that it is easier for me to see what degree I achieved and dig into details to check my performance on different targets.				
	Athlete	#09	• I want to view basic sports data on my activity, such as the duration of the activity, calories burn, distance, based on the specific activity transferred. 1. Ensure the User input is able to request data with the Frontend (React) 2. Backend (Java) is able to retrieve the data required from the MongoDB (Database) 3. Backend (Java) is able to produce output into the Frontend (React) 4. Frontend is able to display the necessary data from the Backend	Must have	8
	Athlete	#10	• I want to be able to see a summary of my activities in a form of a dashboard to be able to see clearly what I have achieved overall 1. Create a statistic graph over time to show calories burnt over each day 2. Create a relational graph of two statistics to show relational data between them	Could have	4

Out-scope

Epics	User	User Stories	MoSCoW Priority	Story Points(1 hour/pt)
A5	Athlete	As an athlete, I want to be able to know the intricate exercise statistics details of my exercise history that are displayed in a visualization way.	Must have	10
A6	Athlete	As an athlete, I want to know how many calories I used during each exercise so that I can know and control the exercise plan. • The website can automatically calculate the calories consumed, based on corresponding sports data.	Should have	5
A7	Athlete	As an athlete, I want to view my exercise history in a visualization way, so that I can get to know my status in an easily accessible way.	Should have	15
C2	Coach	As a coach, I want to be able to see the heat map of the athletes when they are running/cycling so that I can better manage their training.	Should have	8
C3	Coach	As a coach, I want to compare the performance of my athletes in the specific training period, so that I can track the performance of each athlete.	Should have	5
C4	Coach	As a coach, I want to be able to see the distance of athletes when they run/bike/swim so that I can better manage their training. • The website can extract the distance data from the database, which corresponds to the workout timezone.	Must have	10
C5	Coach	As a coach, I want to be able to see an athlete's heart rate, exercise time, and calorie burn as they work out so that I can better manage their training.	Should have	10
C6	Coach	As a coach, I want to be able to see all my athlete's performance data in a collective view so that it is easier to check the athletes' status.	Should have	5
M3	Manager	As a manager, I want to see the status of all the coaches and athletes so that I can track the athletes' feedback and assess whether the coaches are dutiful.	Must have	5
M4	Manager	As a manager, I want the product to be scalable and robust so that historical data can be inherited when users use different types of smart products.	Should have	10
M5	Manager	As a manager, I want the software to support financial and human resource management so that I only need to use other software.	Should have	10

Version 2.0

In-scope

Epics	User	User Story ID	User Stories	MoSCoW Priority	Story Points(1 hour/pt)
E1- Account Administration Features	As an athlete, I am able to create an account, so that I can view and edit my profile and devices.				
	Athlete	01	• The user can create the account, set the password and username.	Must have	8
	Athlete	02	• The user can reset the password, in case the password is forgotten.	Must have	8
	Athlete	03	• The user account can bind with any sports watch.	Must have	6
	Athlete	04	• The user can unbind with any sports watch that they have currently binded with.	Must have	6
E2 - Data Synchronisation	As an athlete, I am able to sync the data from the watch to the website when I finish the workout so that I can govern my exercise data easily.				
	Athlete	05	• Able to choose the option to sync and upload their data from Garmin server to the CoachingMate server.	Must have	8
	Athlete	06	• Able to upload the data consistently from the Garmin server on the CoachingMate website.	Must have	5
	As an athlete, I want to get a notification when the transition is completed otherwise know the fault type.				
	Athlete	07	• When the synchronization succeeds, it should show the latest date of the data that has been synchronised.	Should have	7
	Athlete	08	• When the synchronization fault, it should show a notification that the API is not synchroised.	Should have	7
E3 - Data Visualisation	As an athlete, I want to view my sports data on the website, so that it is easier for me to see what degree I achieved and dig into details to check my performance on different targets.				
	Athlete	09	• I want to view basic sports data on my activity, such as the duration of the activity, calories burn, distance, based on the specific activity transferred.	Must have	10
	Athlete	10	• I want to be able to see a summary of my activities in a form of a dashboard to be able to see clearly what I have achieved overall	Could have	10

Out-scope

Epics	User	User Stories	MoSCoW Priority	Story Points(1 hour/pt)
A5	Athlete	As an athlete, I want to be able to know the intricate exercise statistics details of my exercise history that are displayed in a visualization way.	Must have	10
A6	Athlete	As an athlete, I want to know how many calories I used during each exercise so that I can know and control the exercise plan. • The website can automatically calculate the calories consumed, based on corresponding sports data.	Should have	5
A7	Athlete	As an athlete, I want to view my exercise history in a visualization way, so that I can get to know my status in an easily accessible way.	Should have	15
C2	Coach	As a coach, I want to be able to see the heat map of the athletes when they are running/cycling so that I can better manage their training.	Should have	8
C3	Coach	As a coach, I want to compare the performance of my athletes in the specific training period, so that I can track the performance of each athlete.	Should have	5
C4	Coach	As a coach, I want to be able to see the distance of athletes when they run/bike/swim so that I can better manage their training. • The website can extract the distance data from the database, which corresponds to the workout timezone.	Must have	10

C5	Coach	As a coach, I want to be able to see an athlete's heart rate, exercise time, and calorie burn as they work out so that I can better manage their training.	Should have	10
C6	Coach	As a coach, I want to be able to see all my athlete's performance data in a collective view so that it is easier to check the athletes' status.	Should have	5
M3	Manager	As a manager, I want to see the status of all the coaches and athletes so that I can track the athletes' feedback and assess whether the coaches are dutiful.	Must have	5
M4	Manager	As a manager, I want the product to be scalable and robust so that historical data can be inherited when users use different types of smart products.	Should have	10
M5	Manager	As a manager, I want the software to support financial and human resource management so that I only need to use other software.	Should have	10

Version 1.0

In-scope

Epics	User	User Stories	MoSCoW Priority	Story Points(1 hour/pt)
A1	Athlete	As an athlete, I am able to create an account, so that I can view and edit my profile and devices. • The user can create the account, set the password and username. • The user can reset the password, in case the password is forgotten. • The user account can bind with multiple sports watch. • The user can unbind with any sports watch they currently have.	Must have	30
A2	Athlete	As an athlete, I am able to sync the data from the watch to the website when I finish the workout so that I can govern my exercise data easily. • The app should have a button, that can synchronic the data from the phone to the user profile. • The watch has backup, in case the time when the connection is unavailable. • The watch can automatically resume the synchronization if wireless is available again.	Must have	30
A3	Athlete	As an athlete, I want to get a notification when the transition is completed otherwise know the fault type. • When the synchronization succeeds, there will be a notification shown on the phone. • When the synchronization fault, the fault type can be shown on the phone.	Should have	6
A4	Athlete	As an athlete, I want to view my sports data on my website profile, so that it is easier for me to see what degree I achieved and dig into details to check my performance on different targets.	Must have	24
C1	Coach	As a coach, I want to see the workout data of my athletes, so that I can monitor the athlete statistic.	Should have	12
M1	Manager	As a manager, I want to regulate the Garmin watches data type as JSON, so that allows more types of sports watch to be extended. • The features can be extracted to different queries. • The format of data can be unified when developed.	Should have	10

M2	Manager	As a manager, I want to assign different panel views for users based on user type, so that users can get the information they need.
		• The database has three user types:
		  • The normal athlete can only view his workout history data.
		  • The coach can only view a set of athlete's data from his workout group.
		  • The manager can view all-athletes data, and adjust the content of data, i.e change, move, delete.
		• The database should allow the queries option and return without failure to request.
			Should have	28

Out-scope

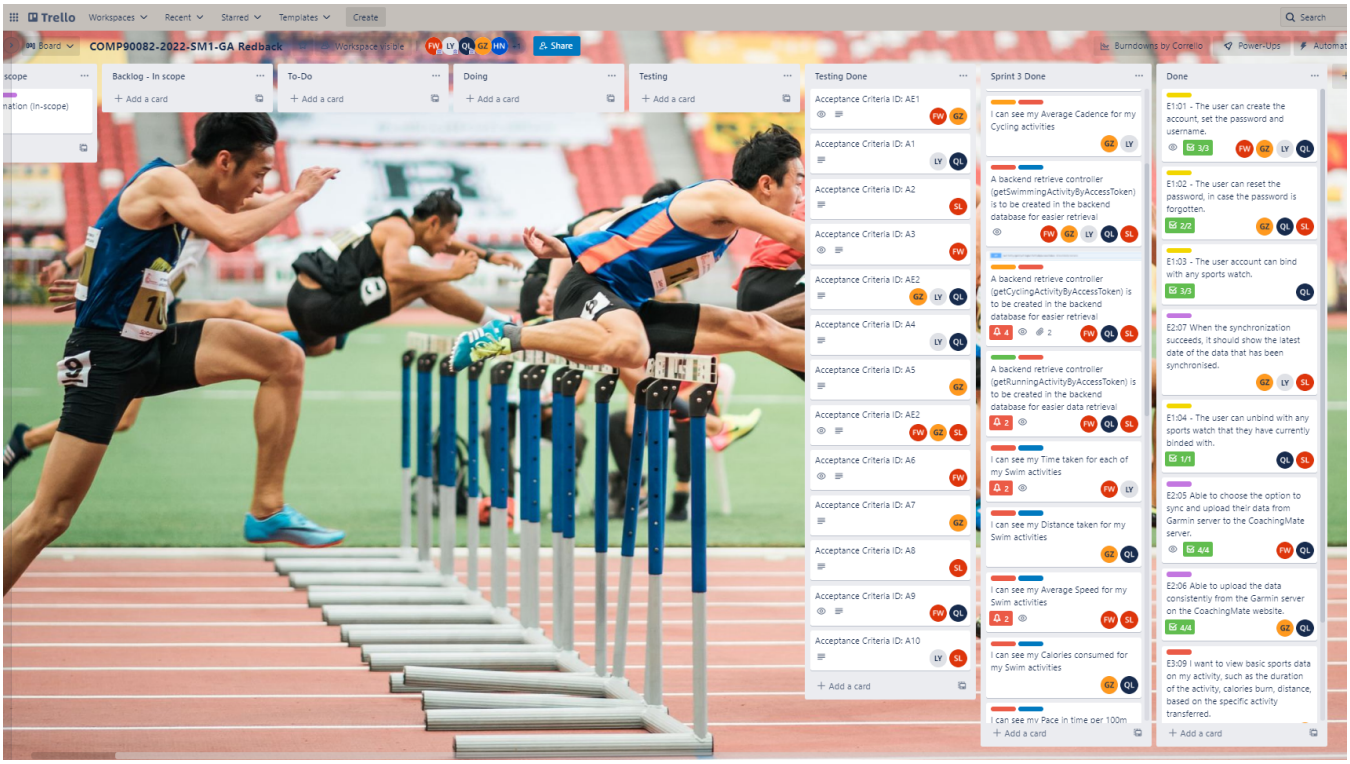
Epics	User	User Stories	MoSCoW Priority	Story Points(1 hour/pt)
A5	Athlete	As an athlete, I want to be able to know the intricate exercise statistics details of my exercise history that are displayed in a visualization way.	Must have	10
A6	Athlete	As an athlete, I want to know how many calories I used during each exercise so that I can know and control the exercise plan. • The website can automatically calculate the calories consumed, based on corresponding sports data.	Should have	5
A7	Athlete	As an athlete, I want to view my exercise history in a visualization way, so that I can get to know my status in an easily accessible way.	Should have	15

C2	Coach	As a coach, I want to be able to see the heat map of the athletes when they are running/cycling so that I can better manage their training.	Should have	8
C3	Coach	As a coach, I want to compare the performance of my athletes in the specific training period, so that I can track the performance of each athlete.	Should have	5
C4	Coach	As a coach, I want to be able to see the distance of athletes when they run/bike/swim so that I can better manage their training. • The website can extract the distance data from the database, which corresponds to the workout timezone.	Must have	10
C5	Coach	As a coach, I want to be able to see an athlete's heart rate, exercise time, and calorie burn as they work out so that I can better manage their training.	Should have	10
C6	Coach	As a coach, I want to be able to see all my athlete's performance data in a collective view so that it is easier to check the athletes' status.	Should have	5
M3	Manager	As a manager, I want to see the status of all the coaches and athletes so that I can track the athletes' feedback and assess whether the coaches are dutiful.	Must have	5
M4	Manager	As a manager, I want the product to be scalable and robust so that historical data can be inherited when users use different types of smart products.	Should have	10
M5	Manager	As a manager, I want the software to support financial and human resource management so that I only need to use other software.	Should have	10

Plan (Trello) - Redback

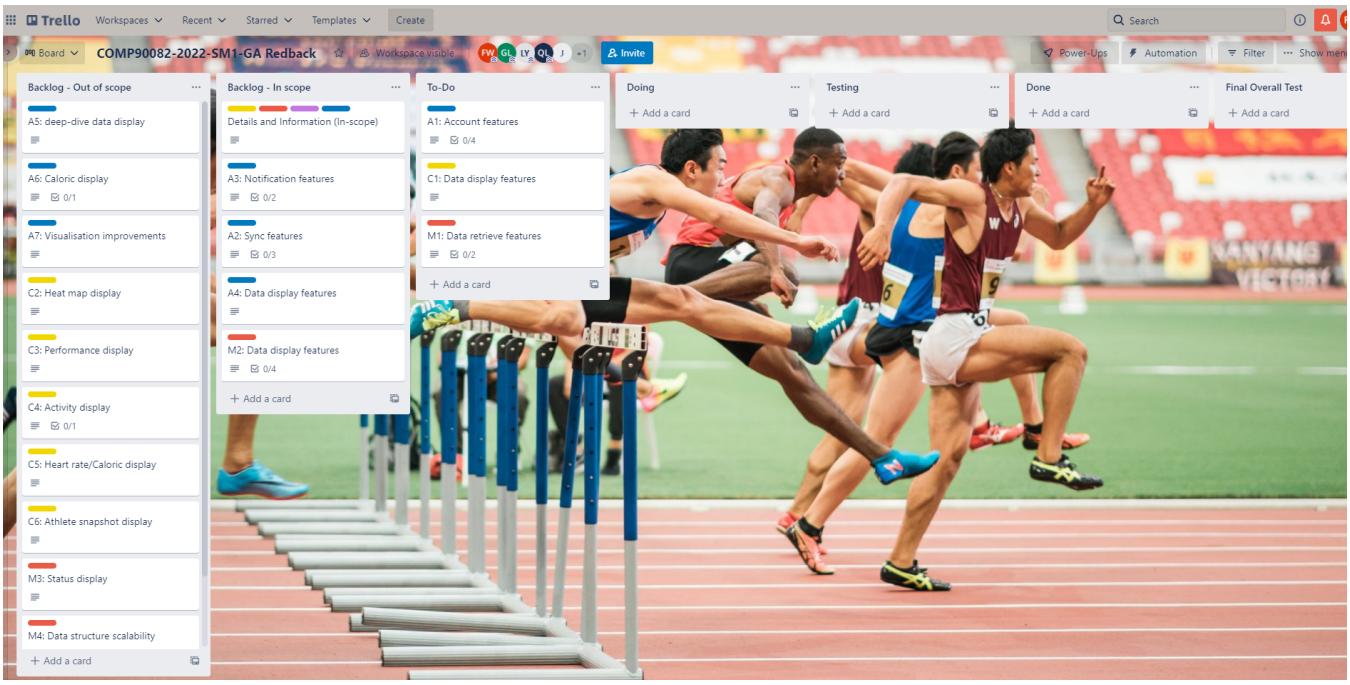
This is the Trello page of team Redback: https://trello.com/b/NcHpFGZF/comp90082-2022-sm1-ga-redback

Sprint3:



Sprint2:

Sprint1:


Trello
Workspaces
Recent
Starred
Templates
Create
Search
Board
COMP90082-2022-SM1-GA Redback
Workspace visible
Invite
Power-Ups
Automation
Filter
Backlog - Out of scope
A5: deep-dive data display
A6: Caloric display
0/1
A7: Visualisation improvements
C2: Heat map display
C3: Performance display
C4: Activity display
0/1
C5: Heart rate/Caloric display
C6: Athlete snapshot display
M3: Status display
M4: Data structure scalability
Add a card
Backlog - In scope
Details and Information (In-scope)
A3: Notification features
0/2
A2: Sync features
0/3
A4: Data display features
M2: Data display features
0/4
Add a card
To-Do
A1: Account features
0/4
C1: Data display features
M1: Data retrieve features
0/2
Add a card
Doing
Add a card
Testing
Add a card
Done
Add a card
Final Overall Test
Add a card

Development

Technical Details

A. Spring framework

The Spring Framework is an application framework and inversion of the control container for the Java platform. The framework's core features can be used by any Java application, but there are extensions for building web applications on top of the Java EE (Enterprise Edition) platform.

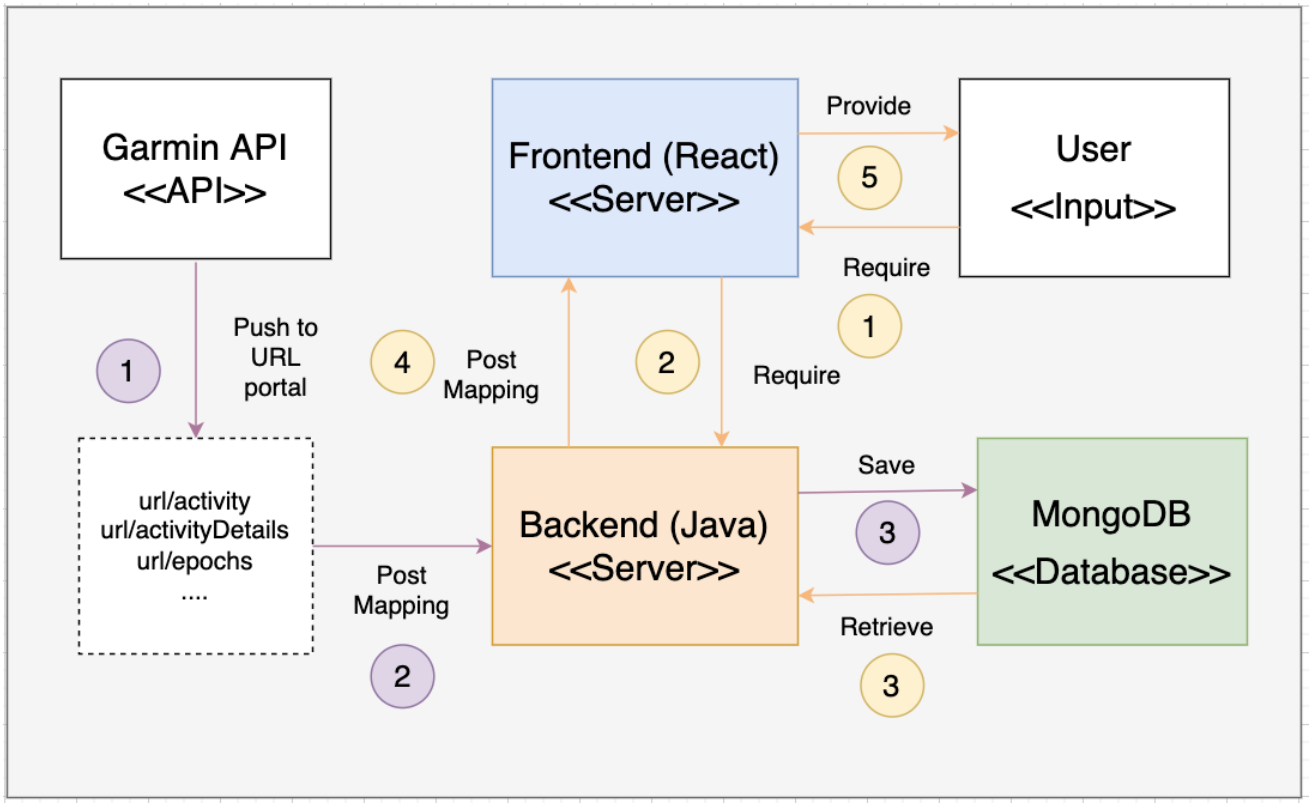
More details:

https://www.baeldung.com/jsf-spring-boot-controller-service-dao

B. class in our project

- **coachingmateanalytics.coachingmate.controller: Front controller layer**
- **coachingmateanalytics.coachingmate.entity: Data entity class**
- **coachingmateanalytics.coachingmate.dao : Data interface access layer**
- **coachingmateanalytics.coachingmate.service : Data service interface layer**
- **coachingmateanalytics.coachingmate.service.serviceImpl : Data service interface implementation layer**
- **coachingmateanalytics.coachingmate.utils : Tool library**
- **coachingmateanalytics.coachingmate.intercepter : used to set the response header for all request**
- **resources/application.yml : Project profile resources/static/ : Static resource directory**

C. Project Flow



Information saving (Purple flow):
1. GarminAPI publishes the data to the URL portal.
2. Then the server backend will then post the data.
2. The server backend sends the posted data and saves it to the MongoDB database in different API data formats according to the categories of the data(See in 'API data format' below).

Information retrieval (Yellow flow):
1. The user, after given data permission through Garmin authentication, can then request data through the frontend.
2. Frontend will send the data request to the backend
3. The server backend will retrieve the requested data in MongoDB.
4. The server backend will do port mapping and send the data to the frontend
5. The server frontend can then get the requested data back to the user.

D. API data format

activity data: reflect the summary of one activity acquire from Garmin bind


```
_id: ObjectId('625becc19b5a832fc5b7c1ba')
durationInSeconds: 10
averageSpeedInMetersPerSecond: 0.233
averageHeartRateInBeatsPerMinute: 92
distanceInMeters: 2.42
activityName: "Running"
userId: "5073e79f-df60-45dc-92f2-bc0ef300f1f0"
deviceName: "forerunner935"
averagePaceInMinutesPerKilometer: 71.53076
activityId: 8654408415
startTimeInSeconds: 1650191523
userAccessToken: "227a7c55-590d-498f-97ad-fb6ba3cb259f"
startTimeOffsetInSeconds: 36000
maxPaceInMinutesPerKilometer: 6.8956003
maxHeartRateInBeatsPerMinute: 95
summaryId: "8654408415"
maxSpeedInMetersPerSecond: 2.417
activityType: "RUNNING"
```


activity details data example: reflect the specification of one activity acquire from Garmin bind, sampling per second


```
_id: ObjectId('625bfb021225d51f33796559')
summary: Object
  durationInSeconds: 10
  averageSpeedInMetersPerSecond: 1.362
  averageHeartRateInBeatsPerMinute: 88
  distanceInMeters: 13.39
  activityName: "Running"
  deviceName: "forerunner935"
  steps: 4
  averageRunCadenceInStepsPerMinute: 25.09375
  averagePaceInMinutesPerKilometer: 12.236906
  activityId: 8654812566
  startTimeInSeconds: 1650195177
  startTimeOffsetInSeconds: 36000
  maxPaceInMinutesPerKilometer: 4.8676014
  maxHeartRateInBeatsPerMinute: 93
  maxRunCadenceInStepsPerMinute: 138
  maxSpeedInMetersPerSecond: 3.424
  activityType: "RUNNING"
activityId: 8654812566
userAccessToken: "227a7c55-590d-498f-97ad-fb6ba3cb259f"
summaryId: "8654812566-detail"
laps: Array
userId: "5073e79f-df60-45dc-92f2-bc0ef300f1f0"
samples: Array
  0: Object
    startTimeInSeconds: 1650195177
    timerDurationInSeconds: 0
    movingDurationInSeconds: 0
    heartRate: 84
    elevationInMeters: 33.20000076293945
    speedMetersPerSecond: 0
    airTemperatureCelcius: 32
    totalDistanceInMeters: 0
  1: Object
    startTimeInSeconds: 1650195178
    timerDurationInSeconds: 1
    movingDurationInSeconds: 0
    heartRate: 86
    elevationInMeters: 33.20000076293945
    speedMetersPerSecond: 0
    clockDurationInSeconds: 1
    stepsPerMinute: 0
    airTemperatureCelcius: 32
    totalDistanceInMeters: 0
  2: Object
  3: Object
  4: Object
```


epho data example: reflect calories active and sedentary consumption, sampling 15 minutes.

```
_id: ObjectId('625c184db2254248a0cc56fd')
activeKilocalories: 23
durationInSeconds: 900
distanceInMeters: 324.31
userId: "5073e79f-df60-45dc-92f2-bc0ef300f1f0"
steps: 346
activeTimeInSeconds: 420
meanMotionIntensity: 2
intensity: "ACTIVE"
startTimeInSeconds: 1650026700
userAccessToken: "227a7c55-590d-498f-97ad-fb6ba3cb259f"
startTimeOffsetInSeconds: 36000
maxMotionIntensity: 2
summaryId: "s46ed0f2-625968cc-6"
activityType: "WALKING"
met: 3.0090992
```

```
_id: ObjectId('625c184db2254248a0cc56fe')
activeKilocalories: 0
durationInSeconds: 900
distanceInMeters: 0
userId: "5073e79f-df60-45dc-92f2-bc0ef300f1f0"
steps: 0
activeTimeInSeconds: 480
meanMotionIntensity: 0
intensity: "SEDENTARY"
startTimeInSeconds: 1650026700
userAccessToken: "227a7c55-590d-498f-97ad-fb6ba3cb259f"
startTimeOffsetInSeconds: 36000
maxMotionIntensity: 0
summaryId: "s46ed0f2-625968cc-8"
```


E. Garmin authentication

Step 1: Acquire Unauthorized Request Token
For each user, the first step in getting user-approved data is acquiring a request token and token secret. This request token does not have the ability to access data, nor is it user-specific yet.

Enter your consumer secret below to generate the signature for getting the Request Token.

Step 2: Authorization of the Request Token
The request token from step 1 needs to be approved by the user. This is done by redirecting to Garmin Connect where the user logs in and grants access to the data. Garmin Connect will redirect the user to the given callback URL and append the OAuth verifier.

Step 3: Acquire User Access Token
After the user authorized the data access, the Request Token needs to be exchanged for a User Access Token. The request needs to be signed with the Consumer Key, the Request Token from step 1 and the OAuth verifier from step 2
Swagger

Backend deployment

Development Environment Instructions

1. Install IDEA from this location.

2. Install maven from this location.

Reference blog: https://toolsqa.com/maven/how-to-install-maven-on-windows/

3. Install Mongodb from this location.

Reference blog: https://medium.com/@LondonAppBrewery/how-to-download-install-mongodb-on-windows-4ee4b3493514

4. Install git version control in IDEA

Reference video: https://www.youtube.com/watch?v=mM_drNdss4c

Installation and Setup Instructions

1. install JDK

Official tutorial for JDK installation

2. install maven

Official tutorial for Maven installation
take mac os for example

- download apache-maven-3.6.3-bin.tar.gz
- tar xzvf apache-maven-3.6.3-bin.tar.gz

- Alternatively, use your preferred archive extraction tool.
- Add the bin directory of the created directory apache-maven-3.6.3 to the PATH environment variable
- Confirm with mvn -v in a new shell. The result should look similar to

```
Apache Maven 3.6.3 (cecedd343002696d0abb50b32b541b8a6ba2883f)
Maven home: /opt/apache-maven-3.6.3
Java version: 1.8.0_45, vendor: Oracle Corporation
Java home: /Library/Java/JavaVirtualMachines/jdk1.8.0_45.jdk/Contents/Home/jre
Default locale: en_US, platform encoding: UTF-8
OS name: "mac os x", version: "10.8.5", arch: "x86_64", family: "mac"
```

3. install MongoDB

Official tutorial for mongoDB installation

```
brew install mongodb-community@4.4
```

4. install git environment

- Linux install

```
yum install git
```

- mac install

```
brew install git
```

- Before you can use Git for version management, you need to configure Git by telling git your **username** and your **email account**

```
//configuration
[root@localhost ~]# git config --global user.name flymegoc
[root@localhost ~]# git config --global user.email 343672271@qq.com

//Check the configuration
[root@localhost ~]# git config --list
user.name=flymegoc
user.email=343672271@qq.com
```

5. Run this project locally

Clone down this repository. You will need `maven` and `JDK1.8` installed globally on your machine.

`git clone https://github.com/agogear/coaching-mate.git`

Modify configuration file: application-dev.properties

- create your MongoDB database with {database name} manually in your local environment

```
spring.data.mongodb.database={database name}

spring.data.mongodb.port=27017
```

- export data from cloud database: cloud MongoDB database
- import data to your local MongoDB database

Installation:

```
cd coachingmate
mvn install
```

To Start Server:

```
java -jar ./target/coachingmate-0.0.1-SNAPSHOT.jar
```

To login App:

```
localhost:8080/login?username={username}&password={password}
```

Deployment guidelines

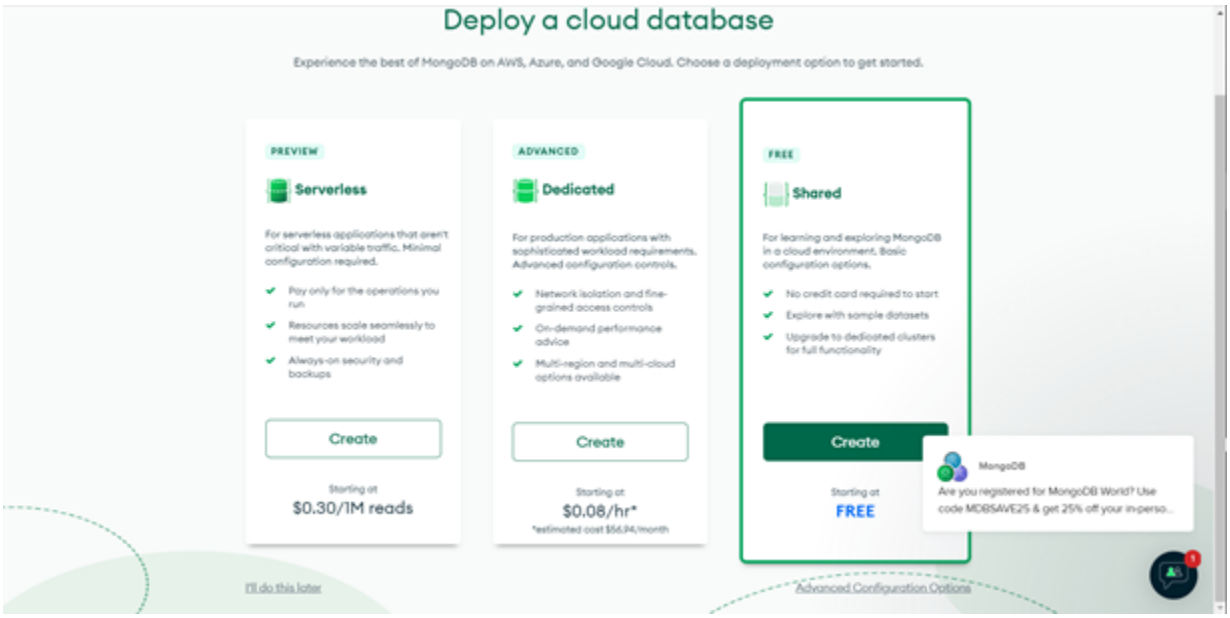
There are three places' variable should be replaced by yours

- MongoDB
- Garmin app API
- Garmin app api portal

MongoDB:

Using a Gmail account create a free cloud space, and Replace the URL.

1.


Deploy a cloud database
Experience the best of MongoDB on AWS, Azure, and Google Cloud. Choose a deployment option to get started.
PREVIEW
Serverless
For serverless applications that aren't critical with variable traffic. Minimal configuration required.
Pay only for the operations you run
Resources scale seamlessly to meet your workload
Always-on security and backups
Create
Starting at
$0.30/1M reads
ADVANCED
Dedicated
For production applications with sophisticated workload requirements. Advanced configuration controls.
Network isolation and fine-grained access controls
On-demand performance advice
Multi-region and multi-cloud options available
Create
Starting at
$0.08/hr*
*estimated cost $56.94/month
FREE
Shared
For learning and exploring MongoDB in a cloud environment. Basic configuration options.
No credit card required to start
Explore with sample datasets
Upgrade to dedicated clusters for full functionality
Create
Starting at
FREE
MongoDB
Are you registered for MongoDB World? Use code MDBSAVE25 & get 25% off your in-perso...
I'll do this later
Advanced Configuration Options

2.

CLUSTERS > CREATE A SHARED CLUSTER

Create a Shared Cluster

Welcome to MongoDB Atlas! We've recommended some of our most popular options, but feel free to customize your cluster to your needs. For more information, check our documentation.

PREVIEW Serverless | Dedicated | FREE Shared

For learning and exploring MongoDB in a sandbox environment. Basic configuration controls.

No credit card required to start. Upgrade to dedicated clusters for full functionality.
Explore with sample datasets. Limit of one free cluster per project.

Cloud Provider & Region

AWS, Sydney (ap-southeast-2)

aws | Google Cloud | Azure

★ Recommended region Paid tier region

NORTH AMERICA
- N. Virginia (us-east-1) ★
- Oregon (us-west-2) ★
- Ohio (us-east-2) ★
- N. California (us-west-1)
- Montreal (ca-central-1)

SOUTH AMERICA
- Sao Paulo (sa-east-1)

EUROPE
- Paris (eu-west-3) ★
- Stockholm (eu-north-1) ★
- Ireland (eu-west-1) ★
- Frankfurt (eu-central-1) ★
- London (eu-west-2) ★
- Milan (eu-south-1) ★

MIDDLE EAST
- Bahrain (me-south-1) ★

AFRICA
- Cape Town (af-south-1) ★

AUSTRALIA
- Sydney (ap-southeast-2) ★

ASIA
- Mumbai (ap-south-1)
- Tokyo (ap-northeast-1)
- Singapore (ap-southeast-1) ★
- Seoul (ap-northeast-2)
- Hong Kong (ap-east-1) ★
- Jakarta (ap-southeast-3) ★
- Osaka (ap-northeast-3) ★

Cluster Tier

M0 Sandbox (Shared RAM, 512 MB Storage)
Encrypted

Additional Settings

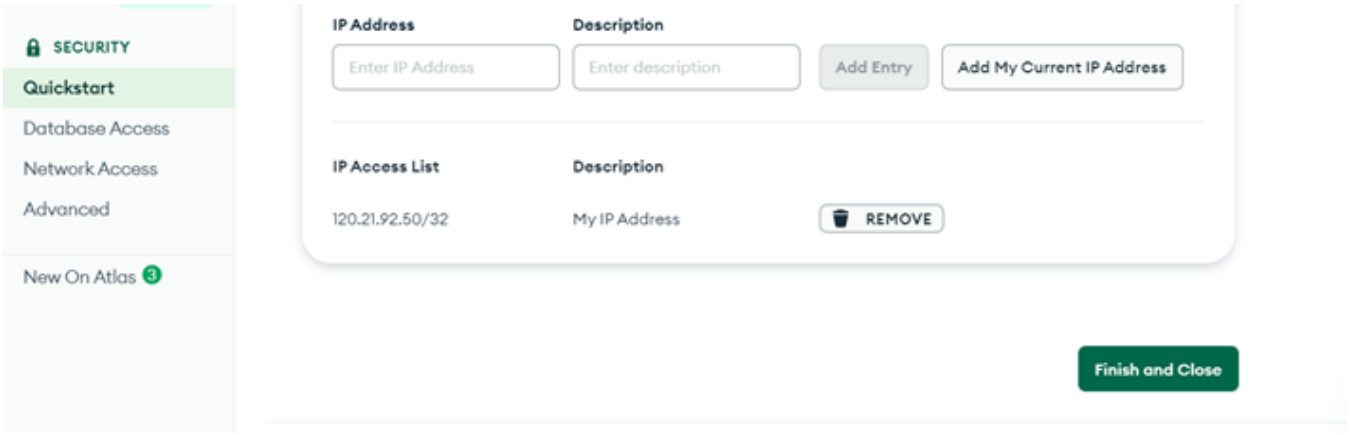
MongoDB 5.0, No Backup

FREE

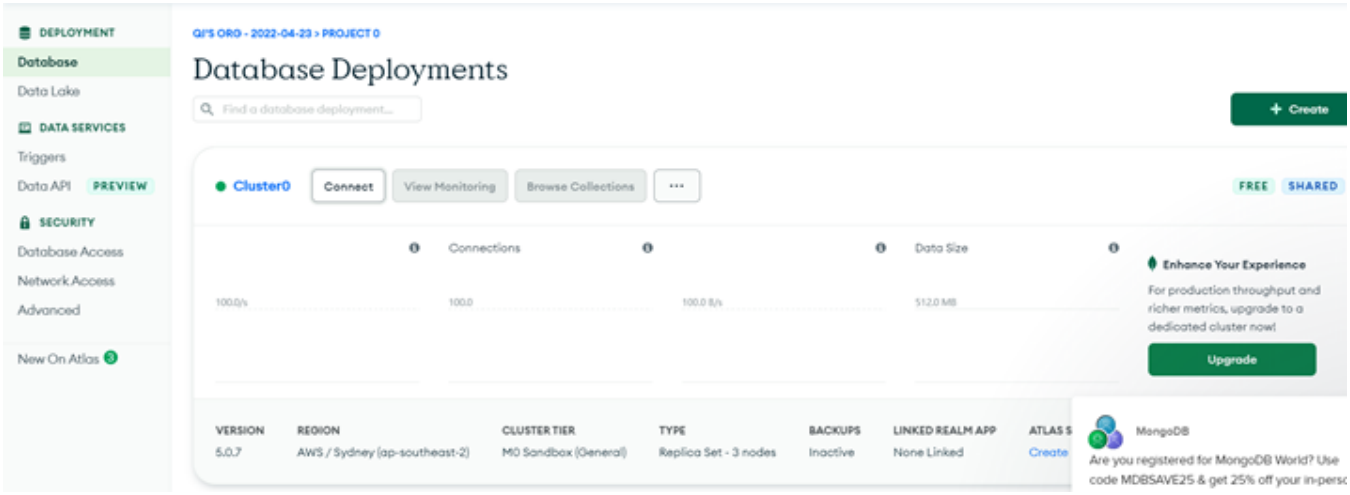
Free forever! Your M0 cluster is ideal for experimenting in a limited sandbox. You can upgrade to a production cluster anytime.

Back | Create Cluster

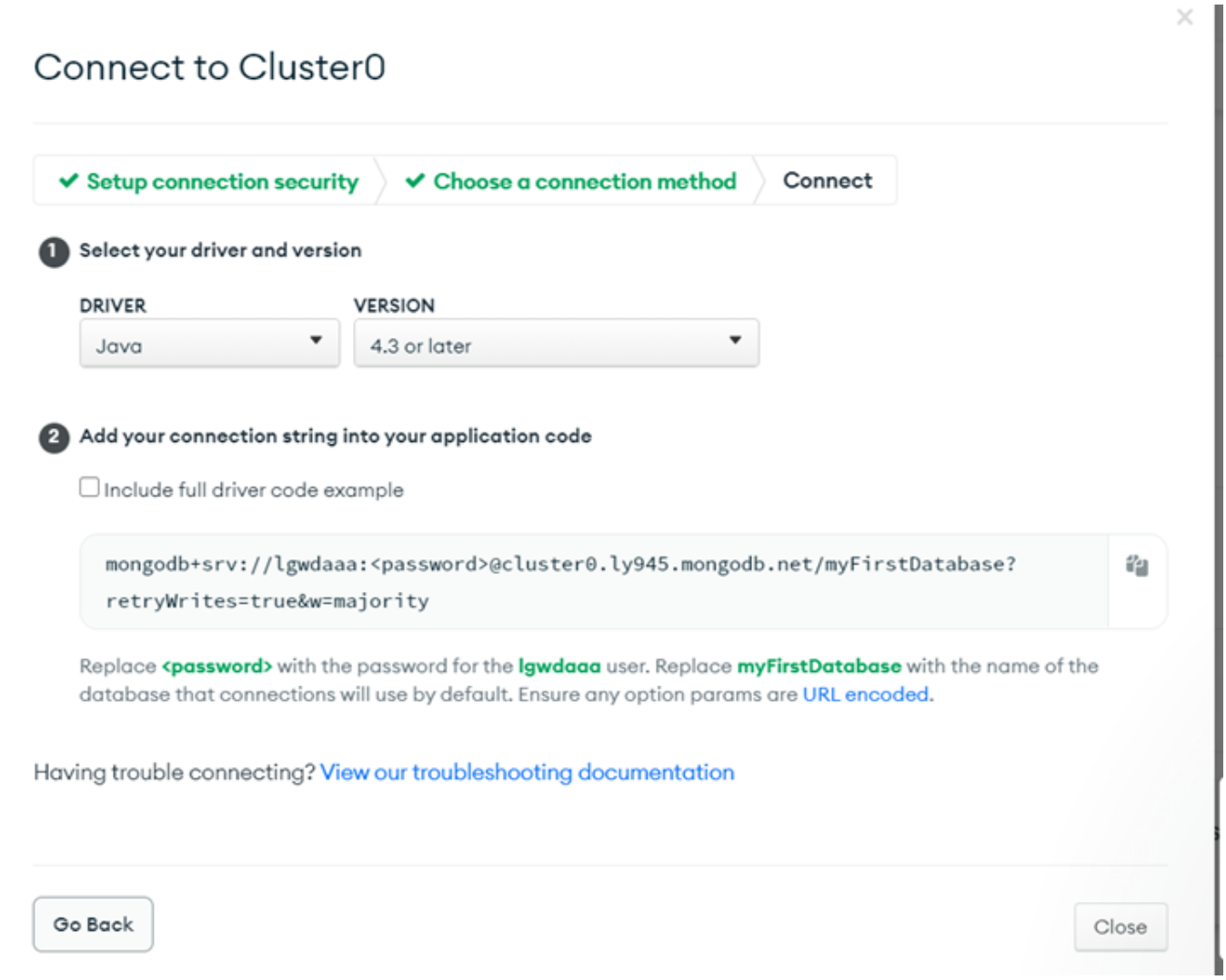
3.


SECURITY
Quickstart
Database Access
Network Access
Advanced
New On Atlas 3
IP Address
Description
Enter IP Address
Enter description
Add Entry
Add My Current IP Address
IP Access List
Description
120.21.92.50/32
My IP Address
REMOVE
Finish and Close

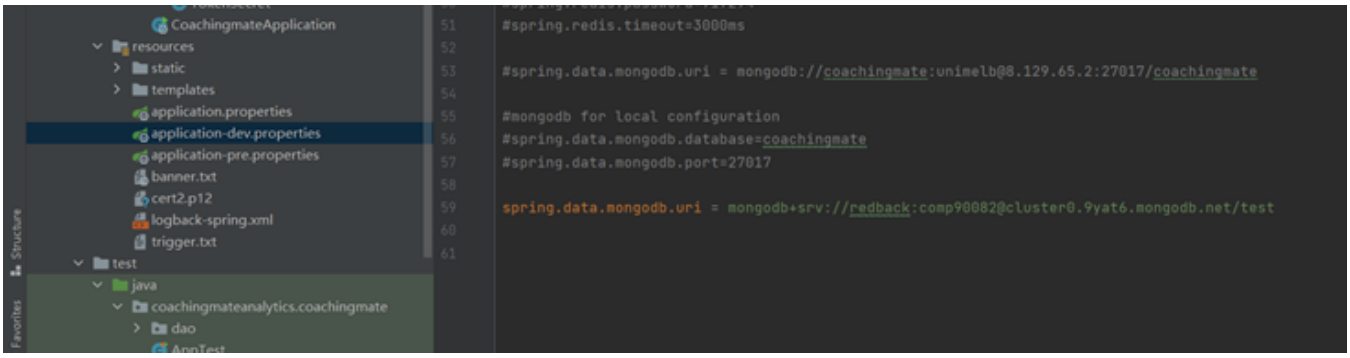
4.


DEPLOYMENT
Database
Data Lake
DATA SERVICES
Triggers
Data API PREVIEW
SECURITY
Database Access
Network Access
Advanced
New On Atlas 3
Database Deployments
Create
Cluster0
Connect
View Monitoring
Browse Collections
FREE
SHARED
Connections
Data Size
Enhance Your Experience
For production throughput and richer metrics, upgrade to a dedicated cluster now!
Upgrade
VERSION
REGION
CLUSTER TIER
TYPE
BACKUPS
LINKED REALM APP
5.0.7
AWS / Sydney (ap-southeast-2)
M0 Sandbox (General)
Replica Set - 3 nodes
Inactive
None Linked
MongoDB

5.



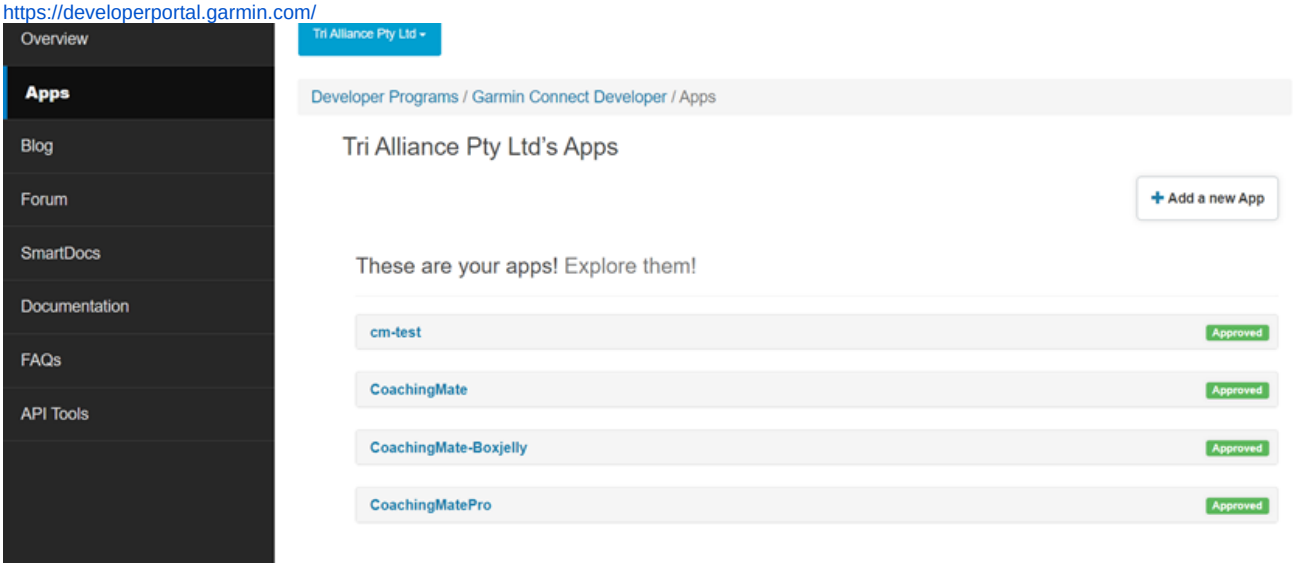
6. Replace the URL with your copy one



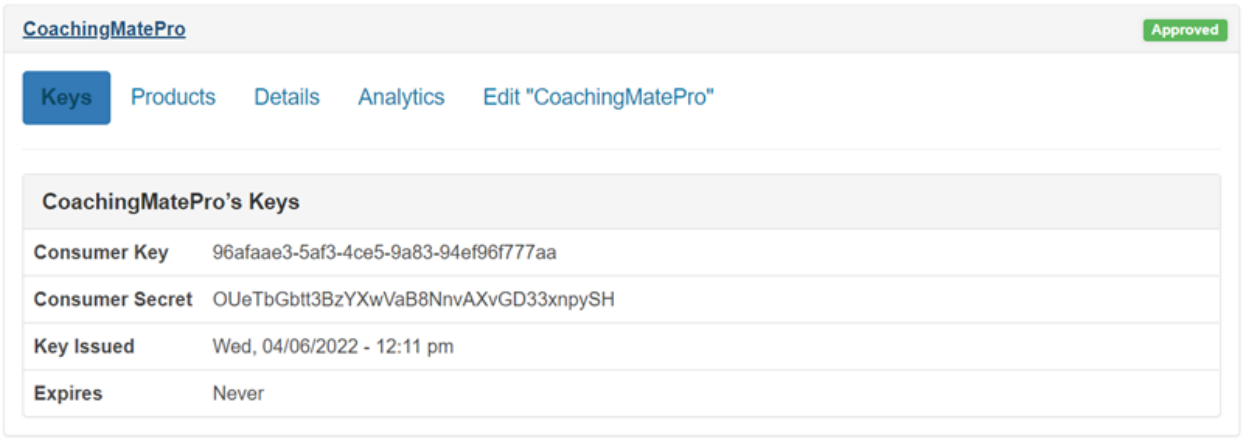
Garmin app API:

1. You need to create a new app in the developer portal of Garmin.

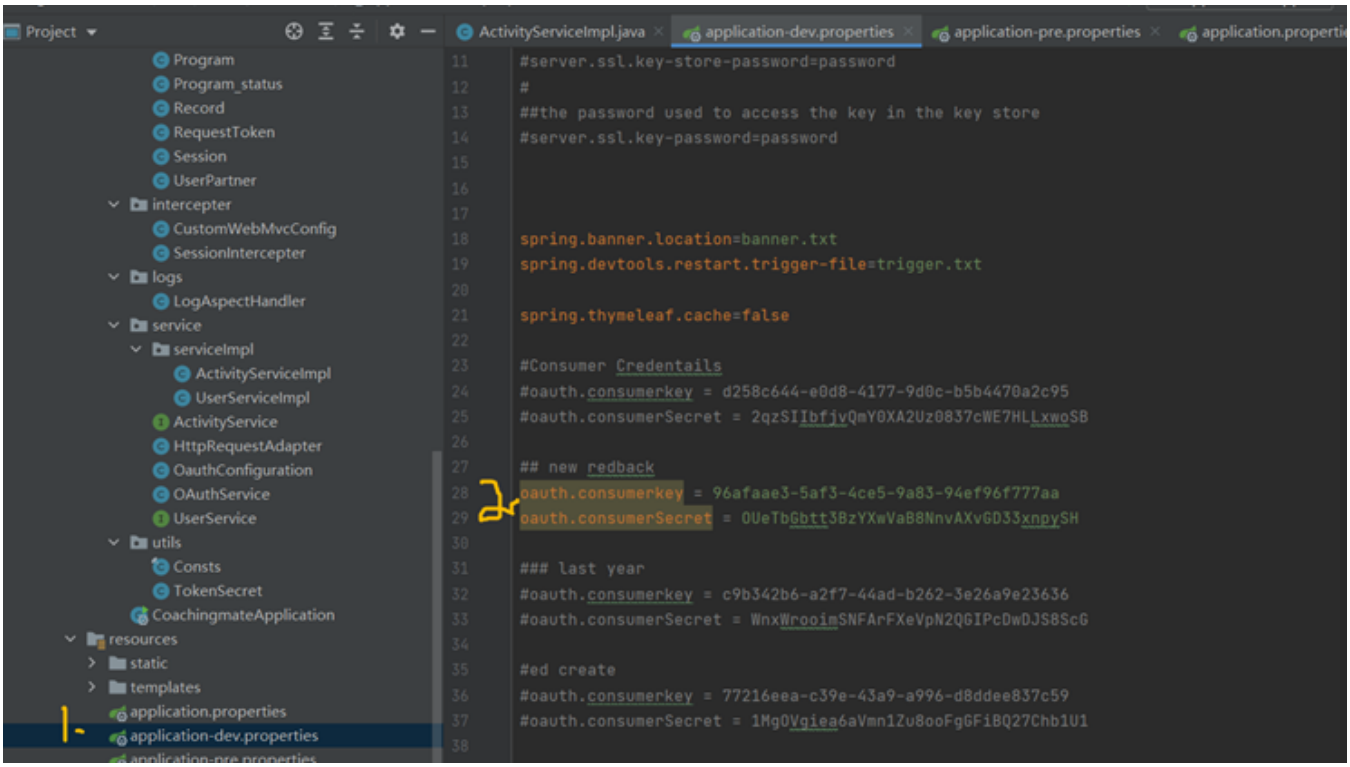
https://developerportal.garmin.com/



2. Then you need to write down the consumer key and consumerSececret



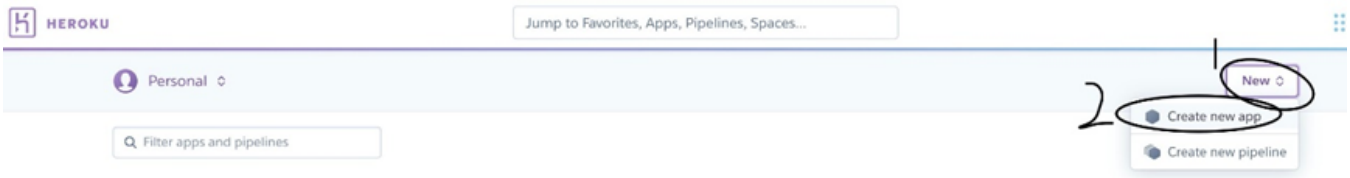
After that process, you own API key and secret. That info should be included in your apps, replaced those vars with yours.



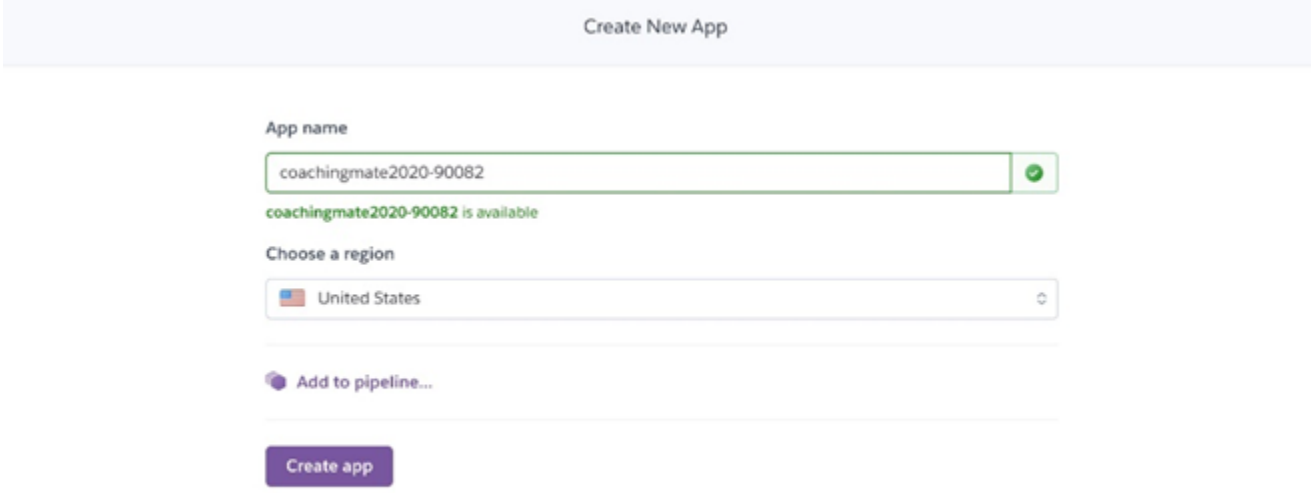
Garmin app api portal:

You need to upload your project to GitHub, and then connect the GitHub repository with Heroku.

1. create a new application on Heroku

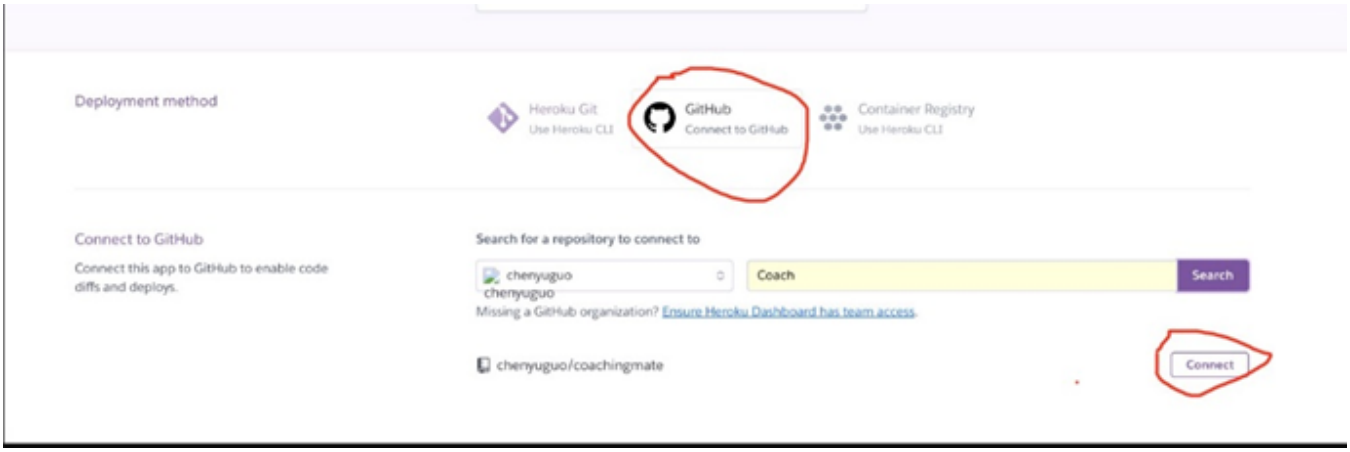


2. text our app name and region

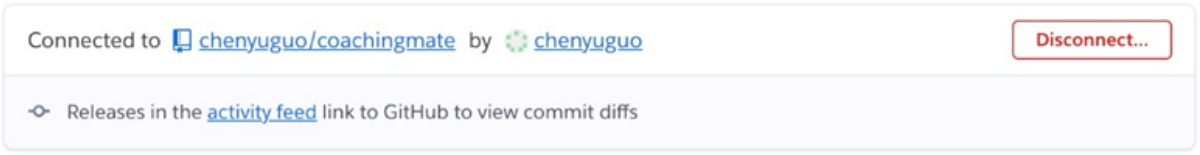


3. fork this repository to your own repository

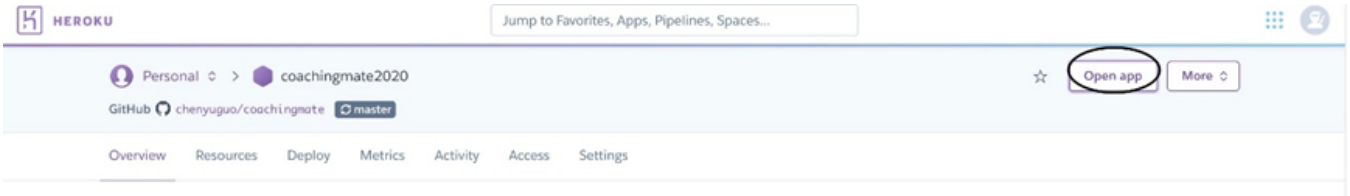
4. Select Github and find this forked project in your own repository



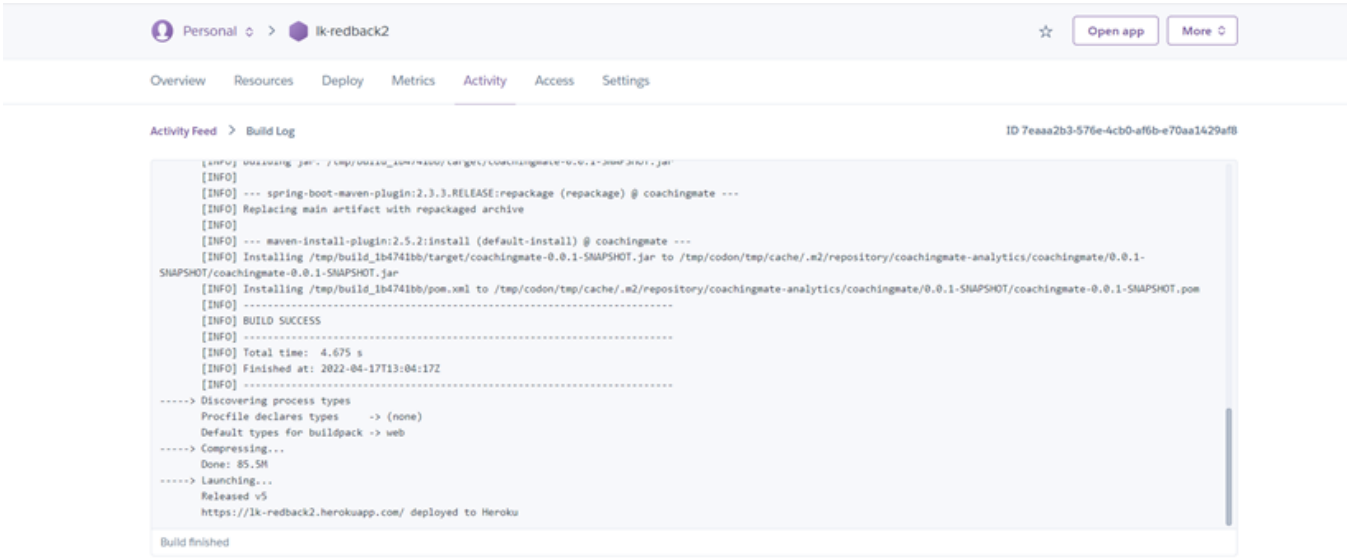
5. connect successfully



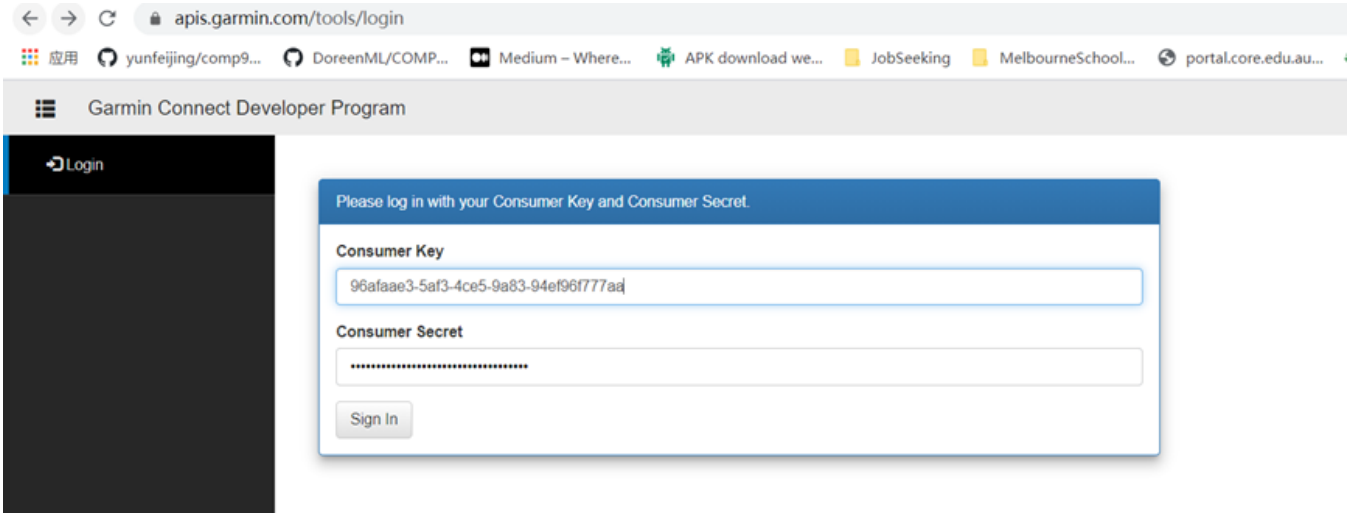
6. find our app and click the Open app button

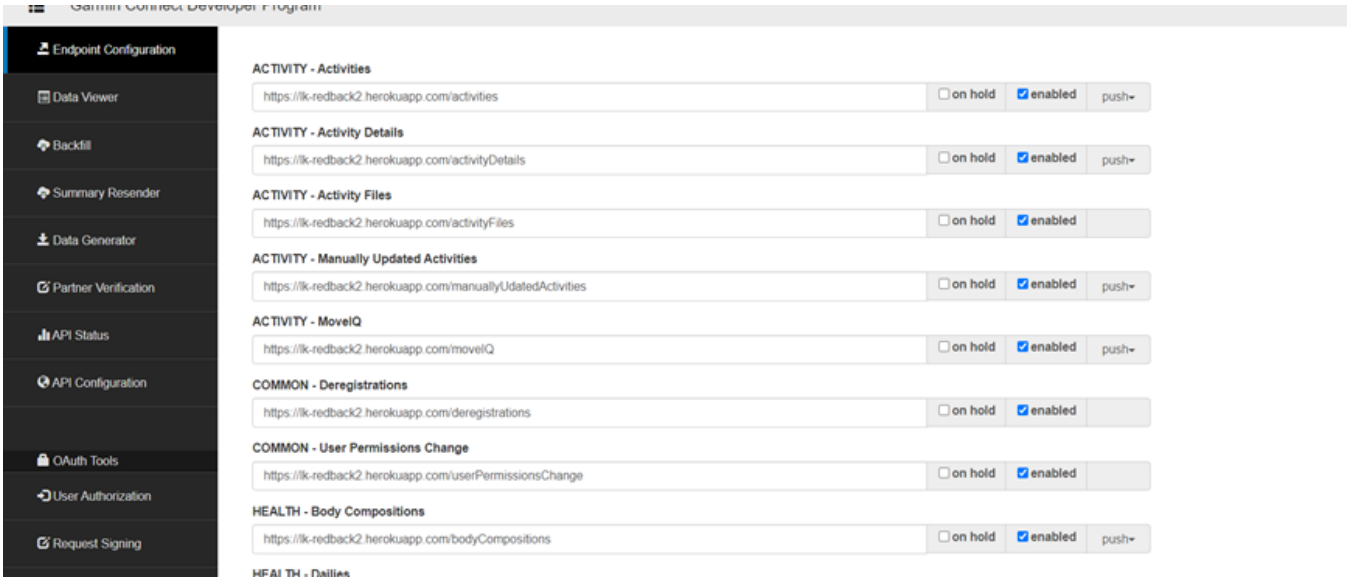


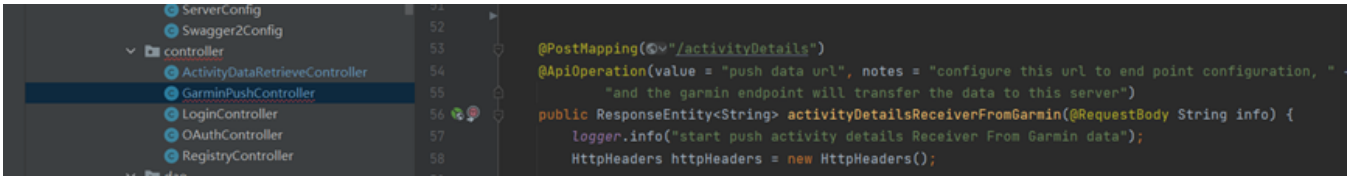
7. find our URL in the build log

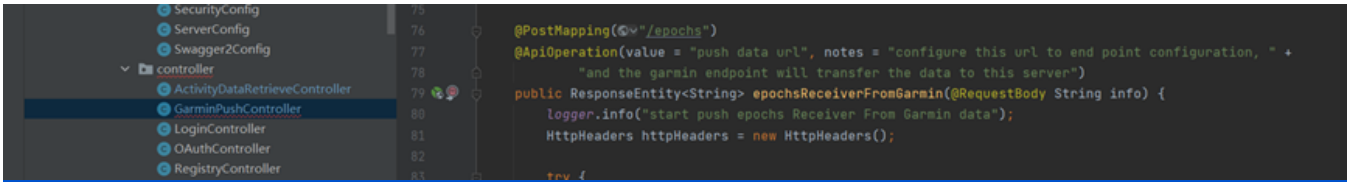


Finally, replace the Garmin developer URL as the Heroku app + path









Frontend deployment

Getting Started with Create React App

This project was bootstrapped with Create React App.

Available Scripts

In the project directory, you can run:

`npm start`

Runs the app in the development mode.
Open http://localhost:3000 to view it in your browser.

The page will reload when you make changes.
You may also see any lint errors in the console.

`npm test`

Launches the test runner in the interactive watch mode.
See the section about running tests for more information.

`npm run build`

Builds the app for production to the `build` folder.
It correctly bundles React in production mode and optimizes the build for the best performance.

The build is minified and the filenames include the hashes.
Your app is ready to be deployed!

See the section about deployment for more information.

`npm run eject`

Note: this is a one-way operation. Once you `eject`, you can't go back!

If you aren't satisfied with the build tool and configuration choices, you can `eject` at any time. This command will remove the single build dependency from your project.

Instead, it will copy all the configuration files and the transitive dependencies (webpack, Babel, ESLint, etc) right into your project so you have full control over them. All of the commands except `eject` will still work, but they will point to the copied scripts so you can tweak them. At this point you're on your own.

You don't have to ever use `eject`. The curated feature set is suitable for small and middle deployments, and you shouldn't feel obligated to use this feature. However we understand that this tool wouldn't be useful if you couldn't customize it when you are ready for it.

Learn More

You can learn more in the Create React App documentation.

To learn React, check out the React documentation.

Code Splitting

This section has moved here: https://facebook.github.io/create-react-app/docs/code-splitting

Analyzing the Bundle Size

This section has moved here: https://facebook.github.io/create-react-app/docs/analyzing-the-bundle-size

Making a Progressive Web App

This section has moved here: https://facebook.github.io/create-react-app/docs/making-a-progressive-web-app

Advanced Configuration

This section has moved here: https://facebook.github.io/create-react-app/docs/advanced-configuration

Deployment

This section has moved here: https://facebook.github.io/create-react-app/docs/deployment

`npm run build` fails to minify

This section has moved here: https://facebook.github.io/create-react-app/docs/troubleshooting#npm-run-build-fails-to-minify

Quality Control

Simple Overall Test Guide

Versions

Version ID	Description	Date
2.0	Redone once frontend is deployed, added more details	2022-05-29
1.0	The first version is based on an initial prototype before frontend is completed	2022-04-28

Version 2.0

The tests are 100% documented here on Confluence, and require both manual and technical testing, used for cross-platform testing between the backend (Java), frontend(React), and database(MongoDB).

The tests here is just a basic summary and guideline of what to do and check, once backend(Java), frontend(React), and database(MongoDB) has been set up.

To Ensure CoachingMate is properly deployed

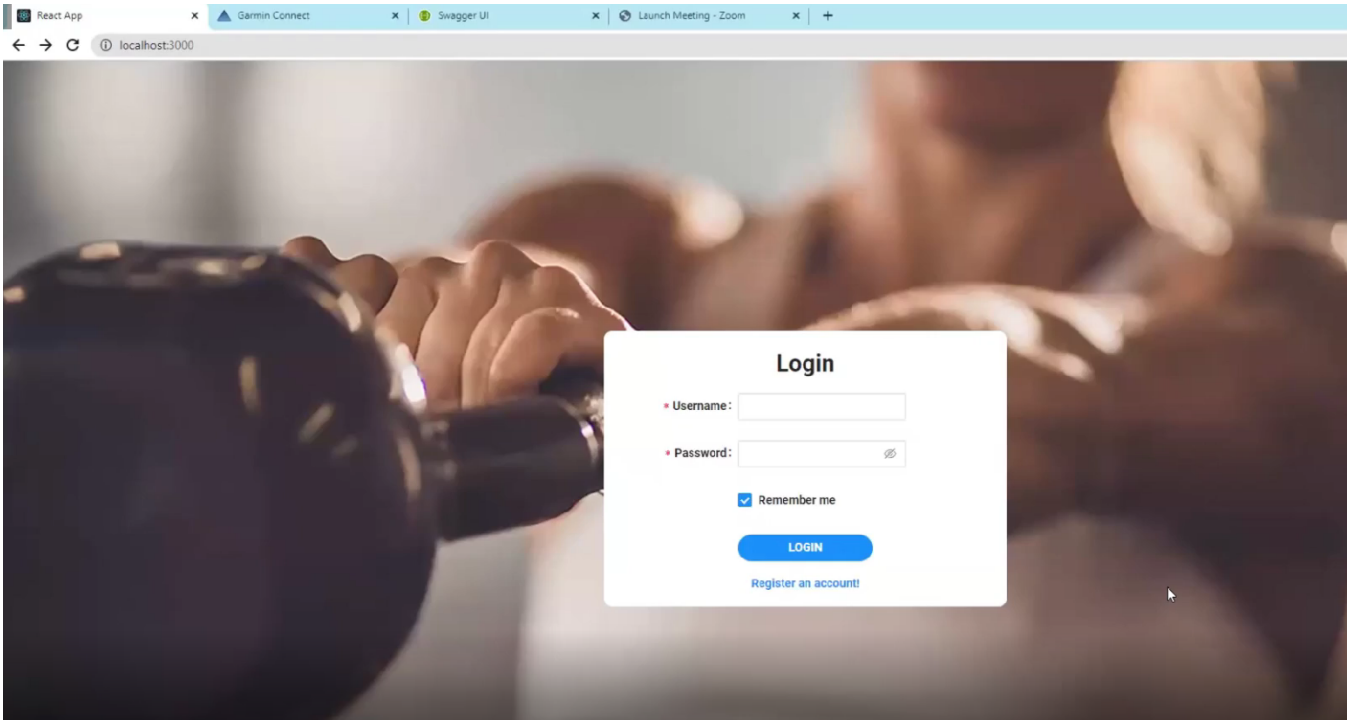
First, the user must first deploy both the backend and frontend. Refer to backend and frontend deployment under development for more details.

Once the backend is deployed, the server would be able to receive data from Garmin connect, and able to display it on the frontend.

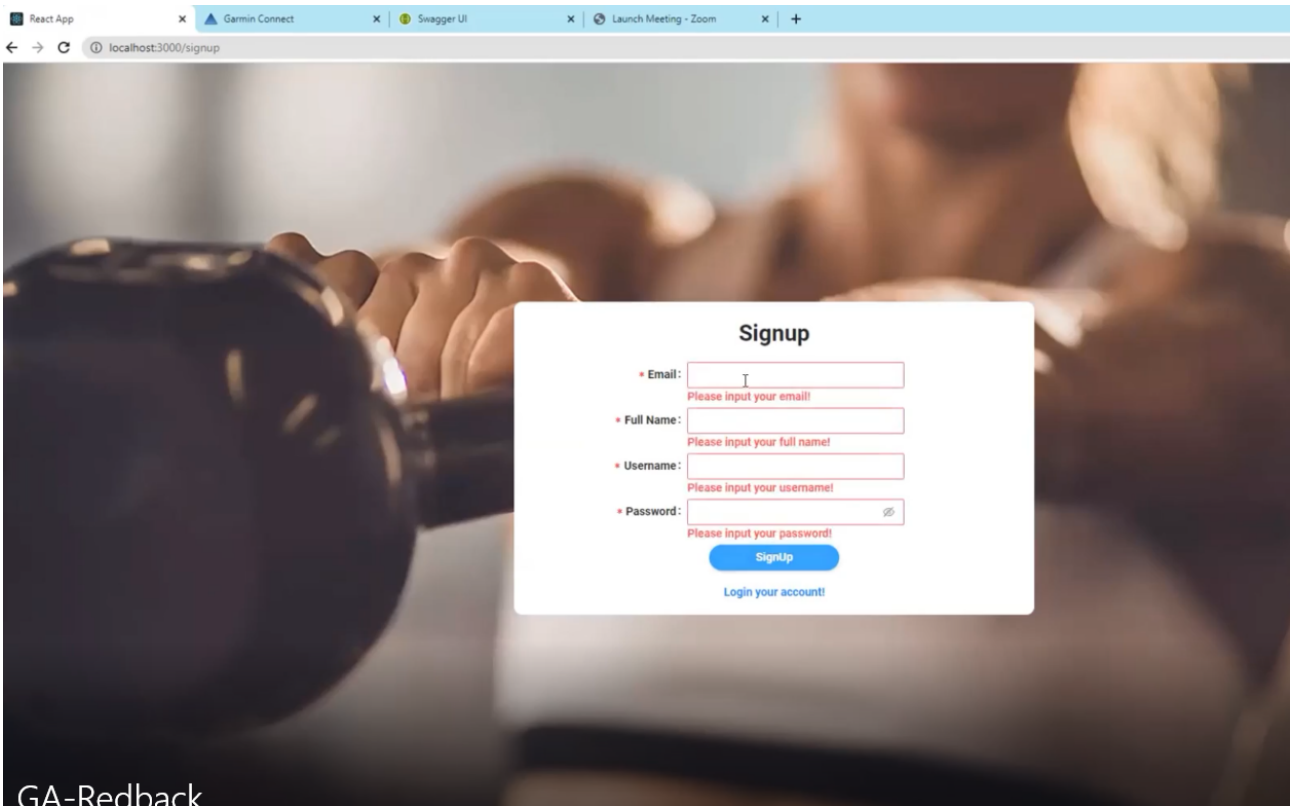
Checking frontend deployment

1. Create an account on the frontend, using a new username and password.

(login photo)

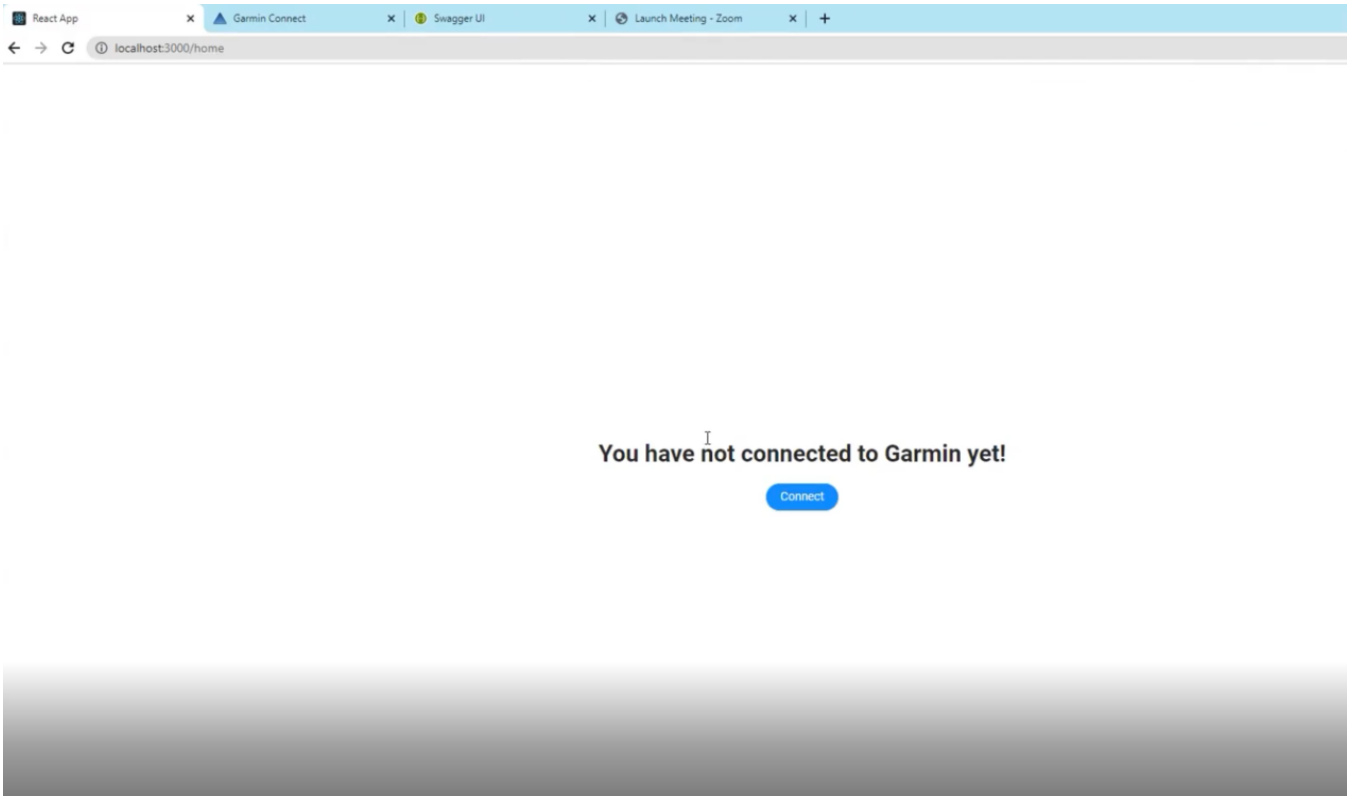


(account creation photo)

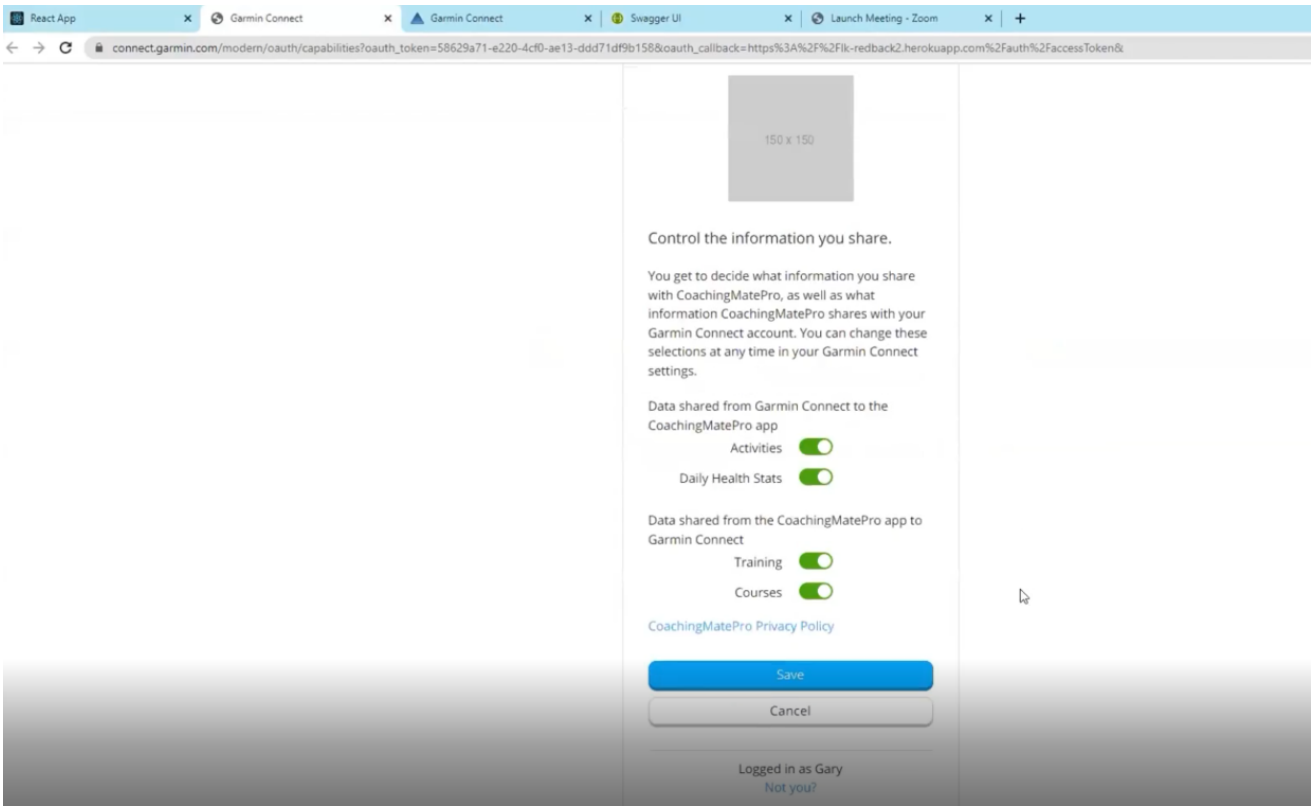


2. Once logged in, sync your CoachingMate account with your Garmin Connect account, so that the CoachingMate account will have access to your watch.

(photo of CoachingMate account not connected to Garmin)



(photo of requesting access to your Garmin account)

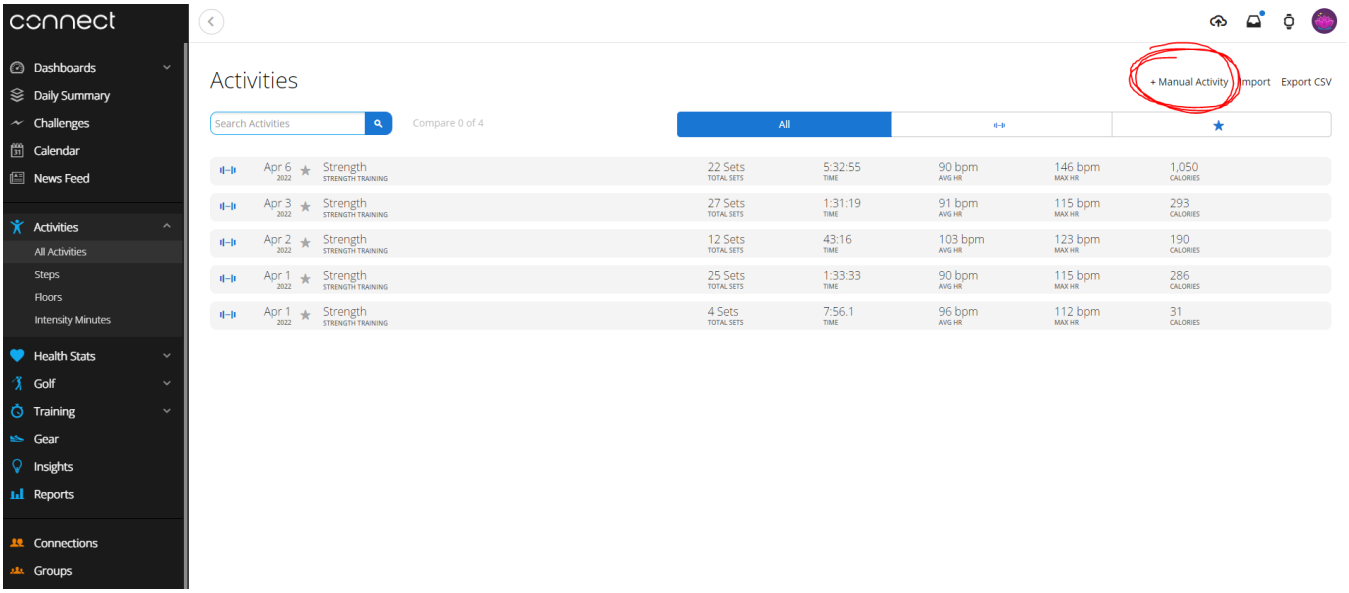


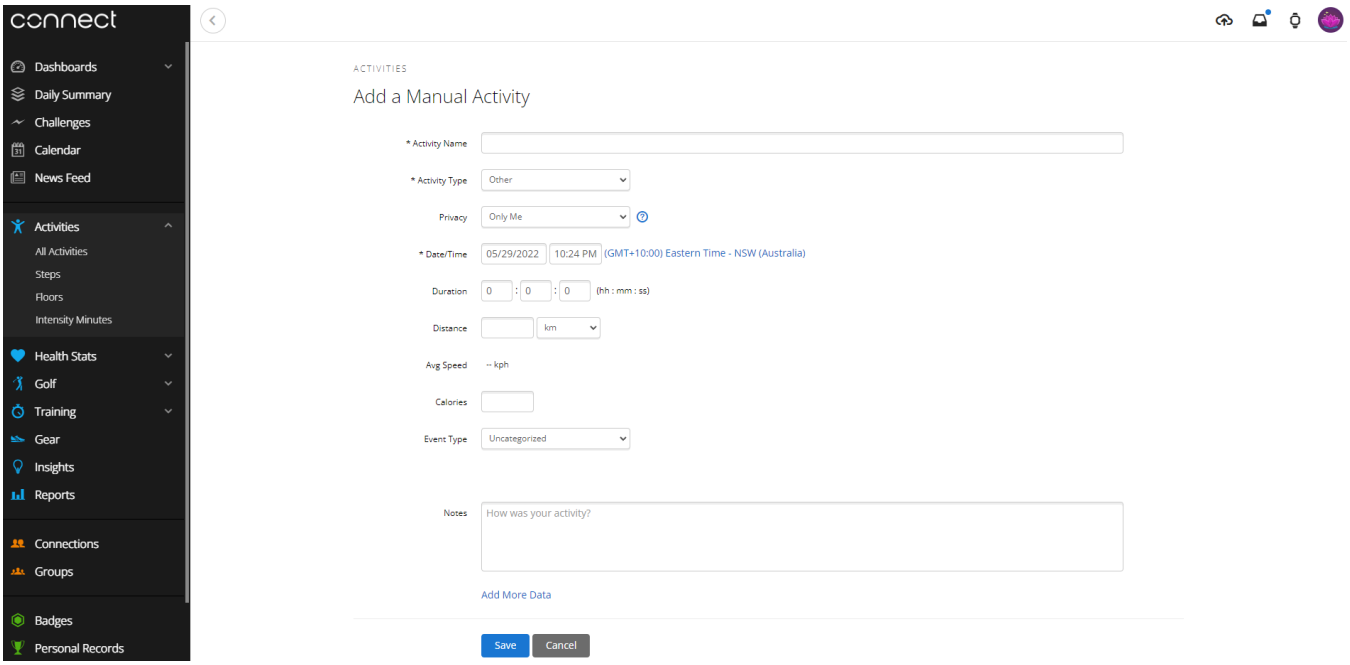
3. To ensure that the server is receiving data from Garmin connect, either

1)push an activity from your watch to the Garmin Connect account, or

2) create an activity

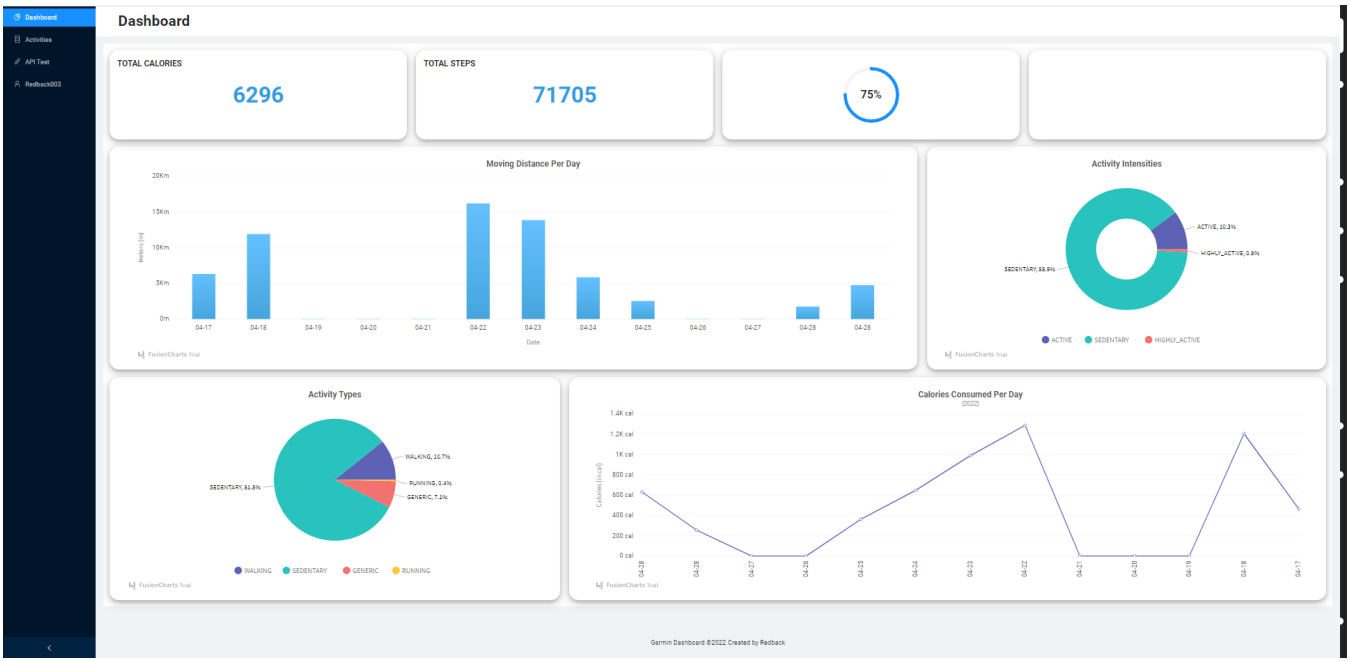
(photos of creating an activity in Garmin)

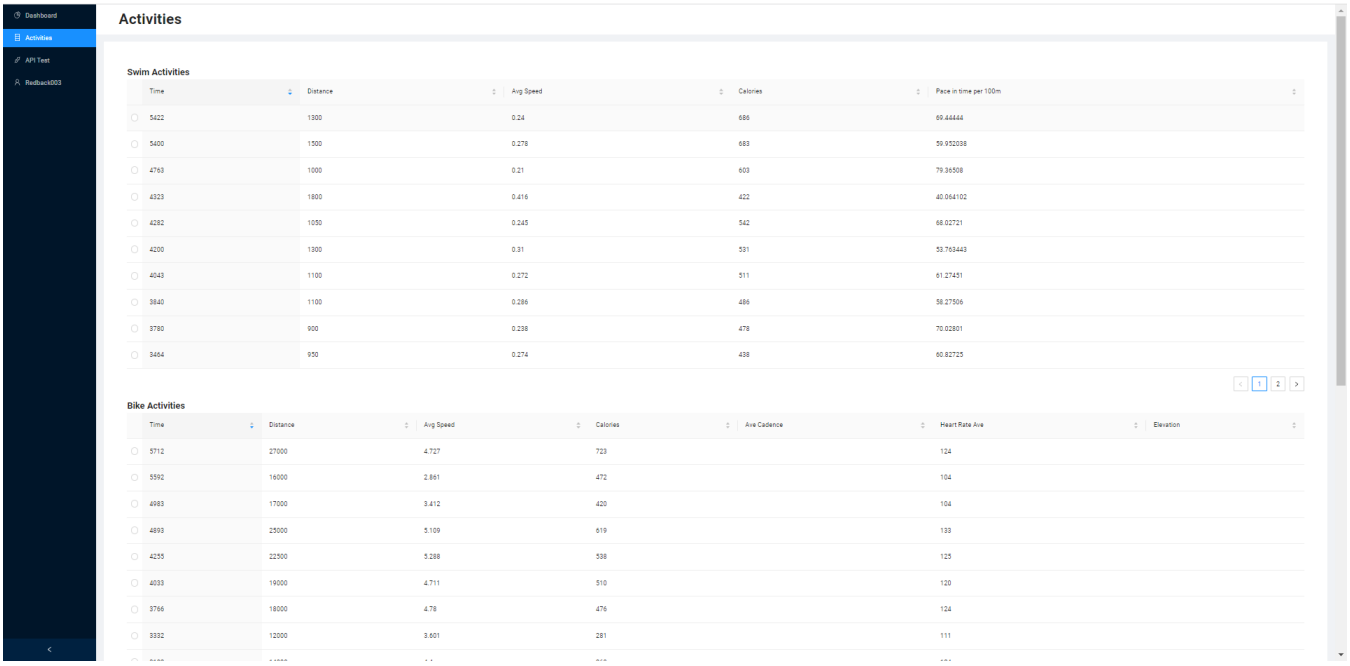




4. The frontend should have 2 tabs, the homepage and the activities tab.

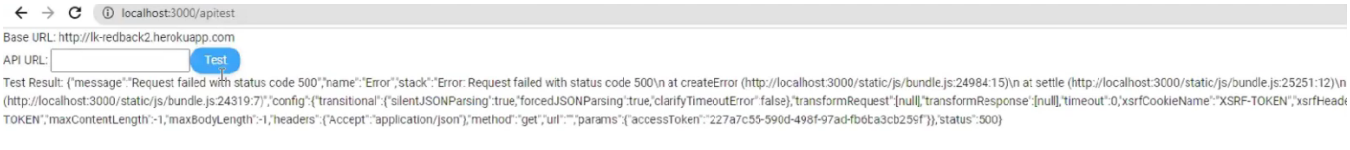
(photo of homepage and activities tab)





5. API Test tab is provided to show that data is properly transferred over from the backend to frontend. Input the API URL to to test retrieval of data.

(photo of API test tab)



Checking backend deployment

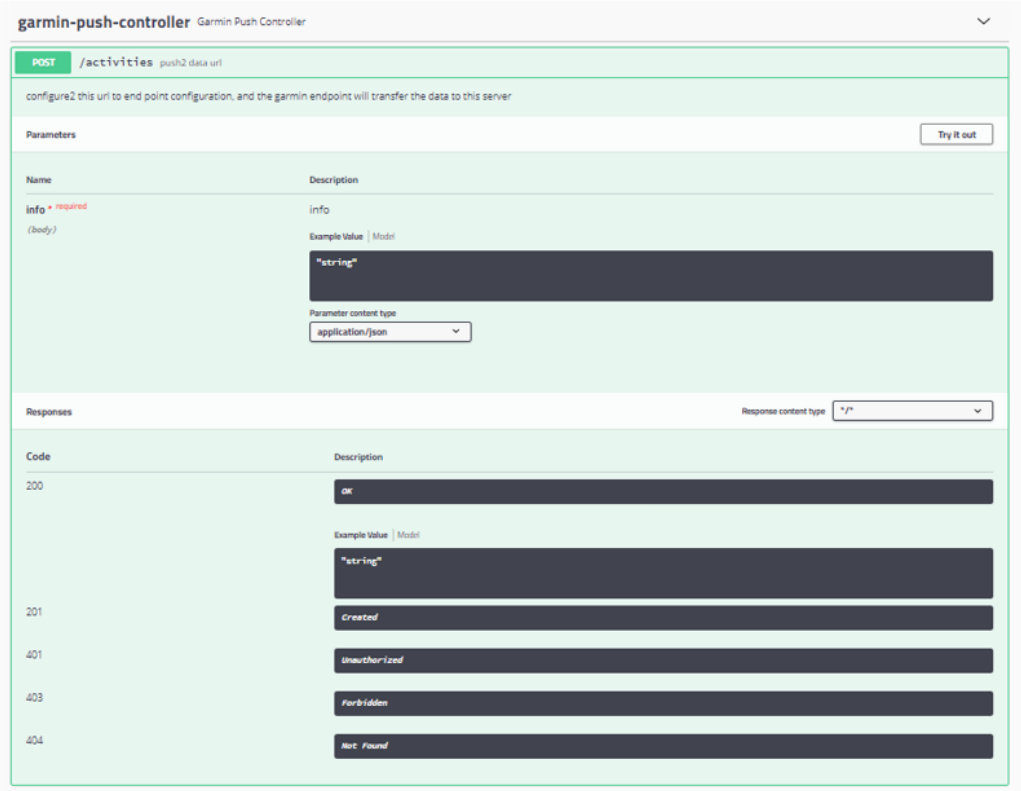
4. Once the activity is pushed to the Garmin Connect account, the CoachingMate Java backend would be able to receive the data shortly.

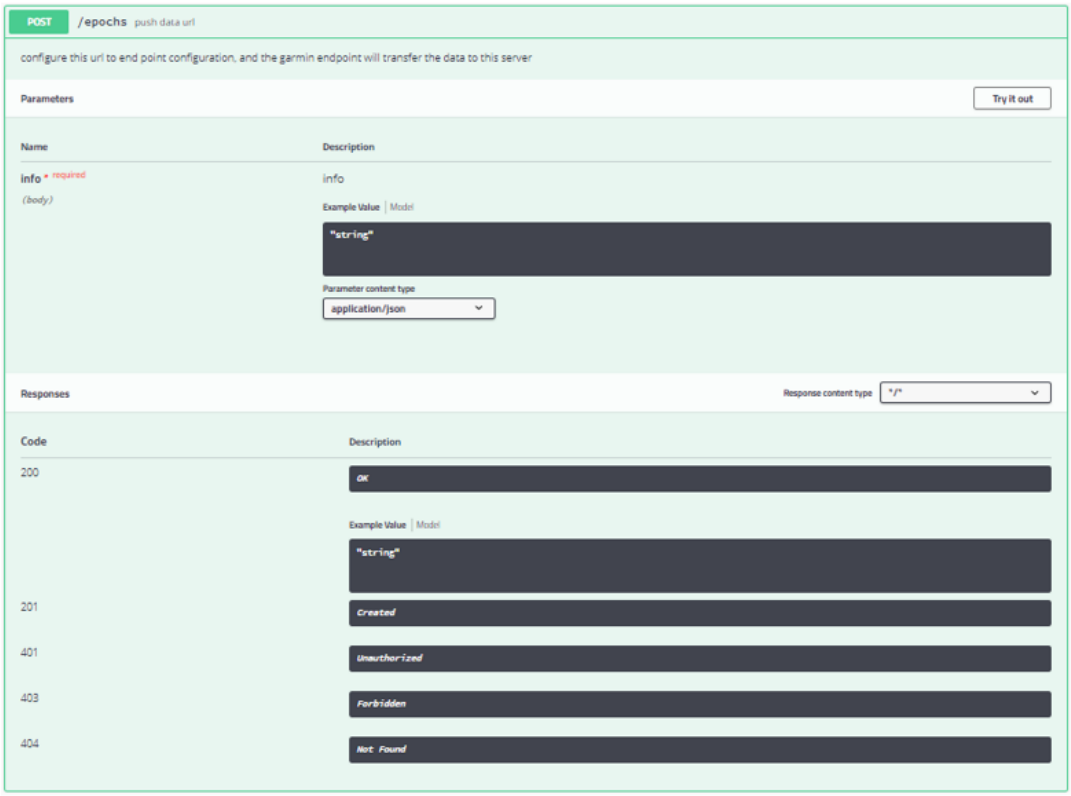
The data received from the API should include the Garmin-push-controller:

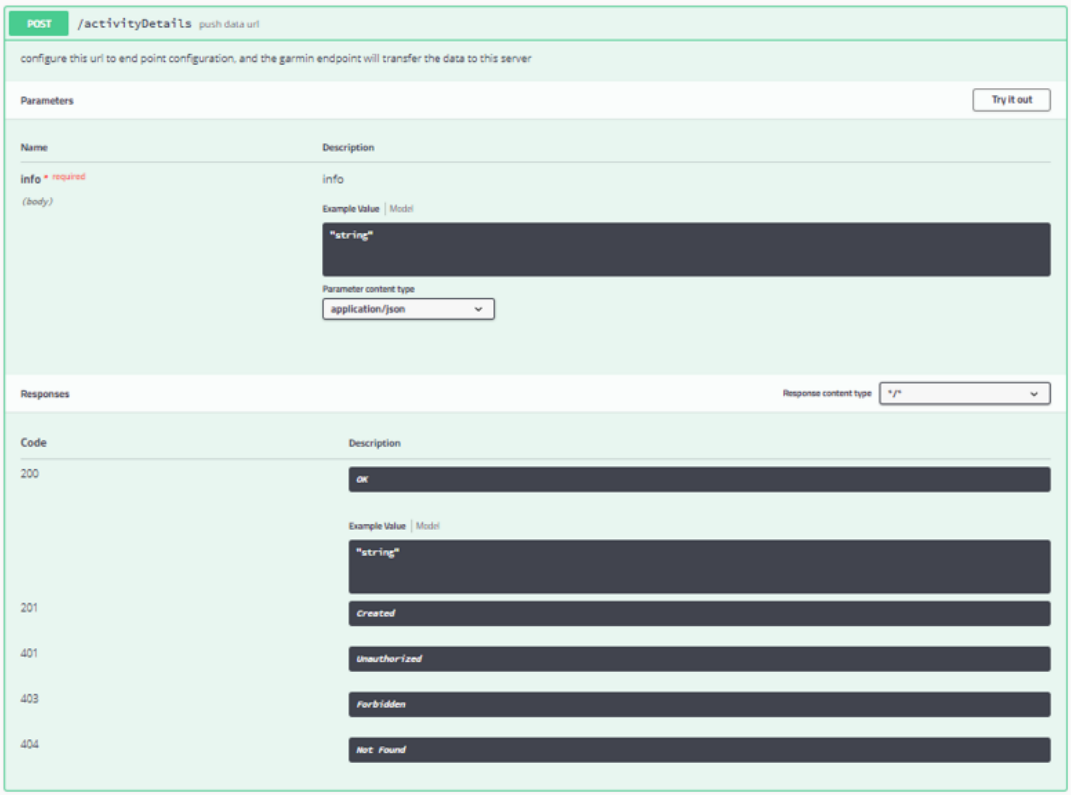
1. activities

2. activity details
3. epoche

The Garmin-push-controller contains the data from the athletes to be used for analysis for CoachingMate.



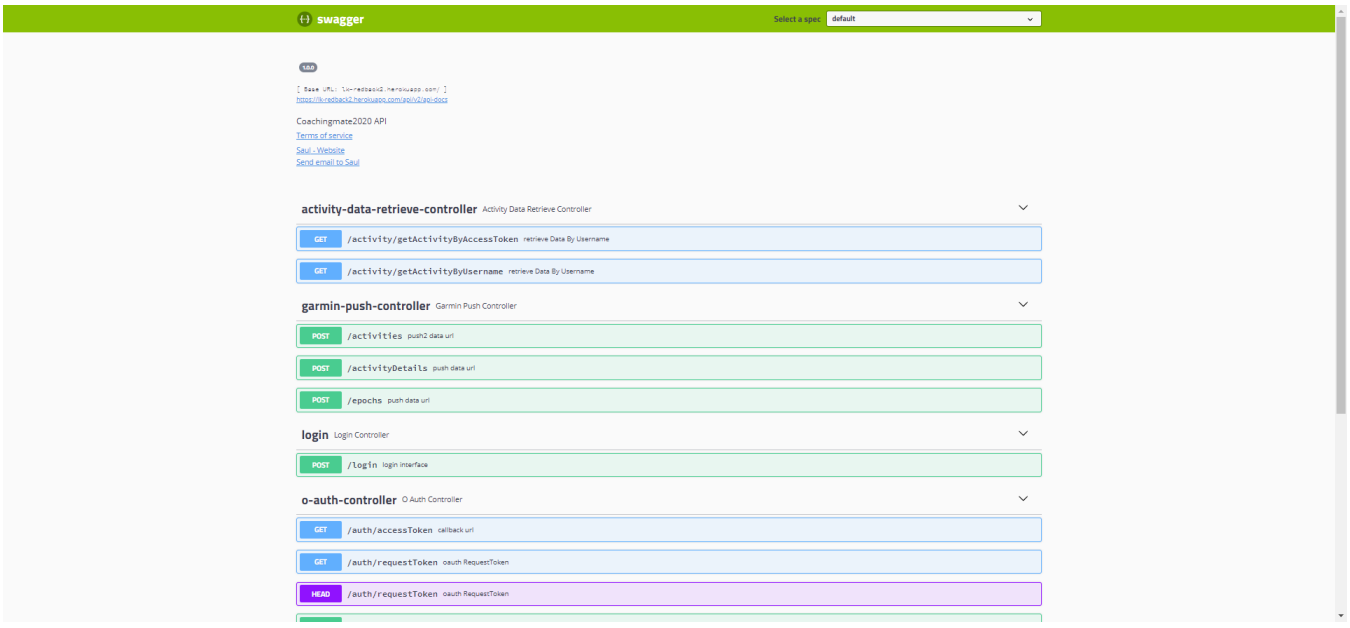


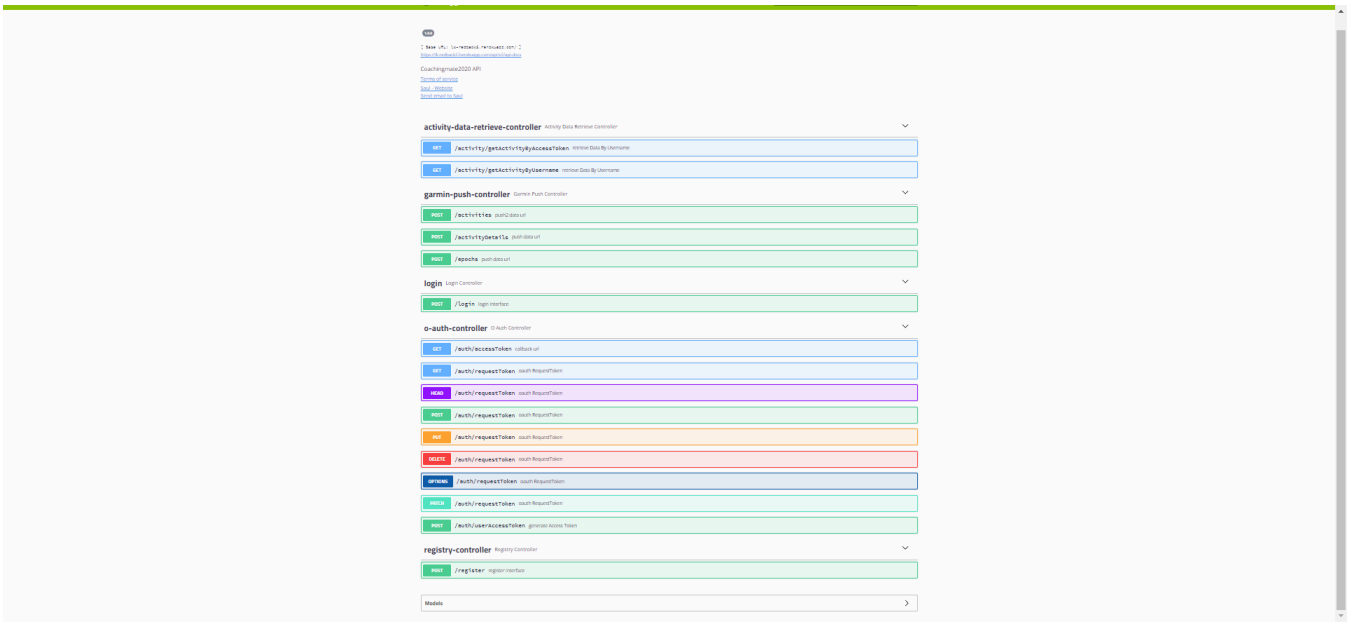


The data from the server can be displayed through the swagger ui deployed within the backend of the heroku server.

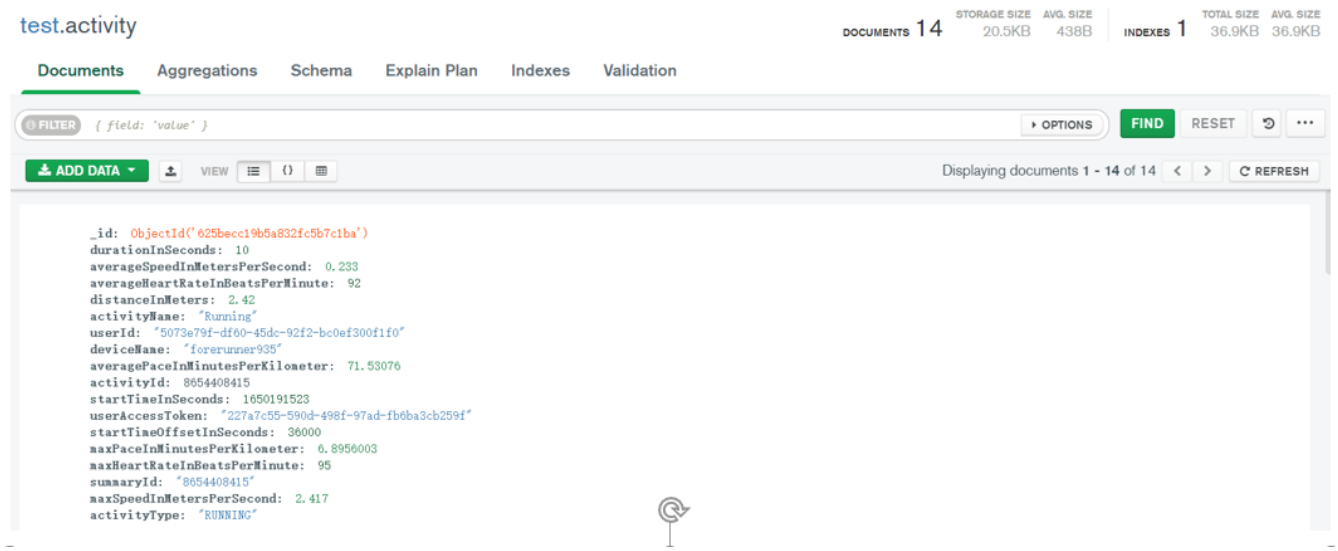
An example of our swagger ui that our team have deployed is located here. https://lk-redback2.herokuapp.com/api/swagger-ui.html#

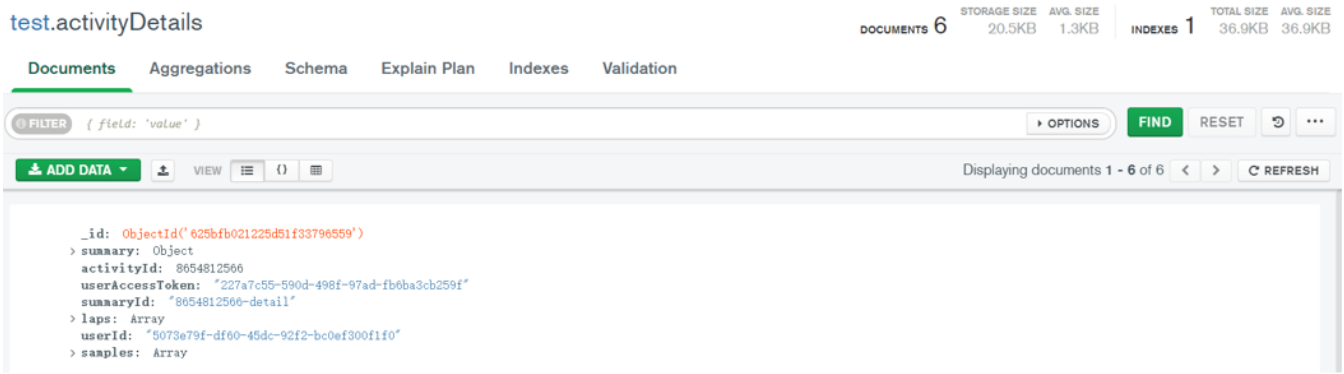
Screenshots of how the swagger ui looks like once deployed from the heroku server below:

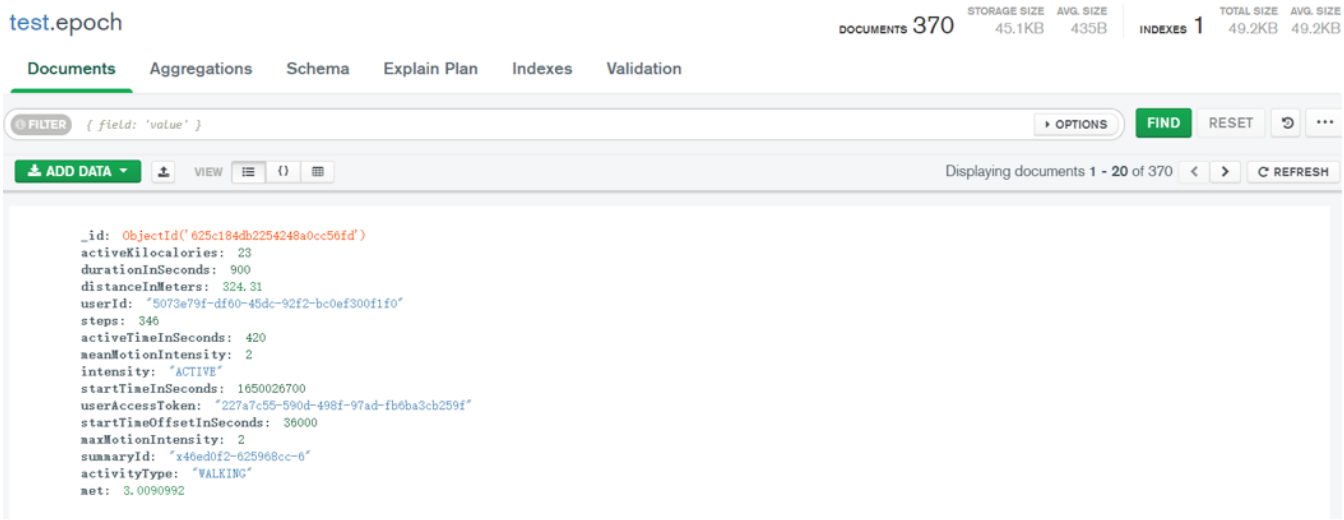




The data that will be displayed in MongoDB will produce the same results as shown in the screenshots below:







Version 1.0 (depleted)

The tests are 100% documented here on Confluence, and require manual testing, due to the cross-platform testing required between the backend(Java), frontend(React), and database(MongoDB).

The following steps below would show that the CoachingMate is running smoothly and that the data from the

Test: Ensure Backend is properly deployed

Once the backend is deployed, the server would be able to receive data from Garmin connect.

To be able to see if the server is able to receive the data, the user must first deploy both the backend and front end.

1. Create an account on the frontend, using a new username and password.

(account creation photo. The photo will be inserted once Frontend is completed)

2. Once logged in, sync your CoachingMate account with your Garmin Connect account, so that the CoachingMate account will have access to your watch.

(photo of requesting access to your Garmin account. The photo will be inserted once Frontend is completed)

3. To ensure that the server is receiving data from Garmin connect, either 1)push an activity from your watch to the Garmin Connect account, or 2) create an activity

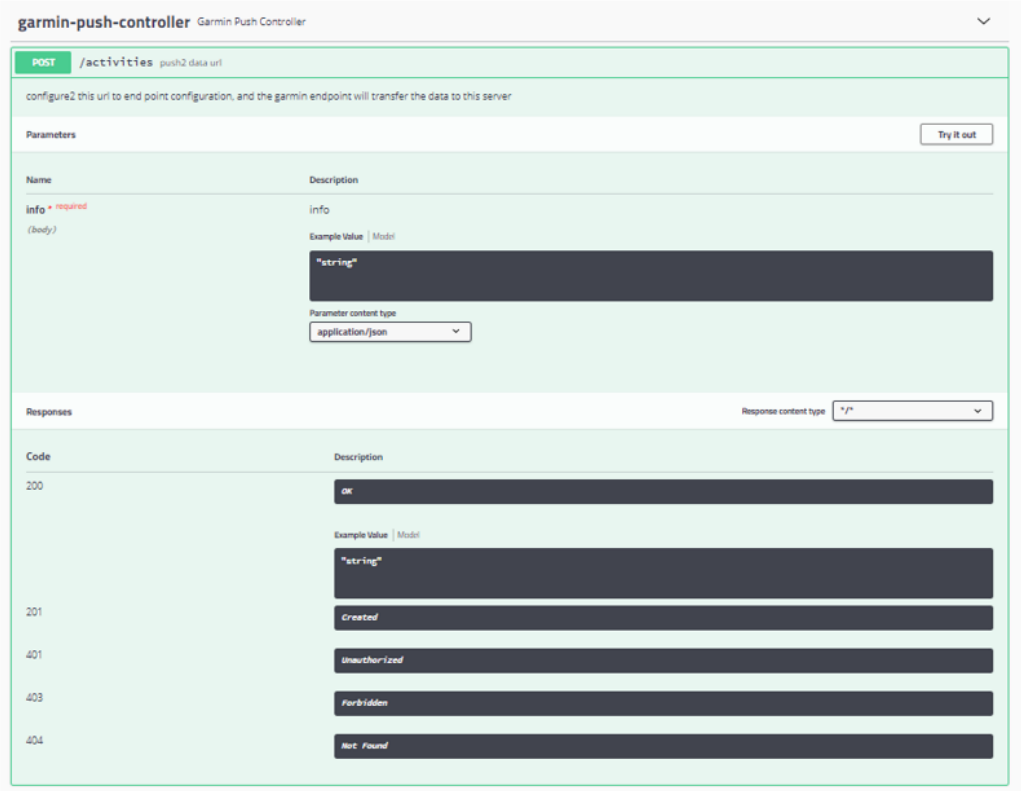
(photo of pushing activity to Garmin. The photo will be inserted once Frontend is completed)

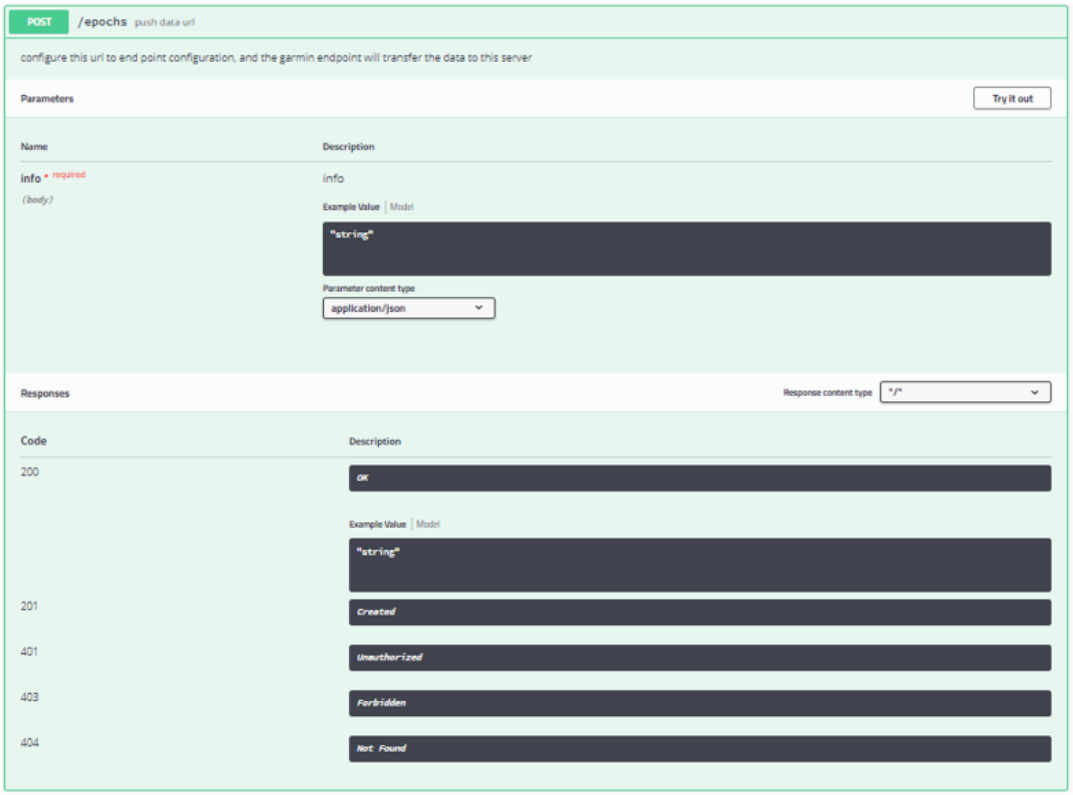
4. Once the activity is pushed to the Garmin Connect account, the CoachingMate Java backend would be able to receive the data shortly.

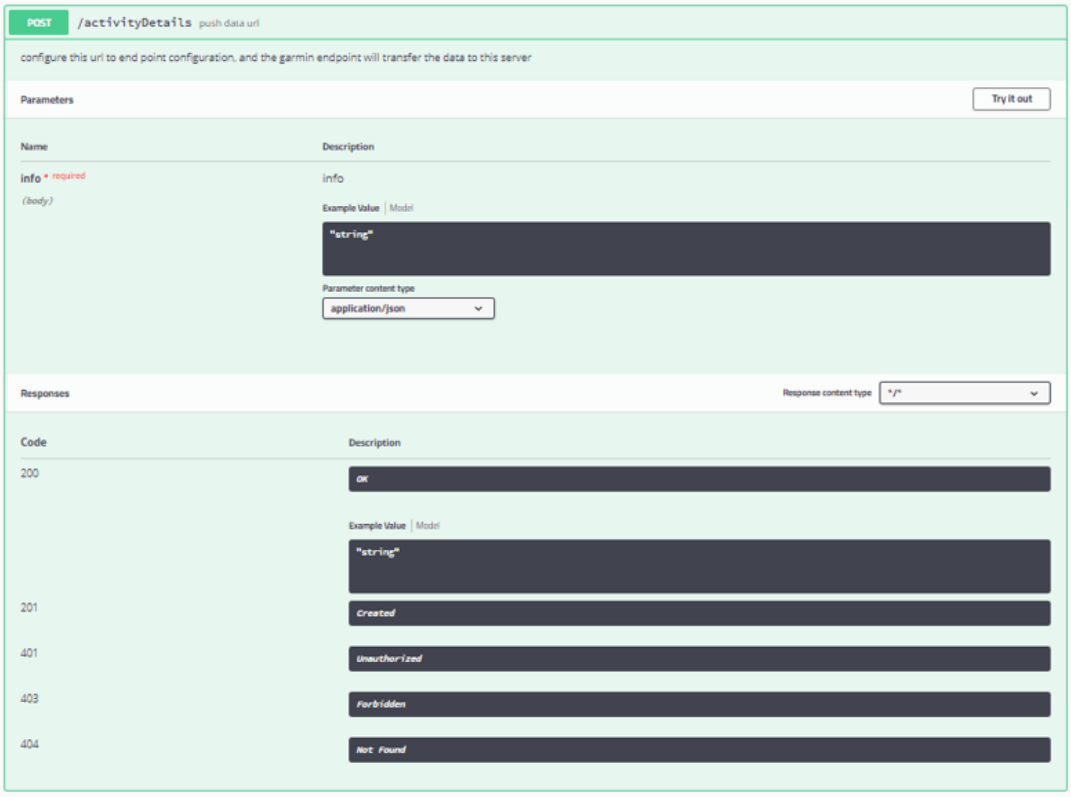
The data received from the API should include the Garmin-push-controller:

1. activities
2. activity details
3. epoche

The Garmin-push-controller contains the data from the athletes to be used for analysis for CoachingMate.



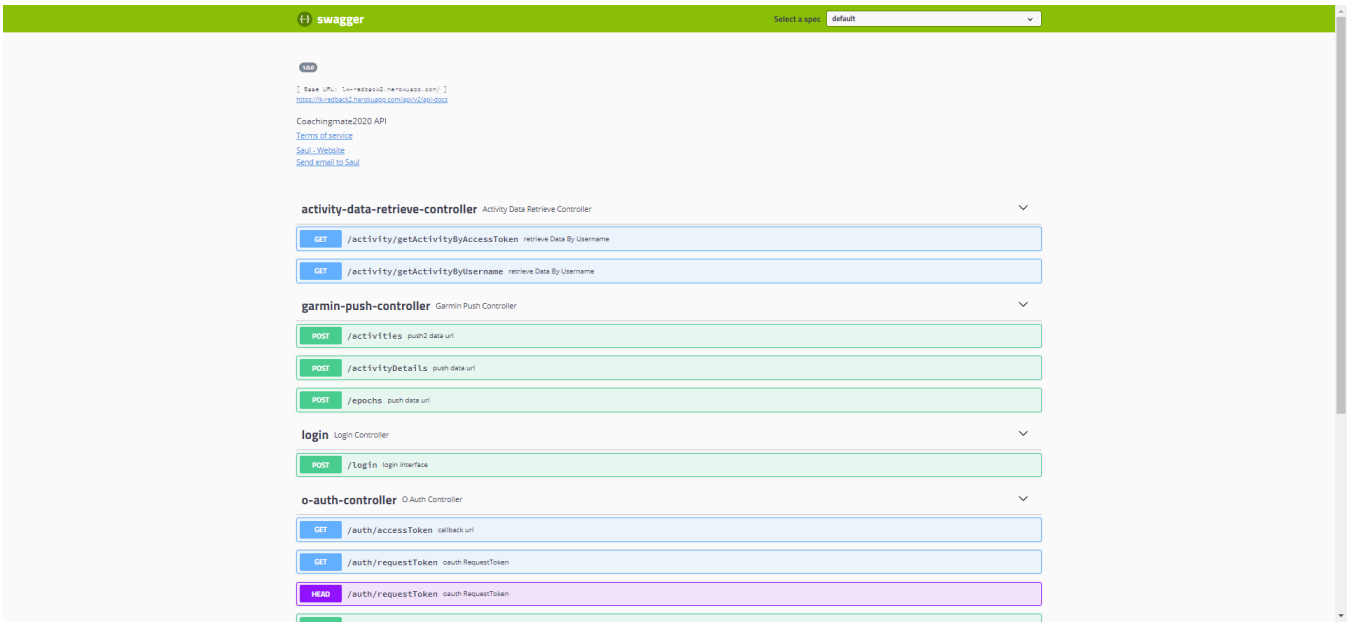


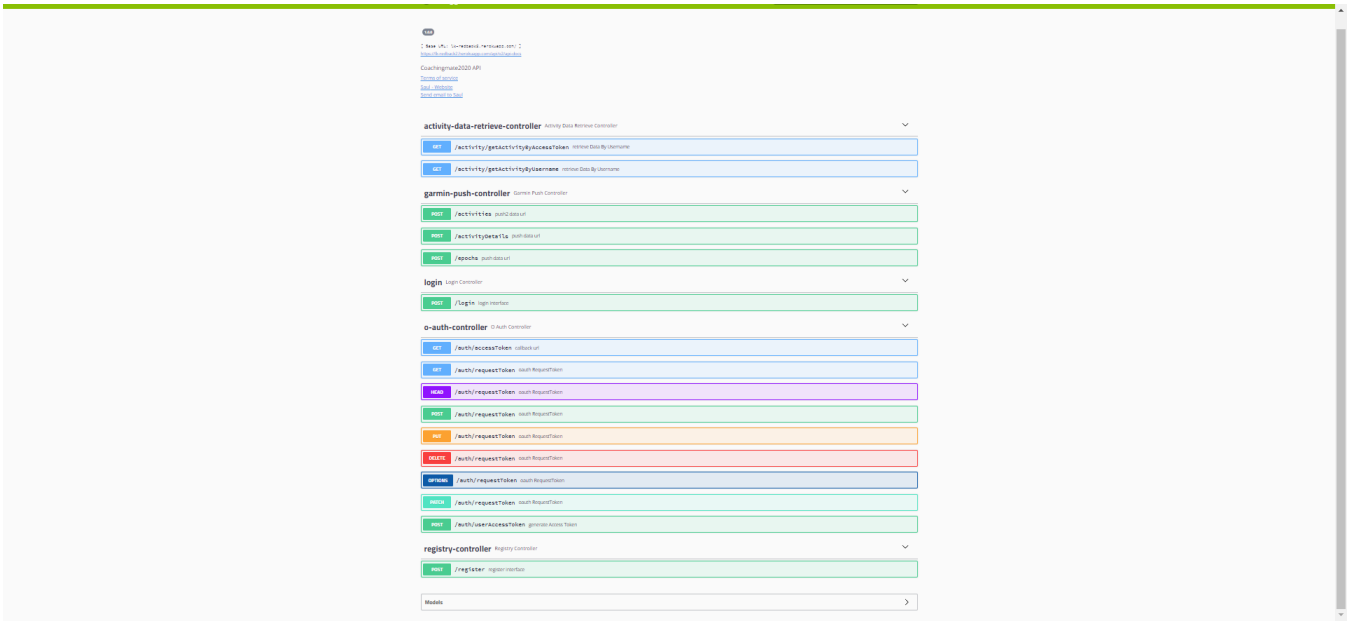


The data from the server can be displayed through the swagger ui deployed within the backend of the heroku server.

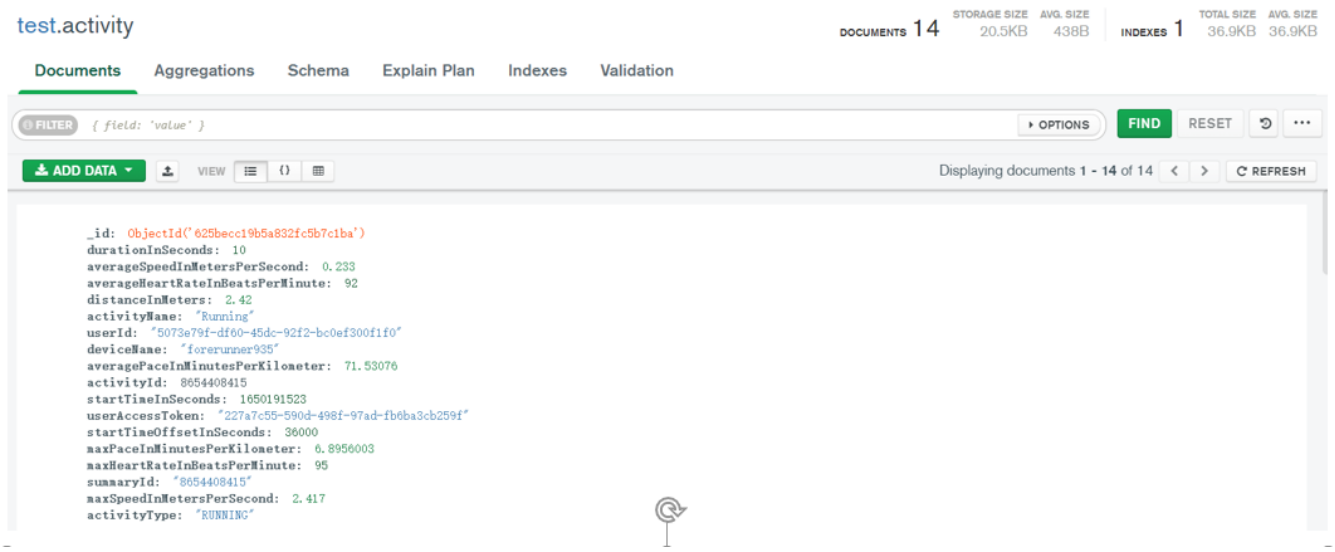
An example of our swagger ui that our team have deployed is located here. https://lk-redback2.herokuapp.com/api/swagger-ui.html#

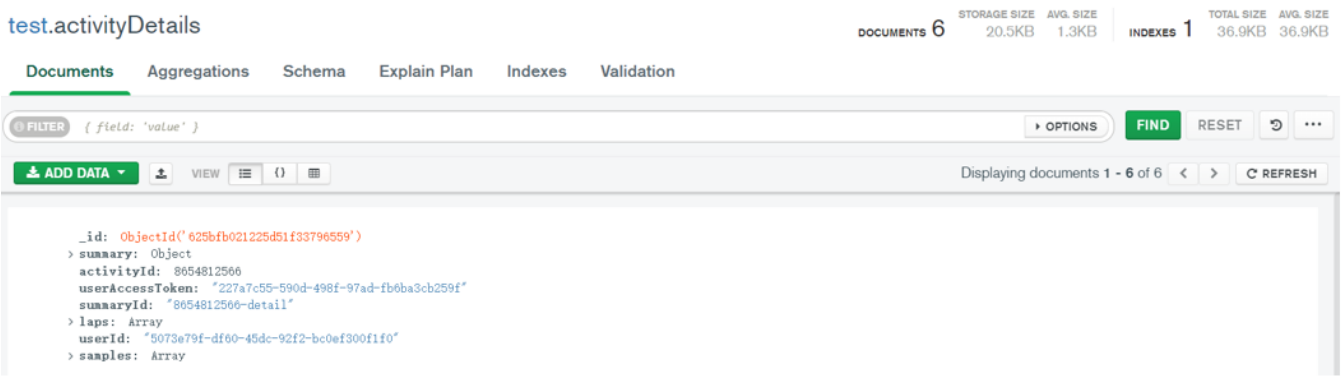
Screenshots of how the swagger ui looks like once deployed from the heroku server below:

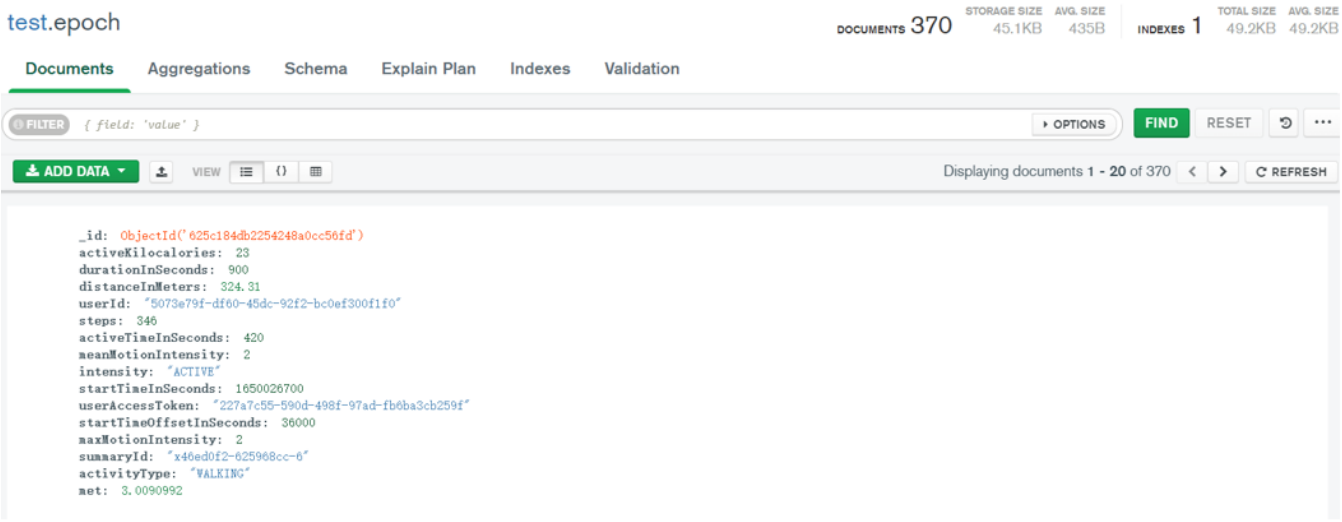




The data that will be displayed in MongoDB will produce the same results as shown in the screenshots below:







Test cases - acceptance criteria(backend)

Test Case	User Story ID	User Story	Test Instructions	Outcome

1	Auth Test	The user is authenticated to gain access to web APIs.	1. User-approved data is acquiring a request token and token secret. In this step, the program acquires an Unauthorized request token 2. Redirecting to Garmin Connect to authorized request token 3. the Request Token from step 1 and the OAuth verifier from step 2 will be used to generate a user access token	Get user access token Use to check the user is connected from the garmin app, OAuth?
2	Database Test	The backend fully connects to the MongoDB database.	1. The data can be transferred to MongoDB 2. The data can be retrieved from MongoDB	MongoDB connection test no error To check if the Java is connected to the MongoDB
3	UpComing DataProcess Test	The upcoming activity data format same as the desired format required before any processing is done	If the front-end data format is specified, then the data should be the same as it is.	The data format is same as front-end required Take information from Swagger, and check if the data is processed into into the MongoDB
4	Function Test		To check if existing functions that are used for data processing are working before usage to the frontend.	

Test cases - acceptance criteria (frontend)

Test Case	User Story ID	User Story	Test Instructions	Outcome
1	#01	The user can create the account, and set the password and username.	Click on "create new account" Type in a username into the username field. Type in a password into the password field. Click on "confirm"	Account creation successful is shown
3	#03	The user account can bind with any sports watch.	Gamin notification popup to sync with the CoachingMate server appears Click the "Connect to Garmin" button Login into the Garmin account to which your sports watch is connected	Sports watch connected to the Garmin account is synced with the CoachingMate account
4	#04	The user can unbind with any sports watch that they have currently bound with.	Go onto the Garmin connect website, under connections, click disconnect CoachingMate	Garmin account will disconnect from the CoachingMate
5	#05	Able to choose the option to sync and upload their data from the Garmin server to the CoachingMate server.	Once Test Case 3 is done, data will be synced to the CoachingMate server immediately. The data received from the API should include the Garmin-push-controller: 1. activities 2. activity details 3. epoche The data from the server can be shown by using the swagger UI deployed. The URL will depend on the Heroku app server your team has deployed. An example of our swagger UI that our team has deployed is located here. https://lk-redback2.herokuapp.com/api/swagger-ui.html#	Heroku app should be receiving the data from the Garmin server live and should reflect on swagger UI
6	#06	Able to upload the data consistently from the Garmin server on the CoachingMate website.	Login into the Garmin server https://connect.garmin.com/ Under the Activities, All Activities, on the top right-hand corner click on the "+Manual Activity". Once a manual activity is added, it should reflect on the CoachingMate website of your account shortly.	Manual Activity that is added from the Garmin server will reflect onto the CoachingMate website in about less than 5 minutes

Acceptance Criterias

Epics	User Story ID	Acceptance Criteria ID	User Story	Given	When	Then
E1- Account Administration Features		**AE1**	**As an athlete, I am able to create an account, so that I can view and edit my profile and devices.**			
	#01	A1	• The user can create the account, and set the password and username. 1. Able to type into the username field 2. Able to type into the password field 3. User data is able to be stored in the backend database	I am is trying to create a new account with CoachingMate	When I want to sign up with CoachingMate	I am able to create an account with CoachingMate with my username, password, and email.
	#03	A2	• The user account can bind with any sports watch. 1. The user account is able to have the option to sync with the Garmin server 2. Create the API required for the Garmin server data to sync with the CoachingMate server data 3. Create the API link from the Garmin server linking with the Backend (Java)	I am connecting a Garmin watch with my new CoachingMate account	I am connecting a Garmin watch with my new CoachingMate account	Able to bind with any sports watch.
	#04	A3	• The user can unbind with any sports watch that they have currently bound with. Options screen of able to unbind the synchronization to be seen displayed on the Garmin Connect website	I want to disconnect the old Garmin watch to connect to the new one	When I found the disconnect button on the Garmin Connect website	Able to unbind with any sports watch
E2 - Data Synchronisation		**AE2**	**As an athlete, I am able to sync the data from the watch to the website when I finish the workout so that I can govern my exercise data easily.**			
	#05	A2	• Able to choose the option to sync and upload their data from the Garmin server to the CoachingMate server. 1. Ensure that the API call from the Garmin server into the Backend (Java) 2. Options screen of the synchronization to appear and for the user to accept to bind the watch 3. Able to choose which data files for the CoachingMate to retrieve 4. Ensure that the data files can be read by the Backend (Java)	I want the training data to be synced to the backend server so that the front end can query them	When I finish my training /exercise using the Garmin watch	Able to sync and upload their data from the Garmin server to the CoachingMate server
	#06	A5	• Able to upload the data consistently from the Garmin server on the CoachingMate website. 1. Create a MongoDB server in the cloud to store the data required 2. Ensure that the MongoDB server is able to Save data from the Backend (Java) 3. Data retrieved in the Backend must be in the proper format to ensure that data is properly stored in the MongoDB server 4. Ensure that the Backend (Java) can retrieve data from the MongoDB server	I want to see my activity data in the Garmin database	When my Garmin watch has collected enough data	Data can be consistently uploaded to Garmin servers on the CoachingMate website
E3 - Data Visualisation		**AE3**	**As an athlete, I want to view my sports data on the website, so that it is easier for me to see what degree I achieved and dig into details to check my performance on different targets.**			

#09	A6	• I want to view basic sports data on my activity, such as the duration of the activity, calories burn, and distance, based on the specific activity transferred. 1. Ensure the User input is able to request data with the Frontend (React) 2. Backend (Java) is able to retrieve the data required from the MongoDB (Database) 3. The backend (Java) is able to produce output into the Frontend (React) 4. The frontend is able to display the necessary data from the Backend	I want to view my activity data on the front end	When I browse the dashboard to do training plans or view data	Able to view basic athletic data about my activity based on the specific activity
#10	A7	• I want to be able to see a summary of my activities in a form of a dashboard to be able to see clearly what I have achieved overall 1. Create a statistical graph over time to show calories burnt each day 2. Create a relational graph of two statistics to show relational data between them	I record several days of my body's statistical data using a Garmin watch and upload the data to Garmin connect	When I try to figure out the consumption of calories as well as the activities summaries over the past days	The pie chart can show the percentage of my activity types regarding to swimming, running, and cycling. While the line chart shows the consumption of calories through the past days
As an athlete, I am able to see the specific details of each of my individual **Swim** activities					
#11	A8.0	A backend retrieve controller (getSwimmingActivityByAccessToken) is to be created in the backend database for easier retrieval	I receive a user access tokens	When I try to retrieve user activities from MongoDB	Able to get the user activities using the access token
#12	A8.1	I can see the **Time** taken for each of my Swim activities	I have finished my swim activity	When I try to find out how much time I used in swimming	Able to see how much time I spend on this activity
#13	A8.2	I can see the **Distance** taken for my Swim activities	I have finished my swim activity	When I try to find out how much distance I taken in swimming	Able to see how much distance I have swum.
#14	A8.3	I can see the **Average Speed** for my Swim activities	I have finished my swim activity	When I try to find out the average speed for one of my swimming activity	Able to see the average speed of the swimming activity.
#15	A8.4	I can see the **Calories** consumed in my Swim activities	I have finished my swim activity	When I try to find out the calorie consumption for one of my swimming activity	Able to see the consumption of the swimming activity.
#16	A8.5	I can see the **Pace** in time per 100m for my Swim activities	I have finished my swim activity	When I try to find out the speed rhythm per 100 m for one of my swimming activity	Able to see the line chart over speed per 100m when I click on the swimming activity
As an athlete, I am able to see the specific details of each of my individual **Cycling** activities					
#17	A9.0	A backend retrieve controller (getCyclingActivityByAccessToken) is to be created in the backend database for easier data retrieval	I have finished my cycling activity	When I try to retrieve user activities from MongoDB	Able to get the user activities using the access token
#18	A9.1	I can see the **Time** taken for each of my Cycling activities	I have finished my cycling activity	When I try to find out how much time I used in cycling	Able to see how much time I spend on this activity
#19	A9.2	I can see the **Distance** taken for my Cycling activities	I have finished my cycling activity	When I try to find out how much distance I taken in cycling	Able to see how much distance I taken for this activity
#20	A9.3	I can see the **Average Speed** for my Cycling activities	I have finished my cycling activity	When I try to find out my average speed I used in cycling	Able to see my average speed in this activity
#21	A9.4	I can see the **Calories** consumed in my Cycling activities	I have finished my cycling activity	When I try to find out how much calories I burned in cycling	Able to see how many calories I burned on this activity
#22	A9.5	I can see the **Average Cadence** for my Cycling activities	I have finished my cycling activity	When I try to find out my average cadence in cycling	Able to see my average cadence in this activity
#23	A9.6	I can see the **Average Heart Rate** during my Cycling activities	I have finished my cycling activity	When I try to find out my average heart rate in cycling	Able to see my average heart rate during this activity
#24	A9.7	I can see the **Elevation** in my Cycling activities	I have finished my cycling activity	When I try to find out my elevation for cycling	Able to see my elevation for this activity
As an athlete, I am able to see the specific details of each of my individual **Run** activities					

	#25	A10.0	A backend retrieve controller (getRunningActivityByAccessToken) is to be created in the backend database for easier data retrieval	I receive a user access tokens	When I try to retrieve user activities from MongoDB	Able to get the user activities using the access token
	#26	A10.1	I can see the **Time** taken for each of my Run activities	I have finished my running activity	When I try to find out how much time I used in running	Able to see how much time I spend on this activity
	#27	A10.2	I can see the **Distance** taken for each of my Run activities	I have finished my running activity	When I try to find out how much distance I have taken in running	Able to see how much distance I have taken for this activity
	#28	A10.3	I can see the **Average Speed** for my Run activities	I have finished my running activity	When I try to find out my average speed in running	Able to see my average speed in this activity
	#29	A10.4	I can see the **Calories** for my Run activities	I have finished my running activity	When I try to find out how many calories I burned in running	Able to see how many calories I burned on this activity
	#30	A10.5	I can see the **Pace in time per km** for my Run activities	I have finished my running activity	When I try to find out my pace in time per km in running	Able to see my pace in time per km of this activity
	#31	A10.6	I can see the **Average Cadence** for my Run activities	I have finished my running activity	When I try to find out my average cadence in running	Able to see my average cadence in this activity
	#32	A10.7	I can see the **Average Heart Rate** for my Run activities	I have finished my running activity	When I try to find out my average heart rate in running	Able to see my average heart rate during this activity
	#33	A10.8	I can see the **Elevation** for my Run activities	I have finished my running activity	When I try to find out my elevation for running	Able to see my elevation for this activity

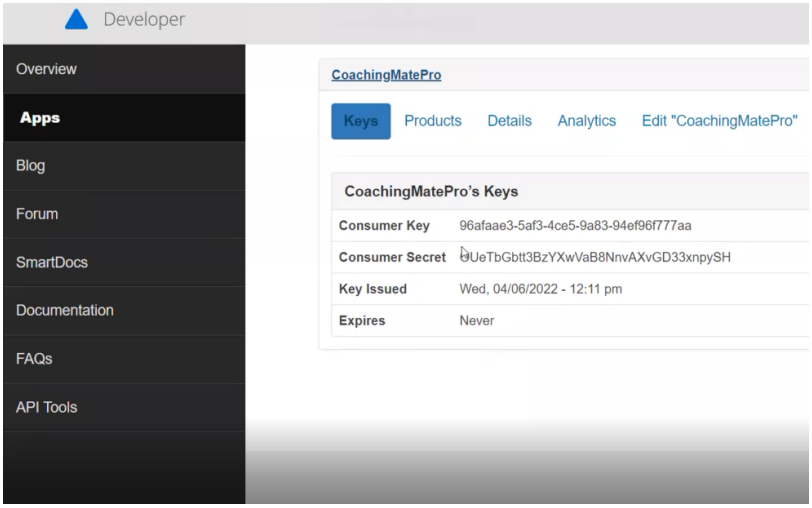
Acceptance Tests

Technical Testing is listed below the Acceptance Test.

Epics	Acceptance Criteria ID	Acceptance Test	Critical		Test Result		Comments
			Yes	No	Accept	Reject	
E1- Account Administration Features	**AE1**	**Technical Testing: authTest.java**	x		x		**Refer to Technical Tests below for details**
	A1	Click on "create new account" Type in an Email into the email field. Type in a name into the Full Name field. Type in a username into the username field. Type in a password into the password field. Click on "SignUp"	x		x		(**authTest.java** technical testing will assist in the user is authenticated to gain access to web APIs)
	A2	Gamin notification popup to sync with the CoachingMate server appears, saying "You have not connected to Garmin yet!" Click the "Connect" button. Information on what to share with CoachingMate should appear. If no Garmin account is logged in, a login prompt will appear. If a Garmin account is logged in, share all data and click "Save". Click "please relogin" to go back to CoachingMate.	x		x		
	A3	Login into the Garmin server website https://connect.garmin.com/ Under connections, click disconnect CoachingMate.		x	x		
E2 - Data Synchronisation	**AE2**	**Technical Testing: databaseTest.java, upComingDataprocessTest.java**	x		x		**Refer to Technical Tests below for details**
	A4	Once the CoachingMate account has been linked to the Garmin watch account (A2), • the user does an exercise using his Garmin watch • or manually adds an exercise through the Garmin Connect website. From the home screen of the CoachingMate website, click the activities tab. The exercise will be listed in the tab.	x		x		For manually adding: 1. Login into the Garmin server website https://connect.garmin.com/ 2. Under the Activities, All Activities, on the top right-hand corner click on the "+Manual Activity". 3. Once a manual activity is added, it should reflect on the CoachingMate website of your account shortly. (**databaseTest.java** technical testing will assist in ensuring The backend fully connects to the MongoDB database.)
	A5	Once A4 is completed, data will be synced to the CoachingMate server immediately and will reflect in MongoDB. The data received from the API should include the following Garmin-push-controllers: 1. activities 2. activity details 3. epoche The data from the server can be shown by using the swagger UI deployed. The URL will depend on the Heroku app server your team has deployed. An example of our swagger UI URL that our team has deployed is located at https://lk-redback2.herokuapp.com/api/swagger-ui.html# Screenshots are available in the "Simple Overall Test Guide".	x		x		(**upComingDataprocessTest.java** technical testing will assist in ensuring the upcoming activity data format same as the desired format required before any processing is done)
E3 - Data Visualisation	**AE3**	**Technical Testing: functionTest.java**	x		x		**Refer to Technical Tests below for details**
	A6	Once the CoachingMate account has been linked to the Garmin watch account (A2), • the user does an exercise using his Garmin watch • or manually adds an exercise through the Garmin Connect website. From the home screen of the CoachingMate website, basic sports data are displayed.	x		x		(**functionTest.java** technical testing will assist to check if existing functions that are used for data processing are working before usage to the frontend.)

A7	Do multiple exercises of varying types from swim, run, cycling and record them into the Garmin Connect or sync from the watch. From the home screen of the CoachingMate website, basic sports data are displayed, and a pie chart can show the percentage of my activity types.		x	x		
As an athlete, I am able to see the specific details of each of my individual **Swim** activities						
A8.0	After a swimming session with my Garmin watch, MongoDB is able to retrieve the user activities using the access token	x		x		
A8.1	After a swimming session with my Garmin watch, under the activities tab, the **Time** taken for the specific swim activity is shown.		x	x		
A8.2	After a swimming session with my Garmin watch, under the activities tab, the **Distance** taken for the specific swim activity is shown.		x	x		
A8.3	After a swimming session with my Garmin watch, under the activities tab, the **Average Speed** taken for the specific swim activity is shown.		x	x		
A8.4	After a swimming session with my Garmin watch, under the activities tab, the **Calories** taken for the specific swim activity is shown.		x	x		
A8.5	After a swimming session with my Garmin watch, under the activities tab, the **Pace** in time per 100m taken for the specific swim activity is shown.		x	x		
As an athlete, I am able to see the specific details of each of my individual **Cycling** activities						
A9.0	After a cycling session with my Garmin watch, MongoDB is able to retrieve the user activities using the access token	x		x		
A9.1	After a cycling session with my Garmin watch, under the activities tab, the **Time** taken for the specific cycling activity is shown.		x	x		
A9.2	After a cycling session with my Garmin watch, under the activities tab, the **Distance** taken for the specific cycling activity is shown.		x	x		
A9.3	After a cycling session with my Garmin watch, under the activities tab, the **Average Speed** taken for the specific cycling activity is shown.		x	x		
A9.4	After a cycling session with my Garmin watch, under the activities tab, the **Calories** consumed for the specific cycling activity is shown.		x	x		
A9.5	After a cycling session with my Garmin watch, under the activities tab, the **Average Cadence** for the specific cycling activity is shown.		x	x		
A9.6	After a cycling session with my Garmin watch, under the activities tab, the **Average Heart Rate** for the specific cycling activity is shown.		x	x		
A9.7	After a cycling session with my Garmin watch, under the activities tab, the **Elevation** for the specific cycling activity is shown.		x	x		
As an athlete, I am able to see the specific details of each of my individual **Run** activities						
A10.0	After a run session with my Garmin watch, MongoDB is able to retrieve the user activities using the access token	x		x		
A10.1	After a run session with my Garmin watch, under the activities tab, the **Time** taken for specific run activity is shown.		x	x		
A10.2	After a run session with my Garmin watch, under the activities tab, the **Distance** taken for specific run activity is shown.		x	x		
A10.3	After a run session with my Garmin watch, under the activities tab, the **Average Speed** taken for specific run activity is shown.		x	x		
A10.4	After a run session with my Garmin watch, under the activities tab, the **Calories** taken for specific run activity are shown.		x	x		
A10.5	After a run session with my Garmin watch, under the activities tab, the **Pace in time per km** taken for specific run activity is shown.		x	x		
A10.6	After a run session with my Garmin watch, under the activities tab, the **Average Cadence** taken for specific run activity is shown.		x	x		
A10.7	After a run session with my Garmin watch, under the activities tab, the **Average Heart Rate** taken for specific run activity is shown.		x	x		
A10.8	After a run session with my Garmin watch, under the activities tab, the **Elevation** taken for specific run activity is shown.		x	x		

Technical Tests

Test Case	Test Name	Test Details	Test Instructions	Outcome	
				Pass	Fail
authTest.java	Authentication Test	The user is authenticated to gain access to web APIs. 1. User-approved data is acquiring a request token and token secret. In this step, the program acquires an Unauthorized request token 2. Redirecting to Garmin Connect to authorized request token 3. the Request Token from step 1 and the OAuth verifier from step 2 will be used to generate a user access token	1. Run the test authTest.java 2. If the test can be run without errors, the previous API is working. Otherwise continue to step 2 and 3 3. Replace line 23 of the code "String consumerkey" with the Consumer Key of your created app from the developer portal of Garmin 4. Replace line 24 of the code "String consumerSecret" with the Consumer Secret of your created app from the developer portal of Garmin 5. Run the test authTest.java again (photo of where to get the Consumer Key and Secret at garmin developer website) 	Get user access token can be obtained and the API is working.	The current API app is not working. A new API is required to be created for a new Consumer Key and Consumer Secret to gain user access token.

					To re-adjust the test for future use, replace code line 23 and 24 with your new consumer Secret and Key.

databaseTest.java	Database Test	This test is to check of the mongoDB URL is working. It is used to check if the Java is connected to the MongoDB.	1. Replace code line 10 of the code "String uri" with your created mongoDB uri. 2. Run the test databaseTest.java	MongoDB connection test no error 1. The data can be transferred to MongoDB 2. The data can be retrieved from MongoDB	MongoDB connnection has error. Re-check if your MongoDB uri is properly established. To re-adjust the test for future use, replace line 10 of the code "String uri" with your created mongoDB uri.
upComingDataprocessTest.java	Upcoming Data Process Test	The upcoming activity data format same as the desired format required before any processing is done for the data received from the API. It should check the Garmin-push-controller of these 3 JSONObjects: 1. activities 2. activity details 3. epochs	1. Run the test upComingDataprocessTest.java 2. If the front-end data format is specified, then the data should be the same as it is. 3. If API data format obtained has changed, an error with be produced based on the code line 17, 18, 19	The data format is the same as the front-end required	The format of the data retrieved from the Garmin API has changed,

The data format no longer the same as the front-end required.

Confirm and check the information from Swagger, and check if the data is processed into the MongoDB.

					To re-adjust the test for future use, replace code line 17, 18, 19 with the new json format obtained from Swagger.

function Test.java	Function Test	This test is to check if existing functions that are used for data processing are working before usage to the frontend.	1. Run the test functionTest.java	the functions coded in the Java are working accordingly	The format of the data retrieved from the Garmin API has most likely changed, resulting in the functions to not work accordingly. To re-adjust the test for future use, add any additional functions used for data processing into this test file.

Sprints

Sprint 2

Sprint 3

Sprint 2

For the Sprint 2 goal, our team is to ensure that the frontend of the code passed down by the previous team is working as intention, re-estabilishing the connection of the backend to the Garmin Watch API, and producing a program that can display simple data.

- Frontend - to be fixed and deployed successfully
- Backend - find the key problem in the API connection and reestablishing it
- Backend - set up an online database server to receive the data
- Frontend - able to receive the data from the backend and display successfully (added due to additional time to work on)

With team members enrolled in the University of Melbourne as full-time students, in consideration of their other units, it is estimated that each student will be able to allocate 1 hour of their time per day on the project. In total, each person will be able to contribute 2 hours per working day, for 5 days a week, with a total of a team of 5 members, which sums up to a total of 50 hours per week.

Sprint 2 – 4 weeks (200 hours)

1 story point = 8 hours

25 story points

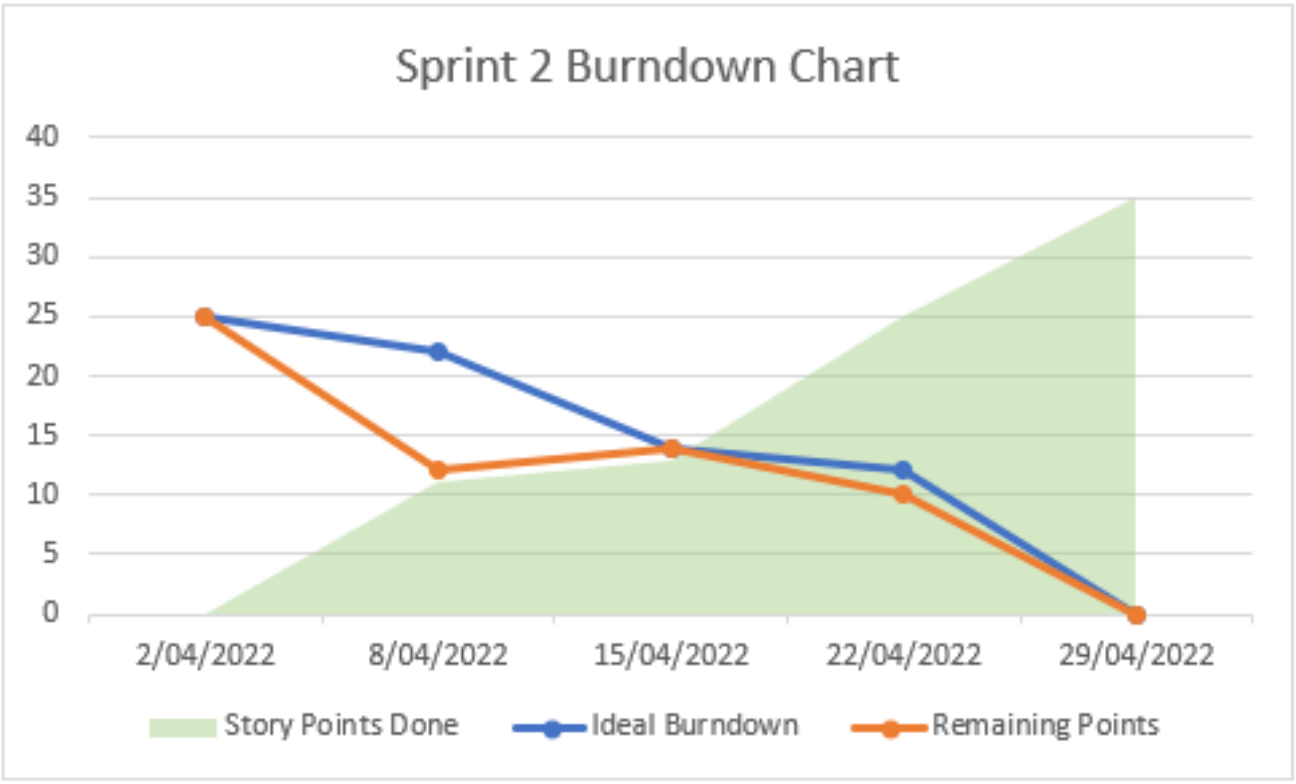
The start date of the project will begin on March 28 and will end on the May 1. In accordance with the bucket system, where the velocity of the team is 12.5 story points per week, which can be calculated by 50 story points divided by 4 weeks.

Our initial plan was to only finish up E1 and partial of E2 User Story 05, however process was smoother than expected, thus leading up to the completion of both E1 and up to E2 (user story E6) during this sprint 2.

User Stories Planned

Epics	User	User Story ID	User Stories	MoSCoW Priority	Story Points (4 hour/pt)
E1- Account Administration Features	As an athlete, I am able to create an account, so that I can view and edit my profile and devices.				
	Athlete	01	• The user can create the account, set the password and username. 1. Able to type into the username field 2. Able to type into the password field 3. User data is able to be stored into the backend database	Must have	3
	Athlete	02	• The user can reset the password, in case the password is forgotten. 1. Enable to change password link 2. Password is updated in the backend database	Must have	4
	Athlete	03	• The user account can bind with any sports watch. 1. The user account is able to have the option to sync with the Garmin server 2. Create the API required for the Garmin server data to sync with the CoachingMate server data 3. Create the API link from the Garmin server linking with the Backend (Java)	Must have	4
	Athlete	04	• The user can unbind with any sports watch that they have currently binded with. Option screen of able to unbind the syncronisation to be seen displayed the Garmin Connect website	Must have	2
E2 - Data Synchronisation	As an athlete, I am able to sync the data from the watch to the website when I finish the workout so that I can govern my exercise data easily.				

	Athlete	05	• Able to choose the option to sync and upload their data from Garmin server to the CoachingMate server. 1. Ensure that the API call from the Garmin server into the Backend (Java) 2. Option screen of the syncronisation to appear and for the user to accept to bind the watch 3. Able to choose which data files for the CoachingMate to retrieve 4. Ensure that the data files that can be read by the Backend (Java)	Must have	12
	Athlete	06	• Able to upload the data consistently from the Garmin server on the CoachingMate website. 1. Create a MongoDB server in the cloud to store the data required 2. Ensure that the MongoDB server is able to Save data from Backend (Java) 3. Data retrieved in the Backend must be in proper format to ensure that data is properly stored in MongoDB server 4. Ensure that the Backend (Java) can retrieve data from the MongoDB server	Must have	10



In accordance to the burndown chart, our team have completed the required story points on 22/04/2022, which an additional week for our team to work on additional story points. Our team added the user story E2: 06 into the final week for completion, adding an additional 10 story points completed, totalling to 35 story points completed within this sprint, compared to the expected 25 story points.

The GA project was to initially target to only setting up the API and backend of the code, up till E2. However, due to the fast progress that the team has achieved. Sprint 3 will incorporate working on a new frontend system in accordance to the new epic E3.

Test cases - acceptance criteria

Test Case	User Story ID	User Story	Test Instructions	Outcome

1	#01	The user can create the account, set the password and username.	Click on "create new account" Type in a username into the username field. Type in a password into the password field. Click on "confirm"	Account creation successful is shown
3	#03	The user account can bind with any sports watch.	Gamin notification popup to sync with the CoachingMate server appears click "Connect to Garmin" button Login into the Garmin account which your sports watch is connected	Sports watch connected to the Garmin account is sync with the CoachingMate account
4	#04	The user can unbind with any sports watch that they have currently binded with.	Go onto the Garmin connect website, under connections, click disconnect CoachingMate	Garmin account will disconnect from the CoachingMate
5	#05	Able to choose the option to sync and upload their data from Garmin server to the CoachingMate server.	Once Test Case 3 is done, data will be synced to the CoachingMate server immediately. The data recieved from the API should include the garmin-push-controller: 1. activities 2. activityDetails 3. epoche The data from the server can be shown by using the swagger UI deployed. The url will depends on the herokuapp server your team has deployed. An example of our swagger ui that our team have deployed is located here. https://lk-redback2.herokuapp.com/api/swagger-ui.html#	Herokuapp should be recieving the data from the Garmin server live and should reflect on swagger UI
6	#06	Able to upload the data consistently from the Garmin server on the CoachingMate website.	Login into the Garmin server https://connect.garmin.com/ Under the Activities, All Activities, on the top right hand corner click in the "+Manual Activity". Once a manual activity is added, it should reflect onto the CoachingMate website of your account shortly.	Manual Activity that is added from Garmin server will reflect onto the CoachingMate website in about less than 5 minutes

Sprint 3

For the Sprint 3 goal, our team is focused on creating an entirely new Frontend interface, that shows the 3 types of activities in detail: Run, Swim and Cycle, using React that is able to connect with the backend and MongoDB.

- Frontend - create a new frontend based on new user stories
- Frontend - able to receive the data from the backend and MongoDB successfully
- Frontend - Able to display a dashboard tab of summary data
- Frontend - Able to display an activities tab of Run, Swim and Cycle

With team members enrolled in the University of Melbourne as full-time students, in consideration of their other units, it is estimated that each student will be able to allocate 1 hour of their time per day on the project. In total, each person will be able to contribute 2 hours per working day, for 5 days a week, with a total of a team of 5 members, which sums up to a total of 50 hours per week.

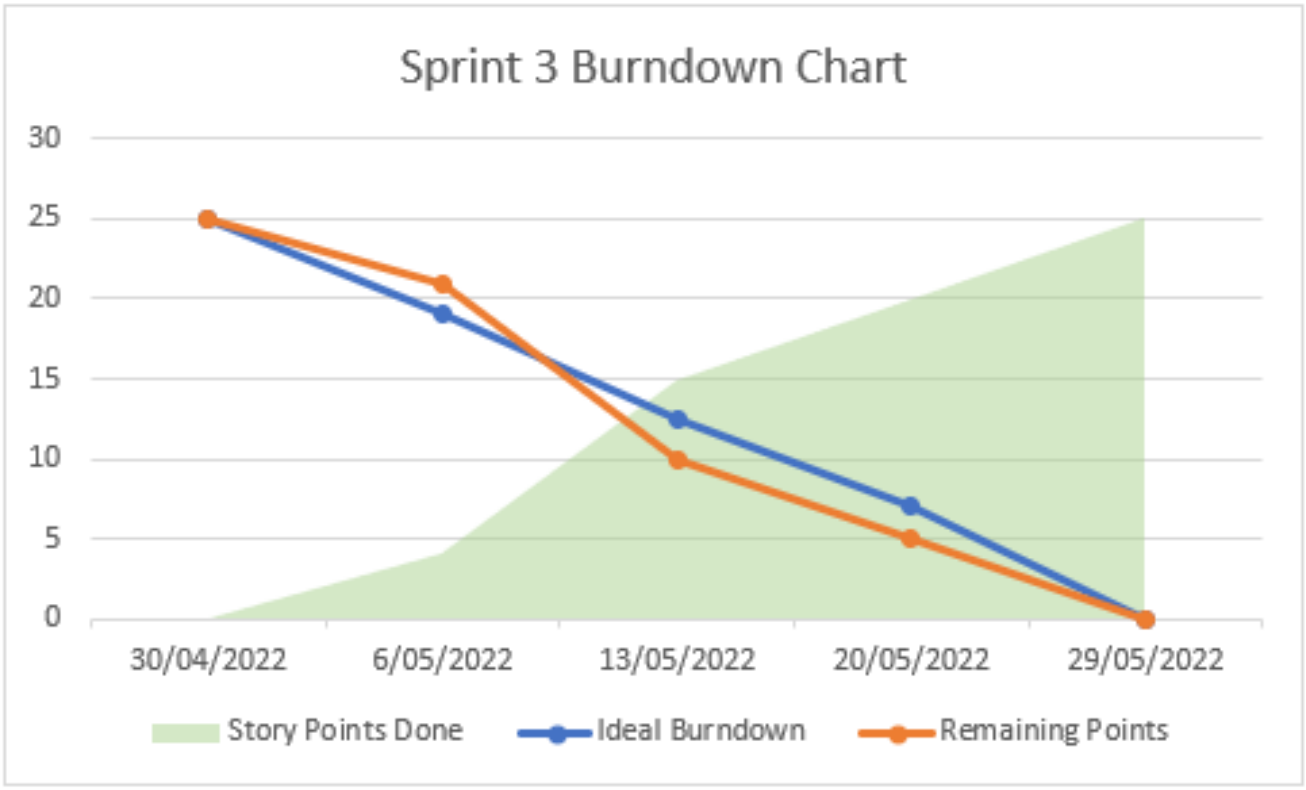
Sprint 2 – 4 weeks (200 hours)

1 story point = 8 hours

25 story points

Epics	User	User Story ID	User Stories	MoSCoW Priority	Story Points(2 hour/pt)
E3 - Data Visualisation **(out of scope)**	As an athlete, I want to view my sports data on the website, so that it is easier for me to see what degree I achieved and dig into details to check my performance on different targets.				
	Athlete	#09	• I want to view basic sports data on my activity, such as the duration of the activity, calories burn, distance, based on the specific activity transferred. 1. Ensure the User input is able to request data with the Frontend (React) 2. Backend (Java) is able to retrieve the data required from the MongoDB (Database) 3. Backend (Java) is able to produce output into the Frontend (React) 4. Frontend is able to display the necessary data from the Backend	Must have	4
	Athlete	#10	• I want to be able to see a summary of my activities in a form of a dashboard to be able to see clearly what I have achieved overall 1. Create a statistic graph over time to show calories burnt over each day 2. Create a relational graph of two statistics to show relational data between them	Could have	2
	As an athlete, I am able to see the specific details of each of my individual **Swim** activities				
	<None>	#11	A backend retrieve controller (getSwimmingActivityByAccessToken) is to be created in the backend database for easier retrieval	Could have	3
	Athlete	#12	I can see my **Time** taken for each of my Swim activities	Could have	0.5
	Athlete	#13	I can see my **Distance** taken for my Swim activities	Could have	0.5
	Athlete	#14	I can see my **Average Speed** for my Swim activities	Could have	0.5
	Athlete	#15	I can see my **Calories** consumed for my Swim activities	Could have	0.5
	Athlete	#16	I can see my **Pace** in time per 100m for my Swim activities	Could have	0.5
	As an athlete, I am able to see the specific details of each of my individual **Cycling** activities				
	<None>	#17	A backend retrieve controller (getCyclingActivityByAccessToken) is to be created in the backend database for easier data retrieval	Could have	3
	Athlete	#18	I can see my **Time** taken for each of my Cycling activities	Could have	0.5
	Athlete	#19	I can see my **Distance** taken for my Cycling activities	Could have	0.5
	Athlete	#20	I can see my **Average Speed** for my Cycling activities	Could have	0.5
	Athlete	#21	I can see my **Calories** consumed for my Cycling activities	Could have	0.5
	Athlete	#22	I can see my **Average Cadence** for my Cycling activities	Could have	0.5
	Athlete	#23	I can see my **Average Heart Rate** for my Cycling activities	Could have	0.5
	Athlete	#24	I can see my **Elevation** for my Cycling activities	Could have	0.5
	As an athlete, I am able to see the specific details of each of my individual **Run** activities				

	<None>	#25	A backend retrieve controller (getRunningActivityByAccessToken) is to be created in the backend database for easier data retrieval	Could have	3
	Athlete	#26	I can see my **Time** taken for each of my Run activities	Could have	0.5
	Athlete	#27	I can see my **Distance** taken for each of my Run activities	Could have	0.5
	Athlete	#28	I can see my **Average Speed** for my Run activities	Could have	0.5
	Athlete	#29	I can see my **Calories** for my Run activities	Could have	0.5
	Athlete	#30	I can see my **Pace in time per km** for my Run activities	Could have	0.5
	Athlete	#31	I can see my **Average Cadence** for my Run activities	Could have	0.5
	Athlete	#32	I can see my **Average Heart Rate** for my Run activities	Could have	0.5
	Athlete	#33	I can see my **Elevation** for my Run activities	Could have	0.5



According to the burndown chart, our team completed the required story points on May 29, 2022. A total of 25 story points were completed in this sprint. As expected, we completed the development tasks on time.

During the first week of development, all progress was a bit slow as the backend was not finished yet. After the team completed all API development in the second week, the front-end development made great progress. Finally, we completed the development of the back-end API and front-end dashboard on time.