Handover

About

This document serves as a formal handover report of the project to another development team to take over and continue if needed.

- Deployment Guideline
- Final Presentation
- Final Release ZIP File
- Development Change Log

Background

Coaching-Mate

The **CoachingMate** is a system for the coach to deliver training sessions to their athletes. And it's now we are specifically required to enable Athletes to review all their Activities and Activity Details through Coaching Mate Web Application

Garmin

The **Garmin Connect** is the tool for Tracking, Analyzing and Sharing health and fitness activities from **Wearable Devices**. So Athletes with Devices like **Garmin Watch** will have their activity data sync to the **Garmin Connect**. And they provide an API that enables us to access those activity data.

- Garmin API
- Garmin Activity API
- Flexible and Interoperable Data Transfer (FIT) SDK

Project Goal

the **GOAL** of our project is about **data integration** with **Garmin API** and syncing the Activity data to the **Coaching-Mate Dashboard**. Our **Coaching-Mate** project is NOT built from scratch, and we are working on the code from the Previous **Garmin API Project**. So our main focus is to maintain and improve the existing platform. And we need to guarantee the data integration from GARMIN and ensure the dashboard display and retrieve the list of activity and activity details from **Garmin Connect**.

System Implementation

Our Scope

The Garmin API is what our project will be focusing on. Garmin is a commercialized product that targets clubs and coaches. Garmin API provides an easy way to integrate a platform directly into their content management website. This project needs to improve the APIs from the existing system, to increase the breadth of activities that are currently pulled by the CoachingMate backend.

Garmin should also be able to connect to external websites so that the workout data can be viewed by other people. Our team is also recommended to develop APIs for Garmin to connect to other external websites.

1. coaching-mate account management
    a. Account register
    b. Account log-in
    c. and log-out
2. Connect **CoachingMate Web Application** with the **Garmin Connect** API
3. Show data from **Garmin Connect** to the Dashboard of the **CoachingMate Web Application**
    a. List of Activities
    b. Activity Details

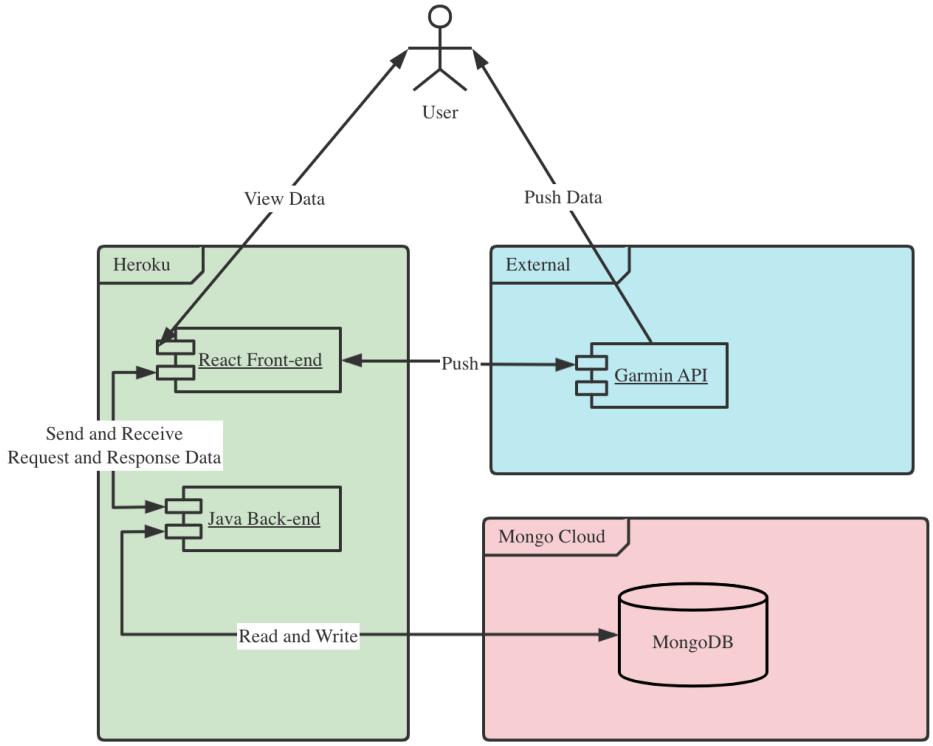
Out of Scope

Our team will not be responsible for data analytics. There are already a lot of functions on the Garmin dashboard. The GA teams will not be involved with the website design. This project will not be involved with how to visualize the data either.

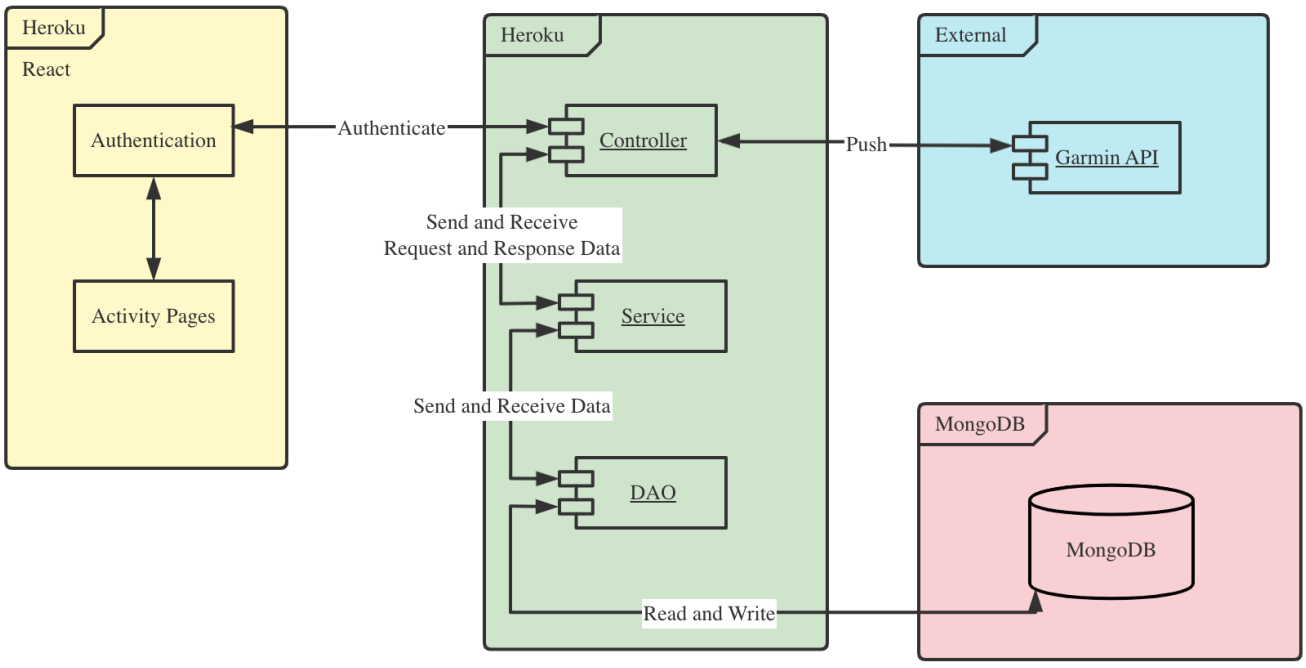
1. Frontend UI Design
2. Data Analysis

System Architecture

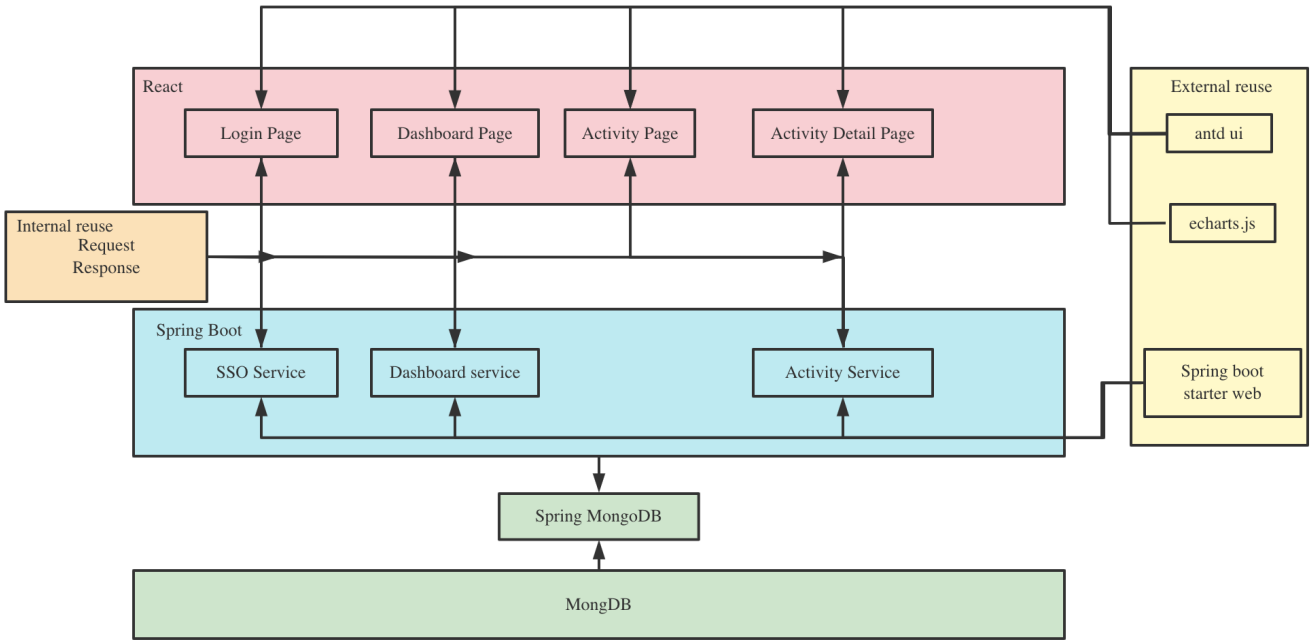
Components diagram



Deployment



System Logical Architecture



Deployment

- Deployment
- Development Change Log

Development Details

API

We made a major update to the front-end and back-end interactions. In the existing system, there is no unified front-end and back-end interaction specification, and error handling must be written every time. In the new system, the front-end and back-end API interactions will take the following format:

```
{
  "statusCode": 200,
  "message": "Success",
  "data": null
}
```

Field	Description
statusCode	The status code of this request, except 200, is a request error. The front end only needs to check this field to determine whether the request is successful.
message	The field for error requests. When the request is wrong, the front end only needs to display this message.
data	This field backend can hold any Java object. such as entity classes. Or the backend can return List<~>. If the backend wants to return different objects, it can use Map<String, Class>.

Frontend Code

GitHub Access

https://github.com/COMP90082SM12022/GA-Boxjelly/tree/master/src/cm-frontend-react

Code Structure

```
public                    # static sources
   favicon.ico            # Website LOGO
   index.html             # html Template
src                       # Frontend source code
   api                    # All requests
   assets                 # images, fronts
   components             # public componets
   config                 # global config
      menuConfig.js       # menu config
      routeMap.js         # router config
   lib                    # others resources
   mock                   # mock data for testing
   store                  # global store
   styles                 # global styles
   utils                  # global tool methods
   views                  # views all pages
   App.js                 # entry page
   defaultSettings.js     # default setting
   index.js               # source entry
.env.development          # development config file
.env.production           # production config file
config-overrides.js       # custom config for webpack of CRA
deploy.sh                 # CI deployment script
.travis.yml               # auto CI config
package.json              # package.json
```

Important files or dirs

- src/api/*.js: All backend APIs are declared here, and other files only need to call the declared functions.
- src/config/menuConfig.js&routeMap.js: Menu and permissions configuration files, menus and permissions need to be configured here before they can be accessed.
- src/store: user information and cookies
- src/views: Pages. After creating a new page file, you need to configure it in the routing before you can access it.
- src/utils/request.js: Advanced encapsulation of Axios, introduced in api/*.js. Contains global exception handling.

Backend Code Structure

GitHub Access: https://github.com/COMP90082SM12022/GA-Boxjelly/tree/master/src/coaching-mate

```
docs                                              # Some documents
lib                                               # other libraries
backup                                            # Backup files
src                                               # Source code
   main                                          # Product source code
      java/coachingmateanalytics/coachingmate
         common                                  # Public components
            config                               # Config files (Including http header config, swagger config, etc.)
            intercepter                          # Session and web intercepter
            utils                                # Request and Response classes
         controller                               # Interact with front-end
         dao                                      # Interact with database
         entity                                   # Entity classes
         service                                  # Handle bussiness logic
         CoachingmateApplication.java             # Application entry
      resources                                    # Config files
   test                                            # Test code
pom.xml                                           # Java library
```

Important files or dirs

- src/main/java/coachingmateanalytics/coachingmate/common/utils: public components. Contains Response, ErrorResponse. Response /ErrorResponse: Follow the Restful API specification.
- src/main/java/coachingmateanalytics/coachingmate/common/utils/ResponseHandler.java: Global return processing. All return values will be encapsulated as Response or ErrorResponse through this method.
- src/main/java/coachingmateanalytics/coachingmate/common/enums: Response enumeration class. Response contains status code and message, which are set here.
- src/main/java/coachingmateanalytics/coachingmate/common/annotation: Custom annotations. Existing annotation: AuthCheck, used to check whether the token is valid.

Deployment Guideline

Deployment Guideline

There are multiple ways to deploy. The following describes the main two ways: deploying on your own server and deploying on the cloud.

Deploying on own server

Deploying on your own server requires checking the running environment:
The following are the recommended operating environments:

- Back-end: JDK 1.8+, Maven 3.6.3+
- Front-end: node v14.17.6+, react 16.9.34+
- Database: MongoDB 5.0.8

Install Softwares

Install Java

Official tutorial for JDK installation

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.

Install node

Offical tutorial for nodejs installation

Install MongoDB

Official tutorial for mongoDB installation

Run this project

Configuration files

backend: src/coaching-mate/src/main/resources/application-dev.properties

create your MongoDB database with {database name}, and set the database name and password to

```
spring.data.mongodb.uri = mongodb+srv://{database name}:{password}@cluster1.mjm2t.mongodb.net/coachingmate?retryWrites=true&w=majority
```

front-end: src/utils/request.js - change the api url.

Backend deployment

1. run mvn install. All files will be packaged into jar files.
2. Execute on the command line: java -jar ***.jar

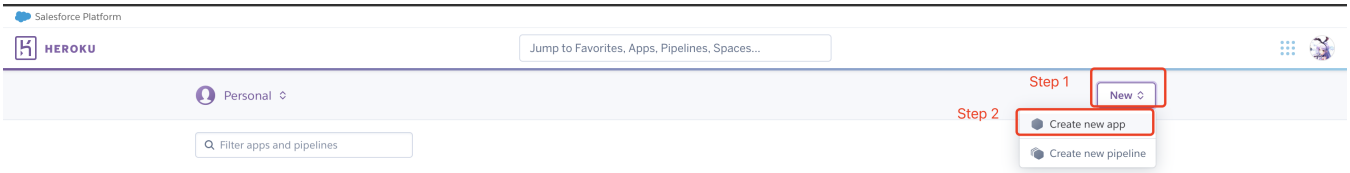
Frontend deployment

1. run npm build. All files will be packaged into dist dir
2. copy the dist dir to the root of the website server(for example: nginx)

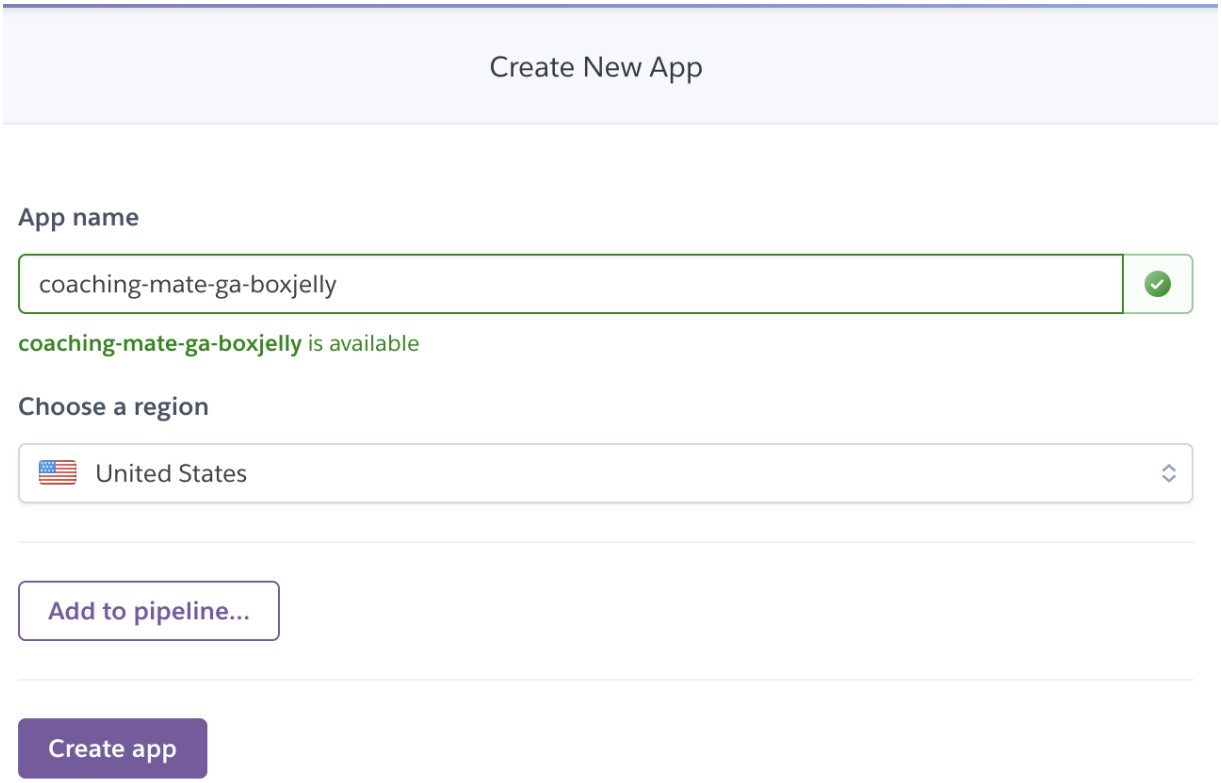
Deploying on Heroku

There are several steps we need to do to run our app on heroku

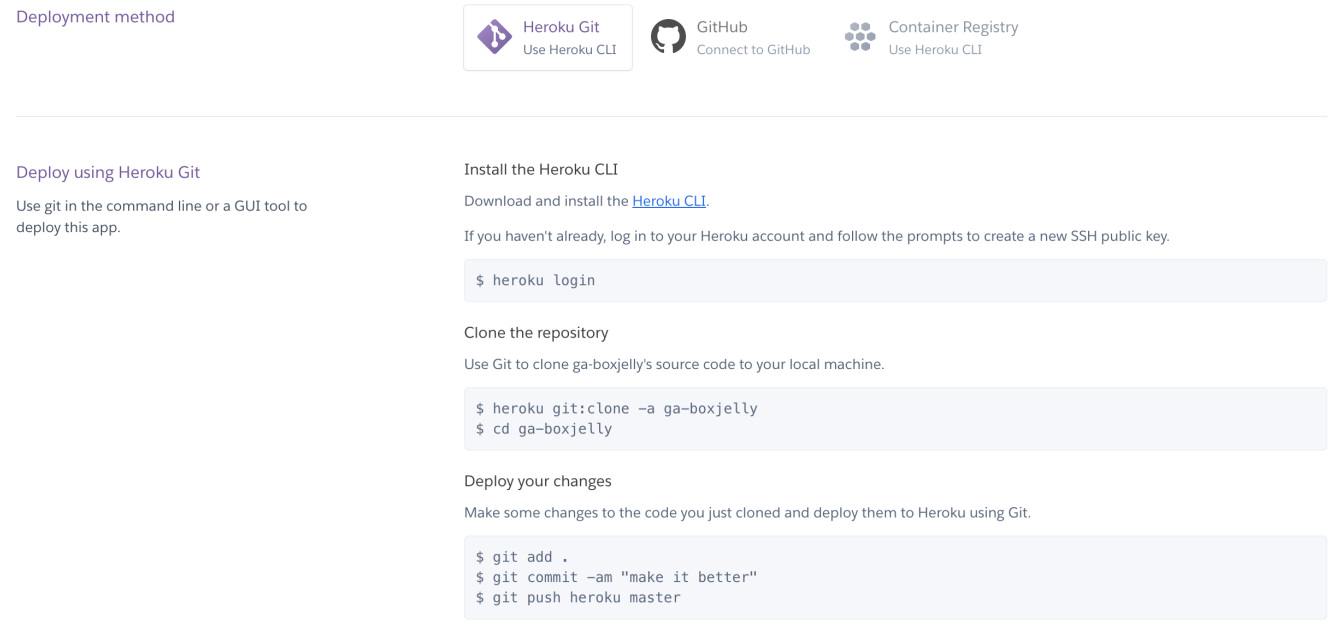
1. create a new application on heroku



2. text our app name and region

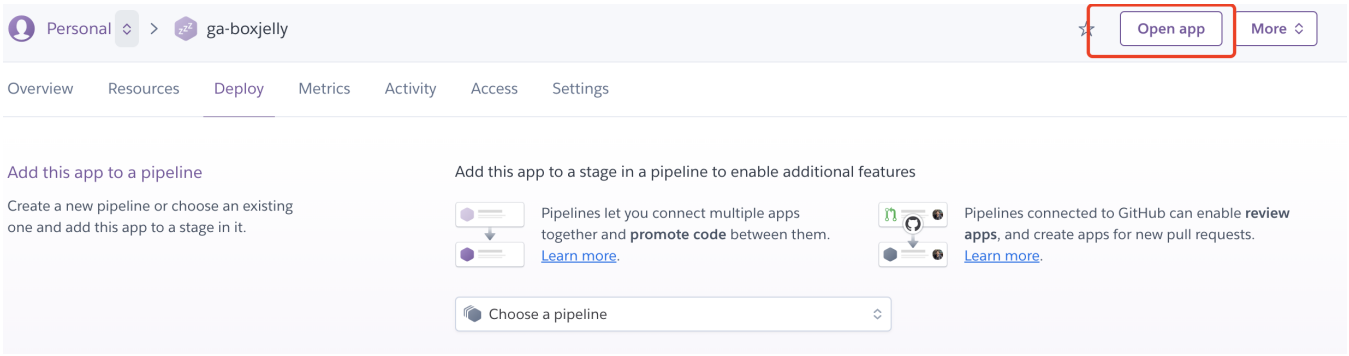


3. Select Heroku git and install heroku cli



4. Run above-mentioned command one by one

5. find our app and click the Openapp button

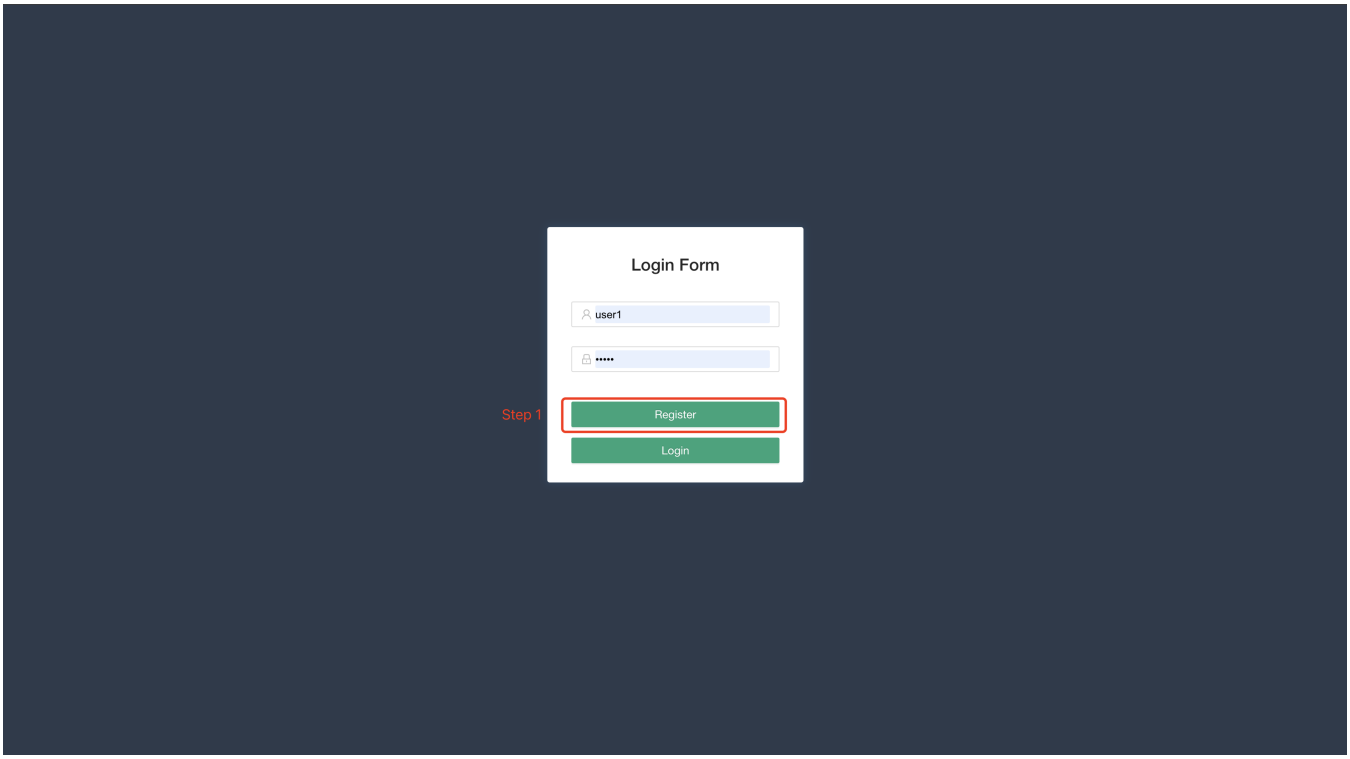


6 find our URL in the build log

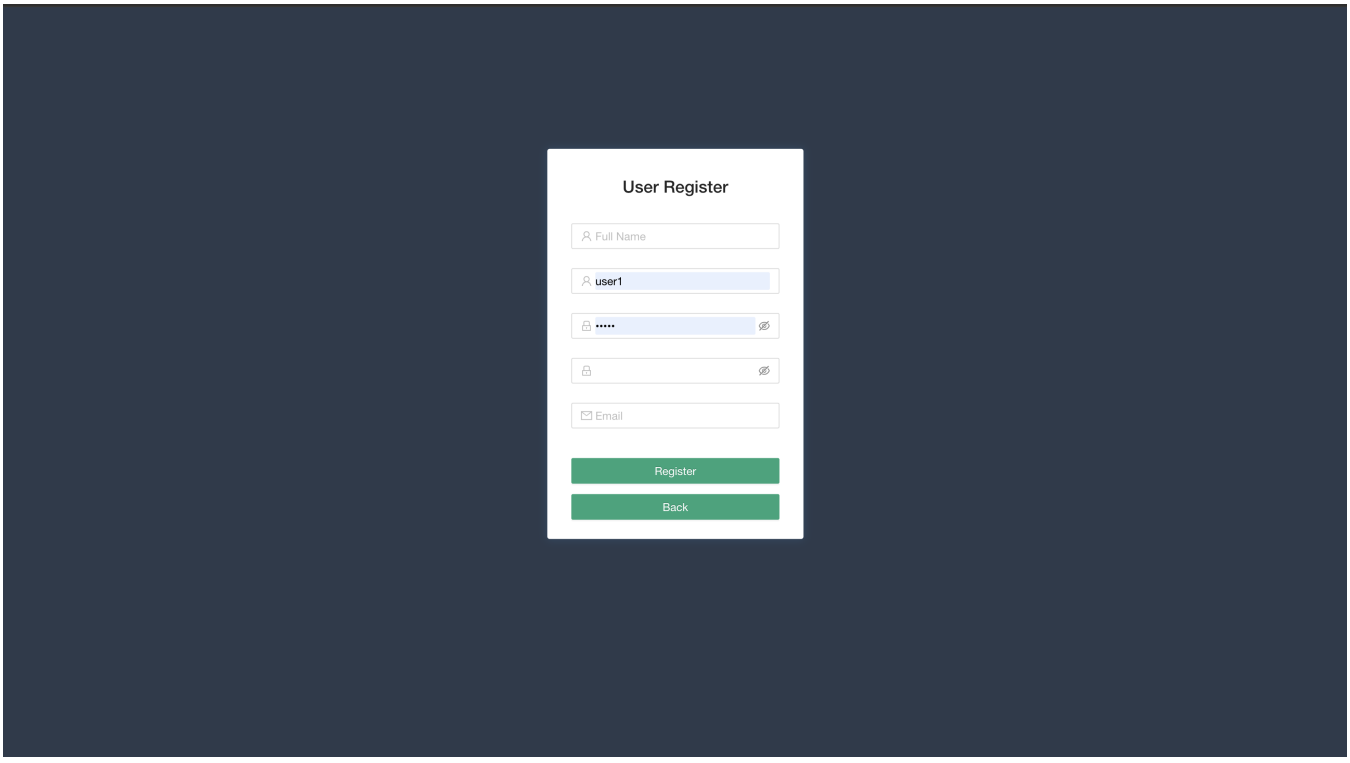
User Guide

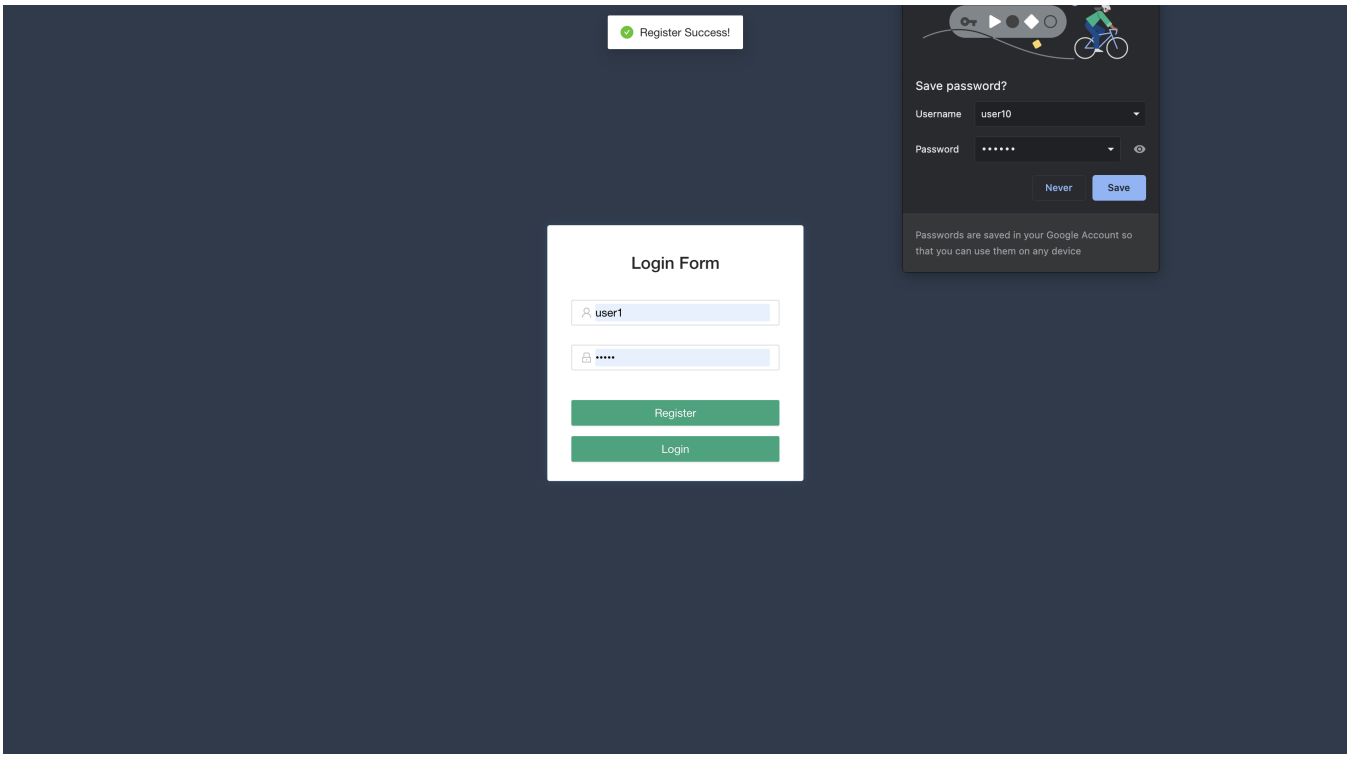
User Register

Step 1: Click the register button



Step 2: Input full name, username, password twice and email address(in the correct format)

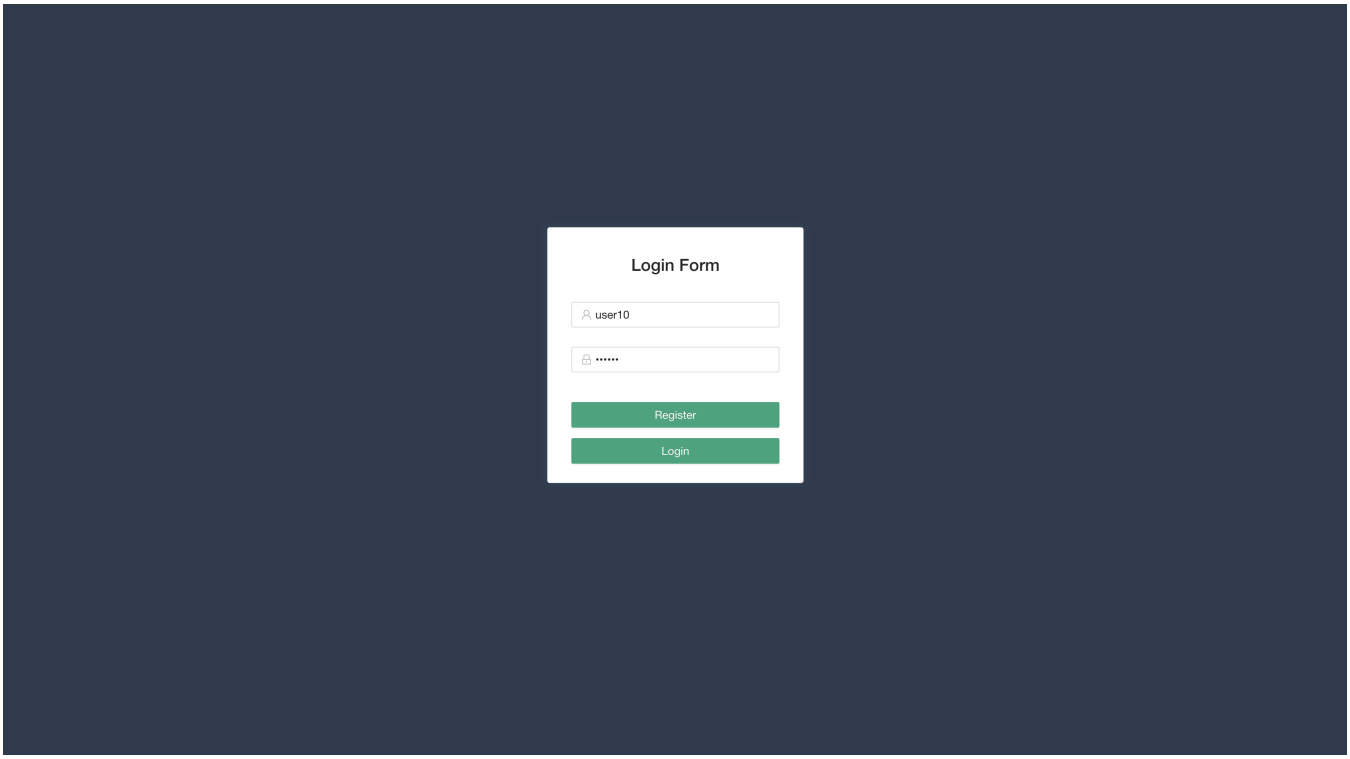




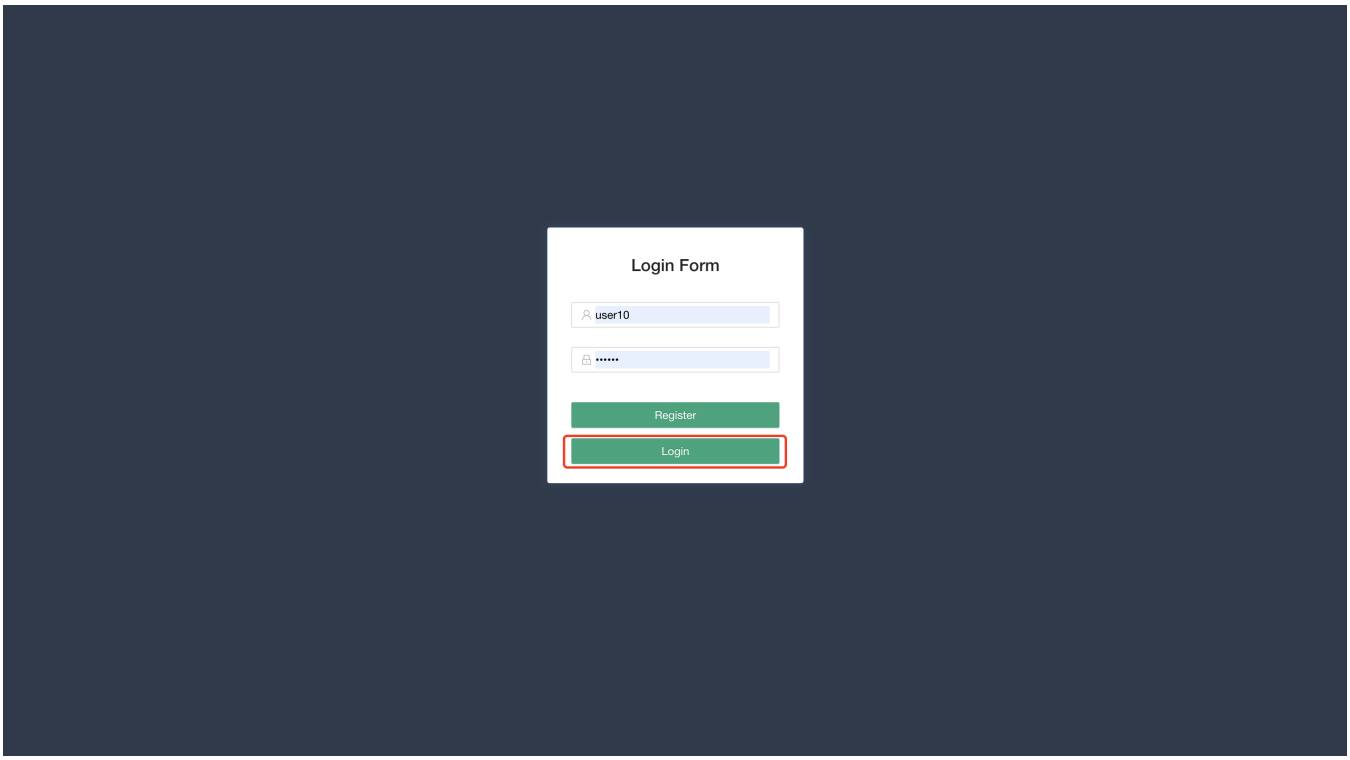
Then, can log in with the username and password.

User Login

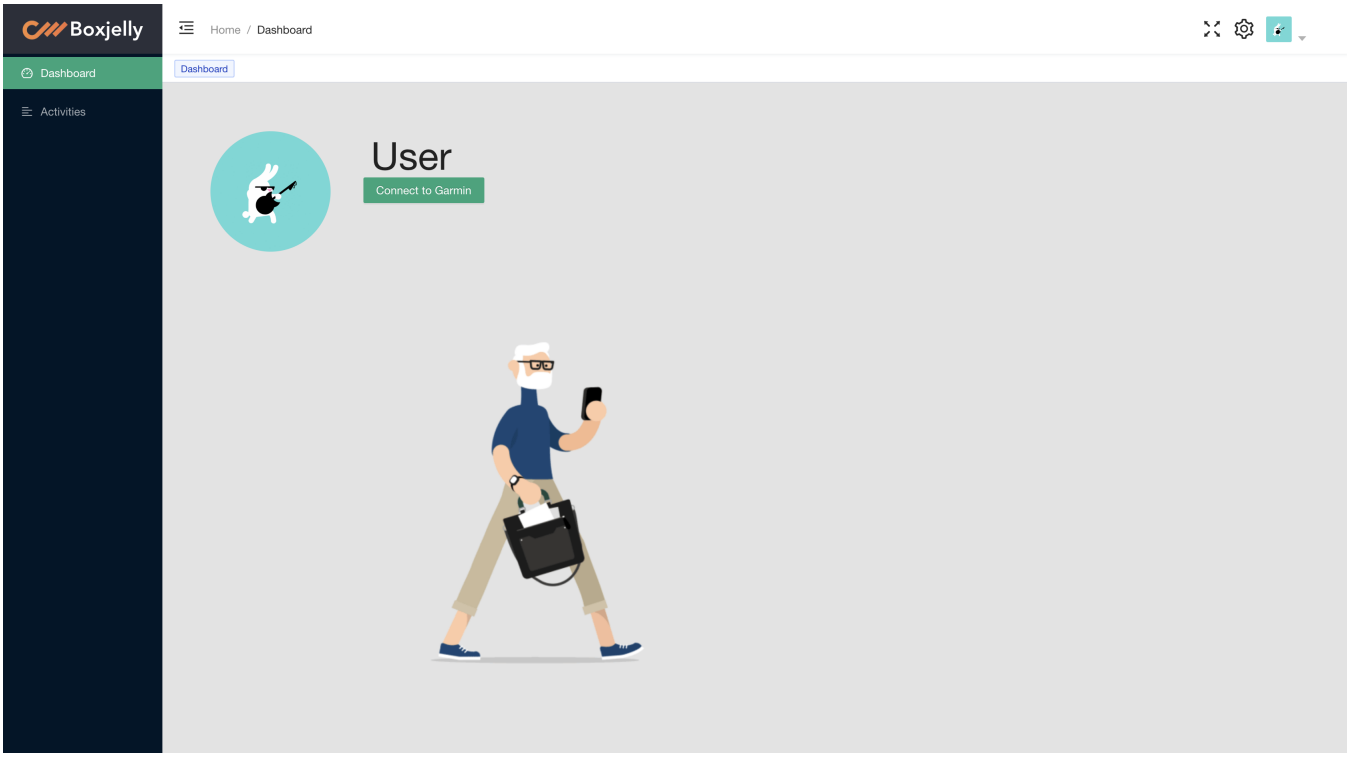
Step 1: Input the username and password



Step 2: Click the login button

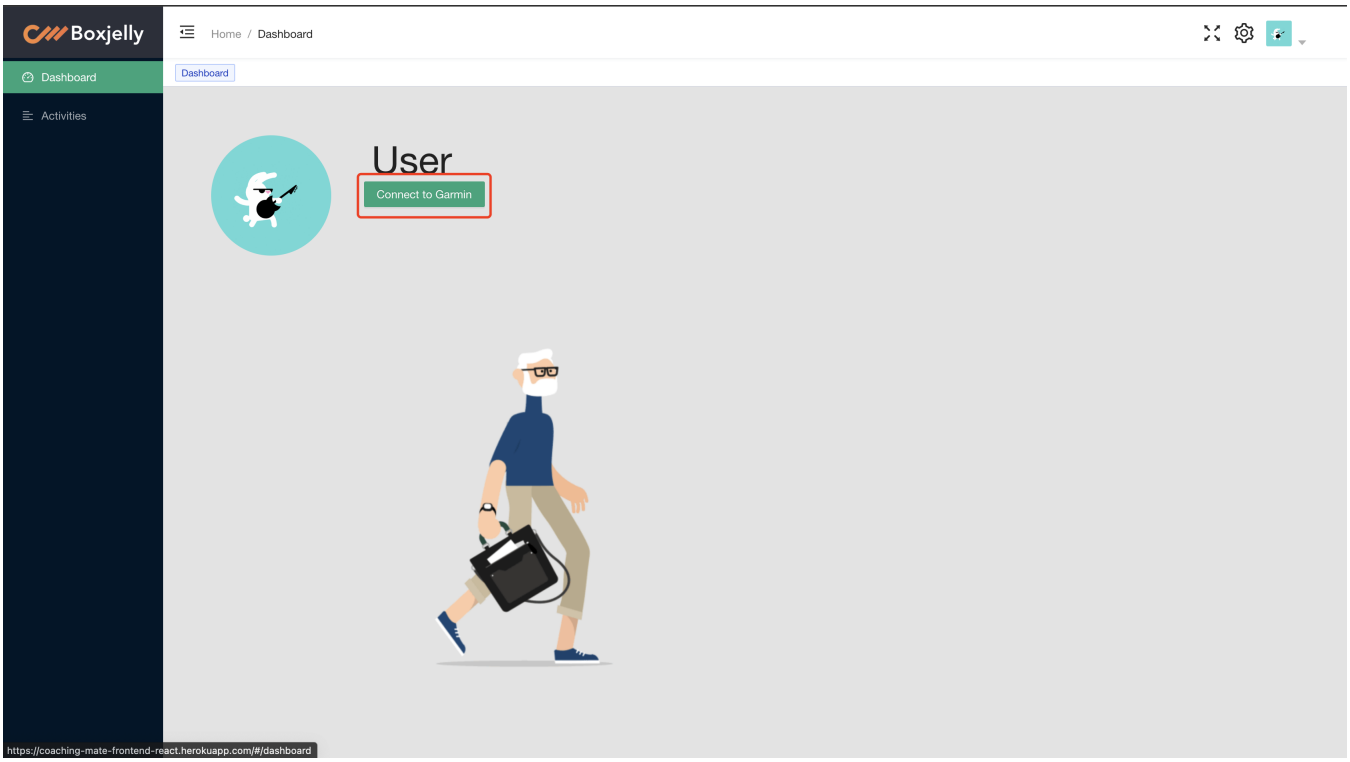


Then, will get to the user main page.

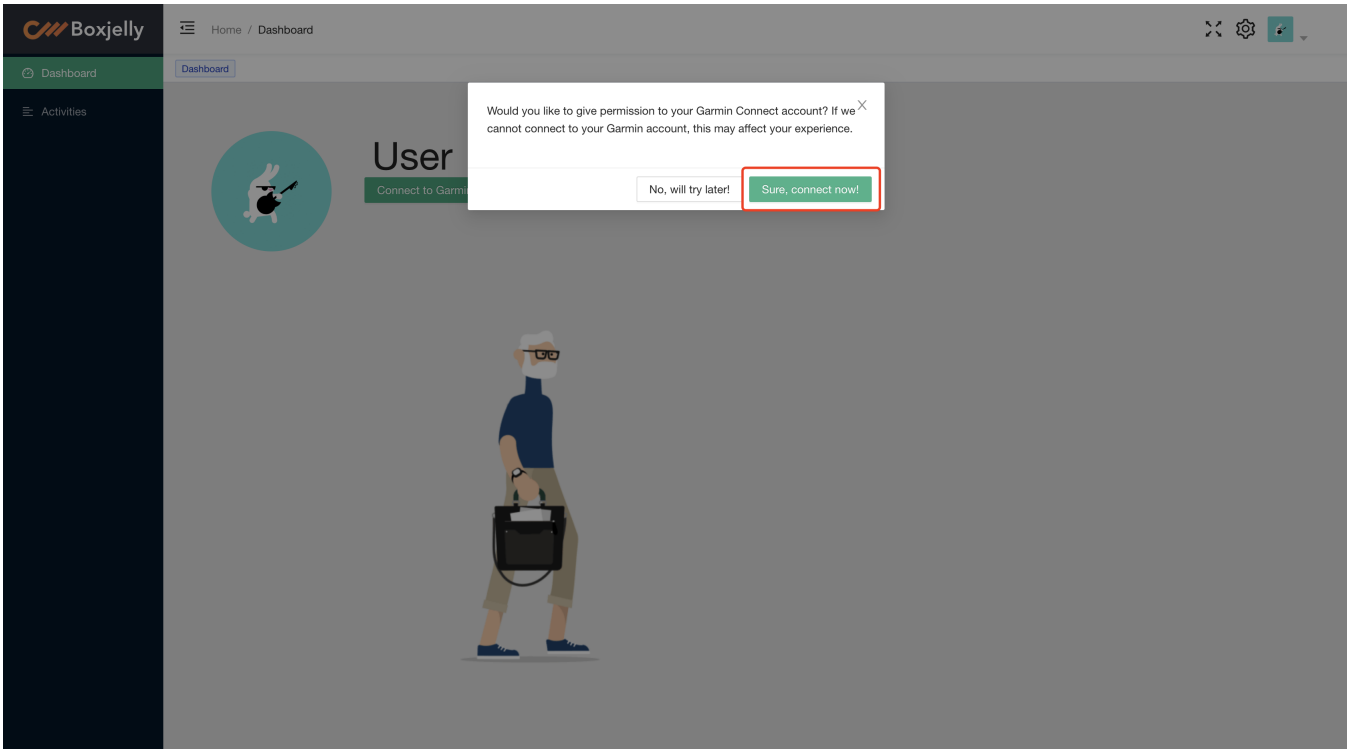


Connect to Garmin

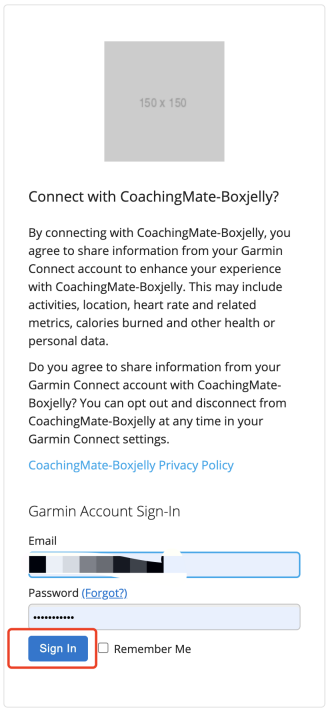
Step 1: Click "Connect to Garmin" button



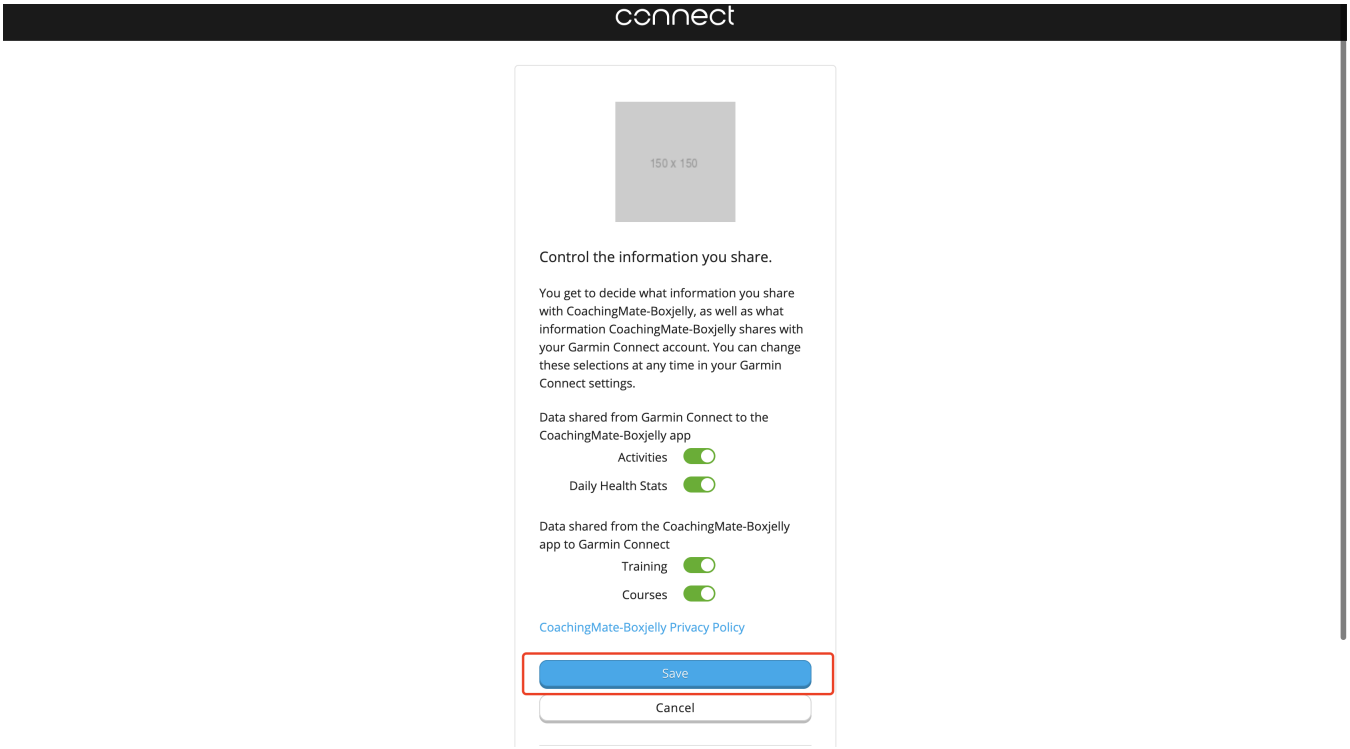
Step 2: Click Sure, "connect now" button



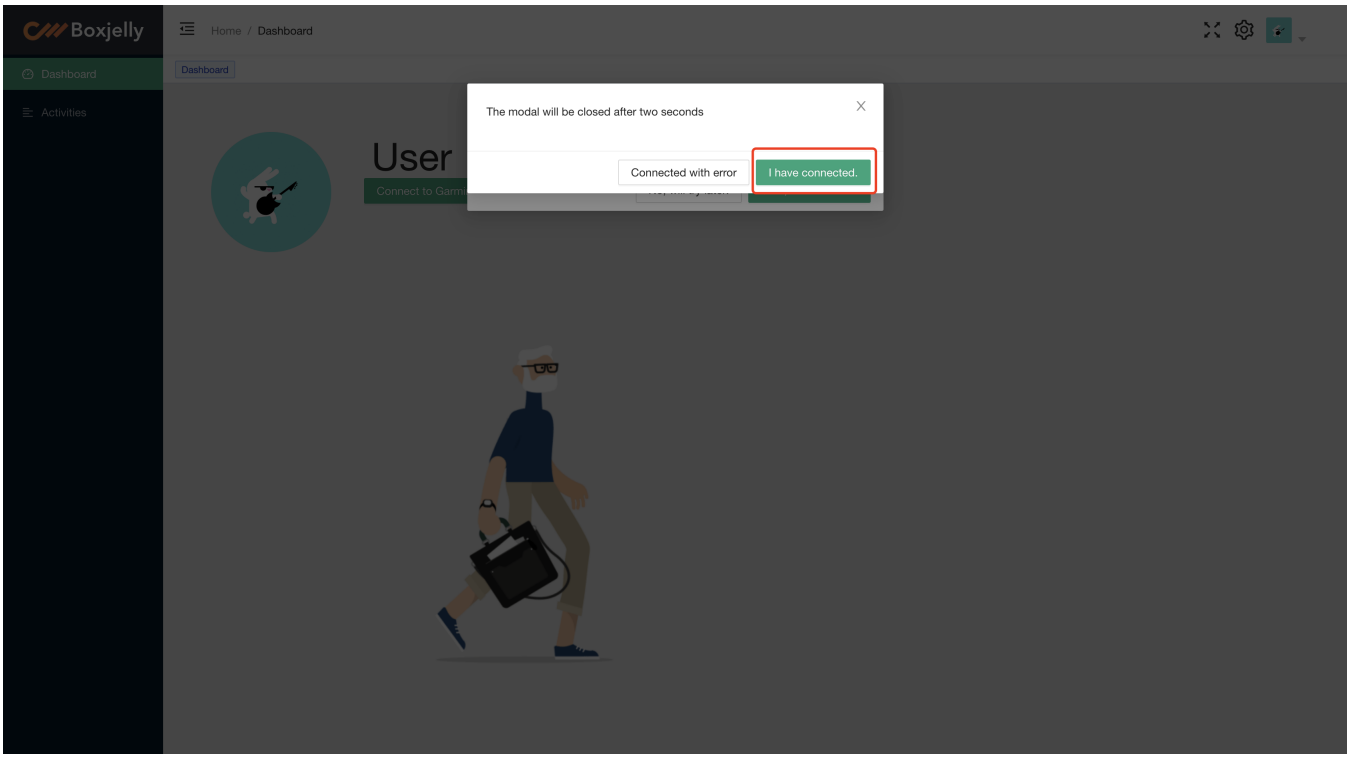
Step 3: Sign in to Garmin



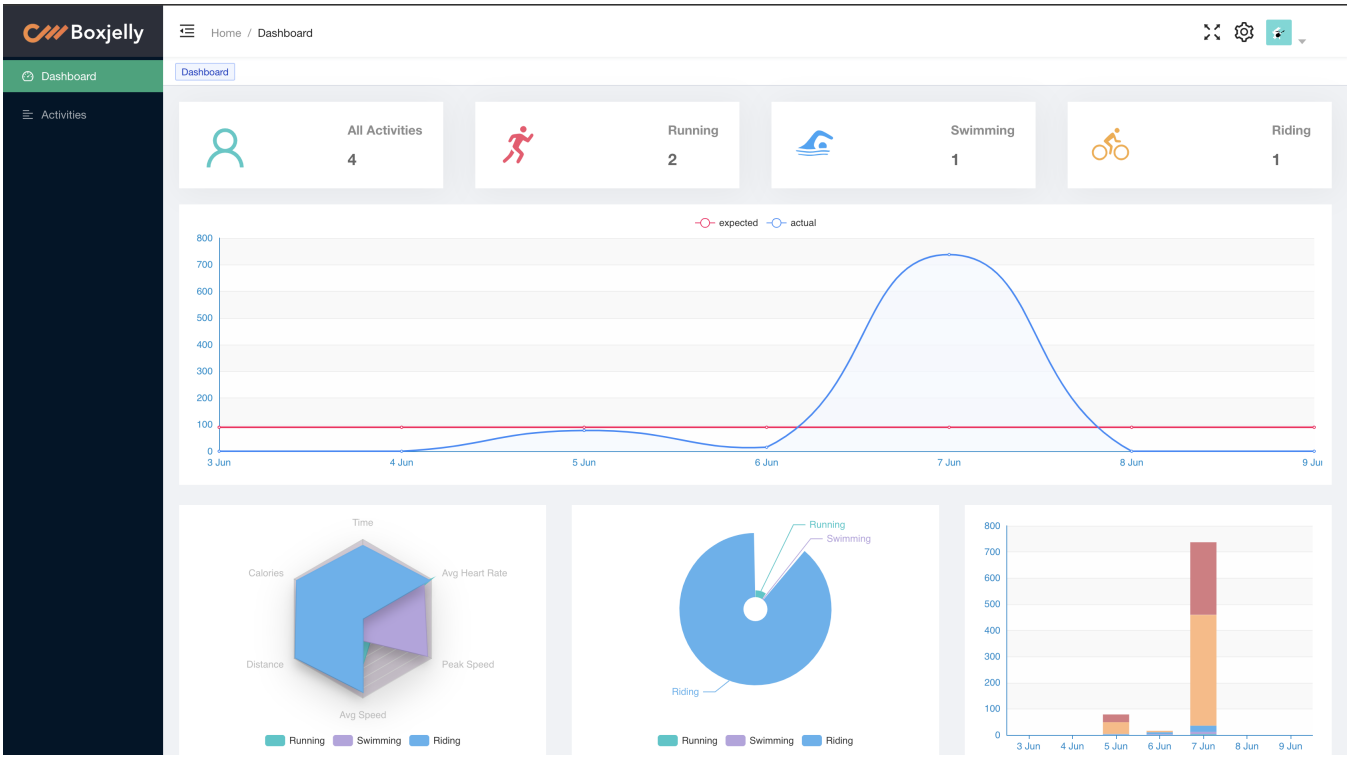
Step 4: Agree with the policy



Step 5: Click the "I have connected" button.



Then will enter the dashboard page.

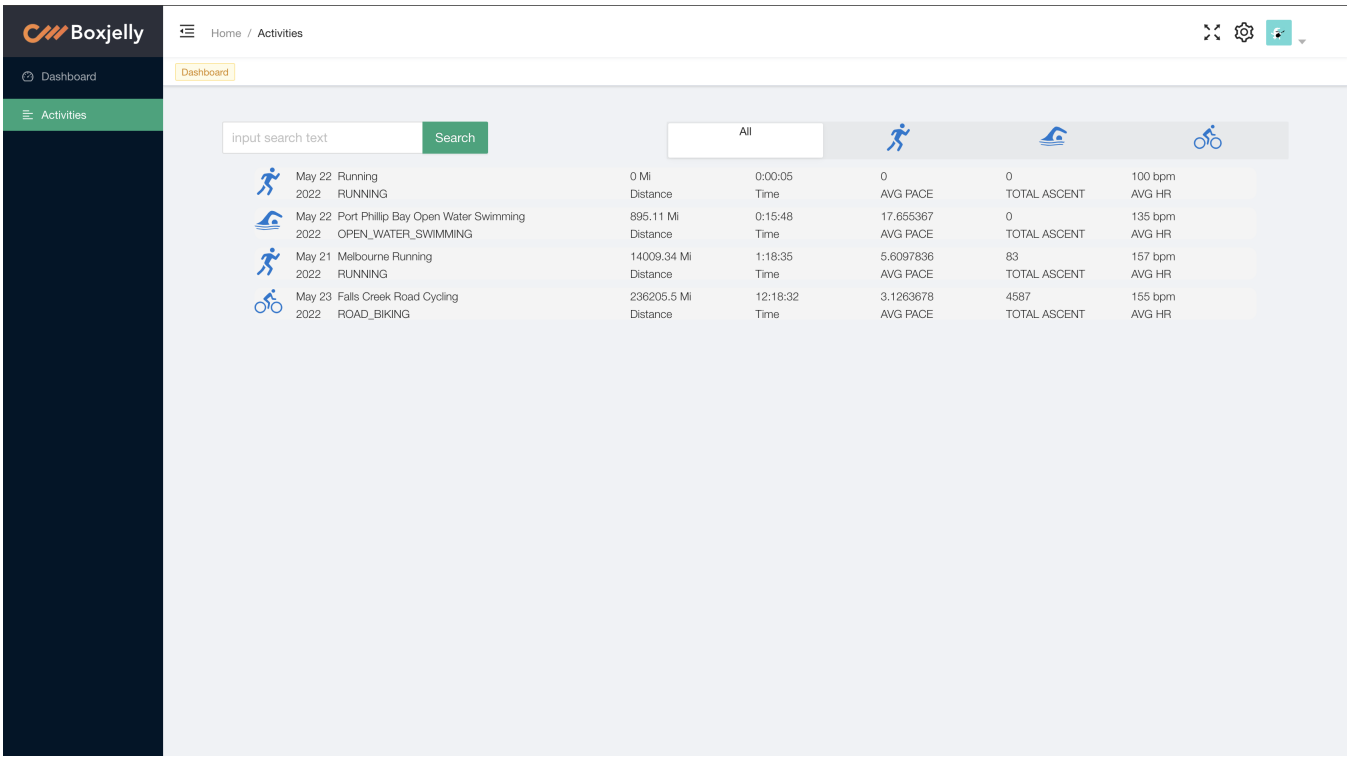


Sync and View Activity Data

Step 1: Start an activity on Garmin watch.

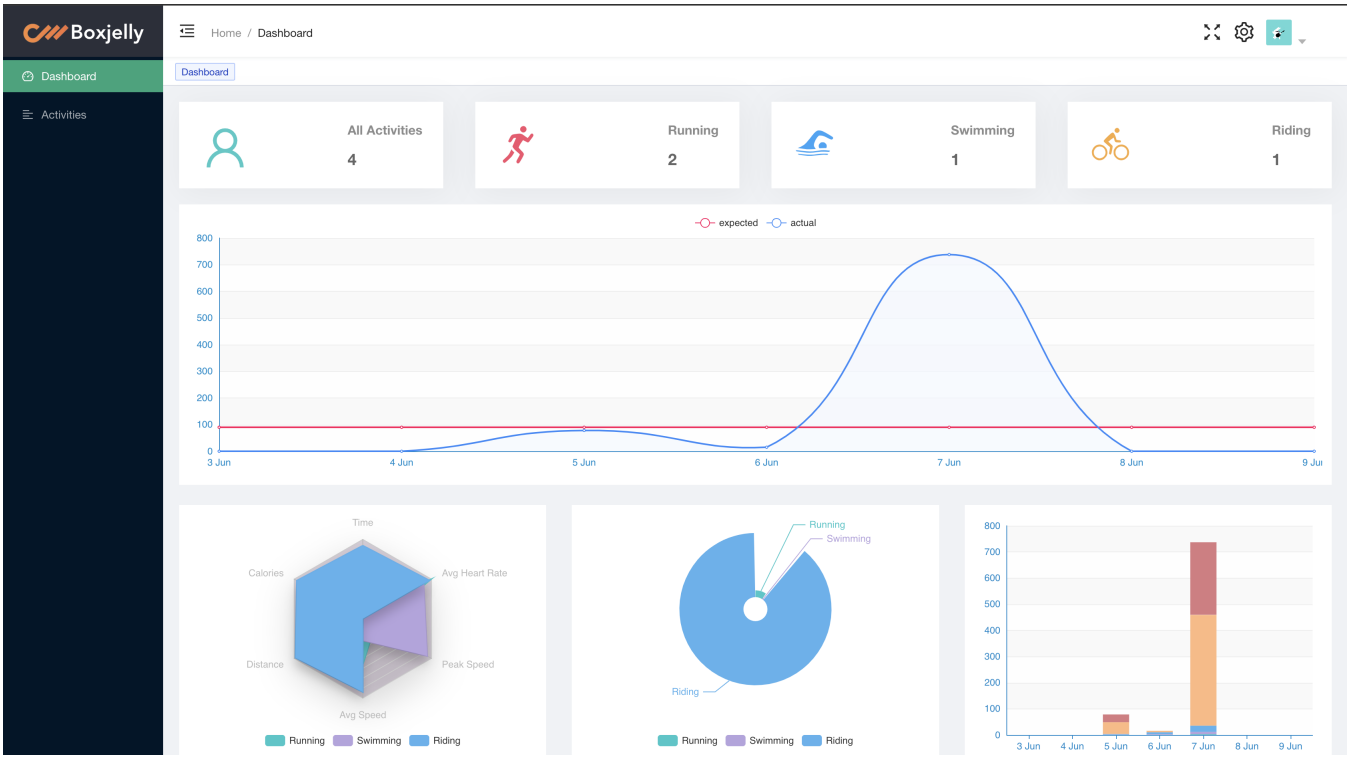
Step 2: When an activity is finished, the activity data will sync automatically

Step 3: Then can view activities on activities page



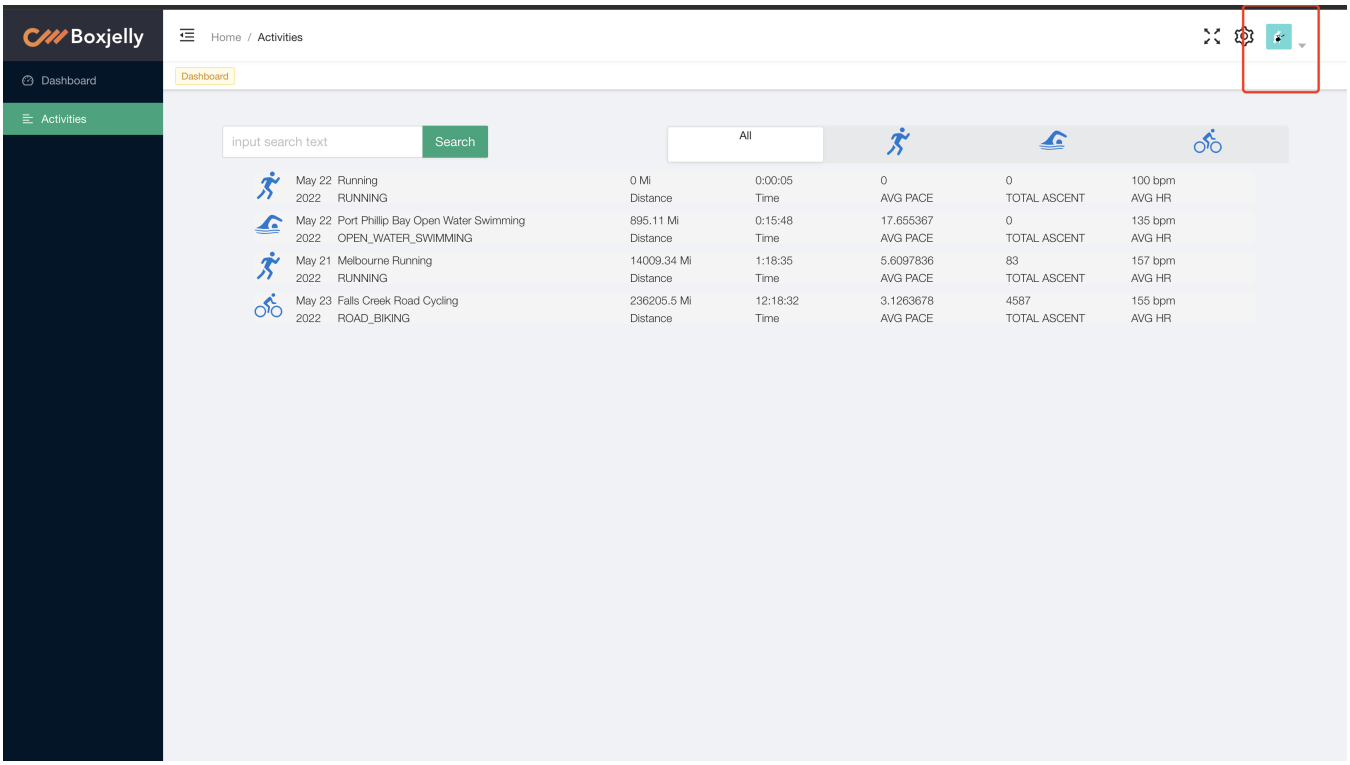
View Dashboard

If there is enough data, the dashboard can display beautiful charts.

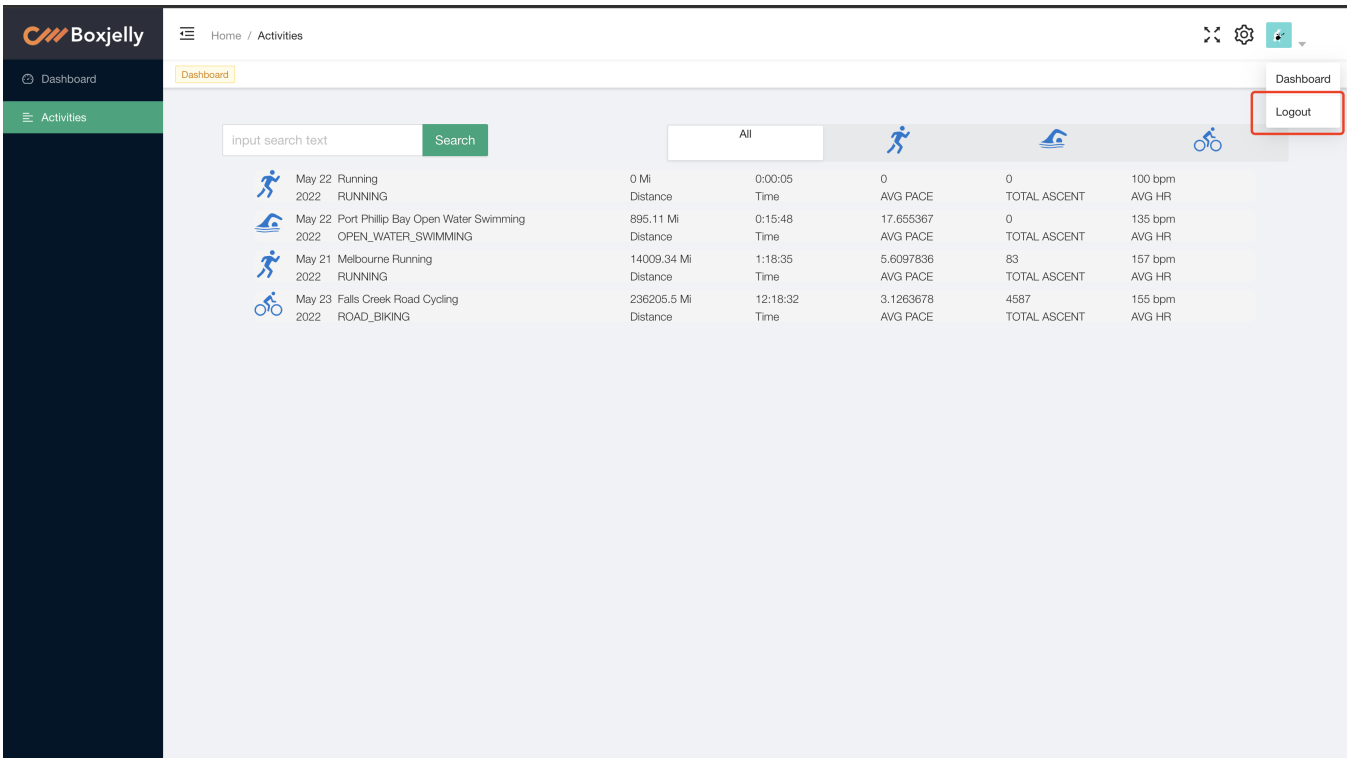


Log out

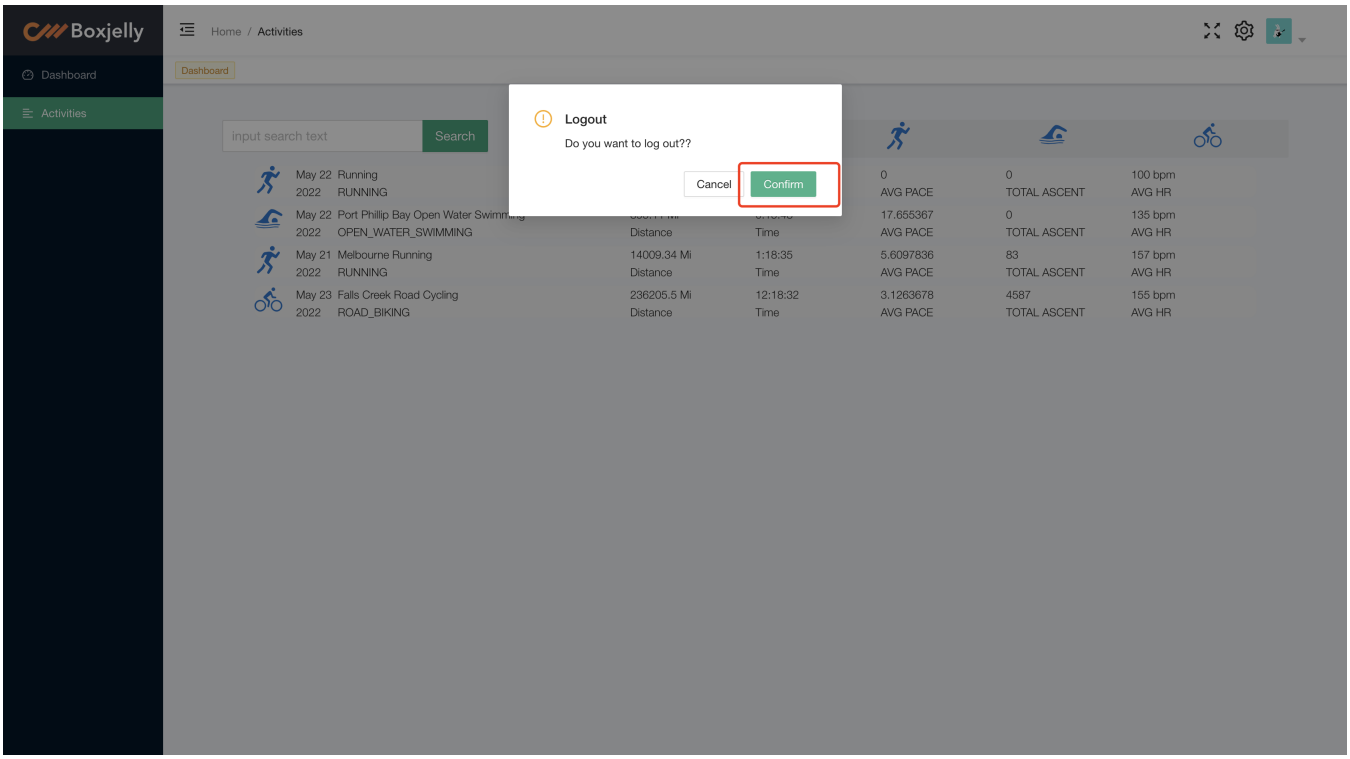
Step 1: Click on the avatar in the upper right corner.



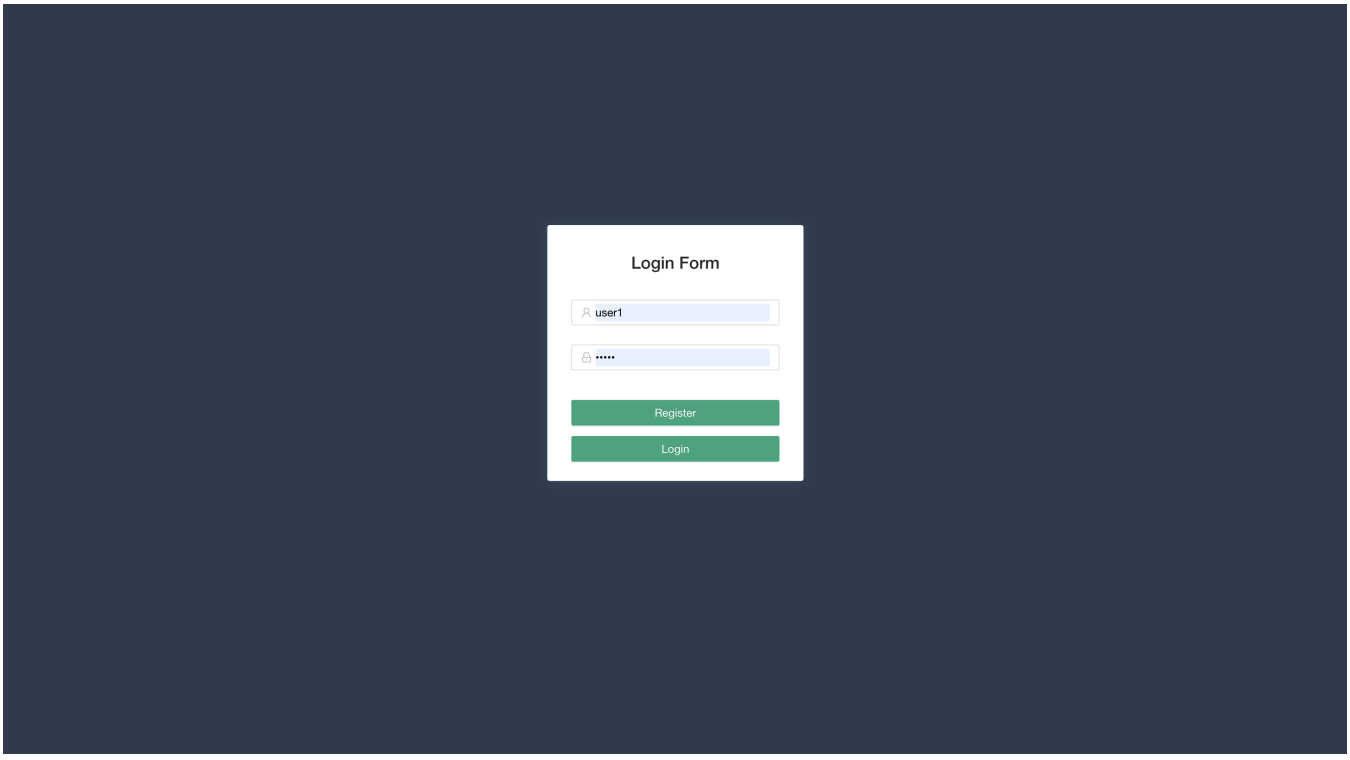
Step 2: Click the "Log out" button



Step 3: Click the "Confirm" button



Then will log out the system



Sprint 4 Release

Summary

The Final Sprint goal is to finalize the project and hand over the final deliverable to the clients.

- [x] Project Documentation Final Update 📁
- [x] Summarized Development and Change Logs 💿
- [x] Summarized Release History 🧾
- [x] Deployment guidelines 🐪
- [x] Final Project Presentation with live DEMO 🖥️ (YouTube link)
- [x] Final Release of the project 📦

Release History

Issue Type	Issue ID	Issue Title
Sprint 1		
Project Inception & Requirement Documentation	#5	Project Background
	#7	Personas
	#6	Motivational Model
	#4	Confluence Setup and Structuring
	#8	User Stories
	#3	Product Backlog
Sprint 2		
Develop Functional Requirement	#31	EP01-01 Account Register
	#10	EP01-02&03 Account login & Logout
	#9	EP02-07&08 Garmin API Connect & Disconnect
	#30	EP02-09 Options (Yes/No) for Garmin API Connection
Research for Decision Making	#22	Figure out what Data is involved during API Connection
	#21	Research On API
Basic Development Setup	#28	Frontend Code Environment Setup
	#17	Backend Code Environment Setup
	#36	BUG: Error on Garmin Push Module

Bug & Fixing	#33	BUG: Error RESTFUL API Post Request not work
	#27	BUG: Activity Data Fail to send from Garmin Watch to Garmin API
	#25	BUG: White Page Error in Swagger UI
	#15	BUG: Activity Files Upload Failed
Acceptance Testing	#20	Acceptance Testing: EP01-01 Account Register
	#18	Acceptance Testing: EP01-02&03 Account Login & Logout
	#13	Acceptance Testing: EP02-07&08 Garmin API Connect & Disconnect
Sprint 3		
Develop Functional Requirement	#14	EP03-10: Activity Dashboard Page
	#44	EP03-11: EP03-11 Activity Detail Page
Acceptance Testing	#11	Acceptance Testing: EP03-10 Activity Dashboard Page
	#12	Acceptance Testing: EP03-11 Activity Detail Page
Bug & Fixing	#37	BUG: Error on Synchronizing Data to Activity Detail bug-fix
	#53	Change frontend from VUE back to React

> No further functional development made in final sprint 4. Mainly working on final handover (Only Add new Documentations and ZIP project up) Details see: Sprint 4 Change Log