Technical Design

All Rise! Project Topicus x Saxion HBO-ICT

Project members:

* Hassan Babiker (462644)
* Bjarne ten Bolscher (466348)
* Yang Lin (450968)
* Wim Luyendijk (445662)
* Ivo Tasev (469297)
* Giovanni Tjandra (445963)

Contents

[Project background 3](#_Toc44528322)

[Implementing the solution 4](#_Toc44528323)

[Fitbit Integration 4](#_Toc44528324)

[Class Diagram 5](#_Toc44528325)

[Database design - MySQL 6](#_Toc44528326)

[Flowchart of the application 7](#_Toc44528327)

[Fitbit 8](#_Toc44528328)

[Challenges 8](#_Toc44528329)

[Daily goal 8](#_Toc44528330)

[Advice 11](#_Toc44528331)

[Challenges 11](#_Toc44528332)

[Daily goal 11](#_Toc44528333)

[Leaderboard 11](#_Toc44528334)

[Avatar 11](#_Toc44528335)

[Shop 11](#_Toc44528336)

[Security 12](#_Toc44528337)

# Project background

As students of Saxion University of Applied Sciences, currently pursuing the software engineering track (HBO-ICT), we are tasked with the course 2.4 project HBO-IT corp. In this course, we are presented with a chance to work on one of the projects relating to the company Topicus.

As a client, Topicus would like us to work on one of their projects that are going to be further used as a solution on the problems of their healthcare sector. The topic concerns improving health in an office environment in terms of sedentary workflow. What is required of us is to design and develop a solution for this. The high-level requirements for the solution require:

* + A social element
  + A way to compete against colleagues
  + It must be educational
  + The solution must be secure and allow for scalability (expressed in the implementation and readiness for an AWS migration)

# Implementing the solution

Because we were provided a lot of freedom when it comes down to the implementation for the product, we decided to use previously known technology – Vue.js, Express.js and MySQL

This tech stack will allow us to speed up production, skipping a large chunk of research and giving the project an explosive start. However, this is also a negative thing – using the same tools that we’ve used before means that we won’t be expanding our tech knowledge. Instead we will further solidify our already existing knowledge, resulting in a more permanent and well-developed solution to our client.

# Fitbit Integration

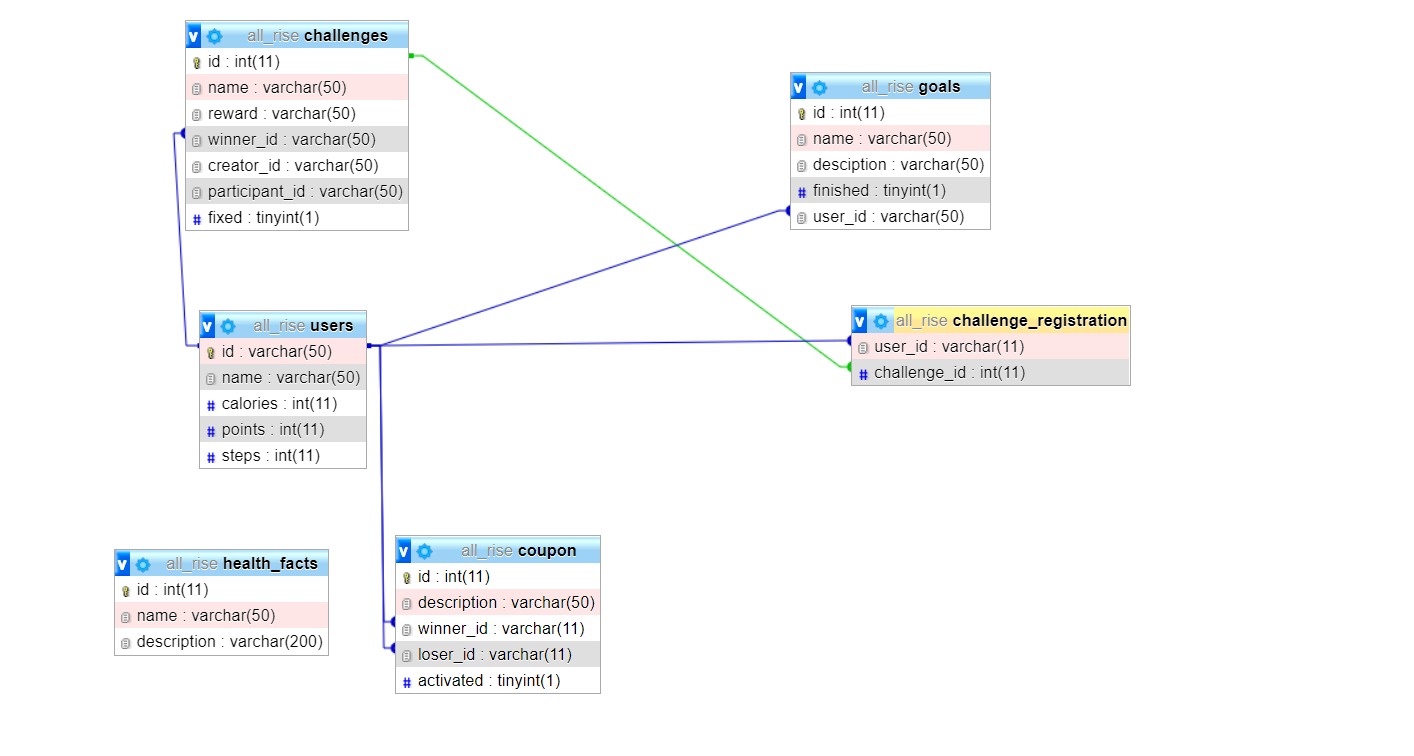
A large part of the functionality will depend on integrating the Fitbit API. Users will have their Fitbit accounts directly connected with our web platform, allowing us to manipulate their data through the Fitbit API. With this we can transform the data into challenges, goals and health tips for the user.

# Class Diagram

A screenshot of a cell phone

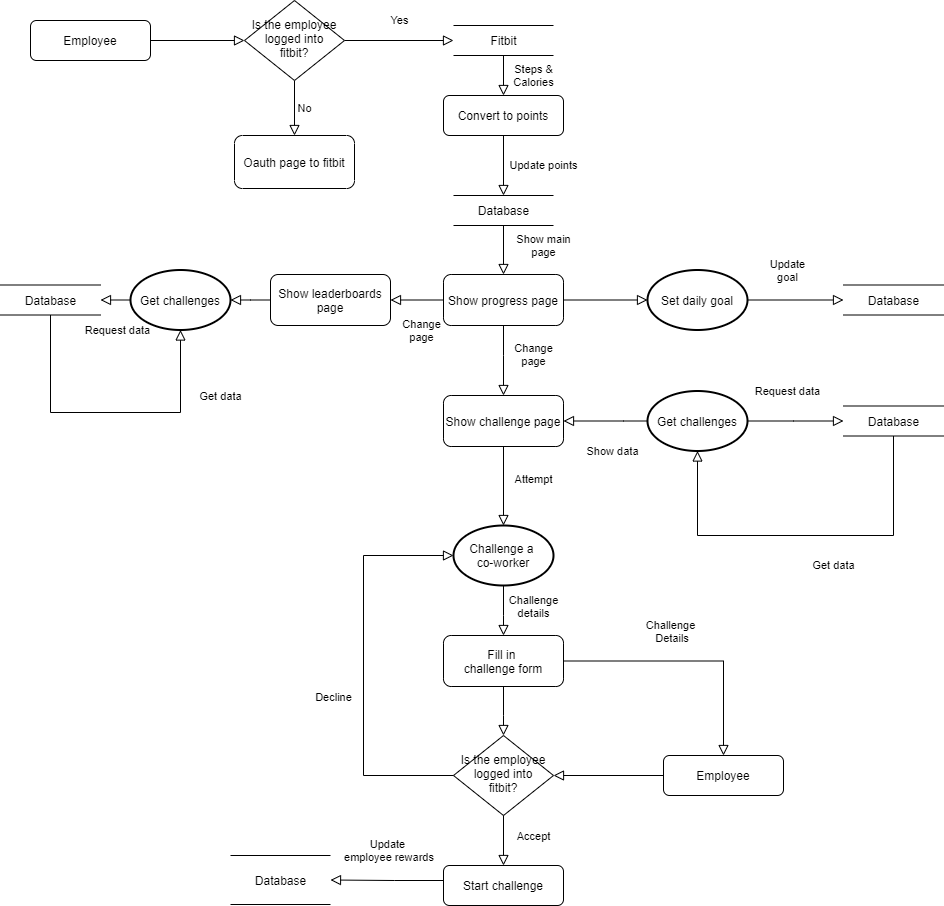
Description automatically generated

# Database design - MySQL



# Flowchart of the application

This flowchart will describe what will happen when the application is launched and we will explain the different parts of the application and why we did it as it is.



Fitbit

To get the user data for our application we will be using the Fitbit API. The data we will be getting from this will be used in our application to work with.

By making the user login to their Fitbit account through OAuth we receive the user info from that account and store that in the database.

We chose for this solution because we would not have to program a way for your mobile device to track your activity and save a lot of time while being sure that it works properly. The drawbacks of this choice is that we do not have full control over what exactly it tracks.

Challenges

When an employee wants to challenge a co-worker they have 2 ways to do so. The first one is to send the co-worker a fixed challenge and the other one is to come up with your own challenge and reward. When the timer for the challenge runs out you and your co-worker must agree on the winner with the UI before the reward will be stored on the winners account.

The way this works is when the challenge is created and stored in the database it will contain the id of both the creator and the participant. When the challenge ends and the users start selecting the winner the system will send a POST request back and update the database entry with the winner. While adding a new database entry to the winner with the reward of that challenge.

We chose to do it this way because it made the most sense to us as we would have to store the challenges and rewards.

Rewards in terms of challenges would be an agreed upon favor, such as getting a cup of coffee.

## Daily goal

Each day the employee will have a goal to reach. This goal can be raised or lowered by the employee him/herself whenever. These goals can be any of the data that the Fitbit API can retrieve for us such as calories, steps or stairs taken.

To change the daily goal we use a pop up with a dropdown menu to choose one of the options and another textbox to fill in the value of it. By sending this with a post we can get the values and post them into the database. Through the Fitbit data that we receive we can check the daily values and compare those to the daily goal to see the progress.

By having a daily goal for the user we give the user the ability to talk to their colleagues about what they want to achieve and how they are doing. This could encourage other colleagues to set a goal they feel is realistic and track their progress while being able to talk about it without it just being raw data that has been collected from the Fitbit API.

Progress Page

The progress page is responsible for getting all data that is relevant to the user and displaying it. The data associated with the user is: Their number of steps, number of calories burned, points, challenge coupons and their name. To help give a learning aspect to the application a random fact will be displayed by this page.

The information displayed here can be requested through the database with simple SQL calls as everything is stored and ready to be used from the moment the user logins.

For displaying how close the user is to completing their calorie or step goal, we will use a circular loading bar component. In the functional design it is shown that the circular loading bar will have a wedge cut out to display the number of steps or calories. This can be accomplished by adding a floating triangle with the same color as the card’s background and have the text on top of that. This does lead to one issue though, by cutting out a portion of the circular loading bar, we will be missing out on a certain percentage of the bar. Meaning that our percentage will not be accurate. To fix the percentage issue, we then simply divide the amount of steps or calories, by the user’s daily goal for each. Then multiply by the percentage of the circular loading bar that is still displayed.

Completion\_progress = (progress / goal) \* percentage\_unfinished

Progress is the users value of the set activity, goal is the value the user set as their goal, Completion\_progress is the percentage that we will display to indicate how close the user is to realizing their goal and percentage\_unfinished is the percentage of the goal that has not been realized yet.

To make our progress circle we need to find the angle of the triangle that is cutting the wedge out of the circle. With CSS it is possible to make a triage by making a div have thick sides and a thick bottom. This thickness will be the height and base of our triangle. By making the sides invisible the remaining shape will be a triangle.



To find the angle coming from the center of the circle, we will theoretically cut the triangle in half, using the same height, but cutting the base in half. This will give us a right triangle, where we can find the bottom right angle by dividing the height by the base. In the application it had a height of 120 and base of 50 which resulted in an angle of 54 degree. In triangles, all angles add up to 180 and since it is a right triangle, we got the remaining angle from 180-(90+54) = 36. To get the full triangle’s top angle we just had to multiply by 2 giving us 72. Now that we have the angle, we can find the percentage by subtracting the angle by 360 degrees giving us 288. Lastly our percentage for the circular loading bar can be found by dividing 288 by 360, which gives us 0.8 or 80%. This means to get an accurately parsed percentage we just need to multiply the result of the user's data by 80 after calculating their actual data from dividing by their goal: (actual/goal)\*80 .

Leaderboard Page

The leaderboard page is there to show the top users, your own rank and the users near your rank. By being able to see the ranks of other users and those near you we want to create a competitive spirit between the users. The leaderboards should serve to show the user how well they are doing compared to their colleagues and give them motivation to improve.

While we can get the list of users from the database, they are not ordered from highest to lowest. To have a correctly working leaderboard they must be in order from highest to lowest. Which can be done through a highest to lowest sorting algorithm, which would then display a set amount of the highest-ranking users. Displaying their ranks can easily be done by iteration through the list and counting from one and so on.

The board that displays the users that are nearest to the user is a bit trickier. In order to get their rank, we need to know the user’s rank. Ranks are not stored in the database since they would be updated far too often. This means that we must calculate the user’s rank ourselves, which can be done by having a counter that increases when there is a user whose points are higher than the user’s. Then to get the users in front of the user, we need to make sure that their points are higher than the user’s but not higher than the user that is a couple of ranks in front of the user. If they meet these criteria, then they can be sorted into the user’s that are closely in front of the user. This list should only hold a certain amount, if it exceeds the specified amount, then the highest-ranking user on this list will be removed from the near user list. Then when displayed, the ranks of the users will be incremented from the main user’s current rank.

Finally, the user that is behind the user can be found by making sure that each user is less than the main user, but more than the current user that is behind the main user. Their rank would simply be one less than the user’s. If no one is in front of, or behind the user, then the board should display that there is no one in that category and can display the rank of the main user.

# Advice

Our application does have room for improvement and some functionality can be extended. We have a good amount of ideas and changes to be implemented however as we did not manage to, we will be leaving this to the future developers.

## Challenges

The challenges system can still be expanded on with other modes such as team vs team or a battle royale mode in the future. By expanding on the modes for the challenge system new possibilities to be competitive you can expand the amount of users that will be dragged into the competitive aspect of the application. By having more competition when the leaderboard or progress page for statistics gets expanded to show more details this could be easily supported by the new data from these challenge modes.

## Daily goal

Currently the daily goal can only be a goal with the fitbit data in mind. How this could be extended would be to have it either be more customizable with different data. Instead of that however, It could also be extended through adding different kind of statistics such as weekly or monthly ones. The goal of the daily goal and statistics is to let the user see what their efforts have achieved and try to motivate them or give them an ability to talk to so showing more statistics in a clear matter would help with that compared to just a daily goal.

## Leaderboard

The leaderboards can be improved on by not only showing points but other statistics. Since points is the sum of your fitbit activity the other things you could make a leaderboard for would be challenges. This could show how someone could be the best at ping pong, standing the longest or burning calories.

## Avatar

To increase personalization in the application we had a thought of avatars for the users. These avatars could be plain at the start however you could get some costumes and customization for them. This plan was scrapped because we did not end up having enough time to implement it.

## Shop

Currently the points that the user gains through activity has no real value besides that you can see it. This could be fixed with a shop where you can spend the points. By giving the points a way to be spend they gain a value. This might be able to encourage some employees to try harder to get more points and be healthier or more active. The shop could be used to personalize your application with different kinds of animations, themes and avatars.

# Security

In any software, whenever you’re working with user's data that means that security will most definitely need to be addressed. Thankfully in our case a big part of security is already handled by Fitbit.

For the most part, the dangers that could arise are from the REST API. We need to make sure that no sort of SQL Injection is possible to executed. To that extend we’ve escaped direct user input when issuing queries by using parametrized queries.

Another part where an attack could arise is the frontend. Cross-site-scripting is a real issue that is thankfully taken care of by the framework that we’ve decided to use. The way that Vue.js is built prevents any sort of script injection into our platform.

# CI & CD Deployment

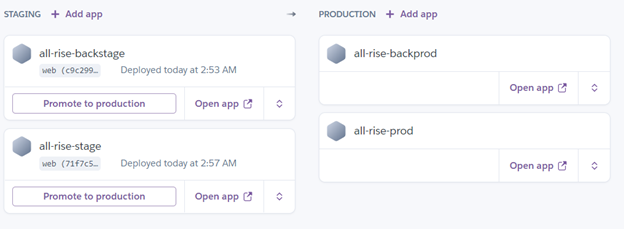
## What is a CI/CD pipeline?

A CI/CD pipeline helps you automate steps in your software delivery process, such as initiating code builds, running automated tests, and deploying to a staging or production environment. Automated pipelines remove manual errors, provide standardized development feedback loops, and enable fast product iterations.

In the absence of an automated pipeline, engineers would still need to perform these steps manually, and hence far less productively.

Below is a diagram to represent our pipeline: -

Design of our CI



## How it works

In most cases the pipeline run is triggered by a source code repository, it has been set now to only run from commits pushed to the master branch. A change in code triggers a notification to the CI/CD tool, which runs the corresponding pipeline.

Failure in each stage typically triggers a notification—via email but can be configured to send to many other channels like Slack, Discord and Microsoft teams—to let the responsible developers know about the cause. Otherwise notifications are usually configured to be sent to the whole team after each successful deploy to production.

### Build stage

We combine the source code and its dependencies to build a runnable instance of our product that we can potentially ship to our end users. Programs written in languages such as Java, C/C++, or Go need to be compiled, whereas Ruby, Python and JavaScript programs work without this step. As we are mainly using JavaScript, we would usually be skipping compiling, but since we are using docker containers to deploy this stage of the CI/CD the pipeline builds the Docker containers.

Failure to pass the build stage is an indicator of a fundamental problem in the configuration of our project.

### Test stage

In this phase we run automated tests to validate the correctness of our code and the behavior of our product. The test stage acts as a safety net that prevents easily reproducible bugs from reaching the end users.

The responsibility of writing tests falls on the developers and is best done while we write new code in the process of test- or behavior-driven development, though for this project we opted to write tests after.

Failure during the test stage exposes problems in code that developers did not foresee when writing the code.

### GitLab snippet

## Heroku

Heroku is a platform as a service (PaaS) that enables developers to build, run, and operate applications entirely in the cloud.

### Deploying

Once we have a built a runnable instance of our code that has passed all predefined tests, we are ready to deploy it. There are multiple deploy environments, a staging environment which is used internally by the team, and a production environment for end users. Below is the current pipeline setup on the Heroku platform: -

### Scalability

How long does it take us to build, test and deploy a simple code commit?

At the current moment the pipeline takes an average of 8 minutes for CI and deployment , this allows developers to push and deploy frequently without having to wait and prevents less frequent and more risky deployments, allowing for more rapid changes that businesses need today.

Do our CI/CD pipelines scale to meet development demands in real time?

Traditionally CI/CD pipelines have limited capacity, meaning that only a certain number of pipelines can run at a given time. As a result, resources sit idle most of the time, while at busy periods of the day developers wait in a queue for CI/CD to become available. Using Heroku on the free plan one is limited in how much one can scale but this applies to most other providers for staging and production. For the CI using GitLab one may configure multiple runners for a project to run their pipelines allowing for an ever-scaling number of pipelines to run if configured.

How quickly can we set up a new pipeline?

Difficulty with scaling CI/CD infrastructure or reusing existing configuration creates friction, which stifles development. Today’s cloud infrastructure is best utilized by writing software as a composition of features, which calls for frequent initiation of new CI/CD pipelines. One can simply create a new app in Heroku and add it to the staging level of the pipeline and with a few configurations push to it to test out these features, it is easily programmable and fits into existing development workflows easily.

## Database

Due to limitations on Heroku with running a dockerised SQL database, for the development and production environment we have opted for a small cloud-based SQL database. The connection properties for the databases are not hardcoded or kept locally on any file ,but rather are set in the environment by configurable variable in the cloud environment ensuring secure access to the database without any sensitive information exposed in the code.

We also have an SQL script which can be run within most SQL databases to set them for use in a few seconds with sample data.

## How to Deploy to AWS ?

* Build the docker Image
* Config docker file.
* Login docker login registry-1.docker.io
* Push to registry docker push reflectoring/aws-hello-world:latest