UNIVERSITETET I TROMSØ
Ifi – Institutt for informatikk - UiT

INF2900 Software engineering
Spring 2023

UiT Norges arktiske universitet

Training application with selective and suggestive workouts. Skooba.

Magnus Dahl-Hansen, Evind Østlyngen, Fedor Semaev, Jørgen Løken and Sigurd Uhre

Training application with selective and suggestive workouts. Skooba.

Magnus Dahl-Hansen, Evind Østlyngen, Fedor Semaev, Jørgen Løken and Sigurd Uhre

May 19, 2023

Abstract

This document is the report that represents our work in the project work for INF-2900-1 23V Software engineering. This report provides a detailed introduction to the methods used and how they have been used in this project process, but it is expected that the reader is familiar with concepts such as agile development. The aim of the report is to give an insight into how the group methodically worked with software development, which features and architecture were implemented, and not least which experiences the team is left with after this project. The project document is another document where the reader can look more closely at tests, user stories, as well as product and sprint backlogs.

Contents

1 Introduction

This project uses the agile software engineering methodology, which is a systematic approach to managing the software development life cycle. With this methodology, the development team begins the project without a fully clear understanding of the final product, instead addressing issues and obstacles as they arise during the development process. Agile development focuses on flexibility, teamwork and producing working software more frequently and faster compared to other development processes. In this report, we will take a deeper look at agile processes and how we have used them during the development of the software. The team have used the agile methods to develop a training application called Skooba. Features and architecture will be discussed and we will take a closer look at lessons learned from this project.

2 Project Overview

2.1 Main content (of the project)

The goal of the project is to develop a workout-application that gives the user opportunity to insert training criteria into the application and receive a proposal for how the training session might look, based on these criteria. The training session must then be able to be carried out with available information and timing. The purpose of the app is that it adapts to and inspires the user's training session.

The scope of the project were five months of software development where a product was planned, developed and delivered. The application is limited to running on a local computer, a mobile app has not been developed but will represent a natural step forward for this software. The approach and methodology in this project is based on agile software engineering with weekly scrum meetings. Agile software engineering methods such as user stories, sprints, tests, effort required, product vision concepts with more, are used in this project, they are explained in section in development process. The deliverable will include this report, the finished software with necessary documentation and instruction, and a project document include user stories, plans and progress at various stages, tests and their results.

2.2 Product vision

The product to be developed is a workout application which allows the customer to enter simple preferences for what kind of body part, how much time, and in what combination they want their training session to look like. The customer must be able to choose the extent to which they will decide what to train through options such as totally random training session, partially random training session, and complete control over the training session.

Compared to other products on the market, this product is based on what the customer wants the training to contain, and not what are necessarily recommended combinations from personal trainers. This product gives total control to the user, and sets no restrictions on the combinations. One of the key visions of our software is that the customer should be inspired to do new exercises, but at the same time they are left with the freedom of choice to decide which body part they want to train and for how long they want to train it. Users of all ages who want inspiration and training on their own terms are the targeted users.

2.3 Market analysis

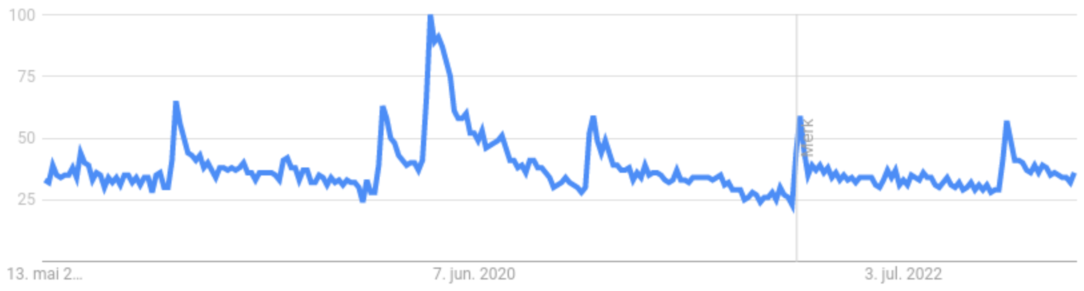

Figure 1: Google trends

In order to see if our workout application stands out from the competition, we have to analyse the market. The goal of this analysis is to find out if our features can gain competitive advantage against the competition, and also find out if similar products already exists, in addition to the demand. To gauge the existence of demand, we utilized tools like Google Trends to examine the search volume for the term ”workout app”. We observe that the peak was in 2020, this is because of the COVID-19 virus which affected all the activities, leading people to seek alternative ways to train. We can also observe that demand is relatively constant, with a spike in January due to New Year's resolutions. Therefore, we can conclude that there is existing demand for workout apps.

The competition in the market consist of other applications, such as Sworkit, Nike training Club and JEFIT. All of these apps has the ability to let the user choose a workout based on users preferences. However, none of these apps allow users to generate a truly unique workout, instead, it gives user a pre-made workout retrieved from a database. Therefore, ”generate random workout” feature will be our main differentiator in the product, and can give us a competetive advantage. In conclusion, this marked analysis shows that there is demand for workout application, and our product has the potential to attract the customers with its distinctive feature.

3 Development process

3.1 Product vision concepts

According to Sommerville (2021, p. 17), a product vision is a brief statement that defines the essence of the product that is being developed. This statement functions as a basis for developing a more comprehensive description of the applied features and attributes of the product. One of the key functions of this statement is that it should explain how this product stands out from current competing products (if any). When new features are suggested, the product vision can work as a guideline to check if the new feature contributes to the product vision.

The team decided to follow the three fundamental questions Sommerville (2021, p. 17) suggested: *What* is the proposed product and how is it different from competing products? 2. *Who* are the target users and customers for the product? 3. *Why* should customers choose to buy this product?

The product vision stated in section 2.2 was planned and designed in an early phase of the project. The team looked to this vision in order not to ”derail” from what was the purpose of the project. This vision shaped the implementations of our features.

3.2 Agile development

Sommerville (2021, p. 31) the purpose of introducing agile development was to create usable and functioning software that could be quickly delivered to the customer. The customer can then respond to the delivery of the product with new or other requirements that they wish to be included in a later version. In this project, the team will be organized in what Sommerville (2021, p. 52) explains to be a self-organizing team. This label indicates that the development team alone, organizes the work to be done by discussing and agreement among team members.

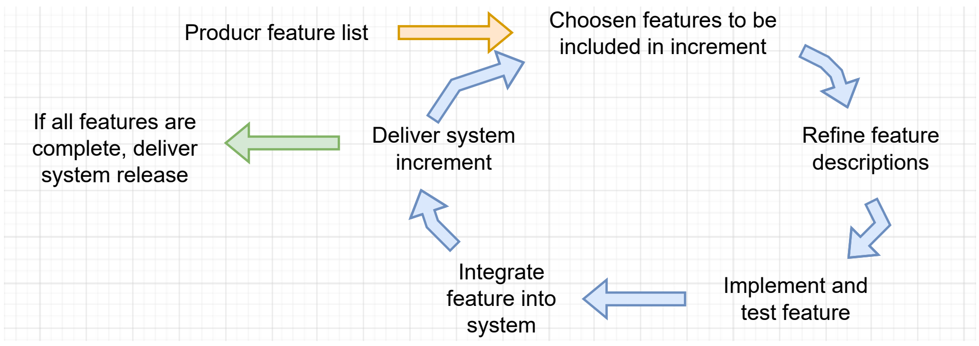
Producr feature list
Choosen features to be included in increment
Refine feature descriptions
Implement and test feature
Integrate feature into system
Deliver system increment
If all features are complete, deliver system release

Figure 2: Incremental development, according to Sommerville (2021, p.32).

With the life cycle of incremental development in mind, the team has made use of several of the tools that agile development offers in its toolbox. In this section, we will take a closer look at the methods and development processes that were used during this project and how they work.

3.2.1 SCRUM

Although SCRUM often is referred to as the working method itself, the scrum meetings (also simply called scrum) is a tool in the agile toolbox. Sommerville (2021, p. 38) explains that the start of each development cycle, the team decides what features to prioritize, how to develop them, and what each team member should do. The key is to keep the planning informal with minimal documentation and no designated project manager. However, this can create an information vacuum between the development team and the organization (management). This vacuum is filled by the Product Owner, who ensures that the development team stays on track, focusing on the mission of the product and doesn’t get sidetracked into technically less relevant work.

In this project, the team has given the teacher assistant the role of product owner. The product owner has been used to advise the team, keep an overview of how the project is going and give the team tips on the use of technologies and methods. Within the team, the tool ***ScrumMaster*** has also been used to keep an overview of the process and to link information to the product owner. This role of ScrumMaster has been rolled on (after sprints) and used to distribute responsibility, ownership and process understanding. Tasks assigned to the ScrumMaster ranged from arranging scrum meetings to instructing in scrum processes.

Some of the tools that the scrum master has available in his toolbox are concepts such as: product backlog, development team, sprints, scrum, stand-ups and effort required. Sommerville (2021, p. 39) explains that in Scrum, a to-do list called the ***Product backlog*** is maintained to keep track of bugs, features, and product improvements that the team has not yet completed. In this project, the team used the product backlog (PB) tool to organize the tasks/features that the team had to carry out. Product backlog does not necessarily only contain features, other ”items” such as user request, essential development activities and desirable refinement improvements can also find a place in this backlog, creating a product backlog items (***PBIs***).

The backlog was created in an early phase of planning and was categorized according to which features and items had to be done first. The content of the backlog was the items that the team was able to foresee that they had to do to complete the product, this also means that new features and tasks were added to PB as the team realized that they were necessary. In this project, we have considered the PBIs a dynamic tool where all members of the team have discussed which items should be included and which are superficial. This PBI has consisted of rough points, which in themselves could often contain many sub-points.

The designated ScrumMaster have the responsibility to organize ***scrum meetings***. The scrum meeting was arranged regularly every week, and if necessary, several meetings were arranged within the same week. The regular meeting was held every Monday and always began with a briefing by the Product Owner, where each individual member of the team carried out a ”stand up” - explaining the task they had, the status and any problems that arose.

These team meetings can be at the start and end (or in the middle) of a scrum process, better known as a sprint. Sommerville (2021, p. 39) explains that one of the key aspects of the Scrum framework is breaking down the work into time-boxed sprints, which normally last from two to three weeks but can also be adjusted if needed. During the sprint execution the team works with the items that they have put together in the Sprint Backlog. Here, work is carried out continuously with the implementation of software or other items from the sprint backlog, which can also be categorized from effort required or story points.

During a sprint, the team had weekly scrums-meetings to review progress and plan the days work. Such sprints make use of the sprint-backlog tool. In this project, the team used both sprints and sprint backlog for implementation of features or the execution of items. The team agreed on the Sprint backlog, which is a separate backlog based on items from the project backlog. In the sprint backlog, the extensive items were broken down into smaller tasks within this item. During the start of a new sprint, the team decided how long this sprint should last based on how extensive the sprint backlog was. The tasks in the sprint backlog were formed from ”logical” items to be carried out in the same sprint, meaning that tasks that build on each other were done in the same sprint.

In this project, the team followed the sprint steps illustrated above. During the scrum meetings, the team processed the sprint backlog and made adjustments if it proved necessary. The next steps in sprints are to develop the software and integrate it with already existing software. When the sprint was over, a sprint review was created. The sprint review was used to evaluate the methods and tools used in the sprint, what worked and what we can be improved, and necessary measures

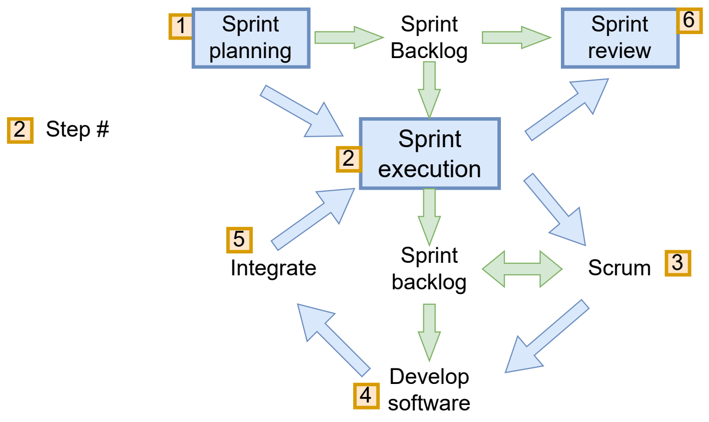

Figure 3: Sprint activities, illustrations inspired by Sommerville (2021, p. 48)

were carried over to the next sprint.

When the sprint is finished, there are two possible outcomes for each item on the sprint backlog: They are either completed or not completed. Tasks that where completed were removed from the sprint- and product backlog, while tasks that were not completed were placed back into the PBI, to be completed at a later sprint. Sommerville (2021, p. 270) explains that ***code-reviews*** is a tool that complements testing in finding bugs and errors in the program. In this project, code review was done on a few integration's to the master branch. Before the code was pushed to the repository, a team member looked through the code to check for errors and any misconceptions by the person who developed the code. This is a powerful tool that both compliments tests and spreads expertise about the software within the team. In this project on the other hand, it turned out that this was a too time-consuming and resource-intensive tool that demanded too much from the team in relation to what it produced. It was therefore decided not to continue with this method.

Sommerville (2021, p. 45) provides an explanation on how to estimate sprint items. The amount of effort required for implementation of each item in the PBI can be expressed in person-hours or person-days requirements, often referred to as "***Effort required***" measurement. "***Story points***" measurements are a more abstract estimation method, taking into account task size, complexity, required technology, and unknown factors. Story points are relative estimates, based on a baseline task agreed upon by the team. In this project, we only used the effort requirement measurement to set the size of tasks in the backlogs. The team decided that one point should be equivalent to 2 hours of work. Before the tasks were assigned to the members of the team, it was discussed how extensive each "item" in the backlog was by putting points behind them. This way we could more accurately distribute the tasks between the members of the team also according to the work effort for each item, and not just the number of items.

3.2.2 Pair programming

Sommerville (2021, p. 35-37) elaborates that pair programming is one of the ten practices in extreme programming (XP). He argues that if two developers create each code unit, they can learn from each other and catch each other's mistakes. Production-wise, there is no evidence that team programming is a more productive method compared to two people working as individuals. In this project, pair programming was used on several occasions. The main reason why we chose to use this practice was because some of the items in the sprint backlog turned out to be extensive and sometimes difficult to set an "effort required" estimate precisely. This is partly because many of the items were a ”first time experience” for the team members.

By using pair programming, the team had the opportunity to spread both workload and knowledge. Essential concepts, such as tests, can be good items to distribute between two team members. Here they can learn the concept together, learn from and catch each other's mistakes, and spread the knowledge further in the team together.

3.2.3 Scenario development

Sommerville (2021, p. 61) explains that when it comes to features that the application should contain, the development team should ideally make a list of which features they want the application to contain. Of course, since this is agile development, redundant features can be removed, and new applications can be added to the list as the product gets a clearer direction. Personas, scenarios, and user stories can be seen as techniques or methods that can be used to visualize and document how users might interact with the application, leading to the development of features that meet the needs of users.

Sommerville (2021, p. 76) elaborates that a user story is a more specific and structured narrative that describes a singular desired action or goal of a user from a software system. Just like scenarios, the user stories also contains a persona developed by the team. User stories follow a standard format of ”As a <role>, I <want/need> to <do something>” or with added justification for the action at the end ”..to <do something> so that <reason>”. They are useful in planning and are often represented as a set of user stories

In this project, the team followed the model that Sommerville (2021, p. 76) explains above to develop user stories and to achieve an understanding of what users want to get out of the software. Writing user stories in the project document was one of the first steps taken in the project, and was later utilized to both find and implement relevant new features.

3.2.4 Kanban

According to the Wikimedia Foundation (2023), kanban is a process method that focuses on limiting initiated work in a production facility. One of the key features is that a new job must wait if the number of started jobs has reached a max limit. As soon as a job is finished, the next pending job can be started. In this project, the development team test out kanban and the kanban board feature on Github. This is a board that visualizes workflow (tasks/items) on a column that represents how far they have come in the process. This kanban board used three columns with relatively ”To do”, ”In Progress” and ”Done” for the tasks moving from the ”To do” column to the ”Done” column.

The Kanban board tool was initially adopted to provide visual clarity on task progress within different stages of the process. It was used during the early sprints, allowing the team to identify the status of tasks. However, its usage was discontinued after a few sprints.

3.3 Code Management

The importance of managing source code is emphasized by Sommerville (2021, p. 309) in the context of modern software engineering practices. Source code management systems are specifically designed to handle the management of an evolving project codebase, enabling the storage and retrieval of different versions of components and entire systems. These systems facilitate parallel work among developers, allowing them to work concurrently without disrupting one another.

In this project, a master file from the remote master-repository was pulled to each team member's local computer, into each team member's personal repository. Instead of all project members working on the same branch, each team member ”branch out” from the development branch ”dev” when they work on the implementation of an item. Branching and merging in code management systems enable team members to work on the same files without interference according to Sommerville (2021, p. 316).

In this project, the team used code management by creating a GitLab repository. Each team member has downloaded this repository to their local computer. The team concluded that a branch called ”dev” should be created, which actually took on the function of a master branch. The master branch was supposed to act as a backup, and as we were supposed to push software every time the team had successfully completed a sprint. The reason for this was that there were several team members who, for the first time, had to work on git repositories with several other team members. The team discovered early on that this could lead to complications, such as pushing the wrong code to the master branch at the wrong time, thereby overwriting newly developed code. The master (main branch) should therefore act as a checkpoint, so that we could always go back to that stage of software if something went wrong.

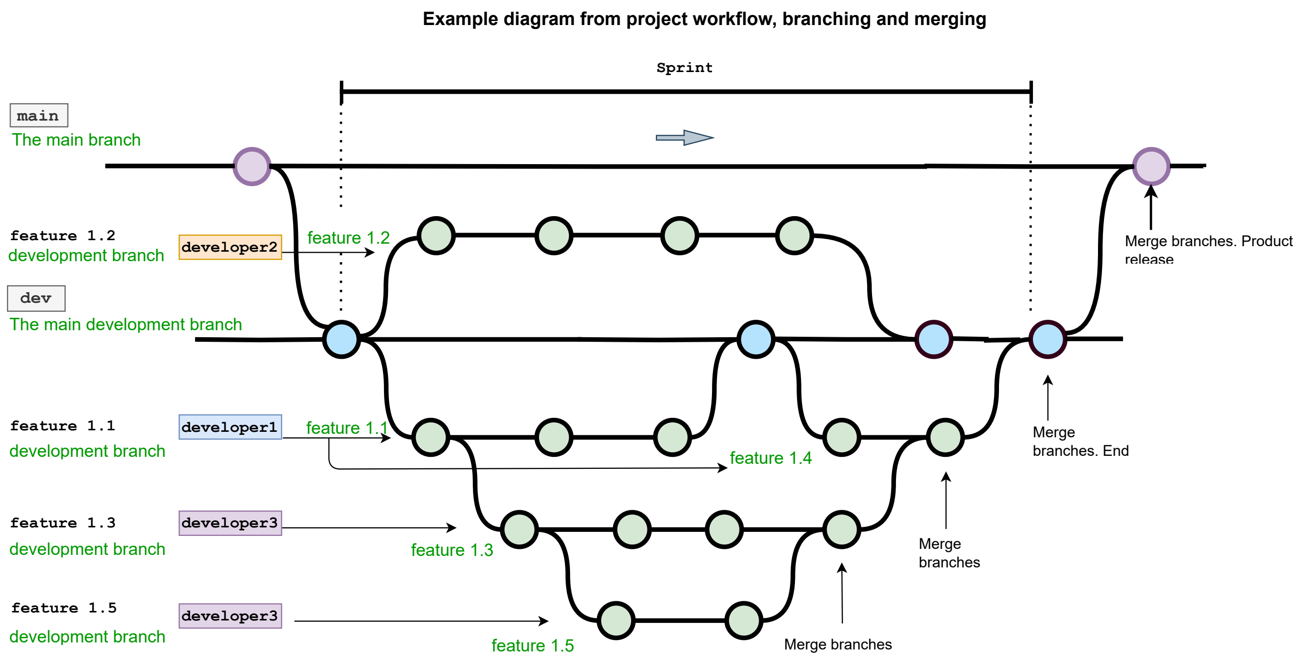

Figure 4: Git workflow

The git workflow figure shows an example of how branching and merging was done in this project, isolated in a sprint. The figure visualizes how team members have branched out from the ”dev” branch at the start of a sprint in their own branch directories. This branch could contain several tasks/items if necessary, and was essentially left to each individual team member for naming and administration. Here you could use the same branch for each task or create a new one and delete earlier ones for each task. After the task i done the branches are merged together, which completes the git flow cycle.

4 Implementation

4.1 Software Architecture

React and TypeScript were used for the front-end development, along with CSS for styling. Django REST Framework was used for the back-end development and SQLite was used as the database management system. Using API requests and Axios JSON data were used to exchange data between different components of the system. The interaction between the frontend and server side components is shown in the figure below.

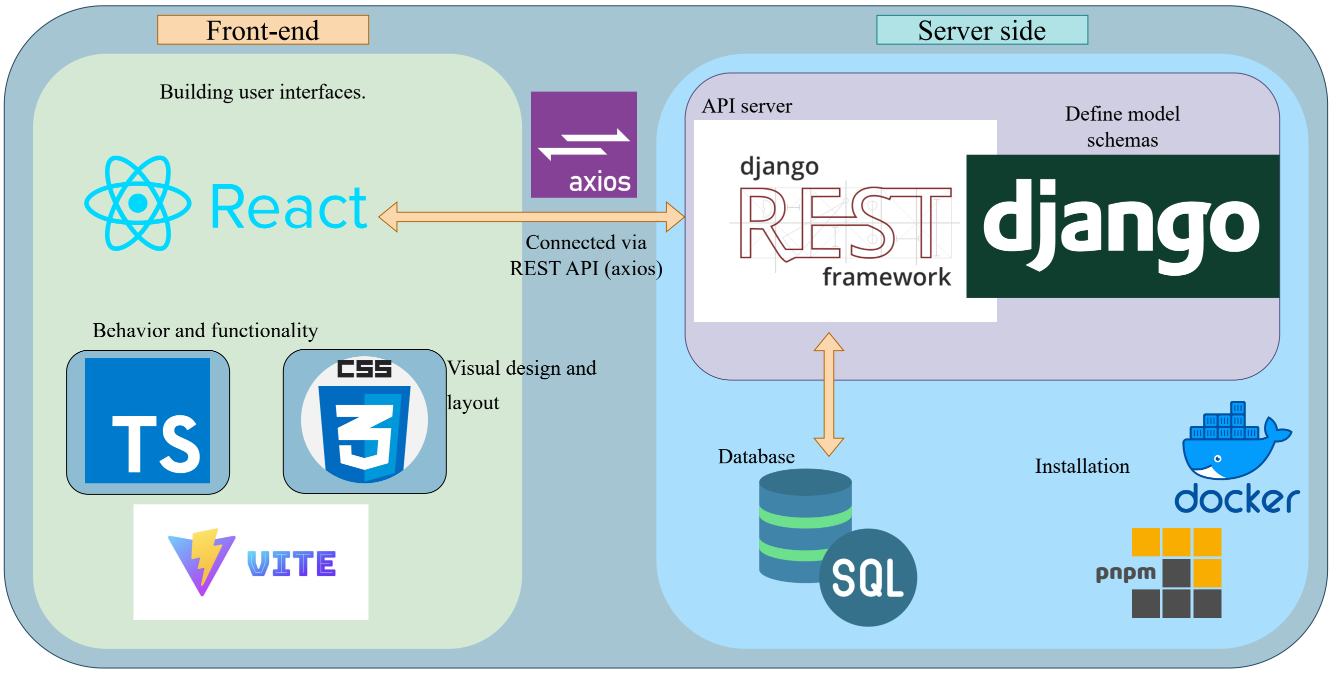

Figure 5: Application architecture

4.1.1 Communication

React allows users to interact with the application by serving as the front-end. The user generates HTTP request in the application, where Axios facilitates the requests from the React front-end to the REST API. The REST API, implemented using Django REST Framework, receives the requests and processes them based on defined API views and business logic. The framework can communicate with the SQL database to fetch or update data. Once the operations are completed, the REST API sends a JSON-formatted response back to the React front-end. React will now update the user interface based on the received data, which in this project can, for example be a training session composed of exercises from the database. This architecture facilitates easy communication between the React front-end and the Django REST framework.

4.1.2 Database

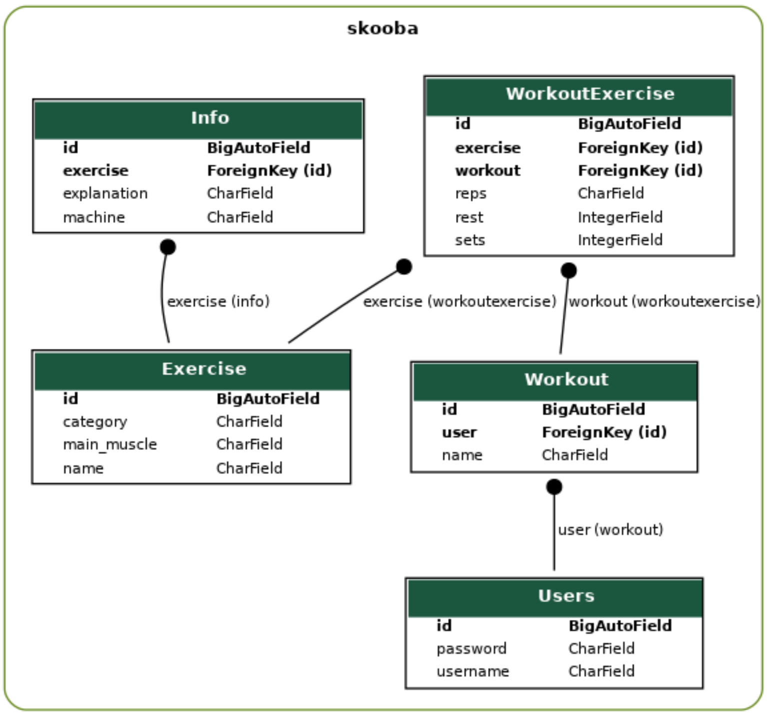

Figure 6: Database schema for Skooba

Displayed in the figure above are all database models used for this project with their respective fields and relations.The tables are generated and usable via the Django ORM and are connected via foreign keys. To define the many-to-many relationship between an exercise and a workout we had to implement a join table ”WorkoutExercise”. This table contains a foreign key to both tables, as a workout should contain multiple exercises and a exercise should be usable on multiple workouts. This join table also contains the additional information about the exercise for that workout.

4.2 Features and functionality

4.2.1 Login & Register

In Skooba, the users have the option of logging in and registering an account, which gives the user privileges that visitors would not have, such as saving a workout for later use. In the login page, the user can input an username and a password, which will be sent as parameters to the backend server in order to check if the account exists in the database. If correct, the user will be redirected to the home page as a logged-in user. Upon a successful login, we set a value in the users browser to differentiate logged in users.

In the register page, the user can input a username and password that will be sent to the back-end server, and if the username doesn't exist in the database, the user is allowed to create and add the account to the database.

4.2.2 Random workout

One of the main features in Skooba is to generate a random workout. To generate such a workout, the user selects their desired workout type, either ”volume” or ”strength,” which will later be used to calculate the reps and rest times, as different workouts require varying amounts of reps and rest.

The user also selects the body parts they wish to train, or they can choose ”random” for both input fields to create a truly random workout. In addition to training type, the user types in the preferable duration. Upon clicking ”create workout”, an API call containing users input is sent to the Django backend server. Based on the selected duration and training type, the number of exercises per body part is calculated, and each body part is filled with random exercises. The number of repetitions, sets and rest is also calculated based on the training type. If user chooses ”random” on both of the input fields, the system will select a random training type, and will randomly pick 1-3 body parts to train depending on the duration. When everything is calculated and the random exercises are filled, a new instance of workout is created inside the database, and then populated with number of repetitions, sets and resting time per exercise.

4.2.3 Custom workout

Compared to random workout, where the user gets a randomly generated workout based on certain specifications, custom workout lets the user choose exercises from different muscle groups. First off, the user is allowed to give the workout a name, which will be saved in the database upon confirming the workout. Moreover, there is a button to show or hide the muscle groups, where each group contains exercises gathered from the database that is related to that muscle group. For instance when clicking the ”Chest” button, the user can click on ”Dumbbell Pullover” or even ”Barbell Bench Press”. These exercises will be added to a table below, where the user can change the amount of reps, sets and rest. Furthermore, the user is also able to delete exercises either from the table or from the related muscle group. Another feature is the option to change the order of the exercises in the table. The exercise located at the top of the table is the first exercise to get started once the simulation runs. When creating the workout, an API call is sent to the backend server, and adds the workout in the database.

4.2.4 Present workout

Present workout is a functionality which displays the created workout, either if it is random or custom made, to the user before it is started. When a workout has been created it is automatically saved with a pre-defined workout name and a unique identification number(ID) and the ID is added as a parameter to the URL. The URL-ID is then used by the ”Present workout” component to issue a HTTP GET-request to fetch the saved workout and display the elements it contains. The exercises are displayed in a table which displays the exercises, repetitions, number of sets and duration of the workout.

For further notice: because the workouts are saved automatically to the database after creation they must be deleted in some cases. A DELETE-request is issued for the workout if the user does not explicitly decide to save the workout or if the user quits the program prematurely.

4.2.5 Edit Workout

In the present workout page, there is a button that lets the user edit its existing workout, which could either come from a workout from saved workout, custom workout or random workout. When clicking the button, the user is linked to custom workout with an URL containing the exercise ID. Once entering the custom workout page, the program fetches workout table from the database based on the exercise ID in the URL, containing the workout name, exercises with the preexisting values for reps, sets and rest as well as being in the correct order. Essentially, the edit workout provides a custom workout page with an existing workout, letting the user change the values. When the workout is confirmed, the program deletes the old workout from the database, and POST's the new to the database.

4.2.6 Load workouts

Load workouts is a functionality which allows the user to inspect, replay and delete the workouts they have created and saved. If a user is logged in then the identification number of the user (user_id) is sent as a HTTP GET-request to fetch the workouts for this user. The response is a list containing all the names of the users saved workouts along with their identification number (workout_id). The workouts are then displayed as a list with replay and delete buttons on the side. If a user is not logged in they are redirected to a page asking them to log in to see their workouts (if any exists).Based on the users input, the workout_id is either passed as a parameter to a navigate function which automatically redirects the user to the ”present workout” page as a replay functionality or it is used in a HTTP DELETE-request which deletes the workout from the database.

4.2.7 Navigation bar

The navigation bar is composed of five buttons which links to different URL's in the application, namely the home page, list of exercises in the database, settings and an option to log in or out based on the login status of the user. The navigation bar is implemented as a parent component while the rest of the components are children components. This means that the navigation bar will be rendered at the top of all pages on the application.

4.2.8 Exercise information

To gain information about each exercise a user can get a step-by-step explanation on a separate page. The page simply contains the name of the exercise and the explanation. The page can easily be redirected to from other pages by using the primary key of the exercise in the URL bar.

4.2.9 Simulation

After the user creates a workout, either random or custom, the saved data can be used to simulate the workout. The exercise names are presented to the users sequentially in the order they are saved, and tells you how many reps and sets to do. If the user forgets how to perform the exercise, a link to the explanation site is easily available. After finishing a set the user can start a countdown timer which matches their saved pause value. The pause timer can also be exited earlier. If the last set of an exercise is completed, the next exercise will be presented.

After finishing the last exercise of the workout, the simulation is complete and the user are presented with multiple options. If the user is logged in, they can choose if they want to save the workout to later repeat it. This feature can be especially useful if a random workout is simulated. If this option is chosen, the user will also be given the ability to give the workout a better name than previously randomly generated. These options are also presented if the user exits the workout before finishing.

4.3 Security

The security of Skooba has not been prioritized since the product was not initially intended to be released to the public. However, there are some security measurements implemented. For example, we encrypt passwords before storing them in the database to defend against dictionary attacks and to ensure developers do not have access to user passwords. In addition, we have an security measurement that prevents a user from accessing directly the workout simulations by changing the workouts ID in the URL. If user does not match the user defined in the workout, the application will be redirected to the homepage, preserving the integrity of the user data.

Despite the security implementations, our system remains vulnerable, since all the communication between server and client is transmitted in plaintext. Consequently, an attacker could easily eavesdrop and access all sensitive information during user registration. In order to make this application more secure, we can encrypt the communication by using SSL/TLS certificates.

5 Retrospective

5.1 Reflections on complexity, problems and choice of solutions

When looking back on the project there are a lot of things that could be different. We opted for typescript instead of JavaScript in order to learn new language. But this created complication, since there were only one person who had experience with JavaScript based languages. Therefore it became messy at the start where people on the project wrote Javascript in TypeScript file, where people did not use features which really make TypeScript stand out like defining the syntax of a variable. But it was to late to change, therefore we have tried to make the code more Typescript like, but there are still areas where there are more JavaScript code then Typescript.

In addition we used pythons Django library for back-end server and as the project went on we found out that Django was outdated and there were much simpler tools available. If we would have repeated the project we would have used Node.js which is JavaScripts back-end framework. And also, we would have used JavaScript instead of Typescript if we would be in the same situation. But we would have used Typescript if we could used the gained knowledge and build on top of it.

Overall the complexity of the project was comprehensive, since both extensive teamwork and agile product development was new to all team members. There is also possibility that the project was to complex, since we did not get the chance to build so many features and make the project more polished. If we would have redone the project, then we would have simplified the minimum viable product (MVP) so we could build extra features.

5.2 Ethic and personal issues

There are a lot of hype in the fitness world to make the smartest and best app. But the smartest apps does not necessary equate to the best apps. If this project would have been launched, we would have to cooperate with a certified personal trainer in order to design workouts and develop an algorithm which prevents the users from injuring themselves. At this moment, the generation of a random workout truly generates a random workout and the exercises may not fit to each other in a healthy way. A more advanced and ethically better solution would be to generate exercises with the previous exercises in mind. For example, if a user want to train chest, there is a risk that all the exercises will be targeting one area of chest, leading to uneven muscle growth and potential injury.

5.3 Efforts in the team

As explained earlier, the team used the ”effort required” method to be able to more accurately estimate how much work an item or task consisted of. This was done at the start of each sprint and the points were always discussed and determined in plenary. In addition to this, the assignment of tasks was always such that each team member could state what he wanted and or what he wanted most. The scrum master would then list the name of this team member on the ”sprint” document. The next team member would then state which task he wanted (if any). This way of assigning tasks gave room for both ”reserving tasks” and at the same time proposing tasks to others. It worked out organically and team members selected tasks based on their skills, expertise, and interests.

The collaborative workflow of the team was centered around scrum meetings every Monday. Here the group had a four-hour work session every week where they started a sprint together or had a scrum meeting in the middle of a sprint. In addition to this regular meeting, the scrum-master had to ensure that other meetings were agreed upon and booked throughout the week if they were needed. Other communication beyond these meetings was done using Discord. This channel was created to share information, ask questions, and arrange meetings within the group.

5.4 Lessons learned

During the start-up phase of the project there was developed a roadmap to help with development, but this could have been created in more detailed. The roadmap was poorly planned because the group was new to the concept and did not fully understand the importance of it. On the other hand, the product was successfully developed because the group had a clear vision of the product and an oral agreement of what the application should do. If the team were to start a new project they would have created a more detailed roadmap mainly to better plan and prioritize work tasks but also to improve communication in the group, allocate resources and anticipate any upcoming obstacles or risks.

During the startup-fase of the project, the team had chosen a SCRUM master who had the responsibility to among other things, distribute tasks, make decisions if there was any disagreement, plan meetings and book rooms in which the members of the group could sit together and work. The role of SCRUM master was intended to be distributed among all members over the development period so that everyone could test this. This was a challenge however, because the members had different view about how components should be designed and members would work mostly from home. As we got further into the project and got better at coding in Django, people became more independent and needed less help from each other. In a way, the communication between the members became worse, but at the same time the members understood what was being done in order for the product to be finished. The lesson learned is that even though this approach worked, it is not a good practice not to have a proper SCRUM master.

5.4.1 Software directory organization

In this project, agile development has been the basis for all methods and approaches to working methods. The essence of the method is that the software should be able to dynamically develop as the project develops over time, which means that you cannot set up all the frameworks from the start because you do not know exactly how the project will develop. With this in mind, there are no restrictions on how you can set up software directories so that it has a known structure and is open to changes at the same time. In this project, we had not defined the directory structure before we started with it. Which resulted in a unstructured file directory.

A lesson learned here is that directory structure have both industry standards and other standards that you can easily follow and familiarize yourself with. The team don't really need to follow these structures if they don't want to, the most important thing is that the team agree in advance how the structure should look and that they set it up before beginning with development. A well-defined directory structure can contribute to better code maintainability, collaboration, and ease of navigation within the project, this should be point number one for any new project in the future.

5.4.2 Unsuccessful agile tools

The Kanban board was an scheduling tool that the team tried to apply to the project, however, it was decided that we should abandon this tool after a few sprints. One primary reason for this decision was the lack of utilization for process evaluation. The team did not find significant value in assessing and optimizing their workflow using the Kanban board. Additionally, maintaining the Kanban board required dedicated resources from the team. The effort involved in logging and updating tasks on the board was perceived as an unnecessary overhead that did not justify the benefits gained. As a result, the team decided to discontinue using the Kanban board as a regular practice. In this particular case, the team determined that the effort required for upkeep outweighed the benefits derived, leading to the decision to explore alternative methods for managing and tracking tasks.

This tool has obvious value for projects, but for this project it was not necessary. If this had been a more comprehensive project with more tasks and team members, then this tool could reach its full potential. A lesson learned is that you have to familiarize yourself with the theory behind the tool you use, and not just use it ”as you think” it should be used. At the same time, not all tools available are suitable for the project you are working on. If you are uncritical of all tools in agile methods, then the tools can end up consuming more resources than what it produces in terms of usefulness.

Code review is another agile tool that the team tried out and decided not to move forward with. In the same way as the kanban board, the team must evaluate which methods give results in terms of simplifying the work, ensuring good software or helping to balance and distribute the work. If the method does not return enough in the form of one of these results, the team must consider whether to continue with it. Code reviews should help ensure that good software is merged with the main branch (dev-branch). We experienced that code reviews took a long time and that they did not produce changes in the code and that the person who reviewed the code had to spend a long time familiarizing himself with it. In essence, it cost too much in terms of resources and time.

5.5 Time lists of the individuals

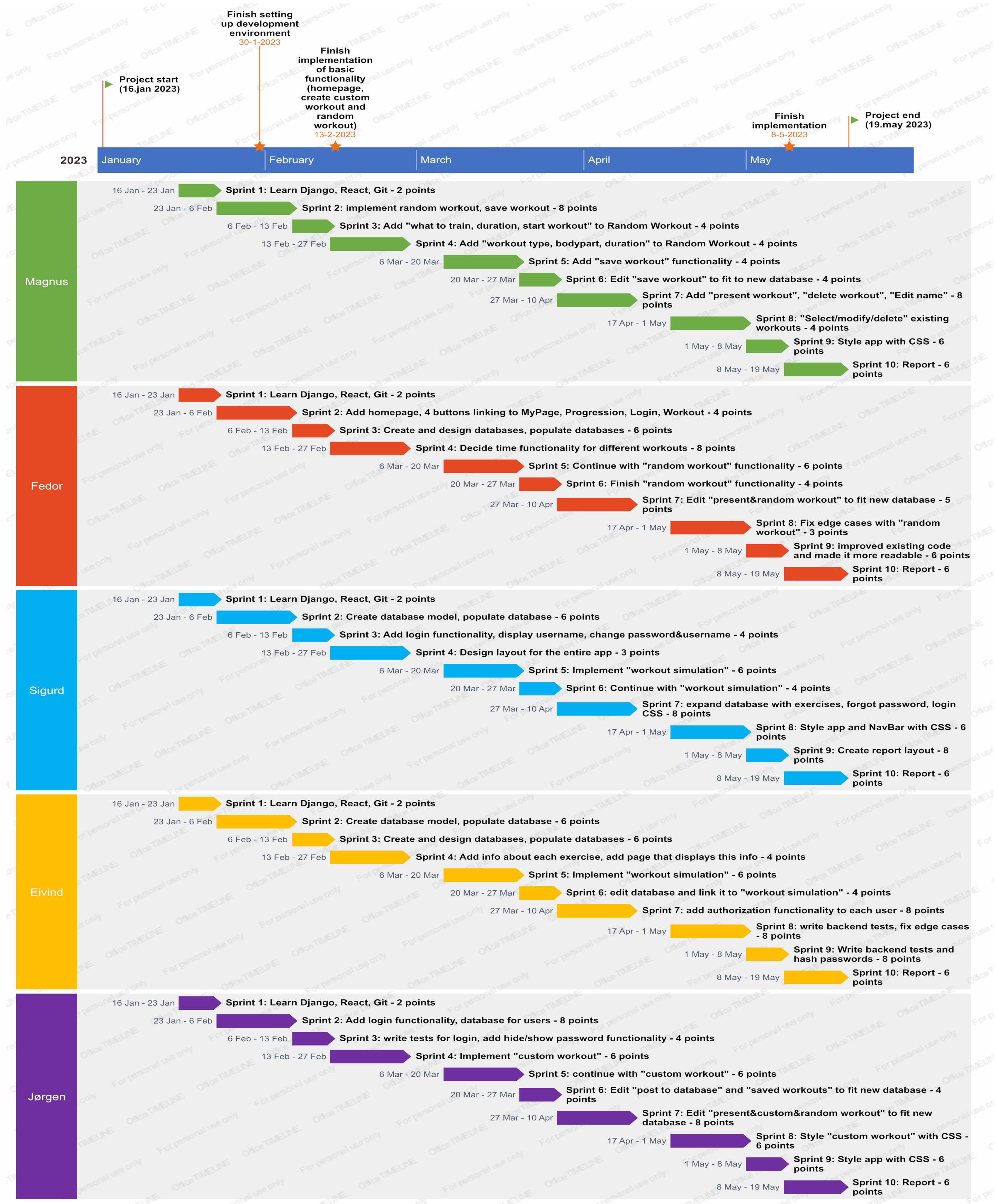

Figure 7: Time list of the individuals

5.6 Alternative approaches, techniques, models.

Extreme programming (XP) offers a wide range of agile practices. Sommerville (2021, p. 36) list some of the most widely used XP practices such as test-driven development and refactoring. Test-driven develop is based on the developers writing the test first instead of writing code and then writing tests for that code. Combined with automation, the code written can be tested frequently during implementation end expose if new code ”breaks” code that has already been written. This is an interesting practice, but it also requires a good understanding of the development of tests at the start of the project.

Sommerville (2021, p. 36) elaborates that refactoring is a practice that emphasizes that the developer is expected to strive to improve the efficiency, readability, security and the overall structure of a software as soon as it is discovered that there is an opportunity to do so. This contributes to the software being simple and easy to maintain. This practice is a tool the team could strive to use, the practice reduce the amount of follow-up errors and make the software more accurate for everyone. However, this mod will most likely also prove to be resource-intensive.

5.7 Future work

These are the points the development team would have developed further if there was resources to do so:

Add new features and functionality. During the planning of sprints and implementation period, the development team discussed functionalities they wanted to implement. Ultimately these functionalities were not implemented because they were overshadowed by other more important functionalities and features. The functionalities which would have been implemented if there was more resources to do give the user ability to: change passwords, view and add personal information such as their current body weight and personal lifting records, chat with other users or coaches and play games in between sets during the resting time. In addition the team wanted to add images and illustrations of each exercise to the database which could have been displayed during the simulation. Adding new functionalities such as these would have contributed to the application becoming more attractive in a possible market.

Optimization of code. Optimizing and improving code is a never ending process. If the development team had more time to spend on this area they would have analyzed the code to find potential components or functions that could have been optimized and measure the codes performance. By doing so the team could have reduced unnecessary resource allocation and improve the overall user experience.

Code readability and standards. If the team were to continue developing the product then it would have been important to make the code more readable and create a standard for how future code should be written. These points are important because it would have been easier to maintain and develop the product further and make it easier for potential collaboration partners or new employees to familiarize themselves with the code. Some ways to achieve this would be to rename variables and functions, commenting the code and organize it into logical folder and modules.

5.8 Challenges and solutions

Throughout the development process, there were challenges which we had to overcome. One of the main technical challenges were to design and adjust the database models to allow the creation of workouts correctly considering our development stage. This had to be implemented using a many-to-many relationship since each workout has to contain multiple different exercises, and each exercise should be available for multiple workouts, a simple solution using default Django behaviour. The main issue arose after finishing the minimum minimum viable product (MVP). Additional information available for each exercise within each workout, such as the number of repetitions and sets, was required for future features. To implement this we had to deviate from the previous default Django many-to-many implementation which took a considerable amount of research to implement properly. We felt this change was a necessity for the functionality of the finished product vision and therefore persevered to find a non-compromising solution. The end solution implements a many-to-many relationship using a defined join table, requiring the use of multiple additional serializers.

Individuality was a challenge the development team had to face. At the start of the project, none of the group members had worked with agile programming before and even though there was selected a SCRUM master who was to delegate tasks and make decisions, everyone was given fairly free rein to influence the development of the application. This led to the members implementing their own functions for the specific components they created, which could perhaps have been collaborated on to create solutions that were compatible with the work of others. The design of the application was also affected as some of the members did not feel they had the right to change and design on the components of other group members and the application ended up having different functionality designs.

6 Conclusion

In this project, the team has used several methods and tools from agile development. The methods have been used as a software development approach that combined emphasizes flexibility, adaptability collaboration, and iterative progress. The software application created using these agile methods is a usable workout application which delivers on the goal we set for the software.

None of the team members had experience with agile product development and with larger teamwork projects at the beginning of the semester. In addition, it was the first time that all the team members worked with the technologies React and Django. The project therefore had a steep learning curve for our team, both the technologies and agile methodology took some time to understand. At the same time, git automation with branching and merging has created unexpected complications in the form of merge conflicts. This was also something the team had to get used to as none of the team members had used git with so many team members on the same master branch.

Although the product works and delivers on the goals we set ourselves, it is not without the team having learned some lessons. All agile methods (and other methods) have a purpose as to why they are used the way they are supposed to be used. If each of the team members does not familiarize themselves with the methods and simply implement them without considerable reflection, the team may risk that the method does not works as intended. In other words, every team member has to immerse themselves in methods, especially when we don't have a real scum-master.

It is important to recognize that not all agile methods may be suitable for this particular project, or these team members. However, this does not mean that it will not work in another setting. You may want to adapt agile methods as the project unfolds. Sometimes you can remove methods because they are superficial, while other times you have to bring in methods to solve ”problems” with the project.

To sum up, our application is a full functional software which can be used for the purpose to generate workouts. The use of user-stories and product vision played a significant role in guiding our development process, ensuring that we maintained focus on our product vision. The development process has been influenced by agile methods and especially the scrum methods. In the end all the requirements were met, but there is still a lot of room for improvement before the application is launched.

7 Refrence APA

1. Sommerville, I. (2021). Engineering software products an introduction to modern software engineering (1st ed.). Pearson.

2. Wikimedia Foundation. (2023d, May 9). React (software). Wikipedia. https://en.wikipedia.org/wiki/React-(software)

3. Wikimedia Foundation. (2023, April 29). Kanban. Wikipedia. https://no.wikipedia.org/wiki/Kanban

4. Wikimedia Foundation. (2023, May 1). Django (web framework). Wikipedia. https://en.wikipedia.org/wiki/D
(web-framework)

5. Hombergs, T. (2022, May 20). Complete guide to axios HTTP client. Reflectoring. https://reflectoring.io/tutor
guide-axios/