Háskólinn í Reykjavík

Instructor: Ólafur Pálmi Guðnason

Examiner: Hildur Andrjesdóttir

Dyngja

Virtual Stock Trading Competition Website

Eggert Orri Hermannsson

Friðrik Steingrímsson

Rúnar Vestmann

Þorri Guðmundsson

Table of Contents

1 Introduction

Dyngja is a new and upcoming company that is looking to fill a certain void in the Icelandic stock market. Dyngja has an unreleased stock trading app that allows users to buy and trade shares of the Icelandic stock market with fake currency, named vISK, which follows the value of the Icelandic Króna. On top of that, the app also teaches its users some fundamentals for stock trading. This allows people to get some stock trading experience without any risk of losing money and is a good opportunity to get to know the Icelandic stock trading market.

Dyngja wanted to expand the usage of their app by creating a website that allows users of the app to compete in stock trading contests. They submitted a final project request at Reykjavík University and a group of students teamed up with Dyngja to turn their idea into reality. The students, who will from now on be called the “development team” or “team”, are Eggert Orri Hermannsson, Friðrik Steingrímsson, Rúnar Vestmann and Þorri Guðmundsson.

The development team’s goal was to develop a website where users of the Dyngja app can create and host stock trading contests that users of the app can participate in. It is possible to view these contests to see how each contestant is performing in that contest. A ranking system was implemented based on the total value of each contestant, some statistics are shown for both the contests as a whole and for each contestant. The performance of the top 3 or the bottom 3 are displayed in an interactive graph where the user can choose from several date ranges to view, depending on how long the contest has been ongoing.

The original idea was that the website would only feature contests and would mostly just be used for administrative purposes. An example of such usage would be a teacher to host a stock trading contest for his class. Dyngja and the development team quickly expanded the scope of the website which resulted in the website becoming an extension of the app since many of features in the app were also accessible through the website with the addition of contest creation and management. Hence the final product is a standalone website where users can trade virtual stocks, follow each other to receive news feed updates of stock trades as well as host and compete in contests.

2 Methodology and Working Process

2.1 Work Schedule

The development team followed an agile methodology very similar to Scrum. The main differences were that the team didn't have daily stand-up meetings but rather had 3 stand-up meetings per week. This was decided because not all members would be working on the project daily due to day jobs and other courses.

As is tradition in Scrum, one of the team members took on the role of a so called "Scrum master". The Scrum master was responsible for keeping the team productive and agile by being the headman when it came to external communication, led the discussions in team meetings, had a bird's eye view of the project as well as guiding the team onward when things didn't go as planned.

The plan was to work in sprints, where in each sprint there was high-level goal and specific tasks already laid out beforehand. Before each sprint the team planned what tasks they would be working on for the next two weeks by looking at the list of requirements, deciding the next sprint's goal and creating tasks that needed to be worked on in order to fulfill the sprint goal. The product backlog is the prioritized list of requirements that was put together based on conversations with the Dyngja team. During each sprint, team members picked any available task he wanted to work on. The team kept up active communication throughout the sprint with stand-up meetings and text messages / brief status reports via Discord.

It was up to each team member how he chose to complete a given task. Some tasks were better solved by more than one person, since it's good to have someone else to bounce ideas off, and other tasks were better suited for a single person. If someone was struggling with a task the team always made room in their schedule so that another member could jump in and help. If any design decisions needed to be made during a sprint, they could be brought up during a stand-up meeting or a specific team meeting might be scheduled to decide how to solve the problem.

After each sprint, the team went over the progress made during the sprint with members of Dyngja by demonstrating the new changes. There was also a sprint

retrospective meeting where the team members reviewed what went well and what could've gone better in the previous sprint to improve the next sprint.

In total there were 8 sprints and each team member spent a minimum of 300 hours in total working on the project. Each member split this time up evenly between each sprint as best he could but there were risk factors which lead to fewer hours spent in one sprint and more hours spent in another. Each team member was responsible for his own working hours and how he wanted to spread the workload over the course of the project.

2.2 Product team

The product team was split into 3 main groups: the product owner, scrum master and the development team. The product owner was the Dyngja team, but Reynir Freyr Hauksson completely oversaw the technical side of the project as he is the developer of the Dyngja app, so all the technical communication went through him. The rest of the Dyngja team were responsible for coming up with a design look for the website and providing the development team with feedback on the developing product.

The development team was composed of Eggert Orri Hermannsson, Friðrik Steingrímsson, Rúnar Vestmann and Þorri Guðmundsson who are all students in computer science at Reykjavík University. They worked together on all aspects of the project and shared the responsibility for the outcome of it.

In the first sprint the development team decided to ignore the role of a scrum master, but quickly realized that it was necessary to have someone with the role of overseeing certain things, like planning meetings and controlling the flow of meetings. The team unanimously selected Rúnar for this role with the possibility of someone else taking over the role at some time during the project.

2.3 Communication

2.3.1 Client meetings

An agreement was made to have a status meeting on Zoom/Teams with Dyngja every two weeks at the end of each sprint and update them on what the team had been working on. Outside of those meetings, the main communication tool with Dyngja was Slack where the team updated Dyngja on various parts of the project and asked for feedback. This proved to be very useful as Dyngja gave useful feedback on most things the development team updated them on.

2.3.2 Instructor meetings

The team had weekly meetings every Tuesday with Ólafur Pálmi Guðnason on Zoom who was the team's instructor throughout the project. During the meeting the team reviewed the current state of the project and were given an opportunity to ask questions if needed. Aside from those meetings the team exchanged emails and messages with the instructor through social platforms.

2.3.3 Standup meetings

In the first 2 sprints the team had no real schedule for standup meetings during the sprints, the meetings were irregular and at different times. The team quickly saw that this was not working because most of the meetings would get moved or cancelled because of short notice. After that, a plan was made to have regular meetings every Monday, Wednesday, and Friday at 17:00. If any member of the team was unable to attend at that time, the meeting would still be held and the member who missed the meeting would be updated later. The number of meetings was increased to every weekday from sprint 5 and onwards due to the team having a lot more time to work on the project.

2.3.4 Sprint retrospective meetings

The team held sprint retrospective meeting every two weeks on a Sunday, the last day of the current sprint. The purpose of these meetings was to evaluate how the sprint went by focusing on these three points:

- What went well
- What could have gone better
- How to adapt/improve for the next sprint

2.3.5 Sprint planning meetings

Following the retrospective meeting the team would then plan the next sprint by following the checklist below:

- Make sure that all completed tasks are marked as done in Jira
- Complete the previous sprint in Jira which will automatically move incomplete tasks to the next sprint
- Look at the backlog, add story points if necessary and move relevant tasks from the backlog to the new sprint
- Create tasks that will aid in fulfilling the sprint goal and make sure story points are assigned to all tasks for the new sprint
- Start the new sprint in Jira

Note: *Jira is a task planning tool that the development team used to manage tasks*

2.4 Workload

Table 1 shows the hours each member worked on the project during each sprint.

Sprint	Eggert	Friðrik	Rúnar	Þorri	**Total**
0	21h 30m	40h	55h 30m	21h 15m	138h 15m
1	21h 10m	14h 30m	38h 51m	13h 55m	88h 26m
2	45h 5m	43h 45m	49h 55m	40h 47m	179h 32m
3	35h 35m	23h 5m	48h 35m	43h 25m	150h 40m
4	30h 25m	26h 15m	27h 40m	31h 20m	112h 40m
5	74h 55m	77h 40m	42h 15m	82h 30m	277h 20m
6	67h 20m	55h 15m	39h 5m	81h 25m	243h 5m
7	61h 55m	71h 36m	28h 25m	64h 35m	226h 41m
Total	357h 55m	352h 6m	330h 16m	379h 12m	1419h 29m

Table 1 *Table showing the workload of each member for each sprint*

2.5 Technical environment

GitHub:

For code hosting and version control. The branch structure for the project was to have two branches that were deployed on the web, “main” and “dev”. The “main” branch was intended to be a stable release of the product which would be updated after each sprint. The “dev” branch was intended to be the branch which would contain all the latest features as soon as they were developed and a way to test out new mechanics as soon as they were merged in. Besides those two branches there were feature branches which were used to develop a specific feature until they were ready to be merged into the “dev” branch and then deleted. To merge a feature branch into the “dev” branch members would create a pull request which had to be reviewed by another group member.

Discord:

For communication between the members of the development team both via text and voice chat during meetings. Discord was also used to keep track of pull requests and there was a specific text channel dedicated for Github pull request links. When a code change was ready the developer created a pull request which essentially meant that he was requesting that his code gets merged with the development branch. Another team member reviewed the code change and if there were no comments then the code change was merged. However, if there were any comments or ways to improve the code then a discussion thread was created on Discord to both let the other team member know that there were some comments that need to be addressed and to have an active thread to communicate with one another. When the developer had addressed those comments and the reviewer is satisfied, then the pull request gets approved and the code is merged with the development branch.

Slack:

For communication between the members of the development team and Dyngja. This served as an active platform where ideas could be bounced back and forth. The development team often started the conversation by sending a screenshot of something they had been working on and Dyngja would then provide feedback.

Jira:

For sprint planning and task management. Tasks were assigned story point estimations which resembled the effort/importance of each task. The team would also use Jira for time tracking and its ability to produce burndown diagrams.

Notion:

Documentation for both the code and technical details, e.g., code snippets, coding conventions and developer guidelines.

VSCode:

Development environment used for writing code.

2.5.1 Code specific tools and environments

Frontend:

- Framework: Next.js[1]
- Programming language: TypeScript[2]
- Host for “main” and “dev” branch: Vercel[3]

Backend

- Framework: Django[4]
- Programming language: Python[5]
- Host for “main” branch: DigitalOcean[6]
- Host for “dev” branch: Heroku[7]

Database:

- System: PostgreSQL[8]
- Host for “main” branch: DigitalOcean
- Host for “dev” branch: Heroku

[1] https://nextjs.org
[2] https://www.typescriptlang.org
[3] https://vercel.com
[4] https://www.djangoproject.com
[5] https://www.python.org/
[6] https://www.digitalocean.com/
[7] https://www.heroku.com/
[8] https://www.postgresql.org/

3 Risk Analysis

The **Likelihood** and **Impact** columns contain numbers of the scale of 1-10 with 1 representing the lowest value of its column. **Risk factor** is the product of **Likelihood** and **Impact**. **Actionee** is the team member who is responsible for making sure that the response is carried out when a risk occurs.

#	Event	Likelihood	Impact	Risk factor	Response	Actionee
1	Increased workload in other courses	6	4	24	Group members will try their best to make up for it later	Þorri

#1 Event log:

31/01/2022 - 07/02/2022: The whole team was under a lot of pressure from deadlines in other courses during this period, resulting in the team not being able to invest as much time into the project as they would've liked.

26/03/2022 - 03/04/2022: The whole team was under a lot of pressure from deadlines and final exam preparations in other courses during this period, resulting in the team not being able to invest as much time into the project as they would've liked.

#	Event	Likelihood	Impact	Risk factor	Response	Actionee
2	Group member could suddenly become unavailable due to unforeseen circumstances	3	5	15	Other members will work on any assigned tasks that are deemed important, otherwise the tasks can wait until the person comes back. When the member is available again then there'll be a short catch-up meeting if needed	Rúnar

#2 Event log:

26/01/2022 – 31/01/2022: One group member had to go out of town for a funeral, so his contribution was minimal for about 5 days.

24/02/2022 – 26/02/2022: One group member got sick and could not work on the project as much as he would've liked to but has since been caught up.

27/02/22 – 04/03/2022 One group member got covid and could not work on the project so his contribution was minimal for a few days

08/03/2022 – 12/03/2022: One group member got covid and could not work on the project so his contribution was minimal for a few days

#	Event	Likelihood	Impact	Risk factor	Response	Actionee
3	Group member can't make it to a meeting	6	2	12	If the member is available at another time during the day, then the meeting time can be moved otherwise the meeting will take place and the member who was not present will be updated	Eggert

#3 Event log:

04/28/2022: one group member was unavailable for a meeting resulting in the rest of the group needing to inform him later on.

#	Event	Likelihood	Impact	Risk factor	Response	Actionee
4	Dyngja could be late with coming up with screen / feature ideas	3	6	18	Then we'll work on other things in the meantime, either do some refactoring or come up with our own ideas and see what Dyngja thinks	Friðrik

#4 Event log:

No log…

#	Event	Likelihood	Impact	Risk factor	Response	Actionee
5	Dyngja could be late with implementing necessary backend logic that our application needs	2	4	8	If we can build a simple mocked version of the backend logic which returns dummy data for us to continue working on the frontend. Otherwise, we will do that otherwise we will work on other features in the meantime	Rúnar

#5 Event log:

No log…

#	Event	Likelihood	Impact	Risk factor	Response	Actionee
6	Dyngja expects something from us that we can't deliver on time	3	6	18	We will explain to Dyngja why we can't meet the deadline. We will set a new, more realistic deadline and put the task as top priority, putting other tasks on hold if needed	Eggert

#6 Event log:

No log…

#	Event	Likelihood	Impact	Risk factor	Response	Actionee
7	Dyngja not satisfied with visuals or a specific feature	5	5	25	Then we'll note down what needs to be fixed and create tasks that'll be worked on either immediately or in the next upcoming sprint	Friðrik

#7 Event log:

04/25/2022: Dyngja pointed out a few things during one of our meetings that they would like us to change visually.

#	Event	Likelihood	Impact	Risk factor	Response	Actionee
8	Group member leaves the project	1	8	8	Then the scope of the project will be reduced in coordination with Dyngja	Þorri

#8 Event log:

No log…

#	Event	Likelihood	Impact	Risk factor	Response	Actionee
9	A feature that Dyngja requested does not seem to be possible or feasible	3	6	18	Explain to Dyngja why this feature won't work out and offer another solution to the problem	Rúnar

#9 Event log:

No log…

#	Event	Likelihood	Impact	Risk factor	Response	Actionee
10	A requested feature can't be implemented due to the group members' technical ability	2	8	16	Look for outsider help. If that doesn't work out then explain to Dyngja why the requested feature can't be implemented and find a feasible solution in coordination with Dyngja	Eggert
#10 Event log: No log…						
#	Event	Likelihood	Impact	Risk factor	Response	Actionee
11	Deployment plan isn't sufficient for our needs (e.g., we run out of database storage)	2	4	8	Ask Dyngja for an upgraded plan or move to another platform	Þorri
#11 Event log: No log…						
#	Event	Likelihood	Impact	Risk factor	Response	Actionee
12	Deployed application is not working as intended	4	4	16	Find out what the problem is, fix the problem locally and deploy the fix when it's ready	Rúnar
#12 Event log: 04/18/2022: A code change was deployed to the development website which contained a bug where on logging in you entered a refresh loop and you need to force quit the browser. This got fixed quickly and only affected the developers since this change got deployed to the development website.						

Table 2 *Table showing the Risk Analysis*

4 Requirement Analysis

4.1 User Groups

Two user groups were defined for this system, Trader and Host, and can be seen in Table 3. The main difference between the two user groups is their technical environment and user goals.

User group	Background	Frequency	Environment	Main user goals
Trader	Age: 13-99 Education: Any Technical skills: Novice	How often: Daily How long: 15 - 30 minutes	Technical environment: Mobile phone, tablet Physical Environment: Any	Partake in stock trading contests. See their own progress and the progress for other users in the same contest.
Host	Age: 13-99 Education: Any Technical skills: Normal	How often: Daily How long: 15 - 30 minutes	Technical environment: Personal Computer Physical Environment: Any	Create stock trading contests that last for a fixed number of days and invite other users to partake in contests.

Table 3 *Table showing the User Groups*

4.2 Requirements list

The requirements list covers both frontend and backend related requirements where each requirement is ranked by priority. The requirements list was put together after meetings with Dyngja where the design of the website was discussed, as well as from prototypes for the website that they provided.

The requirements list was reviewed each sprint to see what requirements were done and if there are any additional requirements needed. The list was also used as a reference to create new tasks for upcoming sprints where each requirement was split into multiple tasks.

Priorities	
A	Critical
B	Important
C	Desirable

Table 4 Table showing requirements priority

Nr.	Description	User Group	Priority	Status
1	Can sign up via e-mail	Any	A	DONE
2	Can log in via e-mail	Any	A	DONE
3	Can invite a user to a contest	Host	A	DONE
4	Can remove a participant from a contest	Host	A	DONE

Nr.	Description	User Group	Priority	Status
5	Can end a contest	Host	A	DONE
6	Can create a contest	Host	A	DONE
7	Can navigate to a user's profile from the contest screen	Any	A	DONE
8	Can see a page for a specific contest you are hosting	Host	A	DONE
9	Can see a graph containing user trading information on a contest page	Any	A	DONE
10	Can see statistics/numbers (i.e., trades today, volume, daily change, total return, total turnover, number of all trades made since the contest started) on a contest page	Any	A	DONE
11	Can see participants on a contest page	Any	A	DONE
12	Can see a user's profile page	Any	A	DONE
13	Can see all contests you are hosting	Host	A	DONE
14	Can see a page for a specific contest you are participating in as a trader	Trader	B	DONE
15	Can sign up via OAuth (Google, Apple)	Any	B	DONE
16	Can log in via OAuth (Google, Apple)	Any	B	DONE

Nr.	Description	User Group	Priority	Status
17	Can edit a contest	Host	B	DONE
18	Can see users appear in search results	Any	B	DONE
19	Can see newest trades on a contest page	Any	B	DONE
20	Can change personal info in settings	Any	B	DONE
21	Can see a graph on a user's profile page	Any	B	DONE
22	Can see statistics/numbers on a user's profile page	Any	B	DONE
23	Can see user's followers on a user's profile page	Trader	B	DONE
24	Can see all contests you are participating in as a trader	Trader	B	DONE
25	Can turn on/off daily notifications	Any	B	SKIP
26	Can turn on/off whether you have an open/closed account	Any	B	DONE
27	Can send users an invite link to your contest	Host	B	SKIP

Nr.	Description	User Group	Priority	Status
28	Can see all the contests you are involved in, either as a participant or an owner	Any	B	DONE
29	Can follow users from the profile page view	Trader	B	DONE
30	Can choose what information you see on a user's profile (what contest is shown, etc.)	Any	C	DONE
31	The website user interface can be scaled down to a mobile view	Any	C	DONE
32	The website user interface can be scaled down to a tablet view	Any	C	DONE
33	Can buy stocks	Trader	C	DONE
34	Can sell stocks	Trader	C	DONE

Table 5 *The requirement list in a table format*

5 User Interface

5.1 Screen prototypes

During the first meeting with Dyngja the development team got screen prototypes that showed what the frontend would look like. After the first 2 sprints the team had managed to create all the screens that Dyngja had provided. These prototypes from Dyngja made the work a whole lot easier as Dyngja had spent some time on making a good-looking design that the development team could follow and use for the base visuals of the website.

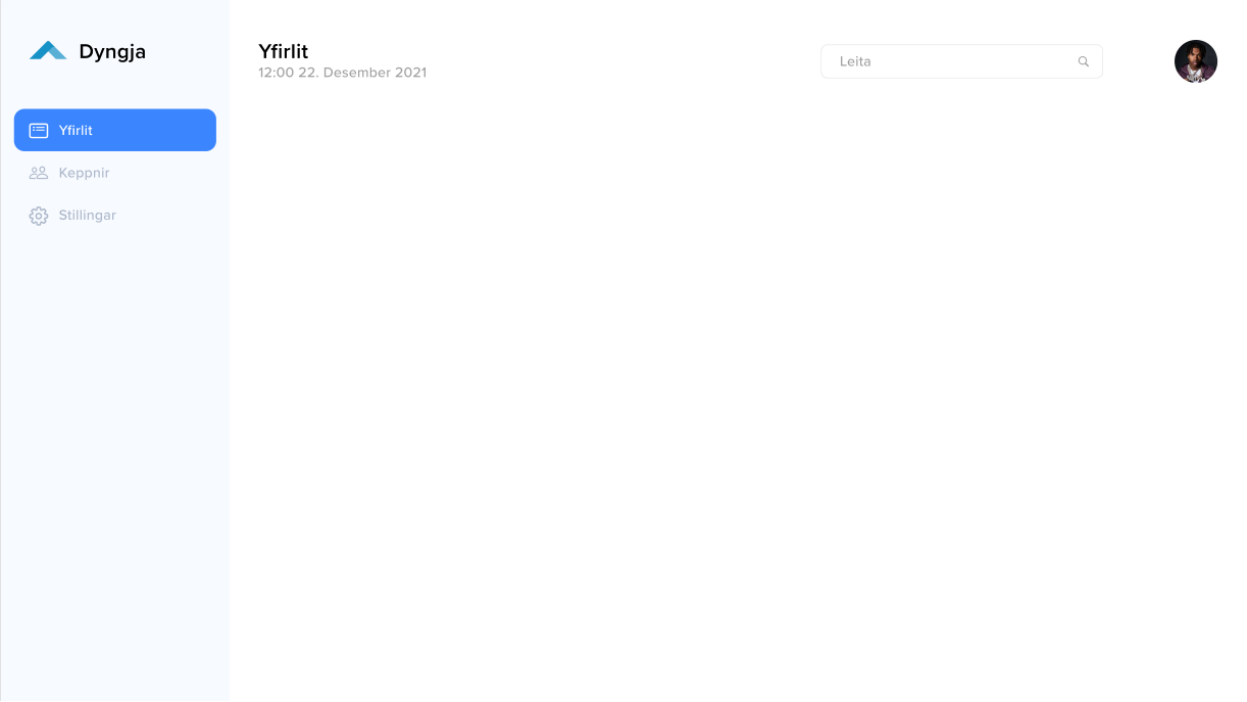


Figure 5.1 1 *The landing screen/overview screen*

Figure 5.1 1 is the prototype of the overview screen, but as can be seen Dyngja had no plans for the overview screen as they did not know what information it should contain but nonetheless wanted to have it on the website and decide later what should be displayed.

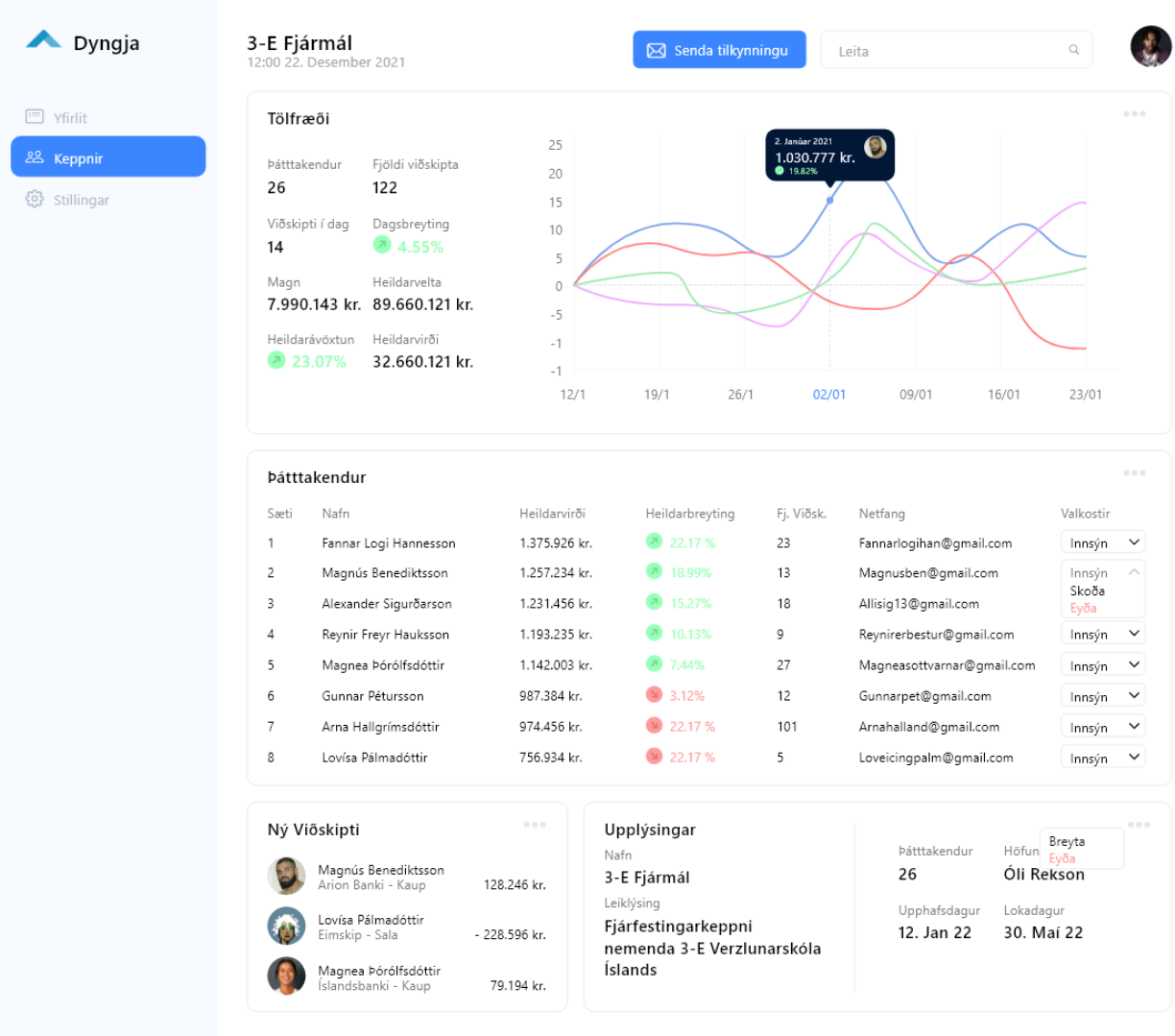


Figure 5.1 2 *Contest overview page*

Figure 5.1 2 is the prototype of the contest screen in action, it would showcase all the most important statistics for the contest and a graph showcasing the statistics “heildarbreyting” (total change) and “heildarvirði” (total value) of the top 5 users in this specific contest.

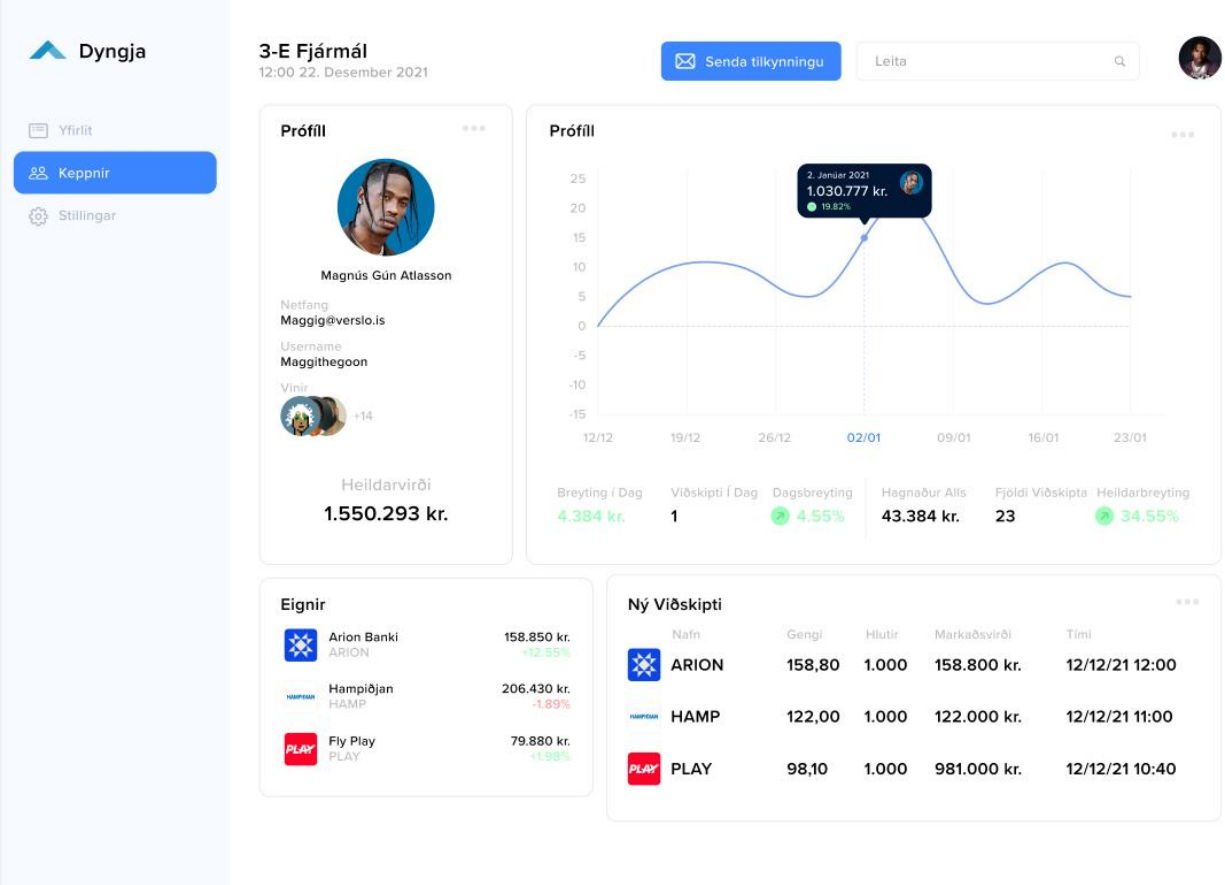


Figure 5.2 3 Desktop view of the overview page

Figure 5.2 3 shows the prototype of the profile page. On this page the user could see all the most important information about any user he either searched for using the search bar in the header, or if he clicked on a participant in a contest which redirects him to that participant's profile.

5.2 Screens of the final product

This section contains the visual designs of some of the most important screens of the final product. The development team tried to match the prototypes that the product owners had given them. Overall, there were a lot of changes made with the help of Dyngja, some screens were significantly changed while others stayed relatively the same.

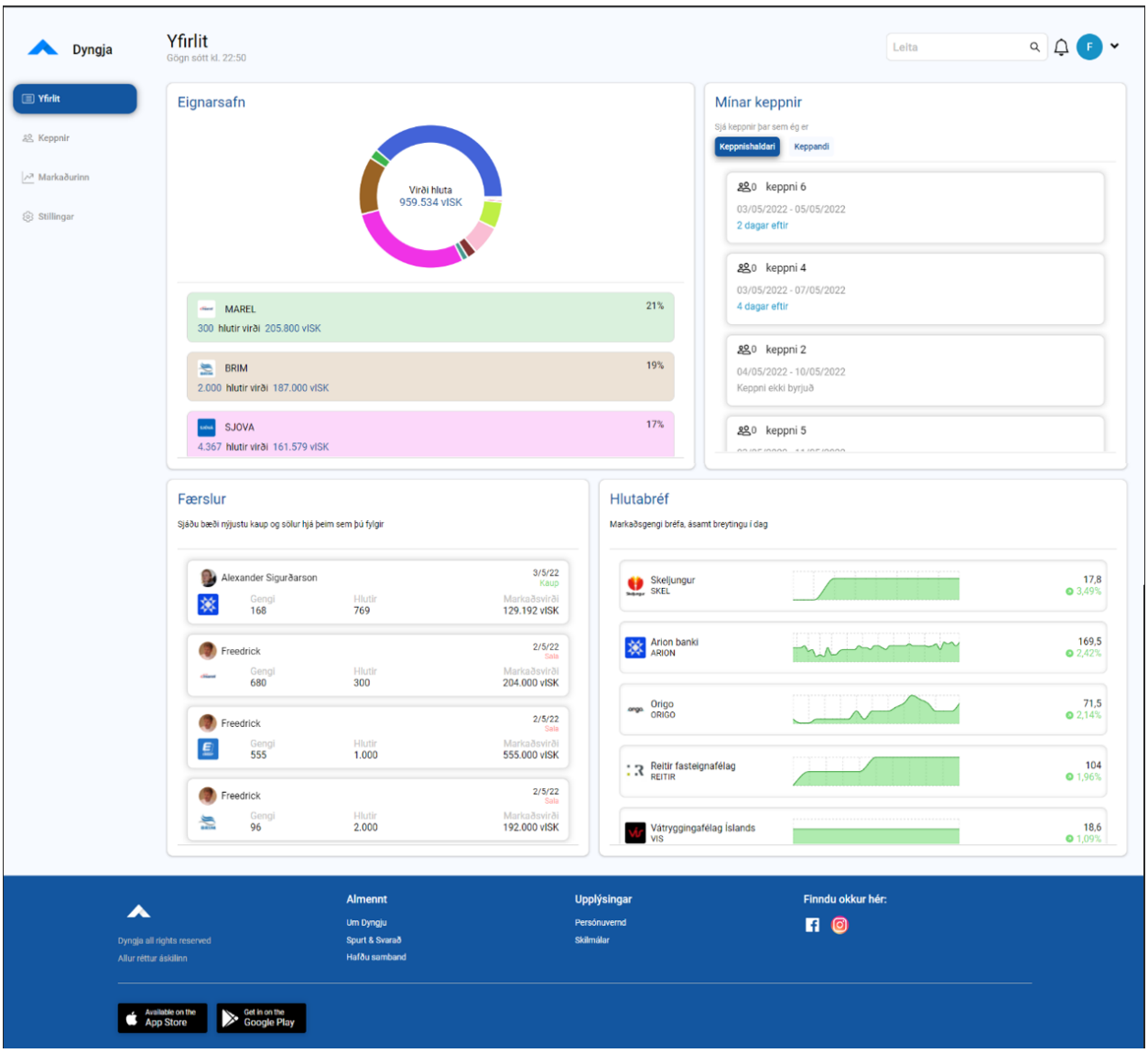


Figure 5.2 4 Desktop view of the overview page

Figure 5.2 4 shows the overview screen and how it changed the most because initially it contained no information or logic, so the developers almost decided to get rid of the screen. But in one of the later sprints both the developer team and the product owners agreed to make the overview screen contain some basic information that was already in the app but was still missing from the webpage. This information includes stock information and recent transactions made in the app.

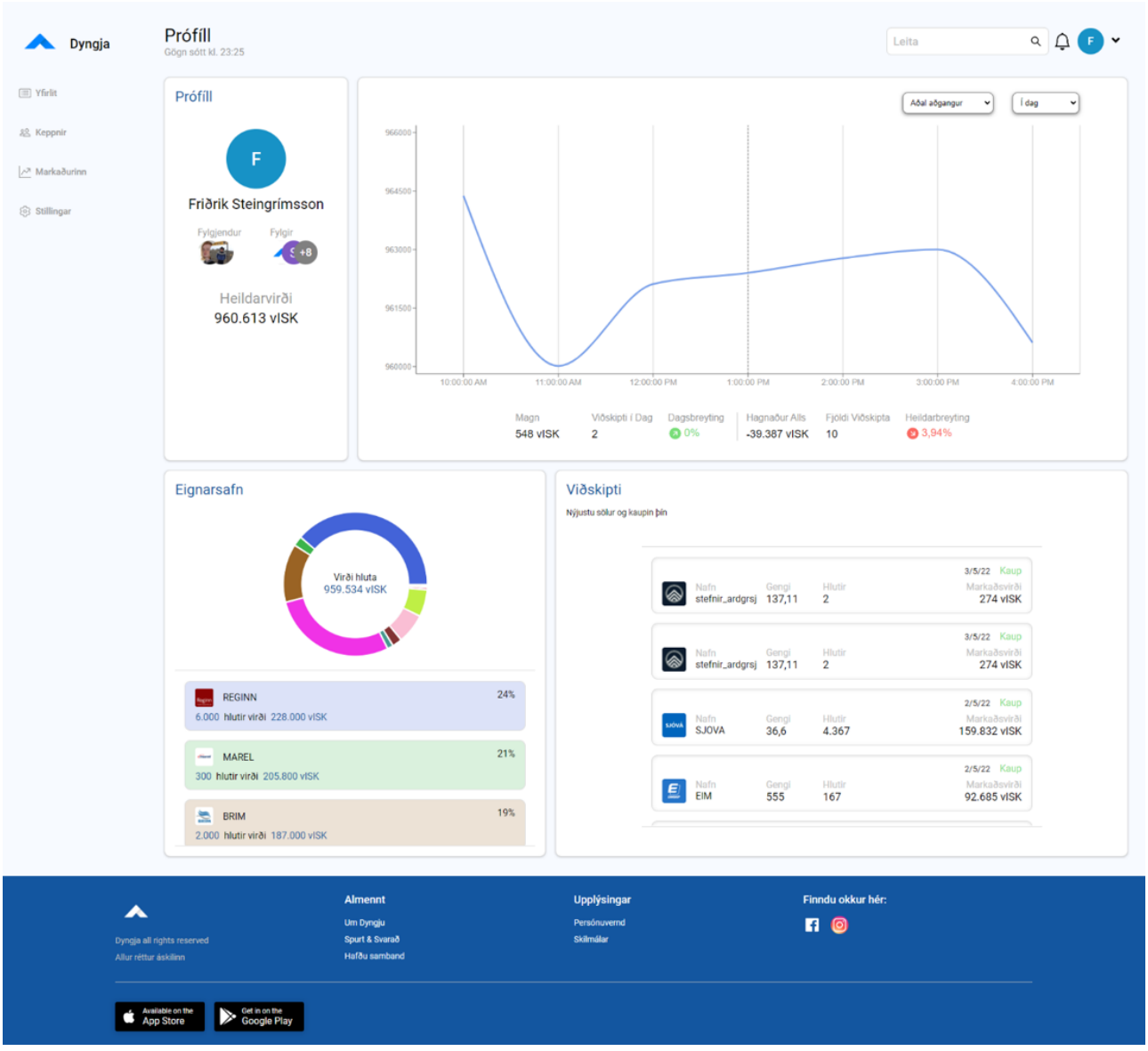


Figure 5.2 5 Desktop view of the user profile screen

Figure 5.2 5 shows how the profile screen ended up looking almost identical to the prototype seen in Figure 5.2 3 with some visual changes and improvements, but the final version still displayed all the same data and information as the prototype.

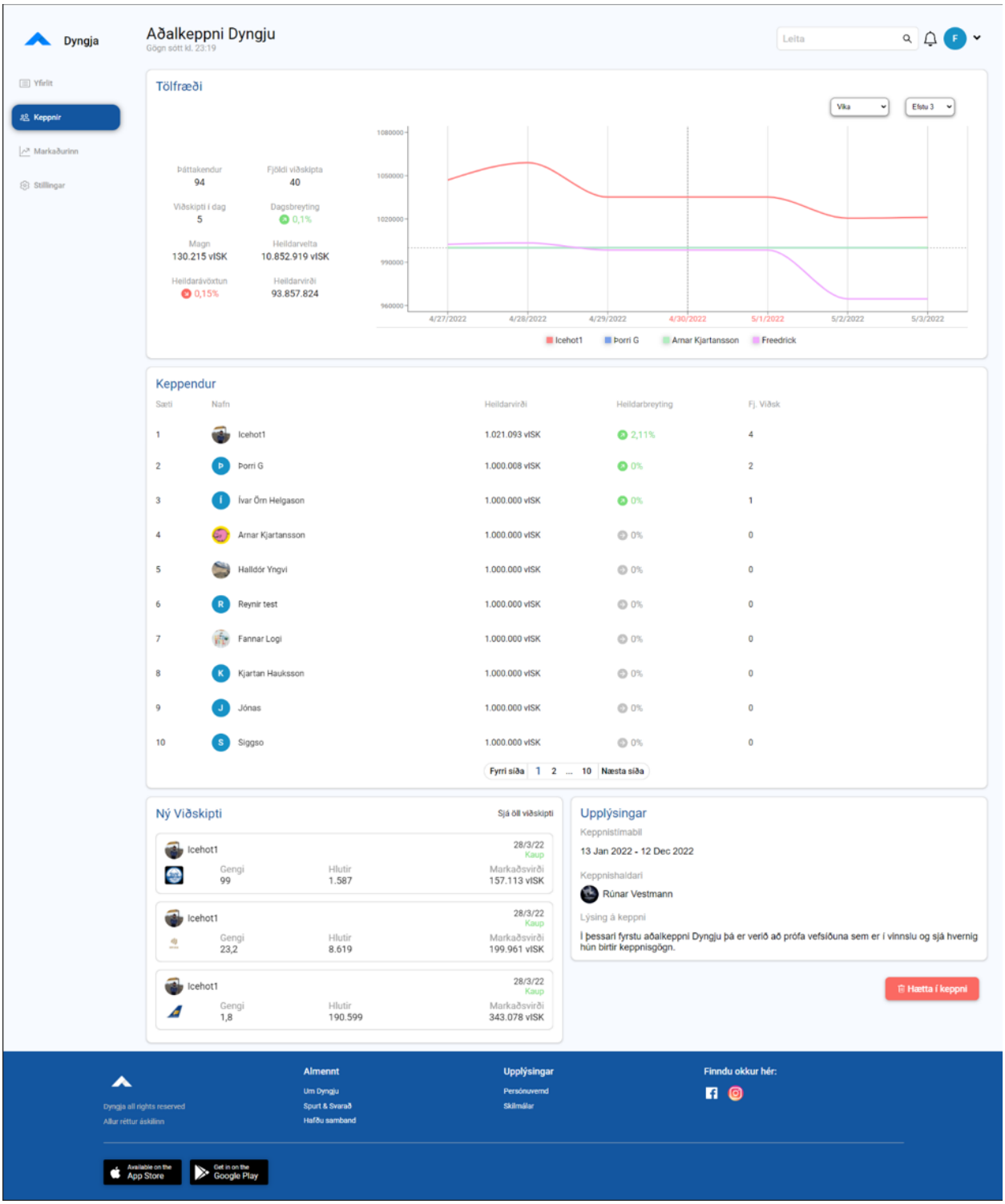


Figure 5.2 6 Desktop view of the contest detail page

In Figure 5.2 6, the final version of the contest screen ended up looking almost identical to the prototype seen in Figure 5.1 2 with only a few visual changes. The graph has some added features, such as a filter to view either the top three or bottom three contestants, selecting a date range and a legend where contestants can be toggled on or off in the graph.

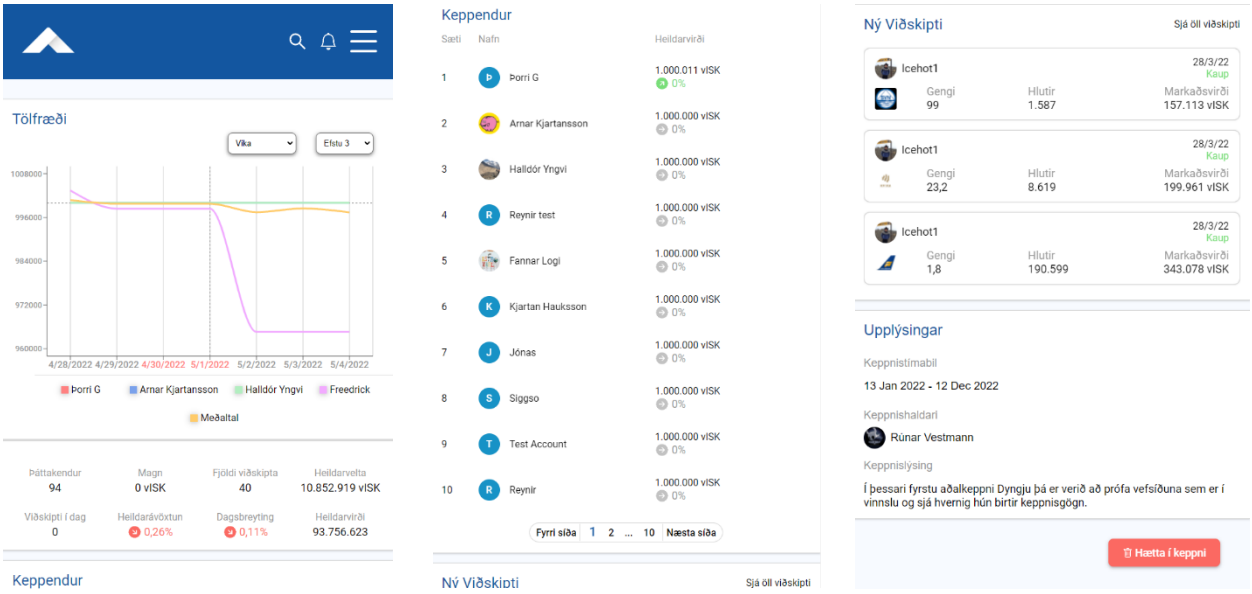


Figure 5.2 7 Mobile/tablet view of the contest detail page going from left to right.

The whole website had to be responsive to all screen sizes and the whole theme of the site changed a bit with the sidebar disappearing and a new blue header appearing at the top. Figure 5.2 7 is an example of how the final version of the contest screen looks like when viewing it on a mobile phone allowing users to open and interact with the website on their phones.

6 Technical design

As was discussed briefly in section 2.5, the project is hosted on two domains. One is the live website where users will interact with the website and the other is simply for the developers and Dyngja to preview things before they go live. This is done by having two specific branches for both the website git repository as well as the backend git repository. The “main” branch of the website talks to the “main” branch of the backend and the “dev” branch of the website talks to the “dev” branch of the backend. The “main” and “dev” branches of the backend have different databases, one is hosted on DigitalOcean and the other is hosted on Heroku. This is so that the development team can easily create test data and not have to worry about the live website also displaying that test data.

6.1 Frontend

The website was developed with the Next.js framework which allowed for page pre-rendering at build time as well as incremental static regeneration. This means that the Next.js server could already have created a page before it gets requested which reduces load times since otherwise the user’s browser would have to go through the effort to populate the page with the relevant data instead of the server having just done that beforehand. This feature is utilized for the contest page since a lot of users will want to access that page and it’s better to reduce the load time for those users simply by having a static copy ready for them.

The styling method chosen for the website was to use vanilla-extract[9] which allowed the team to setup the CSS (which is the styling language of the web) in a type-safe manner with the TypeScript programming language. The TypeScript code would then be compiled into a stylesheet at build time meaning that there was no performance hit while still having the benefit of writing type-safe styles.

[9] https://vanilla-extract.style/

6.2 Backend

Dyngja already had a backend that the mobile app uses to store user information as well as perform activities such as buying or selling stocks. The website also utilizes that backend since many of the features in the mobile app were also implemented for the website and it was decided that the website would communicate with the mobile app backend in order to access these features. This means that all user related activities are performed by the mobile app backend, and this includes user authentication, stock trades as well as being able to follow another user.

The main goal of the website is to provide a more fitting user interface for stock trading contests, therefore the development team had their own backend system which manages all contest specific data. Since the contest participants are the users of the Dyngja mobile app there needs to be some communication between the backend systems.

6.2.1 Backend Communication

Figure 6.2.1 1 shows a general overview of the communication to and from the contest backend.

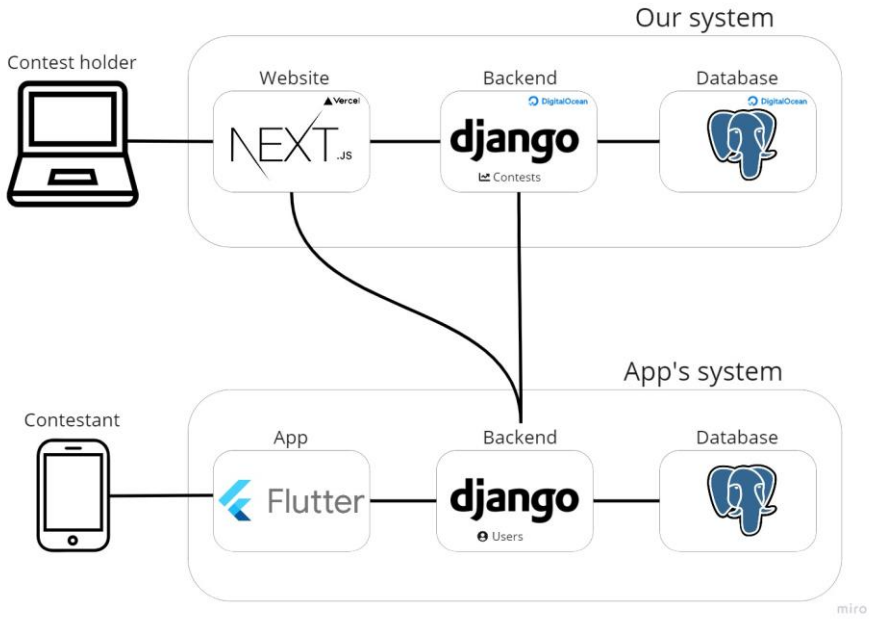


Figure 6.2.1 1 *Diagram of the project's backend communication*

There are two systems at play here, the mobile app's system and the website's system (which is called "Our system" in Figure 6.2.1 1). The development team was responsible for creating the system for the website which includes the website itself, which is developed via Next.js, the backend server, which is developed via Django, and finally the database which utilizes the PostgreSQL database management system.

The database and backend server are both involved in performing actions regarding contests. Since contests include users that could either be logged in the mobile app or on the website it was decided that all user authentications would take place in the mobile backend. Therefore, the web backend and the website need to communicate with the mobile application backend to authenticate users, get user information as well as perform user actions, such as creating a new business account when a user enters a contest. The mobile backend also takes care of stock related actions. It does so by having multiple business accounts per user where each business account holds stock information either for a contest or for the user's own personal possessions.

6.2.2 Database schema

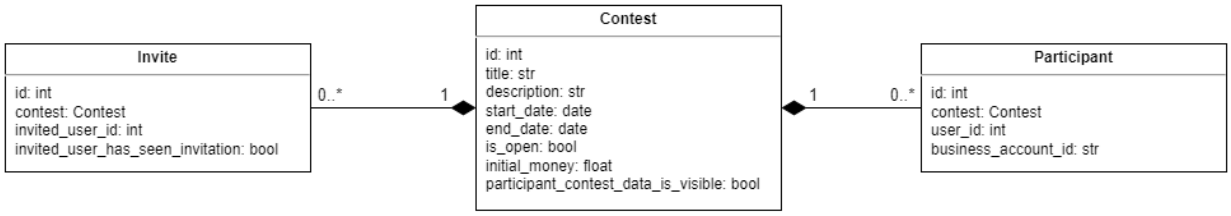


Figure 6.2.2 1 Database schema

The database schema consists of 3 models:

- **Invite**
  - Represents a contest invite
  - Gets created when a contest owner sends out an invitation
  - Will be deleted when the contest gets deleted
- **Contest**
  - Represents a stock trading contest
  - Cannot be deleted after it's started
- **Participant**
  - Represents a user that's participating in a contest
  - Will be deleted when the contest gets deleted

Once a contest has started the owner cannot delete it, but he can however change the end date, meaning that he can end the contest early, but it will still exist on the website and contestants can view its contents after the contest is over. This does however mean that a lot of contests will exist on the website, and this will have to be addressed at a later point in time but without knowing how users will end up interacting with the website it was deemed best to take the approach of not archiving/deleting contests after they're done since there's no real rush at this point to free up space since the public has not been introduced to the website.

7 Testing

7.1 Approach to testing

The team was determined to test the system in some fashion. The team expected to test individual functionality within the system using unit tests as well as having users uninvolved with the project test it in a structured manner. The user tests were planned to have the users complete certain tasks within the system and then give feedback afterwards. However, when the team wanted to conduct these tests, the Dyngja team wanted to make a contest featuring all the users of the mobile app which shifted the focus of the development team towards that contest. As a result, the team discovered problems such as poor performance that took priority, leading to the user tests not being conducted.

7.2 Backend testing

Since the backend uses Django the clear choice was to make use of a Django testing module called TestCase[10], which allowed for unit tests, and additionally the built-in unit test module in Python was used for creating mocked functions that run instead of their real counterparts in order to isolate the execution of a single unit that's being tested.

The backend unit tests were mainly focused on endpoints and helper functions related to contests and the goal was to see if operations were behaving as expected and whether the resulting data contained the correct information.

7.3 Frontend testing

For testing the frontend, unit tests were implemented using the Jest[11] framework. The team was mostly interested in testing the calculations involved with the contests. While these unit tests were used for checking if a calculation function yielded correct results, they also showed the team that some of the data types in the frontend were not

[10] https://docs.djangoproject.com/en/4.0/topics/testing/tools/#testcase
[11] https://jestjs.io/

complete mirror images of the real data, such as missing a value.
There were no issues caught by this in practice but with the unit tests in place, this at least provides a confirmation that the calculations are correct.

7.4 User testing

Throughout the project it was mostly Dyngja and the development team that opened the links to the two different deployments of the website. The “dev” branch deployment of the website proved to be very useful since the developers could quickly roll out a new feature and send Dyngja a link for them to test things out. This was often utilized and proved to be very beneficial since it gave Dyngja an opportunity to interact with the website and try it out themselves instead of having to rely on screenshot or a discussion to estimate the progress of the project. As a bonus, they could also try out newly implemented features and quickly give feedback to the development team.

The “main” branch deployment of the website talked to the production mobile backend, meaning that when some action was performed in the app (like a name or image change for example) then it would also be displayed on the website. This invoked the idea of creating a contest that every single user of the Dyngja app would participate in. Each user would then be able to buy or sell stocks using their main business account in the app. Normally when you join a contest, a specific business account is created but in this case a script was created to add every single user of the app and allow them to use their main business account since the contest functionality had not been added to the mobile app. This was very useful for the development team since having around 90 contestants in a contest really tested the website and it turned out that the graphs and contests loaded very slowly, which needed to be addressed. The solution was to statically generate the page beforehand instead of having users load everything themselves, asynchronously fetch the contestants’ user information, have fewer plot points for the graphs since the number of plot points were redundant and reduce the amount of database queries performed by the backend endpoint.

8 Progress Overview

8.1 Sprints

For each sprint, the team starts off by creating tasks to fill the sprint backlog with, then the team assigns each task with a set number of story points representing both the difficulty and importance of that task. Early on the team agreed to only give tasks labeled “forritun” (programming) story points and leaving the tasks labeled “skjölun” (documentation) and “fundir” (meetings) with no story points assigned. This decision was made since the team felt these tasks were too open-ended and it would be hard to represent them with an accurate story point count. In result this may lead to inaccurate burndown charts since the hours spent and story points do not go hand in hand and can be seen in a few of the charts where the time spent is high, but the team burndown line does not go down a single point.

The burndown chart has three major values, the Ideal burndown represented by the red line, the total hours worked each day represented by the blue bars and the team burndown represented by the yellow line. The yellow line in the sprint burndown charts or the “team burndown” line represents how many story points are left for that sprint. The yellow line only goes down when team members label tasks as “done”. Because of this the yellow line does not always go down at the end of each day since some tasks can take a few workdays. Also bear in mind that tasks could also have been created later in a given sprint resulting in an increased number of story points.

Sprint 0

In this sprint the team focused on setting the project up, planning what tools to use and so forth. The team started using Jira to manage tasks and work hours. For this first sprint no story points were assigned to any task since the team was still mostly setting things up and researching material that would be useful for the next few sprints. At the end of the sprint, the team had a very simplistic website up and running which didn't contain any logic (except the ability to log in / sign up via e-mail) but did however vaguely reflect the prototype screens that Dyngja provided.

Hours spent on sprint 0: 138 hours

Sprint 0 burndown chart

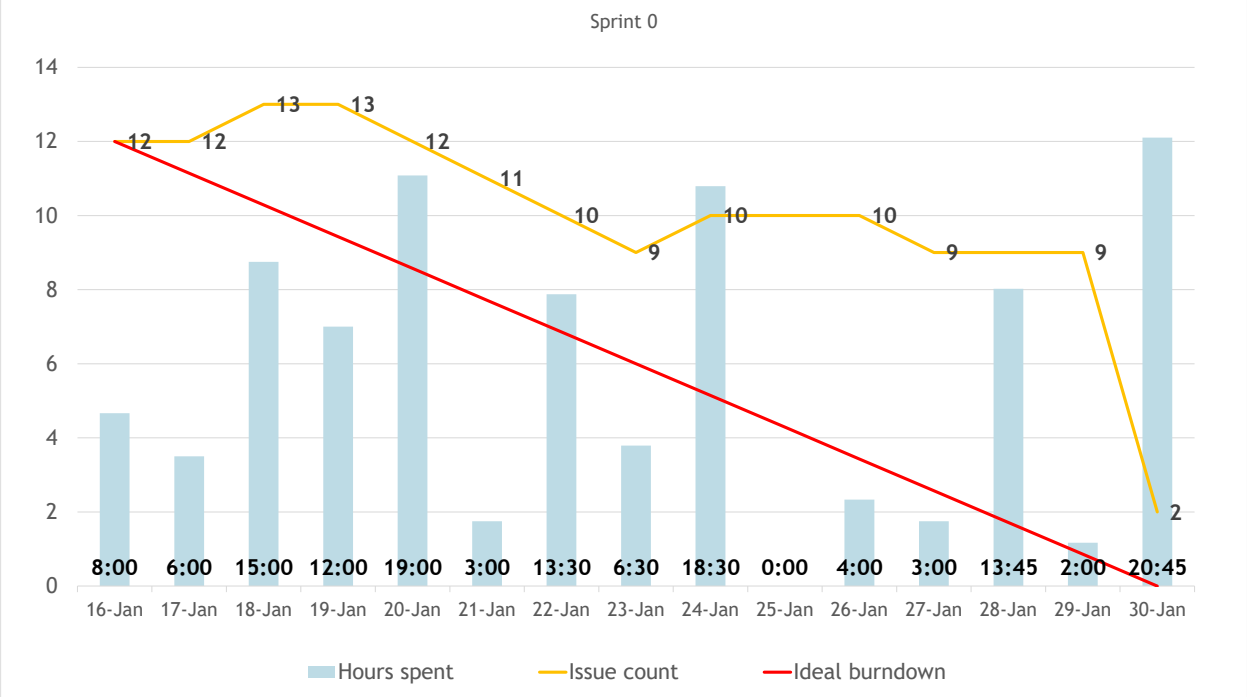


Figure 8.1 1 *Burndown chart for sprint 0, showing the hours spent, ideal burndown and the team burndown.*

What went well

Setting up the project went well, and most members felt that this first sprint was an investment which would pay off in the next sprints since most of the time went into setting things up and learning more about the tools that would be used throughout the project.

What could have gone better

Planning could have gone better regarding schedule and meetings. In general, the team felt that they needed to have a stricter schedule regarding team meetings since in this sprint the number of meetings was random, and the meeting schedule felt all over the place. Since the members were mostly working separately, they felt like the code wasn't consistent between members. The team also could've done more things in pairs, like pair programming, to get a smoother start to the project and to learn from each other.

How to adapt

Have standup meetings 3 times a week so everyone is up to date.
Agree on coding conventions and document those conventions. Have more pair programming sessions, especially when someone is having problems implementing a certain task, so they don't get stuck on it for too long.

Sprint 1

In this sprint the team got the backend and database up to speed and even ended the sprint by deploying both the backend and the database so the previously deployed web could connect to the newly made backend system. Members of the development team felt like this sprint was in general quite underwhelming because of how little time they could invest in the project during the sprint. All members of the team had a handful of large assignments in other courses, among other deadlines, so workload was high which really hurt productivity during the sprint.

Hours spent on sprint 1: 88 hours

Sprint 1 burndown chart

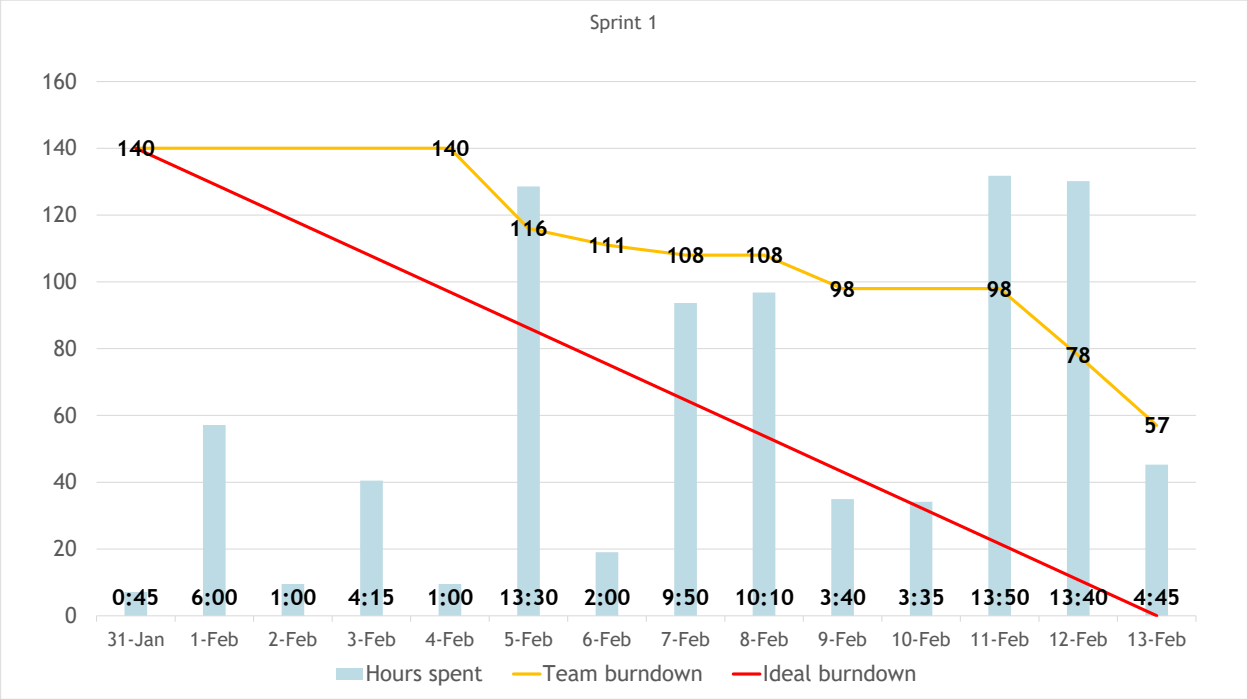


Figure 8.1 2 *Burndown chart for sprint 1, showing the hours spent, ideal burndown and the team burndown.*

What went well

The team set up the backend and a few endpoints that can be called to POST, PATCH and DELETE contests. Also, the team had a few good pair programming sessions and/or group programming sessions that were productive and were able to accomplish a lot despite how little time was spent on the project due to workload in other courses.

What could have gone better

Because of the high workload in other courses members spent a lot less time on the project during this sprint than needed. The team felt like their tasks weren't specific enough and some were way too big in scope and open ended to be a single task, so the team needs to be better at splitting tasks up. The team still needed a stricter schedule for meetings to keep every member informed on the current status of the project. It was not enough to plan to have 3 meetings a week since there was still uncertainty at what time of the day the meetings would take place.

How to adapt

Members of the team often felt like they were out of the loop for some parts of the project. To improve this there needed to be more effort put into keeping other members of the development team updated and making tasks in Jira simpler and more straight-forward in order to get a better feeling on what the project status is. Also, every standup meeting would now take place on Mondays, Wednesdays and Fridays at 17:00.

Sprint 2

All in all, this sprint went very well. There were no big setbacks and each team member contributed similar number of hours to this sprint. A lot of new functionality was introduced during this sprint and the team managed to finish almost every issue in the sprint backlog.

Hours spent on sprint 2: 179 hours

Sprint 2 burndown chart

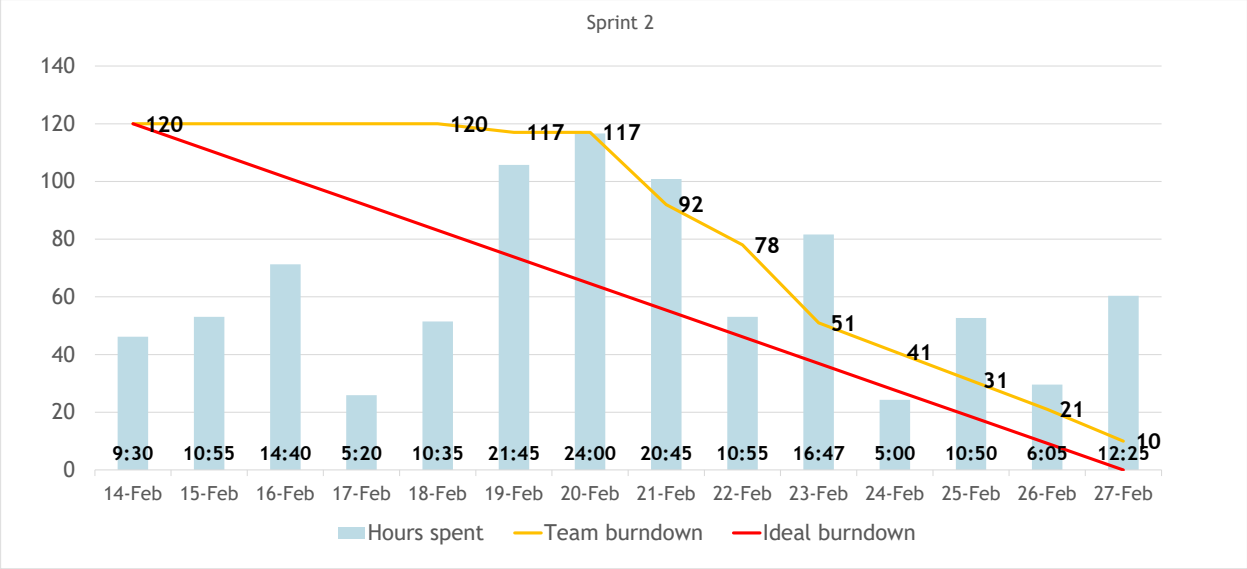


Figure 8.1 3 *Burndown chart for sprint 2, showing the hours spent, ideal burndown and the team burndown.*

What went well

The team made some readjustments to its workflow, such as establishing a fixed meeting schedule that the whole team adheres to. The team would have short stand-up meetings on Monday, Wednesday, and Friday. There were a lot of good meetings, and the team was able to finish most of the tasks they set up. Tasks were a lot better this time around, but still needed some improvement.

What could have gone better

The team still has a hard time cutting tasks down and because of that some tasks were abstract in their description or too big, some tasks were useless or redundant and to some the team gave too few or too many story points.

How to adapt

Make the description clearer for each task and create more smaller tasks instead of having fewer broad ones.

Sprint 3

In this sprint the team started working on producing and showing real stock trading data, both on each contest screen and on each user's profile. A notification system was implemented, where each user now gets a notification in the upper right corner of his screen when he gets an invitation to a contest. A few team members fell ill during this sprint, so their work output was limited at certain points. In general, the team felt like this sprint could have gone better, especially the planning stage. It became clear during this sprint that some tasks were too broad, and the story point estimation was way off in some cases.

Hours spent on sprint 3: 150 hours

Sprint 3 burndown chart

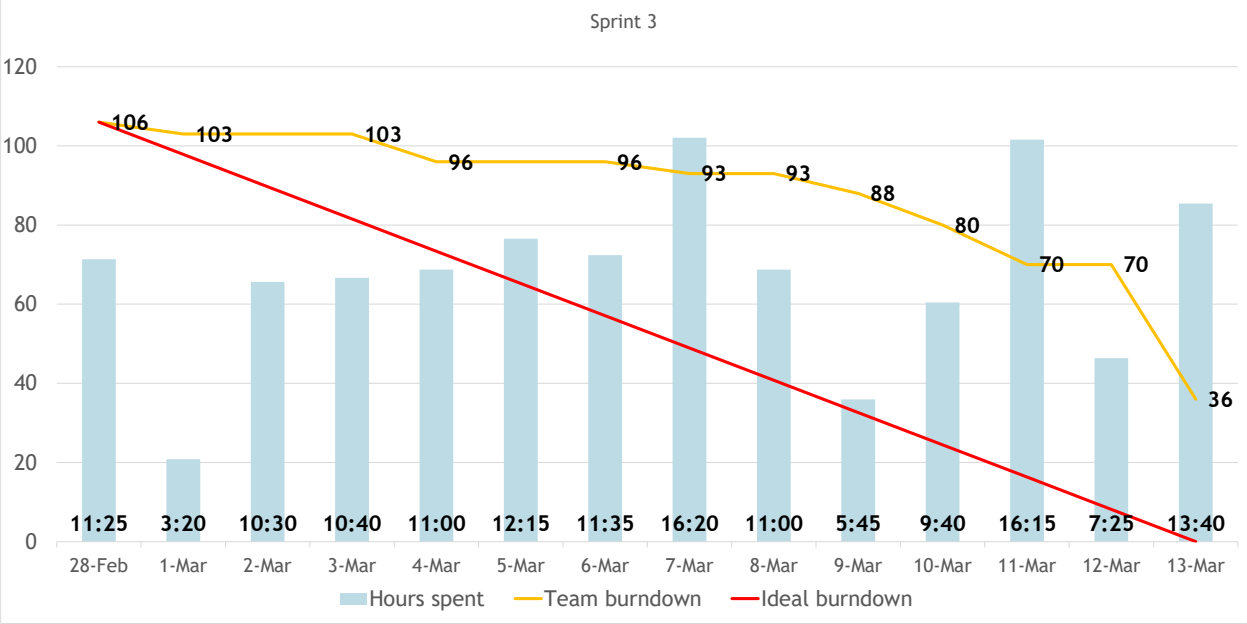


Figure 8.1 4 *Burndown chart for sprint 3, showing the hours spent, ideal burndown and the team burndown.*

What went well

Some features were fully implemented from the ground up, like the notification system. Stand-up meeting kept being a good platform for keeping everyone up to date, especially when team members got sick and didn't participate as much in the project.

What could have gone better

Group members found it hard to determine at what point tasks were done in this sprint. This is due to some tasks having too broad descriptions and some tasks being dependent on other tasks, which was not known until the team started working on them. The team had not remedied this problem fully since this has been a problem in prior sprints. These problems resulted in a very ugly and inaccurate burndown chart as seen in Figure 8.1 4. The team also felt like some stand-up meeting dragged on for too long and felt a bit less concise than in previous sprints.

How to adapt

Set a clearer definition for when a task is done. If tasks are dependent on other tasks, then have a clear working order for those tasks. Have a clearer idea of the scope for each task before assigning story points.

Set a tighter time frame for each team member to speak during the stand-up meetings, except for when they are introducing a new feature/changes.

Sprint 4

The focus of this sprint was supposed to be to continue working on the graphs for both the contest page and the profile page. However, mid-sprint, the team decided to take a break from the project since other courses that the team members were in had final projects that were due during the sprint. So, this sprint ended up being 3 weeks instead of the regular 2 weeks in length due to this break since the team did not want to start a new sprint while the course workload was ongoing.

As can be seen from the burndown chart below, the second half of the sprint still resulted in tasks being completed but that's due to miscommunication regarding when tasks are actually done. It seems that tasks were done but the person working on it did not deem it ready since the task description was ambiguous.

Hours spent on sprint 4: 115 hours

Sprint 4 burndown chart

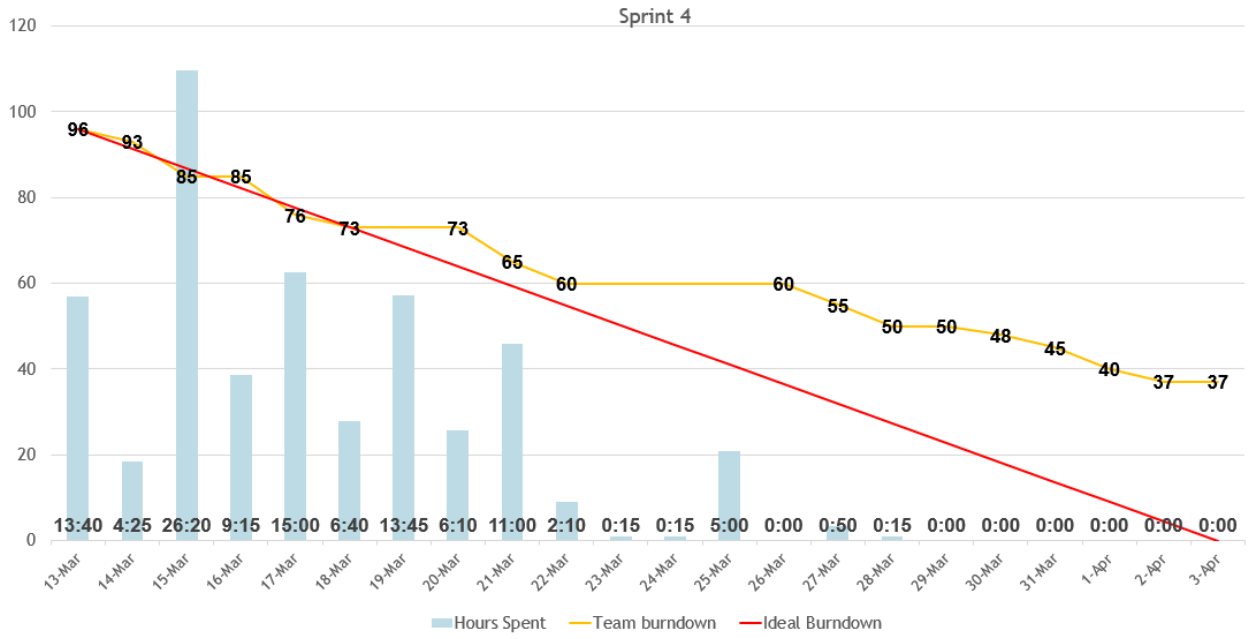


Figure 8.1 5 *Burndown chart for sprint 4, showing the hours spent, ideal burndown and the team burndown.*

What went well

Even though the team almost only worked on the project during the first half of the sprint there was still some progress made on the project as can be seen in the burndown chart.

What could have gone better

All group members were under heavy workload in other courses, so they were not able to deliver as many hours as estimated during the later halve of the sprint.

How to adapt

The team will have more time on their hands in the upcoming sprints so the work that was supposed to get done during this sprint will be addressed in the next sprint.

Sprint 5

The team could focus more on the project in this sprint since other course deadlines were mostly over. Here are a few things that were on this sprints task list:

- Log in support for Google/Apple
- Improved website layout in mobile/tablet views
- Graph calculations on the contest page
- Contest related calculations on the contest page

Hours spent on sprint 5: 277 hours

Sprint 5 burndown chart

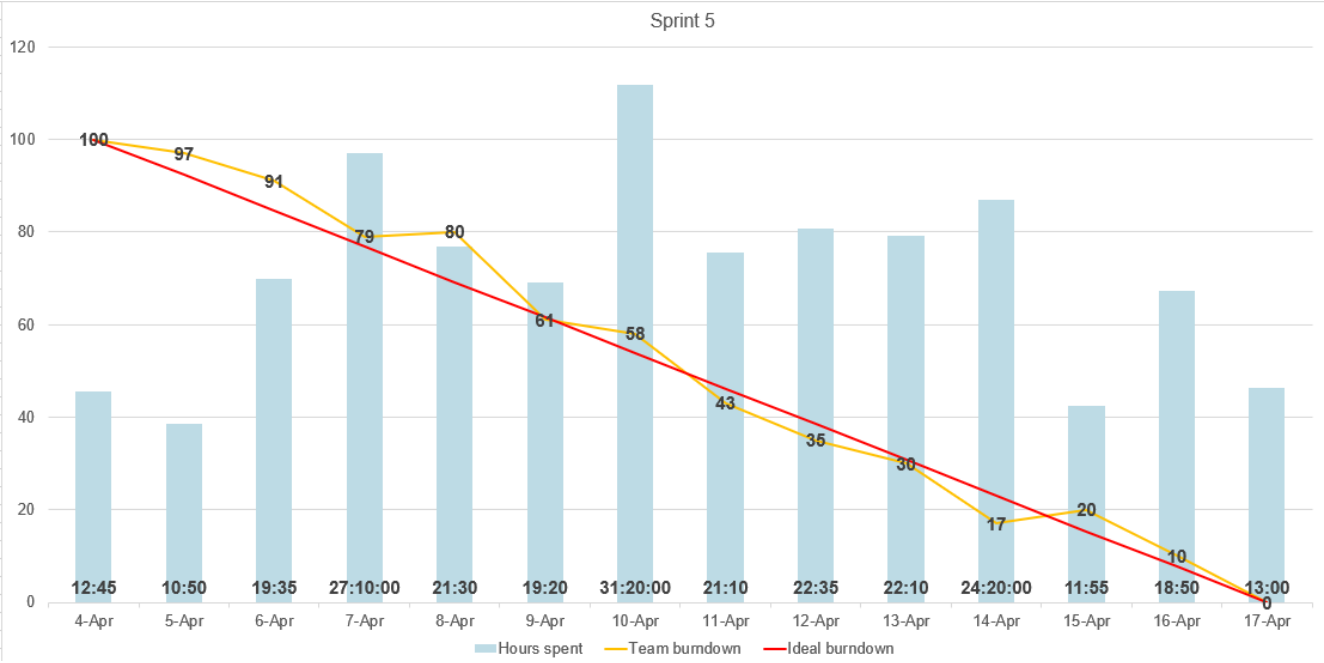


Figure 8.1 6 *Burndown chart for sprint 5, showing the hours spent, ideal burndown and the team burndown.*

What went well

All the team members were able to spend a good amount of time on the project during this sprint and those that were a bit behind on time spent on the project were able to close the gap during this sprint. This results in a near perfect burndown chart as can be seen in Figure 8.1 6.

What could have gone better

Few of the bigger tasks turned out to be too big and too ambiguous and should have been cut up in smaller pieces. A lot of time was spent on tiny visual details on the frontend that could have been resolved by talking it through with the product owners on what they thought was best.

How to adapt

More time needs to be spent on declaring tasks in the sprint planning sessions so that they are clear and not too broad in scope. Team members need to ask Dyngja more often for their opinion on various design decisions to maximize time efficiency on tasks.

Sprint 6

Even though the team had to prepare for their last final exam that was on the 22nd they still managed to put in a good amount of work in the first halve of the sprint. After the final the team was able to focus solely on the project which led to the team finishing all the tasks in the sprint. A lot of work was spent on refactoring some frontend code and functionality, making it a bit more readable and friendly for any outside programmer who wants to keep working on this project in the future.

Hours spent on sprint 6: 243h 5m

Sprint 6 burndown chart

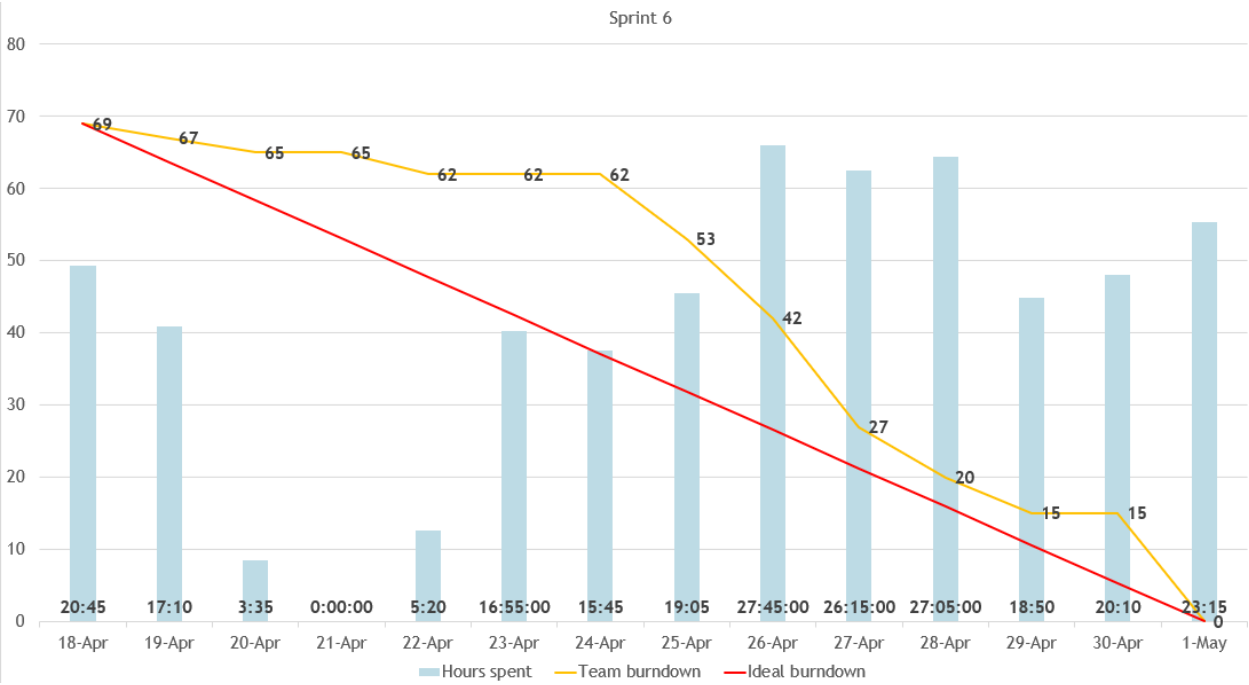


Figure 8.1 7 *Burndown chart for sprint 5, showing the hours spent, ideal burndown and the team burndown.*

What went well

Most of the core functionality and user requirements were finished during this sprint, allowing the team to refactor and clean up the code and even add some additional functionality like the ability to buy and sell stocks on the website just like in the app. This was not a part of the project at first but since the team was able to finish all the requirements it was decided to add the stock market allowing users to buy stocks on the web instead of just in the app. The team also had a good meeting with the product owners to finalize and nail down many of the decisions and concerns on a lot of the visuals for example the colors, button sizes, font sizes and more.

What could have gone better

The team should have finished all the major functionality and designs during this sprint so that the team could focus solely on bug testing, refactoring and such. However, that was not the case, so these things needed to be addressed in sprint 7.

How to adapt

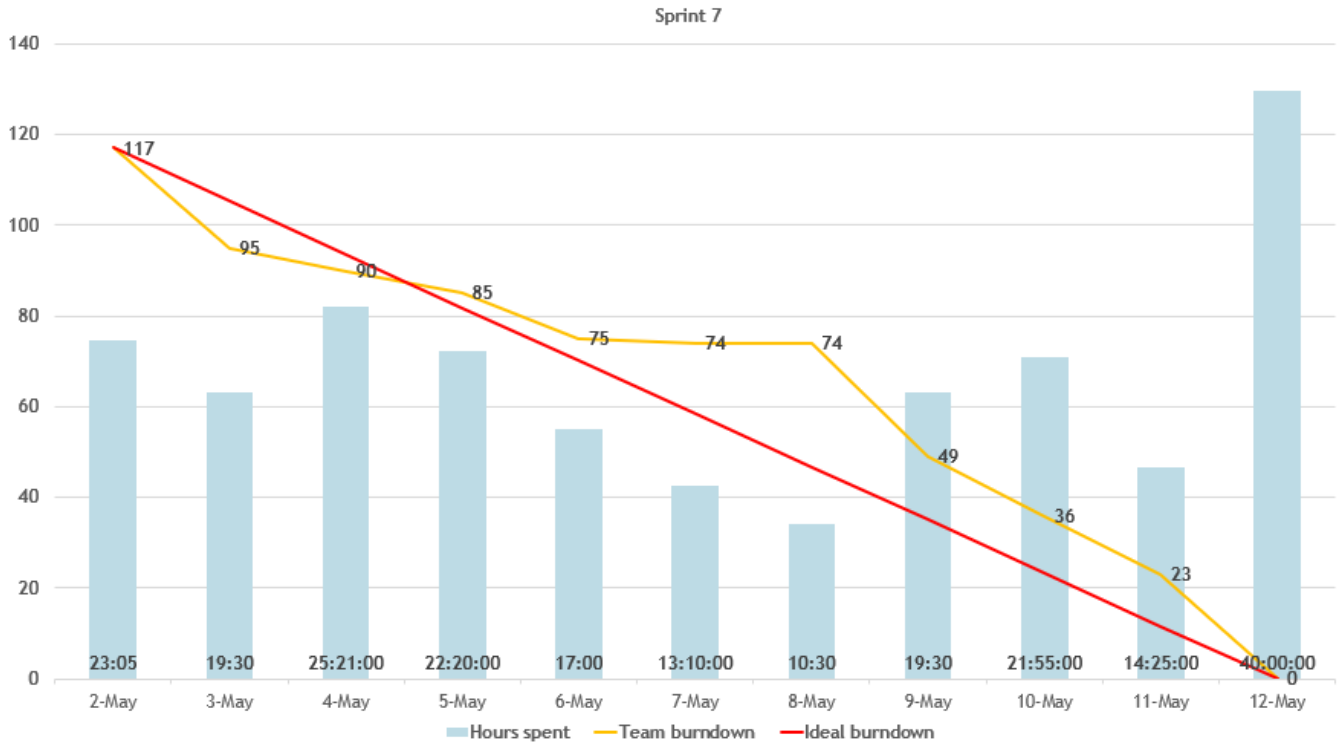
Finish implementing major functionalities as soon as possible so they can be tested thoroughly before the end of the last sprint.

Sprint 7

The goal for this sprint was to tie up all loose ends in the project and prepare documents for future developers of the project.

Hours spent on sprint 7: 226h

Sprint 7 Burndown Chart



What went well

The team managed to get the product to a deliverable state and the communication during the sprint was excellent, there were more and longer meetings than usual, but they were necessary to be able to decide on various aspects of this version of the product.

What could have gone better

There are some minor issues that relate to the scalability of the project that need to be addressed at a later point. This was made aware in the technical documentation and discussed with the technical lead of the Dyngja app who will continue developing the project.

8.2 Project burndown

Here we can see a burndown chart for the whole project. As we can see there are a total of 620 story points throughout the project and the team managed to finish all the story points in sprint 7. We can see a slight dip in logged hours for both sprint 1 and sprint 4, but at those times the team was under heavy workload in other courses as was covered in the risk analysis. Then we see the hours rise significantly in sprint 5 and onwards due to other courses having finished at that point so there was more time to spend on the project.

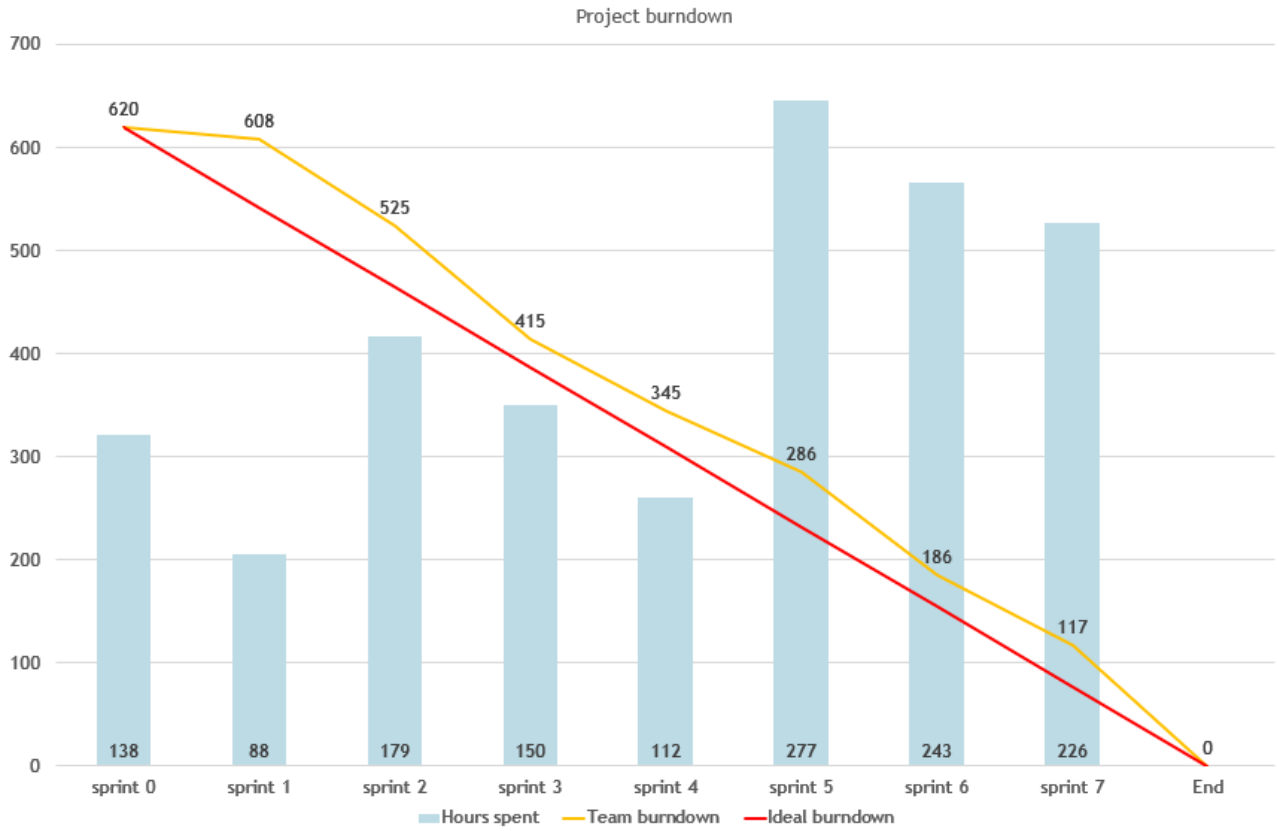


Figure 8.2 1 Project burndown chart

8.3 Division of hours by label

In Figure 8.3 1 we can see the time spent on tasks divided by each of the three labels used in Jira. Programming took up 2/3rds of the logged hours throughout the project. The proportion of meetings rose during the last two sprints since we increased the number of meetings from three times a week to five, so we saw a slight proportional increase during the latter half of the project. Time spent documenting remained about the same in each sprint since the team made sure to keep the report up to date and to document various technical details using Notion.

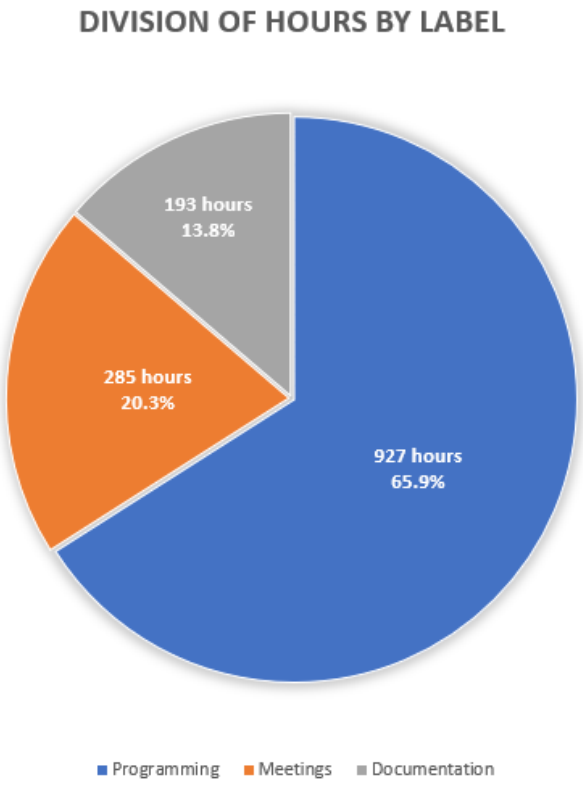


Figure 8.3 1 Division of hours by label

9 Project retrospective

This section serves as the team's reflection on how the project came together and how it went by, looking into the team's thoughts on the project. In general, we feel like the project went very well and was a worthy learning experience. We achieved more than we planned to do initially and there were no major setbacks. However, there were a few things that could have gone better. We would like to address these positives and negatives and then end by discussion potential future work.

9.1 Thoughts on the project

9.1.1 What went well

There is a lot to be said about things that went well. To start, the team worked well together, there were no conflicts between team members and each member contributed equally to the project. The meeting schedule which the team set up in sprint 2 turned out to be very crucial for the development process since that gave each team member a platform to update others on their current progress and seek resolutions on problems they faced. That way, everybody was always up to date on the current status of the project and no team member got stuck on a problem for too long without getting help from the rest of the team.

Only one team member had prior experience with Next.js and TypeScript, but since the whole team had experience with React and JavaScript the transition was relatively easy. Typescript took some time getting used to since it is a lot stricter than JavaScript and it forces the user to use types and interfaces to describe data. We feel that this was well worth the extra effort since using plain JavaScript can cause various errors that could be time consuming to debug. Next.js turned out to be even more beneficial for the project than we initially thought since it's features, like page pre-rendering at build time and incremental static regeneration, helped us in solving certain issues we had and were crucial to enhance the user experience on the website.

The team managed to finish much more than what was initially pitched for the idea for the website. The original purpose for the website was mainly to use it as an

administration tool for creating and managing contests, so only a portion of the users of the app would be using the website. In the final product every user of the app can log on to the website, browse through ongoing contests and even buy stocks, which is way beyond the original scope of the project. This makes the website more of an extension of the app, where most of the functionality of the app can also be found, instead of just an extra tool to use purely for administration purposes.

9.1.2 What could have gone better

There were certain things that could have gone better, especially in the first few sprints. The team had problems with estimating the amount of work for each of the first few sprints so there were a lot of unfinished tasks. That may also be due to unclear tasks, most tasks were too broad in scope and should have been split into a few subtasks. The definition of done for many of these tasks became unclear so many tasks were left “In progress” until the end of the sprint where they were marked as done and another task created for the following sprint if needed.

The team did not make time for the type of user testing we had initially planned, which was to get several individuals to perform certain tasks on the website and then submit feedback through a google questionnaire. This was due to Dyngja wanting to create a contest with all the current users of the app, as discussed in a previous section, which was instead used for testing purposes. Although this test provided very vital information about the performance of the website, we could have benefitted greatly from getting more feedback on things like usability and user experience.

At times it felt like there was some lack of direction regarding the design and functionality of the website from the product owner. This was especially noticeable regarding the page “Yfirlit” (overview) which was present in the original prototypes but never seemed to have any sort of purpose. This issue was brought up numerous times during our meetings with Dyngja but was not resolved until very late into the project. This did however not halt work on the project since there was always something else waiting to be worked on, so we did not feel the need to log this in the risk analysis. Other times we felt like we didn’t get enough critique from the owner when presenting ideas, which at times would have been beneficial to nail down a particular idea and

finish implementing it. Overall, these problems were really minor and are the only critique we have on the otherwise great working relationship we had with Dyngja.

9.2 Future work

Although we did manage to complete more than we had initially planned, there are many things that could be added to this project in the future. To start there was one priority B requirement, found in Table 5, that we did not implement. This was due to the fact that daily notifications were not implemented in any way into the website. The only notifications we have currently are contest invitations. Another feature that was not implemented was “Can send users an invite link to your contest” due to being more problematic than previously expected so it was deemed too time consuming for its benefit to the project. Other possible additions to the project could be:

- Expanding the notification feature, e.g., to get updates from followers
- Selecting contestants to show in the graph, instead of just top 3 or bottom 3
- Make graphs more responsive when changing screen size
- Add a “log in with Facebook” feature
- Adding features to customize profiles for each contest
- Make a more general root page for the website so that users don’t start on the log in page
- Messaging system between users
- Simple chat window/message board for each contest
- Send email to user when he receives an invitation to a contest

These are just a handful of possible additions to the page and at this time it is not clear what would be beneficial to improve upon what we already have. We would not want to overload the user with useless features or make the user interface too bloated or confusing. We should also have in mind that the features that are not directly tied to the contests should maybe also be present in the app. We are not trying to replace the app, we’re only trying to extend its functionality.