Project Team 12 Final Report

The product

Our project scope was to create a *“website game aiming to educate people about the stock market by letting users compete against each other with fictional stock portfolios updated with real-time stock data in a safe environment”*. This has been done, and it is up and running. Our initial milestones were to create a sign-up function, get the stock-api working and to create the user interface. These were all implemented in the planned time frame, and after that we added further features in order to meet the project scope and get the initial milestones working as soon as possible.

Two features that we included in our project scope that were not implemented in the final product are the league chatroom and news forum related to specific stocks. These were a bit ambitious, and were never part of the main features of the app. If we would have had more time we would probably have worked on those later on. Comparing our first mockup with the final product there are many similarities, and also some differences. The core features and the thought behind the different pages are similar, and the overall flow between the pages are the same. However, the specific design differs. Specifying the design was however never the idea with the mockup, and by visualizing the key concept it served its cause well.

Learnings:

1. **Doing the complex, but necessary, things first**
   One important learning we discovered during the project is the importance of prioritising the necessary functionalities for the project and working on them first. While we did follow this approach to some extent during each sprint, we could have had a more structured approach at the beginning of the whole project by mapping out the crucial functionalities needed for the minimum viable product. This means focusing on the core functionality of the product and ensuring that it functions as expected, even if it requires a lot of work and is not as visually appealing as other things. By taking this approach, we can reduce the risk of spending a significant amount of time and resources on features that may not be as critical to the product's success.

   During the project we spent a lot of time in the earlier sprints making sure that we had a nice landing and login page, making sure you could register, making sure you could search for stocks and that the API works. In retrospect, it would have been better to start with the No.1 core feature - participating in the leagues. These leagues would take more than one sprint to complete, and we started with them on sprint one, but we should have weighted the focus of the group more on that instead of having one or two developers there while the rest was working on the other functionalities. As a result, the landing and login page was completed after sprint one and from there we continued working on what we had built, taking incremental steps improving from this end instead of taking incremental steps from the core task as it was not yet developed. However, we also needed to make sure the API could work before investing the time in features relying on the API to work, so only focusing on the

league participating in the beginning could have backfired if the API did not work in the way the thought. Overall, this learning highlights the importance of taking a structured approach to identifying and prioritizing the most important functionalities of the product. By doing so, we can ensure that we are building a product that meets the needs of the users and delivers value to the stakeholders at any time of the development process.

2. **Relying on your foundation**
   Another thing we learnt when it came to building the product was how to apply the more theoretical programming skills we've learnt from previous courses to the more practical and dynamical issues of software development. This was especially the case when it came to using the Java Spring Boot framework, which most had not worked with before. Using it required a necessary foundation of java knowledge and then a lot of trying, experimenting and working together to make sure the application could run. However it turned out to work very well, and we realised that being afraid of using new frameworks (which we were in the beginning) is something you will have to shake off as a programmer, as there will always be relevant frameworks for the product you are building, and the theory you've learnt in other languages will help you manage these as well.

3. **Set-up Environment**
   One learning we discovered during the project is the difficulty of setting up a proper working space and ensuring the development environment works for everyone involved. Each sprint, we faced some issues related to setup, and this ultimately made it challenging to complete tasks as we underestimated the time required. To mitigate these issues, we learned to allocate sufficient time for setup and testing in each sprint. Communication with the team was essential here to address any issues related to setup and ensure that everyone had a functional development environment, especially within the groups. As a result, the product we delivered during the first weeks was not entirely as ambitious as we set out for due to time loss with the set up, at first it made us feel disappointed in ourselves but ultimately we learned that it is something that is bound to happen each week so we learned to handle the bar for the customer and not setting too high goals for ourselves.

   Underestimating the time required for this setup can impact the project's progress significantly. It's crucial to factor in the time required for setup not only at the beginning of the project but also with each update to the main branch. With each update, there is a risk that somebody might lose functionality if a branch with a bug has been merged, leading to issues that can delay progress. By accounting for the time set-up time we can reduce the risk of delays and ensure that the development environment works for everyone involved.

The team

As agreed on in the social contract, we have used Facebook Messenger and Slack as communication tools. Messenger was used for administrative matters such as meeting times and locations, and Slack was used for matters regarding code, programming and software development. Furthermore we decided to strictly communicate during weekdays, and keep weekends off which has worked well. By doing this, we have learned the importance of having dedicated communication channels for different types of discussions. It gave us a more structured organization and ensured that relevant information reached the appropriate team members. Also, if a team member showed up late to a meeting we have used to well-known term “fikapinnar” which resulted in a couple of tasty fika’s during the project, boosting the team spirit.

When it comes to file management and decision making we agreed on implementing majority decisions for overall functionality and look, and individual autonomy for small details and coding. This gave us a good balance between collaborative decision making and individual autonomy. While some major decisions required involving the entire team, other smaller and more specific tasks allowed team individuals to take ownership and contribute with their expertise. All pull requests to the main branch required another group member to review it in order to be merged, this was added as an extra security measure and made it easier to quickly adjust issues and bugs caused by unexpected issues with the pull requests.

Learnings:

1. **Communication and Collaboration**
   One of the most important things that we improved on as a team is effective communication and collaboration. We learned that frequent and open communication is essential for sharing information, addressing challenges, and keeping everyone informed about the project's progress, especially when the team members have their strength in different programming languages. TypeScript programming with the React framework was a bit troublesome to learn in the beginning for most members but once the team got comfortable with asking even the “dumb” questions, the effectiveness of the group increased.

   Throughout the sprints, we have had regular meetings, and used Facebook Messenger and Slack as communication tools and for exchanging ideas, feedback, and updates. During the first meeting of the week we had a brief discussion about last week and made sure that everyone was up to date. We also discussed the upcoming sprint goals, and have used planning poker as a way of evaluating our different user stories. As much as possible, we have preferred face to face communication over written communication as we found it easier to collaborate on code when sitting next to each other. By doing these things we could increase the communication which we found was one of the key factors for success in the agile team. Over all, our teamwork has worked very well throughout the project, especially after a couple of weeks when the different roles within the team were clarified.

2. **Adaptability and Flexibility**
   As the project has focused on agile methodologies, the importance of adaptability and flexibility in response to changing requirements and priorities has been very relevant.

   When our TA has given us feedback and wished for specific features we have learned that it is essential to be open to changes and embrace them as opportunities for improvement. In order to do so, we had to adapt our plans, reprioritize tasks, and adjust our strategies based on feedback and evolving project needs. This would have been difficult if we would have used a waterfall project approach, and it made the advantages of agile methodologies clear to us. We learned to see the client as a resource rather than an obstacle, which was useful in aiding our decision making process as we could discuss different features to see which we would prioritise.

3. **Continuous Improvement and Retrospectives**
   At the end of each sprint we have conducted retrospectives to reflect on our processes, identify areas for improvement, and make necessary adjustments. These retrospectives provided a valuable opportunity for the team to learn from our experiences and adapt our practices. Throughout the sprints we have discovered new ways to optimise our workflow, enhance collaboration, and overcome challenges encountered during the project. For example, while implementing pair programming we have tried to work closely between the pairs in order to keep the information flow running.

   By consistently focusing on continuous improvement, our team has been able to refine our processes and deliver higher-quality software over time. It was not, however, a walk in the park to make sure everyone actively participated in the retrospectives as we in the beginning were not used to such frequent reflections. After realising this we started going “around the table” to make sure that everyone had at least one reflection, when we in this way “forced” all to participate we could learn more from each other. From this we bring with us the value of making sure, even if it might seem silly, that everyone gets the time and attention to say what's on their mind.