



Fished App

IS-304: 2022

Title:

Subject code:	IS-304
Subject name:	Bachelor thesis in information systems
Course coordinator:	Hallgeir Nilsen
Supervisor/Mentor:	Janis Gailis
Client:	Oxidane Venture AS

Students:

Surname	First name
Andersen	Ole Marius
Brødsjø	Markus
Granås	Ole Bjørnar
Gustavsen	Niklas A
Valen	Michael Herland

I/we confirm that we do not cite or otherwise use other people's work without this being stated and that all references are given in the bibliography.	YES <input checked="" type="checkbox"/>	NO ___
Can the answer be used for teaching purposes?	YES <input checked="" type="checkbox"/>	NO ___
We confirm that everyone in the group has contributed to the answer.	YES <input checked="" type="checkbox"/>	NO ___

Abstract

This report summarizes a project done in collaboration with Oxidane Venture as the last part of a bachelor's degree in IT and Information Systems. The project spanned through the Spring semester of 2022, starting in early January and ending in late May. The project was offered by Oxidane Venture in regards to developing a mobile application for one of their investment companies, Fished. The main focus of the report is on the process, central decisions, and reflections.

As mobile application development was new to the majority of the group, it offered a steep learning curve. The curve flattened out as the project progressed and the group acquired more knowledge surrounding mobile application development. During the project, every member of the group has participated in all aspects to some degree. This resulted in an even learning outcome, although some specializations occurred during the project mainly regarding development and report writing. The state of the mobile application reached a satisfactory level of functionality, however there was still a long way to go before a release could be considered.

Preface

This report is written for people with knowledge of technical concepts in system development and the agile framework Scrum.

We would like to take this opportunity to express our gratitude to all the people who have been involved in our bachelor's project. It has been a tremendous educational experience where knowledge and tips from those involved have been of great help.

Thanks to Janis Gailis for guidance on the project. You have given us sound advice related to project management, technical advice, and more. You have also been available when we required guidance and constructive critique. We would also thank Hallgeir Nilsen who gave great advice on how to ensure good quality and process during his lectures and notes.

Thanks to Tor Oskar Wilhelmsen and Thomas Prestvik from Oxidane Venture for their expertise in this project. They helped us a lot with the technical aspect of development and gave us good insight into creating a better product and processes.

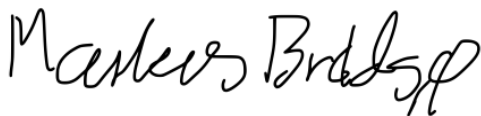
Finally, we would like to thank our family for helping us during a challenging and exciting project. We owe part of our success to them because of their continuous support and encouragement.



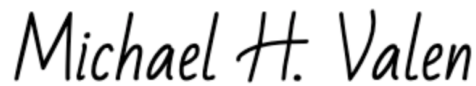
Ole Marius Andersen



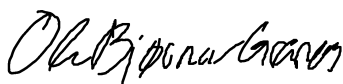
Niklas A Gustavsen



Markus Brødsjø



Michael Herland Valen



Ole Bjørnar Granås

Table of contents

1 Introduction	7
1.1 About Oxidane Venture	7
1.2 About the Project	8
2 Product	8
2.1 Product Requirements	8
2.2 Product demonstration	9
3 Project Management	9
3.1 Project Methodology	9
3.1.1 Roles	11
3.1.2 Backlogs	11
3.1.3 Sprints	12
3.1.4 Daily Scrum	12
3.2 Time Management	13
3.2.1 Time Estimation	13
3.2.2 Time Tracking	13
3.3 Communication	14
3.3.1 Discord	14
3.3.2 Microsoft Teams	14
3.3.3 Physical meetings	15
3.4 Administration	15
3.4.1 Azure DevOps	15
3.4.2 Google Drive & ClickUp	16
3.5 Risk Management	16
3.5.1 Risk Matrix	17
4 Analysis	19
4.1 Requirements	19
4.2 Design Criteria	20
4.2.1 Security	21
4.2.2 Reliability	22
4.2.3 Testability	22
4.2.4 Interoperability	22
4.3 Use Cases	23
5 Technology	24
5.1 Auth0	24
5.2 Platform Alternatives	25
5.3 Flutter & Dart	26
6 Architecture	27
6.1 Design Pattern	27
6.2 Deployment Diagram	28
7 Design	28
7.1 Design Principles	29

8 Quality	30
8.1 Definition of Quality	30
8.2 Quality Requirements	30
8.3 Assessment of Quality	31
8.4 Quality Assurance	31
8.4.1 Code Standard	31
8.4.2 Testing	32
8.4.3 Version Control	33
8.4.4 DevOps	33
8.4.5 Workshop	34
9 Sprints	35
9.1 Pre-Sprint (10.01-30.01)	35
9.2 Sprint 1 (31.01 - 20.2)	37
9.3 Sprint 2 (21.02 - 13.03)	38
9.4 Sprint 3 (14.03 - 03.04)	39
9.5 Sprint 4 (04.04 - 10.04)	39
9.6 Sprint 5 (11.04 - 24.04)	40
9.7 Sprint 6 (25.04 - 08.05)	41
9.8 Sprint 7 (09.5 - 15.05)	42
10 Reflection	42
10.1 Challenges	43
10.2 Previous Knowledge	44
10.3 Learning Outcome	44
10.4 Changes	45
10.4.1 TDD & Testing	45
10.4.2 Version Control Management	46
11 Group Evaluation	48
11.1 Individual Evaluation	48
References	51
Appendix	56
Appendix 1: Statement from Oxidane Venture	56
Appendix 2: Map of the Requirements	59
Appendix 3: Use Cases	60
Appendix 4: Definition of Done	62
Appendix 5: Best practice Flutter & Dart	64
Appendix 6: Git Cheatsheet	67

Figure list

Figure 1 - Project Characteristics	10
Figure 2 - Clockify example	13
Figure 3 - Risk matrix	17
Figure 4 - Design Criteria Table	21
Figure 5 - Use Case Marketplace	23
Figure 6 - Sequence Diagram Auth0	24
Figure 7 - Google Trends	26
Figure 8 - Deployment Diagram	28
Figure 9 - DevOps Flow	34
Figure 10 - TDD Benefit table	46

Part 1 - Project Overview

1 Introduction

This bachelor's project aimed to develop further and explore the knowledge and skills the group has acquired throughout the study. The project gave an authentic setting with opportunities to plan, analyze, and develop a product. This undertaking was the last challenge the group had to pass before achieving a degree in IT and Information Systems.

The undertaking consisted of developing a mobile application for an existing web solution. It included developing an understanding of requirements, pre-existing APIs, and design elements given by a product owner. The focus of the project was not the product but the process.

This report contains documentation of the undertaking structured into three parts; project overview, analysis, and process. The reader will be introduced to the client, product, and project management decisions in the project overview. A further analysis of the project requirements and choice of technologies based on them follows in part after. The last part of the report lays forth the group's process through sprints, followed by a reflection on the project.

1.1 About Oxidane Venture

Oxidane Venture is a venture company that invests in the intersection between water and technology. One of many things they can offer the entrepreneurs they invest in is knowledge of how to build the technology and software necessary for the ideas to become successful (Oxidane, n.d.). Digital readiness is, in other words, one of the main points of focus for Oxidane Venture. This is evident from the projects they involve themselves in. Examples are Intoto, Eco Trawl, and many others, including the project the group is currently developing a mobile application for, Fished. Intoto is about gathering and analyzing data about water levels, particularly in rivers, to prepare for climate change regarding draughts and floods (Intoto, n.d.). Eco trawl seeks to create more environmentally friendly technology for industrial fishing by using underwater drones (Eco Trawl, n.d.). Fished is about creating a digital marketplace for fish between sellers and buyers (Fished, n.d.).

1.2 About the Project

Fished is the daughter company of Reinhartsen and Oxidane Venture (Proff, n.d.). The company seeks to digitize and simplify the process of buying, selling, and auctioning fish. It connects users in a larger marketplace, giving them a better overview of conditions and prices. The feature-packed marketplace empowers the users by giving them enhanced opportunities to view, trade, and transport seafood (Fished, n.d.).

The group's task was to create a mobile application to support the current web solution. Finishing the application was not essential for the product owner, but rather that it tests the waters for Fished. This gave the group an excellent opportunity to explore and experiment in a low-risk environment with new technologies. These technologies included Auth0, Flutter/Dart, asynchronous programming, state management, Postman, and Azure DevOps. For a statement from the product owner see appendix 1.

2 Product

The following chapter describes the project requirements in light detail. After that, a product description will follow with a link to a product demonstration.

2.1 Product Requirements

Oxidane Venture had a few requirements for the development of the Fished mobile application. These requirements can be split into three categories; design, function, and technology. Design requirements were expressed in simple terms; use the assigned mockups and the pre-existing web application as examples in the design process. Function or functional requirements were expressed through the input and output of the web application and Fished's API documentation. Finally, technology had two rigid requirements; the application had to be a mobile application, Android, iOS, or both. Furthermore, it had to utilize the Auth0 platform for authentication. In communication with the API, retrieving and sending information would be impossible without Auth0 authentication. These requirements will be discussed further in part 2, the analysis part of this report.

In addition to the requirements above, Oxidane expressed some more flexible preferences. These were to use Azure DevOps as a version control platform and documentation hub and Microsoft Teams for communication. Additionally, Oxidane expressed interest in utilizing

cross-compilation technology in this project. This was due to their plans to have the mobile application on both platforms. For a map of the requirements see appendix 2.

2.2 Product demonstration

Before going into the next chapter about project management, let us present the current product. The product was programmed using the framework Flutter, and the programming language Dart. It consists of eight pages displaying the product's core functionality. These pages are:

- Auth0 login
- My Order (Homepage)
- Marketplace (Requests and offers)
- Order description
- Order creation (Buy and sell)
- Auction (Not yet implemented by the product owner)
- About page (User information)
- Menu page (Changing company and log out)

Click this link for a video demonstration: [Product demonstration](#).

3 Project Management

Project management is crucial for a good project process, and it is the glue that sticks the project together. It gives the group realistic plans, clear objectives, control, and better time estimation(Lucidchart, n.d). The group has utilized multiple tools to ensure structured and organized project management. These will be dealt with below, and they are structured into the following categories: project methodology, time management, communication, administration, and risk management.

3.1 Project Methodology

The contents of this chapter were in part retrieved from assignment one, in the concurrent subject IS-305, "Current IT-related Topics, Digitalization and Sustainability". When choosing project methodology, there were multiple things to consider, such as the project's documentation needs, liability to change, experience with technologies, time sensitivity, and more. In this project, the group weighed three methodologies against each other; Waterfall,

Kanban, and Scrum. Each of them has its use cases. Kanban and Scrum are examples of agile methodologies (Rehkopf, n.d.), while waterfall stands apart as a more traditional approach to project management (Radigan, n.d.). A summary of the discussion can be viewed in the table below.

Experience with technologies	Low	The group had no experience with Azure DevOps and mobile development previous to this project.
Liability to change	Medium	The web application was in an early stage, and there was a possibility that features could be added.
Documentation needs	High	Bachelor's report detailing the process of the project was to be written in addition to four assignments in an adjacent subject.
Time sensitivity	Low	While the project had a clear deadline, finishing the mobile application was not a priority for any stakeholders in this process.

Figure 1 - Project Characteristics

During some of the first meetings with the product owner, he said that their web solution was unfinished and could gain additional features in the near future. This meant that the requirements could change, meaning another iteration could be added to the project. Alternatively, it could force the project back to an earlier phase and back to the start phase in the worst case. This added some uncertainty to the project. Further uncertainty was added because the group had no experience with mobile development, Azure DevOps, and Auth0. The alternative of using Github instead of Azure DevOps was given, but the group reasoned that it would be an excellent opportunity to learn it. This uncertainty bound to the project rendered the project's scope too unclear to use the waterfall methodology. A more agile approach appeared to fit this project better; this is where Scrum and Kanban came into the picture. While there are plenty of other agile methodologies, only Scrum and Kanban were familiar enough for the group to consider them.

The bachelor project required documentation of the development process. Because of this fact, it was evident that the project could not merely consist of programming and documentation of code. It required documentation of events and decisions, as well as making plans where deadlines, presentations, and meetings are planned out. Kanban's flexible methodology was considered perfect for fixing issues as they occurred. However, in the group's opinion, its way of prioritizing and planning work was considered too sporadic for a

project like this. On the other hand, Scrum was thought to be the stronger of the two in this area. This is because the framework sets down concrete events, each with a dedicated purpose. These events were considered beneficial in ensuring a good process and creating room to document it. As a result of the discussion above, Scrum was chosen to be the project methodology for this project. Following this, a few choices remained on how to use the framework.

3.1.1 Roles

Any Scrum process consists of at least three roles (Schwaber & Sutherland, 2020); product owner, scrum team, and product owner. For product owner Tor Oskar Wilhelmsen was the natural pick, as he was the primary contact for the project. The Scrum team was the author of this report, and the Scrum Master was Markus Brødsjø. If the scrum master were ever absent, Ole Bjørnar Granås would take the lead.

3.1.2 Backlogs

The first choice regarding backlogs was what they should contain. Because the semester consisted of two subjects, and both were relevant to the same company, the group reasoned it would improve the overview if both subjects were included in the backlogs. In other words, all tasks related to everything in the project code, text, and documentation, were to be added to the backlog.

Three options were considered for keeping the product backlog; Trello, Google Sheets, ClickUp, and Azure DevOps. Trello was one of the first tools to be suggested by internal discussion. The group had experience with Trello previously. While its board is intuitive and straightforward, previous experience has highlighted one main issue concerning this project; the board's card interface quickly gets messy when adding a few dozen tasks. This project was not considered huge, but it was deemed large enough for Trello to become cluttered. Google Sheets was also considered because it allows for a great deal of customization. However, the last two alternatives, ClickUp and Azure DevOps, appeared more feature-rich from the get-go. They would let the team focus on the task at hand instead of spending time customizing a spreadsheet. In the end, the choice was between Azure DevOps and ClickUp. One week during the pre-sprint was spent testing these tools to decide which to use. Pros and cons were found with both, rendering the wish to learn how to use Azure DevOps better the deciding factor.

3.1.3 Sprints

Starting off each sprint, the group prepared a sprint backlog. Elements from the product backlog were broken down into smaller subtasks and then put into the sprint backlog. Before the end of every sprint, the group held a sprint review. In most of these, the group included the product owner and the group's mentor as these were both interested parties contributing with guidance, requirements, and opinions. These sprint reviews also served as a great opportunity to keep the interested parties updated on the project's progression. After sprint reviews, the group gathered for sprint retrospectives marking the end of the sprint. Here the team evaluated the sprint, discussed positive and negative experiences, improvements from previous sprints, and areas to improve for the next sprint. Sprint planning, sprint review, and sprint retrospect were all documented.

If tasks were not completed during a sprint, they were forwarded to the next sprint. While each sprint consisted of sprint goals, it was not deemed detrimental if tasks were not completed before the end of the sprint. This is mainly because of the unique situation the group found itself in, with process and learning outcomes being weighted above product.

The duration of sprints was not set before the sprint planning meetings. Although, its duration was often briefly discussed in the preceding sprint retrospective. The primary reason which governed the decision to avoid a set duration for every sprint in the project, was to experiment with how different sprint lengths and adding tasks with high unpredictability affected the iterations.

3.1.4 Daily Scrum

Every morning a Daily Scrum was arranged. It generally lasted twenty to thirty minutes. During these events, the Scrum Master took the lead and asked the individual team members three questions; what they had done the day before, what they planned to do today, and if they were expecting any challenges or hindrances. The group often caught itself being engaged and interested in the challenges people faced the day before, suggesting solutions, giving advice, or engaging in troubleshooting. This was the primary reason the event often took up more time than it was supposed to. However, the group thought it best not to curb the enthusiasm in most cases, except when the problem seemed too complex to solve quickly. The result was an environment focused on learning, improving, and solving problems.

3.2 Time Management

Time management can be seen as managing progression made on tasks and other activities related to the project. It is crucial to plan, schedule, monitor and control activities related to the project (Wrike, n.d.). Hence, the group used different techniques and tools to carefully manage the time spent throughout the project. This chapter's content is about how the group managed time estimation and organized the time-tracking.

3.2.1 Time Estimation

To accurately estimate time has been a significant challenge due to the unfamiliar tools and technologies used in the project. The group has never been involved in developing a mobile application, and therefore purely using the comparative estimation method proved difficult. Previous development projects consist mainly of web applications, so comparing the two might not provide the ideal time estimation. Therefore, it was decided that the group would use two estimation methods in an attempt to estimate with greater accuracy.

The finalized method for time estimation in the bachelor's project is a mix of planning poker and comparative estimation. Comparative estimation compares typical projects with similar past work (Indeed Editorial Team, 2021) while planning poker allows the group to average the agreed-upon estimation for the Sprint tasks. The reasoning behind the modified version was to answer the concern above while also creating a solution that enabled agile teamwork without adding extra administrative workload to the project.

3.2.2 Time Tracking

Clockify is a time-tracking tool that offers an organized platform for time-tracking, with features such as timekeeping, reporting, time management, and more (*Features*, n.d.).

Clockify allows the user to select a project, then choose a sub-task within a said project, followed by a task description. In conjunction with standardization rules, these features made the time spent on administration very brief, which gave more time for completing tasks.



Fri, Apr 29	Total: 06:42:32
1604 - Muligheden til å legge til utstyr, behandling og dato • Bachelor: Koding	\$ 14:30 - 17:12 02:42:32 ▶
Forelesning 304 • Bachelor: Forelesning	\$ 10:00 - 14:00 04:00:00 ▶

Figure 2 - Clockify example

Each member was responsible for tracking time and using the standardized setup mentioned above. This allowed time-tracking to be a part of the documentation, making it easier to do less overall administrative work and hold other members accountable for their tasks.

Another great feature was the ability to visualize the previously tracked time. It gave an insight into trends that occurred during the project; an example of this is that the week started strong but slowly trended downwards when approaching Thursday and Friday. The ability to gather these statistics gave a better insight into how to tackle unseen problems within the group.

3.3 Communication

Communication is essential to achieving good results in any project (Joubert, 2020). The group has communicated in both digital and physical contexts throughout this project. Discord was used during internal digital communication, utilizing its text, voice, file-sharing, and screen-sharing features. However, Microsoft Teams was used during external meetings, meaning communication including product owner and or the group's mentor. Additionally, some communication took place with the group's mentor through emails. Physical meetings mainly took place on campus, with one exception during the startup phase, where the group met up with the product owner to become familiar.

3.3.1 Discord

As mentioned, Discord was used for internal digital communication. The primary reasons for using Discord were the group's familiarity and daily use of it. Using this tool meant that getting hold of group members for comments, help, or sharing information was easier. Discord was also used to share links to documentation or sources and generate quick consensus outside of meetings. Features like pinning messages, splitting communication into separate channels depending on the topic, and screen-sharing were invaluable for making the communication structured and efficient throughout the project.

3.3.2 Microsoft Teams

Microsoft Teams was the product owner's preferred tool of communication. Oxidane uses Microsoft teams as their primary communication platform within their organization. The product owner suggested that the group should contact him over teams to get quick responses to questions. Therefore, the group and the product owner actively used a shared teams-chat

throughout the project. Status meetings and other external meetings were held over Teams for the product owner's convenience.

3.3.3 Physical meetings

Physical meetings on campus took place every week, with few exceptions throughout the semester. These meetings were primarily used to discuss, communicate, and plan out topics with high complexity. In the group's experience, physical meetings significantly contributed to creating essential discussions around the decision-making and planning process. The tradeoff was that the process took more time and could drain some of the members' social batteries. This tradeoff was, in most cases, considered worth it.

3.4 Administration

When it comes to administering a project, selecting the right tools could be essential for the project's progression. The tools for administration in this project were Azure DevOps as a project management tool, Google Drive for storing documentation, Clockify for time management, and ClickUp for documenting the day-to-day activities.

3.4.1 Azure DevOps

Azure DevOps is a developer service used for planning, developing, building, and deploying applications (*Azure DevOps*, 2022). It was Oxidane Venture's platform of choice for developing and collaborating, and therefore it was preferred that the group used the same platform. Azure DevOps gave the product owner unique insight into the project's progression and allowed ongoing feedback.

The platform offers a multitude of features, with the main ones used in the project being Overview, Boards, Repositories and Pipelines. Overview feature delivers functionality such as a Wiki, where the group has created guidelines for the project (*Azure DevOps Wiki*, 2022). These guidelines include cheat sheets, best practices, and other miscellaneous documentation

Boards feature delivered functionality such as Backlogs and Sprints (*Azure DevOps Boards*, 2022). The Backlog feature had some use in the early phases of the project as a planning tool but did not adapt well to the Scrum framework due to its limited functionality. The Sprint's feature offered more flexibility surrounding Sprint planning and execution, as well as adapting to the Scrum framework. In the later stages of the project, the Sprints feature ended up replacing the Backlog feature.

Repos acted as the main codebase for the mobile application during the project. With features such as Branches, Commits and Pull requests, it allowed for collaboration to build better code (*Azure DevOps Repos*, 2022).

The Pipeline feature delivers functionality such as Pipelines, Environments, and Library (*Azure DevOps Pipelines*, 2022). The group wanted to explore the functionality offered by the Pipeline feature more, but due to a lack of resources, it could not be prioritized. This feature will be explained later in the report.

3.4.2 Google Drive & ClickUp

Google Drive and ClickUp were the preferred services for document management and storage of miscellaneous files. Cloud-native apps enable the group to work seamlessly and efficiently in real-time, allowing multiple users to collaborate and cooperate on the same document.

Google Drive is a cloud storage service that offers the ability to share documents and files. In addition, Google Drive offers excellent flexibility due to being free and accessible in the browser (*Google Drive*, n.d.). ClickUp is a cloud-based collaboration and management tool that offers various features for communication, collaboration, task assignments and more (Evans, 2020).

Using separate platforms for organizing documents was a conscious choice because of ClickUp's poor performance while collaborating between multiple users simultaneously. Therefore, ClickUp was deemed more optimal for storing and organizing smaller documents, while Google Drive was more suitable for longer and more structured documents, where collaboration often occurred.

3.5 Risk Management

Risk management is defined as all measures and activities carried out to manage risk (Aven, 2015, s.4). The analysis process is a central part of risk assessment, and there are several ways of presenting a risk analysis (Aven, 2015, s.15). The group decided to create a risk matrix to identify and describe risks associated with the project, and the risks that could prevent the project's goal from being accomplished.

3.5.1 Risk Matrix

The risk matrix establishes a picture and an overview of possible risks, risk probability, consequences, and counteractive measures to deter or handle said risk (Aven, 2015, s.45-48). One of the product owner's suggestions was to update the risk matrix with possible risks for each new Sprint. These reevaluations would happen during sprint planning to see if there were any new elements to add. Furthermore, it was decided not to delete any of the risks from the previous sprint due to the chance that they might appear again.

The group experienced several event risks becoming a reality during the project. An example of this was the pandemic; it was highly likely that group members could be infected during the project. If group members were to become infected, it could result in delays and complicate cooperation. Therefore, this is something the group must take into account. However, the group overcame these issues by following the risk matrix and updating it for every sprint.

Risks	Sprint 1	Sprint 2	Sprint 3	Sprint 4	Sprint 5	Sprint 6	Sprint 7
Talking takes place instead of work when meeting in person	12	12	12	9	9	6	6
Group forgets a deadline	5	10	5	5	4	4	9
Group member could do permanent damage to repository	16	8	4	4	4	5	4
Group member does not get enough sleep	12	8	8	8	8	12	12
Group members could work less effective at home	1	8	8	8	8	3	3
Group member could become sick	8	6	9	9	9	9	5
Computer related does not work as intended	12	6	6	9	9	8	6
Too stressful for the individual group member	10	6	6	6	6	3	3
Group member could leak out secrets	8	4	4	4	4	4	4

Group member overslept	4	4	4	4	4	4	8
Late answer from both mentor and product owner	6	3	3	3	4	6	2
Work does not get documented	NaN	4	4	4	4	4	4
Implementation of code does not work as intended and uses a lot of time	NaN	16	16	16	8	6	4
Group member wastes time on irrelevant tasks	NaN	NaN	2	2	2	6	4
Waste time on administrative work	NaN	NaN	2	2	2	4	2
Fished system is not working correctly	NaN	NaN	NaN	10	4	4	2
Failing to finish all the tasks set up for the sprint	NaN	NaN	NaN	8	8	8	6
Group does not complete the the report in time	NaN	NaN	NaN	NaN	8	8	6
The group fails to deliver the report	NaN	NaN	NaN	NaN	5	5	12
The product fails to meet the product owner's expectations	NaN	NaN	NaN	NaN	4	4	2

Figure 3 - Risk matrix

Part 2 - Analysis, Technology & Architecture

4 Analysis

Previously, in chapter 2.1, the project requirements were briefly laid out. In this chapter, they will be explored and interpreted in-depth. First, the requirements will be presented and explained. Then a table with the prioritized design criteria will be presented, followed by an explanation of the prioritization. After that, follows a set of use case models, showing the internal logic of the application. Based on the requirements in this chapter, technologies and architecture were chosen for the project, these are presented in chapters 5 and 6 respectively.

4.1 Requirements

The six requirements set by Oxidane were:

- Create a native mobile application of the web application Fished.com
- Use Fished's development environment
- Use Fished's REST API
- Use predefined design elements
- Use Auth0
- Create unit-tests
- Allow users to change active company in the mobile application

Fished.com is a web application functioning as a marketplace for buying and selling Fish. The web application makes calls to a backend through a REST API, these calls can be viewed through the network tab in the inspect tool of a web browser, however in order to gain access to the application one first needs to be registered. The group registered test users which were admitted to Fished.com's development environment by Oxidane. In this environment, the group could interact with and test the application. This made it possible to develop an understanding of how it functioned, and how the mobile application should function. Based on these tests the use cases in chapter 4.3 were made.

In order for buy and sell orders to be available from both the web and mobile applications, they would have to use the same backend. This made using Fished's APIs a non-negotiable requirement. To request from and post to the API, a valid Auth0 token was required. This

meant that Auth0 authentication was required for the application's most basic functionality. Thus, the requirement of implementing Auth0 into the mobile application was also non-negotiable. Inside the web application's settings page the product owner noted that in the mobile application, the user should not be allowed to change any account information. Change of account information was reserved for the web application. However, the user should be able to change the active company, meaning the company the user was currently conducting for.

The product owner expressed that the predefined design elements, meaning colors, buttons, logos, and fonts should be used. These were supplied both in the form of high-fidelity mockups and the web solution itself. Some room was left for interpretation, as long as it could be argued for, as long as changes did not break familiarity or consistency. One of the concrete examples the product owner provided of things that could be changed was two graphs on the "My Orders" page. He did not consider either of these graphs to be essential whatsoever to the mobile application. Finally, the last requirement was creating unit tests. Although the product owner did not expect full test coverage, he expected that at the very least, some components and units should be tested. Preferably, tests were to cover all core functionality.

4.2 Design Criteria

In order to create a common understanding of the product requirements, prioritization of design criteria was performed. Design criteria and their definitions were retrieved from the book *Object-Oriented Analysis & Design* written by Mathiassen et al. (2018, p.180). These design criteria were used to make decisions for technology, architecture and prioritization in the project. The table below displays the criteria prioritization. Below the table, is a short description of the most important criteria.

Criteria:	Very important	Important	Less important	Irellevant	Easy to implement
Usability		x			
Security	x				x
Efficient			x		
Correct		x			x
Reliability	x				
Maintenance		x			x
Testability	x				
Flexible			x		x
Comprehensible		x			x
Reusable				x	
Portable			x		
Interoperability	x				

Figure 4 - Design Criteria Table

4.2.1 Security

The first criterion with high prioritization was security. It is concerned with who has access to read and write data in the system (Mathiassen et al., 2018, p. 180). It was considered easy to implement, primarily because the application was completely dependent on Auth0 for authorization. If the user failed to be authorized by Auth0, it would be impossible to retrieve any data of importance from Fished.com's APIs. Oxidane was responsible for the implementation of Auth0 in their backend, and therefore most security-related issues would be out of this project's scope. In other words, if Auth0 was successfully implemented, most of

the security-related issues would be considered solved. However, it was still important to test the output and functionality of unofficial packages if they were to be implemented, to make sure dubious code was not included in the app. There were two reasons this criterion was rated very important; first Auth0 was a requirement, and second, the application dealt with business-sensitive data and funds.

4.2.2 Reliability

The second criterion with high prioritization was reliability. In essence, it means that the system should provide the same expected output every time given identical input (Mathiassen et al., 2018, p. 180). Due to the system being a marketplace, it is paramount that user input is interpreted and sent correctly to the backend. If the system failed to treat the customers' information correctly, its reputation would be badly damaged. Additionally, the user should never have to be insecure whether the data they appear to be interacting with on-screen is the data being interacted with. For instance, if they order an amount of fish through the application, then the correct amount, type of fish, price, etc. should be displayed and processed. The mobile application would be unreleasable without this criterion in order.

4.2.3 Testability

The third criterion with high priority was testability. Testability ensures that a system functions as intended (Mathiassen et al., 2018, p.180). Testability was a clear requirement from the product owner, rendering it very important. In the group's opinion, regardless of testing being a requirement, the system's nature rendered testability a high priority. The system contains many operation-critical functions, and failures or inconsistencies would be detrimental to the system's integrity and user base. Testing among other things can help increase the system's reliability and maintainability (Myers et al., 2011, p. 6).

4.2.4 Interoperability

The fourth and final criterion with high priority is interoperability which refers to the system's ability to work together with other systems (Mathiassen et al., 2018, p.180). This criterion was essentially a requirement from the product owner expressed through having to use Fished's API and Auth0. If the application was unable to work together with Auth0 it would be unable to function. If it was unable to work together with the API or web application, frequent crashes, inconsistent actions, results, or faulty transactions could be expected, and once again the product would be unusable until made interoperable.

4.3 Use Cases

To illustrate the different functions of the mobile application, four use cases were created. A use case diagram “shows the relationships among actors and use cases within a system” (Ambler, 2013). It adds value to the project by easily explaining through a model how the system responds based on actors' behavior (HHS, 2013). The use case diagrams which were created do not deal with every step of the system in extensive detail. Instead, they provide an overview making the flow of events easily available, so the group has a common model for system events and actions based on user interaction. The Marketplace use case was the most complex of these use cases, and it is presented below, while the rest can be found in appendix 3. These use cases are presented below: “Login and Registration”, “Marketplace”, “My Orders”, and “Menu”.

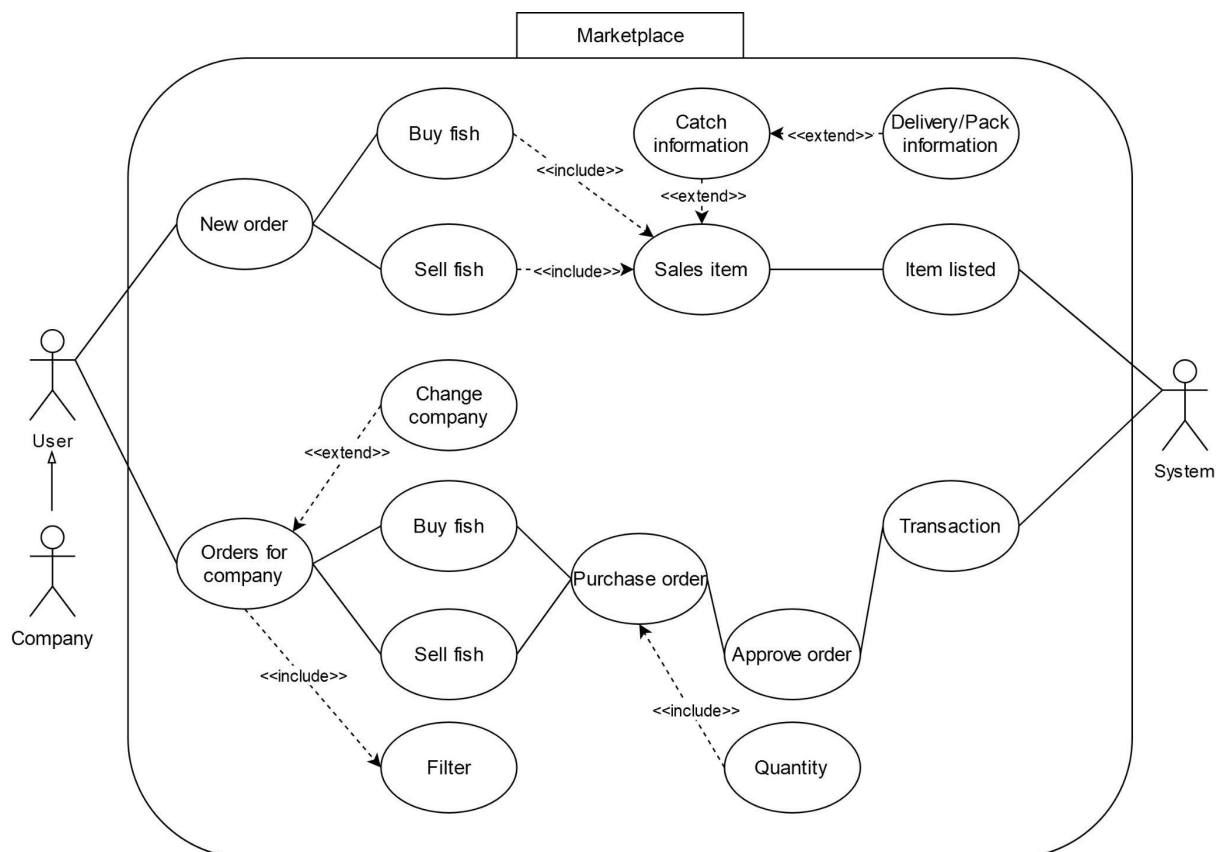


Figure 5 - Use Case Marketplace

5 Technology

The group spent numerous hours exploring, testing, and carefully considering the different technologies during the pre-sprint. In addition, after thorough discussions with the product owner, mentor and other professionals, the group landed on a decision. The chapter contains a short introduction to Auth0, a deliberation about platform alternatives, and further details about Flutter and Dart.

5.1 Auth0

One of the requirements from Oxidane Venture was Auth0 to authenticate users. It allowed the group to focus on other aspects of software development instead of using a lot of resources to build complex security solutions from scratch. Auth0 is an adaptable authentication and authorization platform that offers several different authentication methods, such as multi-factor authentication (MFA), single sign-on (SSO) and email/SMS authentication (Auth0, n.d). The deliberation about the design criteria for security can be read more about in 4.2.1.

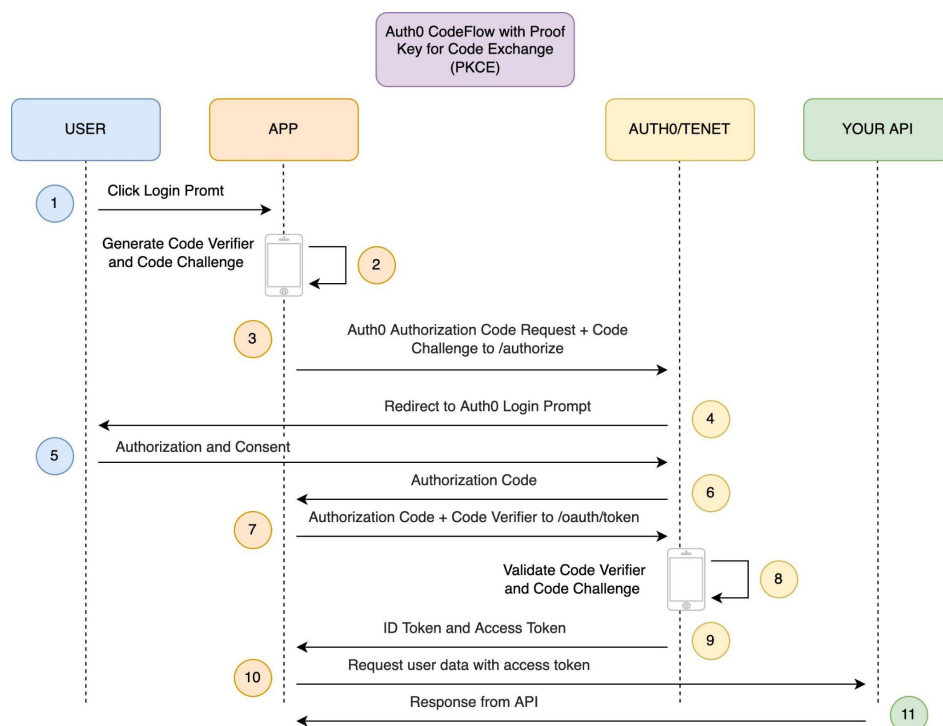


Figure 6 - Sequence Diagram Auth0

The group created a sequence diagram before implementing Auth0. This was done to understand better and visualize the Auth0 process; it also assisted in the implementation process.

5.2 Platform Alternatives

Choosing which technology to develop the mobile application was done during the pre-sprint while the group figured everything out. It landed between two categories, native, with languages such as Swift for iOS and Kotlin for Android or cross-compiling platforms such as Xamarin and Flutter. The group had limited knowledge about these categories and decided to ask other people with more knowledge and experience. The first discussion was with the product owner, and he emphasized doing a simple test of the alternatives and then coming to a conclusion of what the group thinks fits the project best. The other talks were with the mentor and a professional developer; both had concerns due to encountering problems with cross-compilation before, especially when it came to implementing specific features, working on a lower level of abstraction, and the toolkit had some issues that needed ironing out.

The simple test had some criteria when testing the languages and platforms. These were to check how easy it is to set up an application from the ground up, what the toolkit offers, and whether it feels familiar to other languages used in the past. The native languages were excluded relatively quickly due to the group's internal requirement of wanting the mobile application on both platforms. Besides native, both of the cross-compiling platforms were easy to use and set up because the group was already familiar with similar programming languages and paradigms. Another critical factor when deciding on a platform was the documentation and community surrounding it. Surprisingly, both had decent documentation and support from their respective platform, the only essential difference was their age.

Xamarin is a much more mature platform that has been ironed out because it came out in 2011 (Ridland, 2021) , while Flutter came out in 2017 (Flutter, n.d.). The most significant contributing factor in choosing Flutter over Xamarin is that MAUI will replace it at some point in 2022 (Ramel, 2021). Therefore, Flutter has been deemed a safer option in making a future-ready application.

Another thing to factor in is requirements from a technical standpoint. The analysis is discussed earlier in the report; thus, this part will not contain a deeper investigation. The app has to support mobile platforms such as iOS and Android. These platforms come with

external interface requirements for software and hardware. Mobile software and hardware requirements are interchangeable due to the operative system constraint when creating a mobile application for iOS and Android platforms. The hardware is dependent on what operative system it can support; dependent, each of the respective platforms, decides this. The baseline for the mobile application was determined by using pre-existing requirements from other mobile applications with close to the same feature set (Finn.no). Thus, the baseline requirement for the operative system is iOS 13 (Apple, n.d.) and Android 7 (Google, n.d.).

At last, the group decided to check Google Trends to see how popular each of the respective platforms or languages scored. As shown below, it is clear that both native languages, such as Kotlin and Swift, have been doing quite well throughout the years. On the cross-compilation side of things, it is another story, Xamarin is currently on a downward trend in the past years, but Flutter is flourishing on the top and is still increasing since its release (Google, n.d.).

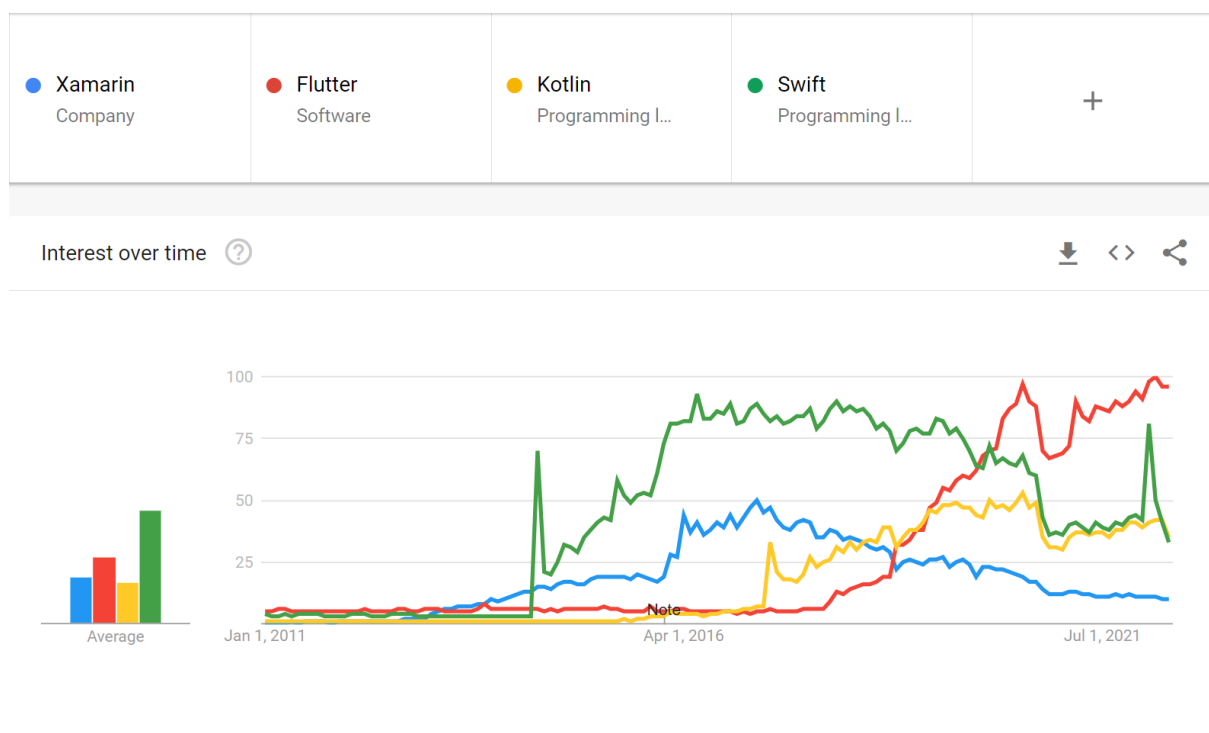


Figure 7 - Google Trends

5.3 Flutter & Dart

Flutter was the chosen framework for developing the mobile application. Flutter is an open-source and multi-platform framework based on the programming language Dart that Google made. One of the main features of Flutter is that it supports cross-compiling for both Android and iOS devices (Flutter, n.d.).

One of the many factors for choosing Flutter is the programming language the framework builds on. Dart as a programming language is similar to the syntax seen in Java, which is well known among the group members and simplifies learning as the group can recognize parts of the language. Flutter also provides a wide array of widgets with functionality such as styling, visual effects, interaction, and more. These widgets are easily modified and offer great flexibility when building applications for different phone types and operating systems. There are also plenty of packages developed by the Flutter dev team and other developers. These packages are quick and easy to implement and offer thorough documentation, making them easier to explore and work with. The state management offered by Flutter has also proven to be valuable for the project and has been used throughout the application. These were all crucial parts when choosing to use Flutter & Dart to develop the mobile application.

6 Architecture

This chapter describes the architectural design, and this process bases itself on understanding how a system should be organized and designing the overall structure based on that understanding. The output of this process is an architectural model that shows how a set of communicating components are organized in a system (Sommerville, 2011, p. 148). The following chapters describe the chosen design pattern based on design criteria and the deployment diagram.

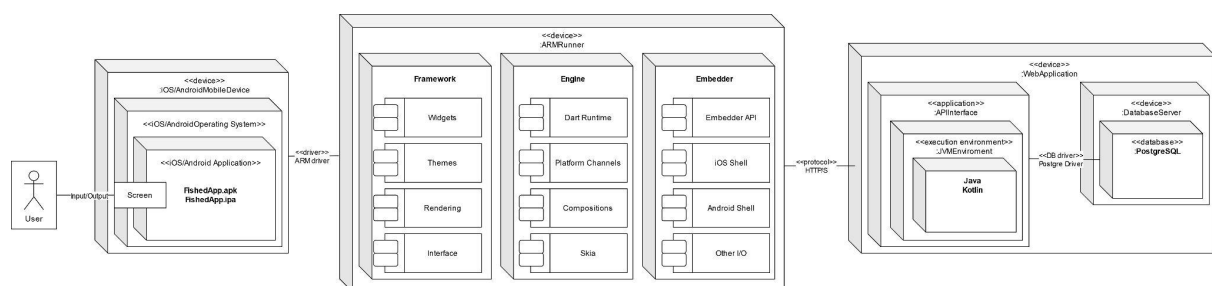
6.1 Design Pattern

In the decision-making process of choosing a design pattern, the group considered three alternatives. These alternatives were MVVM (Model-View-ViewModel), MVC (Model-View-Controller) and DDD (Domain Driven Design) pattern.

It was decided to conduct further research to strengthen the group's knowledge. The group compared the three design patterns against each other and weighted them based on impacted design criteria; maintenance, testability, and reusability. Additionally, the group added another subjective criterion which referred to how easy the group members thought the design pattern was to implement; ease of use. After careful consideration, the group decided to use the MVVM pattern for the project. MVVM is a software architectural pattern that helps to separate our graphical oriented code from business logic (Microsoft, 2021).

The maintenance design criteria are important because the Fished app is an extension of the web solution. Due to that reason alone, it should be easy and cost-effective to maintain for Fished in the future. The MVVM design pattern is the appropriate choice due to being highly modular and testable (O'Reilly, n.d.). Testable systems are easier to maintain because the effects of changes are highlighted when tests are run; this is also true when refactoring is done. Testable systems are easier to test. Furthermore, testability is a requirement from the product owner, thus benefiting from these benefits mentioned above.

6.2 Deployment Diagram



7 Design

good starting point, a decision was made to exclude some processes from the design phase such as sketching, wireframing and prototyping.

The mockups lacked a login page, and the web application's profile page contained functions that were not relevant to the mobile application. Therefore, a simple login page and profile page were created to fill this gap. In addition, the group thought it necessary to alter the "my orders" and "market" pages to better fit a mobile screen. Some other changes were made to the design of the application, but they are not worthy of note. Preceding the work done on designing and redesigning the mobile application, some work was put into understanding the functionality and context of the system. This work is described in chapter 4.1 about requirements.

7.1 Design Principles

User experience (UX) and user interface (UI) are all about the total experience when a user operates a digital product, service, or system (KnowIt, n.d.). In some cases, the group concluded that the mockups in Figma and web app pages do not fulfill the properties to be considered as good UX and UI. To achieve an application providing a good user experience and user interface, some of Benyon's 12 design principles have been applied. Benyon's principles are grouped into four main categories - learnability, effectiveness, safety, and accommodation (Benyon, 2019, p.116-117).

From Benyons four design categories, learnability is the primary category that is applied in the system. Learnability consists of design principles "concerned with access, ease of learning, and remembering" (Benyon, 2019, p.117). One of these design principles is visibility. This principle is about ensuring functions and that the system is currently visible to the user (Benyon, 2019, p.117). Adjusting the web app design into a significantly smaller mobile application format has been an aspect of creating visibility. The web design contains information, input fields, figures and pictures which were challenging to implement into a mobile format and still achieve high visibility.

As mentioned earlier, the group decided to take inspiration from the web application's design and improve it. Many of the symbols, colors and layouts were reused in the mobile application design. The idea behind this decision was to match the web design with the mobile application. Having a too big contrast between the mobile and web design could lead to an application where the user does not feel any familiarity and is misled regarding

functions. This decision is about ensuring high learnability and recognizability, which creates a good user experience making the system satisfying to utilize (Benyon, 2019, p.104-105).

8 Quality

Quality can be complicated to describe as it varies between products, services, individuals, and organizations. In some instances, quality can be described as performance to standards, and in others, it can be described as meeting the customer's needs (Reid & Sanders, 2012, p. 162-163). The definition of quality for this project has been defined through internal discussion and conversation with the product owner.

8.1 Definition of Quality

Defining the group's definition of quality has been necessary before planning any activities, or creating any quality requirements. Quality in respect to this project has been defined as the system's ability to satisfy the product owner's requirements for the system relating to the product and the process.

8.2 Quality Requirements

Securing quality has been ensured by planning activities to achieve a product that fulfills the quality requirements. Standards and routines have been defined within the group to raise quality; these standards have worked as a basis in terms of supplying the project with manuals, instructions and control actions. Examples of this are; the “definition of done” (DOD), code standards, and miscellaneous cheat sheets for development. These standards have been followed to the best of the group's abilities, and have contributed as a basis for the quality requirements.

The “definition of done” (DOD) is an agreed-upon set of terms that must be fulfilled before a task can be considered complete (Madan, 2019). In principle it is the official gate, separating things from being “in progress” to “done”.

After internal discussions, the “definition of done” has been split into three formats: one for the administrative process, one for the development process, and a dedicated guideline list for reports and assignments. Separating the guidelines of the “definition of done” was done out of necessity, as the requirements for the administrative, development and report-related tasks are different. For the Definition of Done see appendix 4.

8.3 Assessment of Quality

Assessment of quality can be complex without predefining what quality is. The “definition of done” mentioned above has played an essential part when approaching how to assess quality in administering, development, and report writing. Using the “definition of done” can assist in setting the standard for quality and ensuring that a task satisfies the quality requirements before being defined as done.

An example taken from the “definition of done” for development is that at least two group members must review any pull request before a merge can occur. Another example from the “definition of done” for report-related tasks is that every group member has to proofread and give feedback before a task can be defined as done.

Sprint reviews and retrospectives have also played a part in assessing quality. The Sprint review enabled the group to have discussions and obtain feedback from the product owner and mentor. Retrospectives allowed the group to reflect on the positive and negative sides of the sprint. These reflections gave the group a better insight into changes that had to be made to achieve a better product.

8.4 Quality Assurance

Quality assurance or QA for short can be described as any systematic process that helps determine if a product or a service meets the specified requirements (Gillis, 2019). This project implements QA activities to ensure that the product and its services meet the needs, expectations, and requirements. These activities are elaborated on in further detail in the following chapters.

8.4.1 Code Standard

A code standard can be described as guidelines set for a developer to follow when writing source code (*Code Standards*, n.d.). The internal Azure DevOps wiki previously mentioned contains a set of guidelines created for development in this project.

These guidelines include best practices, naming conventions, and more. Examples for these guidelines are: The use of descriptive names for variables and methods, avoiding writing more code than necessary, and following best practices for Flutter and Dart as closely as possible (see appendix 5).

By following these guidelines to the best of the group's abilities the aim was to create code with high quality, using a predefined baseline that assists in keeping the code maintainable and readable.

8.4.2 Testing

Testing is vital for any successful development life cycle to ensure that a system works as intended (IBM, n.d.). As mentioned in chapter 4, writing tests was a direct requirement from the product owner. Both the programming language Dart and creating practical unit tests were relatively new concepts to the group, hence the lack of priority of testing in the early sprints. However, because the application is a marketplace that handles personal data and real currency, this entails an additional responsibility upon the development team. There is a lower acceptance for significant errors, as it could lead to great consequences such as dissatisfied users and a loss of customers if something were to go wrong. Therefore, testing is crucial to verify and validate the application before an eventual release.

Testing during the project was primarily oriented around the exploratory, unit, and manual testing. As the general experience level surrounding testing was low within the group, something needed to be done. By conducting research and watching guides on YouTube, the group managed to build a greater understanding of how to test and why testing is essential.

The exploratory testing was primarily done using Postman and Swagger; these are software for developing and testing APIs. Exploratory testing helped the group map all the required APIs needed for the mobile application. This step was crucial when creating Models for the MVVM pattern, as fetching APIs data requires that the structure and data types be identical to Fished's Data Transfer Objects (DTOs).

The unit testing focused on testing services that fetched data from the APIs and mocked the data to ensure that the function's output was correct according to the mocked preset. Output testing has been critical to ensuring that the data displayed to the user is accurate. Flutter offers an extensive library of packages. The Mocktail package has been used as a template for unit testing, as it provides plenty of functionality and documentation while being easy to implement and use.

Manual testing was the go-to method to use as it gave a significant return in relation to the effort required. The testing was done using emulators for the respective platforms or installing the application on smartphones in conjunction with developer tools offered by

Flutter. Installing the mobile application was a more suitable solution when the focus was on testing the whole mobile application instead of just testing a particular function.

8.4.3 Version Control

Version control is the practice of tracking and managing changes to the codebase (Atlassian Bitbucket, n.d.). As mentioned earlier in the Azure DevOps chapter, it was the preferred platform for the product owner and was chosen as the version control platform for this project.

At the start of the project, the group had a workshop with the product owner. As the process has been the project's main focus, the group has been open to every suggestion made by the product owner. Using Rebase & fast-forward instead of the standard merging practice was one of the suggestions made by the product owner during this workshop. As the group was open to learning, this suggestion was implemented into the group's practices. The main issue with the implementation has been that Rebasing could potentially be more destructive in practice and thus require a more thoughtful approach. Hence, strict rules and guidelines were created to counteract any mistakes, some examples being the use of a modified version of GitFlow as a branching model, Git cheat sheets and more(see appendix 6). The topic surrounding choices of version control management, GitFlow and branch modeling will be further elaborated in chapter 10.4.2.

8.4.4 DevOps

DevOps is a mix of development (Dev) and operations (Ops) that combines cultural philosophies, practices and tools to create a superior process for software development (Azure, n.d.).

In the early stages of the project, there was an attempt to implement continuous integration (CI) by using the Azure DevOps Pipelines to automate the build process and run through unit tests. The implementation was successful in the earlier phases, but as the project progressed and other tasks had to be prioritized, it became harder to maintain. The lack of maintenance and CI not being a requirement from the product owner resulted in it being down-prioritized and semi-functional.

Using Azure DevOps for the project made it more accessible to share the install file from artifact storage and let the product owner be more up to date on the development. Artifact

storage lets developers share their code efficiently and manage all their packages from one place (Azure DevOps, 2022).

The figure below shows a simplified DevOps workflow used for the project. It considers the continuous delivery phase, but it was not relevant due to reasons stated earlier.

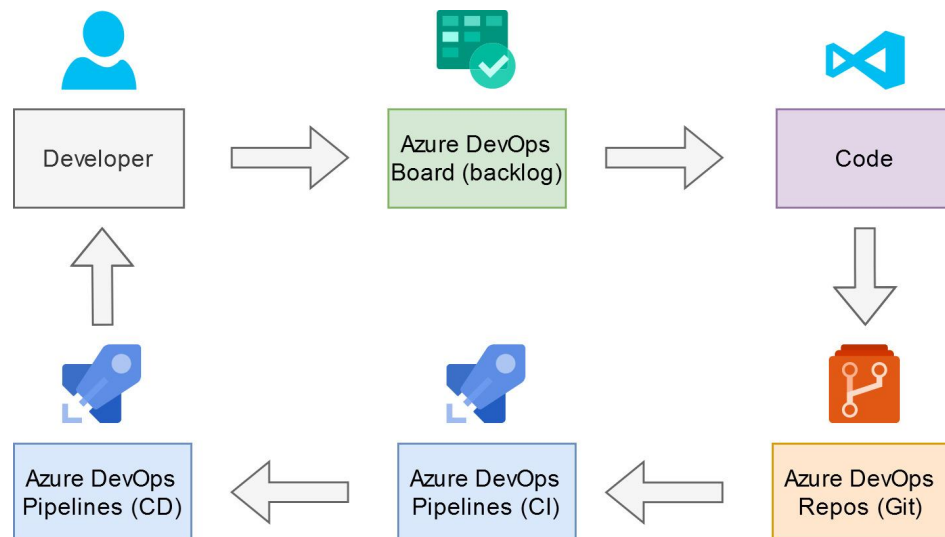


Figure 9 - DevOps Flow

8.4.5 Workshop

Workshops were put together whenever it was deemed necessary. These workshops enabled the group to remove uncertainties with a supportive team-building exercise. As a result, it enabled the group to participate in discussions, whether technical or analytical. It also acted as a great opportunity for the product owner to engage in the project and partake in the progression and development of the project.

Part 3 - Process

9 Sprints

The following chapter describes the sprints from the project in detail. Every sprint including the pre-sprint is accounted for, and important elements from each sprint are reflected upon.

9.1 Pre-Sprint (10.01-30.01)

Before the project could be set in motion, there had to be a foundation of planning and preparation. First, there had to be an understanding and documentation of the scope, the architecture, and the requirements. Next up were the major decisions surrounding project management and agreements regarding standard procedures, project goals, and choices in technologies and tools. These activities happened during the pre-sprint that spanned from the 10th of January until the 30th of January.

The pre-sprint started with several digital meetings between the group members, where the issues mentioned above were introduced and discussed. The group agreed upon the conditions of the group contract, and the group prepared questions for the semester's first meeting with the product owner.

The project's scope was dynamic, and the focus was primarily on the process and the argumentation of choices taken during the project. The task was to create a mobile application for the Fished platform based on the already existing web application. The group got access to the web applications development environment, the documentation of its APIs, and a design for the mobile application. It was also a requirement from the product owner to use Auth0 for user authorization, as a token generated by Auth0 was required to communicate with the APIs. Most of the application logic is done through communicating with their APIs, meaning the application would be dependent on using asynchronous functions. In addition to the project's core, the product owner wished to add ease-of-use functionality to the application. Examples of this functionality were face recognition and fingerprint recognition for a more effortless login experience.

The group was given the freedom to choose relevant technologies for the application development. However, the highlighted question was whether to use a cross-compilation

language framework such as Flutter or Xamarin or develop natively for iOS or Android through Swift or Kotlin.

Based on the information presented in the paragraph above, the group split up to learn which technology to use. Two members chose to learn about Flutter, two about Xamarin, and the last one looked into Swift. When weighing the technologies up against each other, the group focused on five key features; ease of use, syntax, performance, compilation time, and deployment on the phone. Ease of use should be interpreted as if the framework and toolkit are easy to learn and powerful. Syntax, in this case, means if the programming paradigm is familiar and whether functions are wordy. By performance, the group means that the compiled application has good test results and that the application's response time is well within the three main time limits as described by Nielsen (Nielsen, 1993, p. 135). The group was aware that the project's code and API response time would be the main sources of delay. However, having it further delayed by a potentially slow framework was considered a risk, particularly on slower hardware. While short compilation time is not essential to the application's quality, it is a luxury in projects requiring frequent recompiling. Finally, when looking at releasing it for the phone, the application needed to behave and look as similar as possible to the application on the emulator.

The group made the final choice regarding technologies at the end of the pre-sprint, where the choice ended on Flutter/Dart. As many aspects of the projects were unfamiliar to the group, it dedicated much time to explore new technologies, design patterns and various elements related to the project. While the effort dedicated was high, it quickly became apparent in later sprints that merely the group had scraped the surface, and there was plenty new to learn.

In the pre-sprint retrospect, the group highlighted some challenges. The first challenge was all the different choices that the group had to make on an uncertain basis. Of course, there was no way to know if the choices were good, but the time spent exploring during the pre-sprint certainly made some of the choices easier to make. An example of this is the choice of moving forward with Flutter and Dart, as this could have a significant impact on the project.

The second challenge was that the group experienced it as challenging to avoid chit-chatting while sitting together. To avoid breaking each other's concentration and to get more work done, the group decided to experiment with splitting into smaller units whenever meetings were over.

Other challenges included that the installation of Flutter was not cooperative with all the group members' computers and that the unfamiliarity with the project's contents caused some uncertainty. In the following sprint, the group started working on developing and setting up a proper project and report structure.

9.2 Sprint 1 (31.01 - 20.2)

Sprint one began with a physical meeting at Oxidane Venture's offices in Kristiansand. All group members and the product owner were present at the meeting, and it was used as sprint planning. During the planning, backlog items were prioritized and added to the sprint backlog with the assistance of the product owner. Based on internal discussions and dialogs with the product owner, the defined goals of sprint one were Auth0 implementation, developing front-end views, structuring DevOps, a time tracking policy, a report structure, a project structure and defining a definition of done policy for tasks. The group achieved every sprint goal except the Auth0 implementation during the three-week-long sprint.

In the development aspect, the group started with the project's structure. First, the group chose the design pattern MVVM (see chapter 6.1 for reasoning) and created the folder structure based on the MVVM design pattern. Then, in the sprint planning meeting, Auth0 and authentication was a task the group needed to finish before the app could use the APIs. As mentioned in the pre-sprint, the APIs require a token generated by Auth0, which sets the implementation of Auth0 as a high priority. The group researched how to implement Auth0 with a Flutter application, which proved to be more challenging than initially expected. The group used the Fished web application and used the inspect tool in the browser to monitor network activity on the web application. Using these tools gave insight into the different APIs called by the web application and helped gain insight into how the mobile application could potentially use the same APIs. As a result, the group learned more about how Auth0 and the Fished web application work.

The time tracking aspect of sprint one was documented using Microsoft Excel before switching to Clockify, which offered a more suitable solution for the project. Next up, the group spent a reasonable amount of time setting up the repository, the backlog, and the version control in Azure DevOps. The group had minimal experience with Azure DevOps, so setting up the different features was a challenge at first.

Sprint one was planned to last two weeks, but the group extended the sprint from two to three weeks due to a group management meeting with the product owner and the supervisor. During this meeting, the group showed the current state of the Auth0 implementation, and the product owner was optimistic about the progression. At this stage of the project, the workload per group member was set at 35 hours a week. However, it became apparent that 35 hours per week was unrealistic, and it was adjusted down to 30 hours during the sprint review. This adjustment was more realistic and aided the group in making better time estimations. During the sprint, the group also created CI for the app with Azure DevOps's tool Pipeline. CI/CD was not a requirement for the product owner, but there was some interest in learning about CI/CD. The group got a good start with the CI and pipelines through the sprint.

9.3 Sprint 2 (21.02 - 13.03)

As sprint two started, there was a desire to add more functionality to the application. Unfortunately, the implementation of Auth0 had not been successful, and the group had to contact the product owner for some assistance. After a meeting with the product owner and some more time spent developing, the group finally implemented Auth0 successfully. The successful implementation led to a significant boost of knowledge as the general understanding of how the product owner wanted the application to work was strengthened.

The group completed more of the backlog development tasks, leading to an application with front-end login pages, a my-page for the user, and routing between the application pages. There was also time for some experimenting with extracting information from the APIs, and to test the APIs, the group used Postman.

There was also a big focus on the project report and documenting processes during the sprint, with significant changes to the report structure. Changing the report structure was done because the group considered it to lack clarity. The solution was to have more describing chapter headings and split more of the content into several subchapters. It is also worth mentioning that the group members produced models during this sprint. The first model was a deployment diagram, and the second was a requirement specification.

In the sprint review, the team agreed that the Auth0 implementation had required too many resources. If similar problems occurred later in the project, it was determined to ask for help sooner. There was also a change in tools used for document storage. For document storage, the group initially used ClickUp, but after encountering problems with multiple users

collaborating on the same documents, the group decided to use Google Drive. ClickUp still stored all the documents related to scrum activities, such as daily scrums, reviews and retrospectives, while Google Drive stored large and structured documents.

9.4 Sprint 3 (14.03 - 03.04)

As sprint three started, the focus was on application development and quality assurance. Four out of five members were assigned to develop the application during this sprint. The tasks for development in this sprint were to give the user the ability to view orders, create a homepage, create a new buy-order, and the ability to change their company. As the functionality slowly increased, it was a fitting moment to start implementing various code tests, which is also an important activity for quality assurance. Manual testing was the primary way of testing until this point, but the group wanted to implement more automated testing. Finally, the focus shifted towards unit testing, so the group tried to create unit tests for the functioning functionality. Unit testing is something the group has barely done throughout the study, so it was pretty hard to start.

Therefore, conducting research and writing good tests was characterized by trial and error and has shown to be time-consuming and challenging. As a lot was happening in the project, the group figured the codebase needed refactoring, giving the entire group a prime opportunity to review the code together and gain a common understanding. Refactoring the codebase also made it easier to read and more manageable for future development. Midway through the sprint, the entire Fished development environment had issues, resulting in all the APIs being down for a while. Unfortunately, this was an unforeseen issue, and the group needed to rethink what tasks to do without access to the APIs. The result was an updated risk matrix, where the group added technical difficulties from parties over which the group had no control.

9.5 Sprint 4 (04.04 - 10.04)

The goal of this sprint was to complete the remaining tasks from the previous sprint. However, as Easter was coming up, the group decided to make this sprint short and focused on finishing the tasks remaining from the last sprint and a few others added to the sprint backlog.

During previous sprint reviews, the group members found it hard to calculate the workload when deciding how many tasks a sprint backlog should contain. Calculating the workload was hard primarily because the group experienced more time-consuming tasks than expected, especially development-related tasks. Therefore, the group set the sprint duration to one week to be more productive.

The group completed the homepage, created methods to get the current company, and access management during the sprint. These tasks were almost finished in the previous sprint but needed a little more time to be completed. The next challenge to overcome was the functionality surrounding the creation of a new buy order, and the group believed it to be challenging to finish before the sprint ended. During sprint review, the group agreed that a one-week sprint was a little short, but the level of productivity was higher than usual and stable for the entire sprint.

9.6 Sprint 5 (11.04 - 24.04)

The sprint started on Monday of the Easter week and was planned to be a two-week sprint, as three weeks felt a little long and one week was a little short. The group agreed to continue working Monday to Wednesday and then take a vacation for the remaining days of the easter. The holiday was necessary for the group to ensure motivation and productivity moving forward.

When starting the sprint, the group updated the risk matrix. The risk: "Implementation of code does not work as intended and uses much time" was updated from a 16 to an 8. This risk decreased because the group was more familiar with the development and the project, so it spent less time finding information. The other reduced risk was "Fished systems do not work correctly," which went from a 10 to a 4. The group has dealt with this problem earlier and knows how to handle it if Fished's systems are not working.

Even though the focus was on the report, there was still an incentive to finish developing the Fished marketplace and complete the development tasks. The marketplace shows all types of relevant orders to the user's chosen company and gives the possibility to make new buy or sell orders for the market.

Creating new order tasks required large amounts of time and resources. Therefore, the group did a lot of manual testing to determine how much information was needed to successfully

create a new order using the API. The testing started with mock data until the developers successfully made the order and the API was satisfied with the data.

During the sprint, the group's Pipelines encountered an error when building for Android. iOS has not been an issue during the project and has been successfully built at every iteration. There was an attempt to fix the problem, but as time became scarce, the group decided to down-prioritize the issue as it was not of great value to the product. The group worked on tasks related to the report. These focused on the technology used during the project, how the group conducted testing, and documentation for every finished sprint iteration. In the sprint 5 review, there were still some unfinished tasks. The remaining tasks were mainly due to inaccurate time estimations. Even though the time estimations were not entirely accurate, the group felt more comfortable estimating time, and for every sprint, the time estimations were more accurate.

9.7 Sprint 6 (25.04 - 08.05)

The sprint started with a change in weekly capacity, which was reduced from 30 hours a week to 26 hours a week. Based on the previous sprints and tracked time, the group reflected that the high capacity often resulted in little to no breathing room when tasks took longer than expected. In addition, the group rarely managed to fill the set capacity, which reduced motivation and the group members felt they worked hard every week but never reached the set capacity.

In this sprint, the group successfully created orders by sending mock data to the API. Next, the group shifted the focus to using user input to create a new order. As orders were created using mock data, the group knew what data the API required and what was optional. The new order object consists of 40 different values and multiple objects. The group supplied this object with mock data and user input. The data from the user input consisted of fish type, treatment, size, price, currency, and gear used to catch/raise the fish, and the mock data used consisted of various static Strings and or null values. This shortcut was a conscious shortcut taken by the group to finish the functionality, save time, and showcase to the product owner how it would work. If there were more time to finish this functionality, the group would make a proper solution for the shortcut taken.

The group conducted no user testing in the project for multiple reasons. Firstly, in the group's opinion, the application never reached a state where it differed from the web application to a

level where user testing would bring value. Second, the product owner also stated that there are no solutions for the user to reference and that user testing might create more noise than value for the project. As the project was coming to an end, the group invited the product owner to review the project and the product. The group and the product owner reflected upon the project during the discussion. Of course, the group could have done some things better, but overall the product owner appeared pleased with the process and the product.

9.8 Sprint 7 (09.5 - 15.05)

This sprint was the last sprint of the project, and at the start of the sprint, there was only one week left. The focus shifted heavily towards the report, and the number of tasks related to development was not many. The project's development phase did not reach the level that the group was working towards, and this was due to many unforeseen issues early in the project that drained many resources.

During this sprint, the group created a demonstration video of the product on a physical smartphone. As mentioned, the focus was mainly on the report and in regards to the report, the group created an excel sheet. This sheet contained all the chapters of the report and their status. The reasoning behind using an excel sheet over Azure DevOps is that the group had some bad experiences with using Azure DevOps to organize report-related tasks in this fashion. The group had a meeting with the supervisor to review the report and discuss its content. The meeting was deemed necessary, and the supervisor gave good feedback and input, so the group knew what to prioritize before the deadline. The feedback was related to the report's structure, flow and the importance of justifying and reflecting upon the choices made throughout the project. As the delivery date is at the end of the sprint, completing all the tasks related to the report is crucial. The development aspect of the project will continue beyond the sprint and till the date of the exam.

10 Reflection

Reflection is an essential aspect of learning. It can help with further developing your skills and increasing their effectiveness (The Open University, n.d.). During the course of this project, reflection has been an important part. It has resulted in increased learning and has been a helpful tool in optimizing different processes and increasing effectiveness.

10.1 Challenges

The start of this project had a steep learning curve, and it offered a wide variety of challenges. It was a big challenge to develop good routines during the pre-sprint as there were many uncertainties surrounding the project. Most of this time was dedicated to learning different alternatives to choosing a programming language, a design pattern, and a framework. When exploring these different technologies, there were many challenges related to installing and learning as they were new to the majority of the group.

Time estimation was also a big challenge during the pre-sprint, as time was the only set factor for the project. The usage of time-tracking tools was lackluster in the starting phase of the project, which made it hard to keep track of the time spent on different tasks. The utilization of Clockify helped the group track time in a structured manner, which made it more accessible and practical to manage.

As the project had many uncertainties, planning early on was essential to give the project a strong starting point. The earliest planning phase started with discussions surrounding the functionality deemed necessary for the mobile application. The discussion progressed forward as functionality upon functionality was added to the list. Azure DevOps boards were utilized to make a structured list of the planned functionality, which became a problem as there was too much functionality to implement compared to the time available. A lot of the planned functionality was not proposed nor required by the product owner, and using this type of approach resulted in hampering the project's progression.

After the early planning phase and choosing technologies, the next step was to explore the development aspect of the project. The first major challenge regarding development was implementing the third-party authorization platform delivered by Auth0. To access Fished's pre-configured APIs required a key generated by Auth0, and configuring Auth0 to generate the correct key was a more significant challenge than initially expected. The early implementation of Auth0 did not issue the correct key, thus access to the APIs was not granted. The implementation of Auth0 was an issue throughout Sprint 1, but it was successfully implemented in Sprint 2. Through discussion with the product owner, some tweaks to the code and some tweaks to the Fished APIs. Due to the severity of this issue and its impact on the project, the group would have benefited heavily from just asking the product owner at an earlier stage. This was reflected upon later in the project, and it required a lot of

unnecessary time and effort. After this challenge was solved, the general understanding of how the product owner wanted the system to function was strengthened.

As the project progressed there were some problems with the planned lengths of the Sprints, and the length of the planned tasks. On multiple occasions, planned tasks spanned more than one sprint, which made branch merging and time estimation more complicated than they needed to be. Following the discussion surrounding the length of the tasks and the Sprints, the decision was made to test different Sprint lengths to find what suits the project best while also trying to narrow the time required per task down to a workday or less.

During the duration of this project, the group has faced countless challenges of varying difficulty. Solving these challenges has given the group an increased knowledge and experience within the group. It has been a valuable lesson that can be used in future projects where similar challenges might occur. Discussions, reflection, and close contact with the product owner have been essential in solving a majority of issues faced in this project.

10.2 Previous Knowledge

Through the bachelor project, the group has utilized the knowledge acquired through the various bachelor's degree courses. The overall impression is that the group has experienced previous knowledge as very relevant to the project and highly transferable. Although the knowledge from most of the subjects in the bachelor's degree has contributed to the project, the group will especially highlight the courses within IS-310 project accomplishment, IS-202 programming project, IS-200 system analysis & system development, IS-104 interaction design.

10.3 Learning Outcome

The bachelor project has led to great learning benefits and raised the competence within the group. The group has raised their experience in project management, cooperation, personal development, and technical skills throughout the project.

The members have developed their first mobile application, contributing to technical learning outcomes as a new programming language, framework, and tools have been used during the project. The main technical learning outcome has been to use Flutter, Dart, and MVVM to develop an application. The group's security tool in the application is Auth0, and

implementing this into a mobile application has been a new experience for the group. This has given the group valuable knowledge that may prove helpful after the study.

In order to work efficiently and carry out the project in a good way, a good collaboration has been crucial. To be available above each other and ensure a good communication flow, the group created a discord channel intended for group members and a Microsoft Teams room/chat to keep in touch with the product owner. These channels have been shown to increase efficiency in both work and communication. There was a low threshold to ask for help and share experiences through these channels, which improved the project throughout. This has benefited from the good collaboration and maintaining active communication within the group and company.

All group members have worked together on earlier projects and therefore knew some of each other's strengths and weaknesses. To maximize the learning outcome, each group member has been given the opportunity to participate in both technical and report-based tasks. The ability to continuously update each other on what has been done, how it was done, and why it was done has been essential to share knowledge and raise the learning outcome within the group.

10.4 Changes

Even though the project was deemed a success by everyone in the group, there was still some room for improvement. This chapter will go through a deliberation about how the project would have been better with TDD, Testing and version control management.

10.4.1 TDD & Testing

Test-driven development is a form of development that ensures that all production code is written in response to a test case (Martin, n.d.). In hindsight, developing using a TDD mindset would help a lot with problems the group faced during the development. These problems range from accessible development to production halt due to third-party systems not working as intended. The examples below show use cases where and how the group would have benefited from TDD.

Auth0	Accessibility, this one was a major gripe in the production due to always having to write both username and password every time a cold reboot or error was triggered. With TDD, it would allow the group to modularize the authentication and authorization by setting up pre-existing values (mocking the data). Thus, skip the login with every hotfix or error that would occur. In addition, it would also give the group a secure flow of production due to not being highly dependent on Auth0 systems for developing the rest of the application.
Fished	As mentioned in the Auth0 example, this would also benefit a secure production flow. Unfortunately, the mobile application development was co-dependant that the Fished system was up and running, making it high risk. An example of this is authentication APIs not working as intended. The group used a lot of hours trying to figure out the problem without any luck. On the bright side, it only happened once during the project but would have high ramifications in the real-world scenario.
API	The group would benefit from testing the response from the APIs at the start of the project instead of blindly following the documentation. Using a program such as Postman would save the group a lot of hours spent debugging code and refactoring models.

Figure 10 - TDD Benefit table

Another thing the group could have applied while writing tests is using the vital testing principles described in the book “The Art of Software Testing”. In conjunction with the product owner’s rule about not testing data objects, these principles would have given a clearer picture of unit testing. To further add to this point, the author mentions that these principles are often overlooked even though they are apparent (Myers et al., 2011, p. 12-13).

10.4.2 Version Control Management

The group had a meeting with the product owner to go over the basics of version control and testing early stages of the project. During that meeting, the product owner showed a new method of approaching version control instead of the standard merging method learned at the university. The new approach was rebasing and squashing commits to make the commit tree straight and more organized. The group favored this new method because it was recommended by the product owner and made it possible to use the commit tree as part of the documentation.

After experimenting with the new method, it was discovered to be more destructive than anticipated because it let the developers rewrite and delete their commit messages. To circumvent that happening, the group decided to go for a safer and more familiar VCM

option that resembles a modified version of Gitflow. The group's modified version uses a main, dev, and feature branch to ensure that code can be recovered after a mistake. In addition, the branches had documented rules for each of the branching stages, such as; master would contain a working version of the sprint production, dev would have been the primary target for rebasing, and feature would work as a development branch.

Gitflow worked flawlessly in the early stages of development during the uncertainty, because it gave the security for the group to experiment. Most of the mistakes could be reversed and would not contribute to a mental overhang on group members due to lack of experience. However, the version control management had its flaws, and it was apparent that it required extra bureaucracy even when the project only had two active developers. Thus, the reason for this chapter discussing Trunk-based development. Trunk-based development is a set of practices for version control management where developers focus on small merges and frequent updates to the core branch. It streamlines merging and integration phases, helps achieve CI/CD and increases the rapid software delivery. These factors could help the group increase organizational performance if done right (Zettler, n.d.).

11 Group Evaluation

The group is pleased with the communication throughout this project. There were a lot of new technologies, and all group members took the challenge with a smile. The members have worked together before and know each other well. This gave plenty of room for discussion and questions. The group was familiar with its dynamics, and it became better at building on each other's strengths and weaknesses throughout the course of the last year.

Creating a mobile application was new for all group members. Moreover, the group did not know what the project should look like, so there were challenges. The biggest of which was to estimate the time for tasks and the project as a whole. Especially in the start when knowledge was lacking and the scope of the project and tasks was uncertain. However, this project was a good experience, and the group gained lots of new knowledge, and members got to develop their individual skills.

Because of the lack of knowledge and experience with creating a mobile application, it was reasoned that continuously making short-term plans and iterating on a product was the best idea to reach the project goals. Scrum was instrumental in this endeavor; it made it easy to keep the progression up, and to create events which fostered and reinforced communication. The sprint reviews were particularly valuable because they created a great opportunity to present the current progress and get feedback and guidance from both the product owner and the group's mentor.

11.1 Individual Evaluation

Ole Marius Andersen

During this project, my primary work area has been in development, but I have worked with a wide variety of tasks ranging from programming and modeling, to assisting in report writing and editing. This has provided me with a wide variety of learning outcomes and challenged the skills I have built during my study.

I have gained relevant knowledge regarding development on a general level and utilizing Azure DevOps and other tools for project management and implementation. Working with a group of like-minded people over a more extended period has provided me with valuable knowledge that will be relevant to take with me in my work life. The project has also shown

the importance of a good working environment and how important it is to have knowledgeable people around you.

Markus Brødsjø

Under this project I primarily have the responsibility for the technical parts. In this project, I operated a role as Scrum Master, and the role included supporting the other group members. I have taken the lead in the daily scrum and essential meetings within the group and with the product owner and supervisor to increase the progression through the project.

The technical part contains the implementation and planning of the architecture and code. Since none of the group members do have experience with a mobile application, the learning curve was steep. The framework Flutter was a little confusing at the start, but after a month of coding, it was easier to understand the framework and how it works. One requirement for the product owner was to use Auth0 for authentication to use the Fished API's. I got a good overview of how third-party authentication works and will take the experience further in my career.

Ole Bjørnar Granås

Throughout the project I contributed to a high degree with analyzing and editing documents relevant to the bachelor's project. When group members asked for feedback and advice on report related stuff, I was one of the primary sources. I also relieved the group by taking the primary responsibility for the concurrent subject, so they could put additional effort into the bachelor's subject. Lastly, I have participated in parts of the coding and done some code reviewing. I have made many valuable experiences when it comes to project management and development. When it comes to report writing I've improved my skills significantly, and I've also become better at balancing critique with praise when giving feedback,

Niklas A Gustavsen

During the bachelor project, I have been working with a wide variety of tasks. In terms of developing, my main tasks at the beginning of the project were front-end development in the framework Flutter. This task gave me an understanding of a new programming language and framework. My competence was also raised through some pair programming and working agile through a DevOps environment. Further, documenting the process and writing report content has been my main responsibility throughout the project. Tasks that have been included in this work are the creation of mockups for design as well as making explanatory models to include in the project report.

During the project, I have contributed with my strengths and developed my weaknesses for the better. I have both learned from my group members and taught knowledge in the areas where I have been able to contribute. If I am going to highlight something I especially want to bring with me further from the bachelor project it is the project collaboration and the project execution of an mobile application. I am convinced that this experience will benefit me in later projects and that it will contribute to surpassing challenges that may arise.

Michael Herland Valen

At the beginning of the project, I contributed to the initial planning and framework while also experimenting with different roles. As the project matured, I took more of a quality assurance and DevOps developer role in conjunction with writing the report and creating models. That role suited me better because it enabled me to work on tasks I would like to work on during my career.

I have garnered valuable experience in projects by working on my strengths and weaknesses. I am proud of the product and process that we achieved while also glad to witness the growth of my peers as our development process matured.

References

- Ambler, S. (2013, April 19). *Use Case Diagramming Guidelines*. Agile Modeling. Retrieved May 15, 2022, from <http://agilemodeling.com/style/useCaseDiagram.htm>
- Ambler, S. W. (2004). *The object primer*. Cambridge University Press.
- Apple. (n.d.). *FINN.no on the App Store*. App Store. Retrieved May 16, 2022, from <https://apps.apple.com/no/app/finn-no/id526541908>
- Atlassian Bitbucket. (n.d.). *What is version control*. Atlassian. Retrieved May 15, 2022, from <https://www.atlassian.com/git/tutorials/what-is-version-control>
- Auth0. (n.d.). *Auth0 Overview*. Auth0. Retrieved May 15, 2022, from <https://auth0.com/docs/get-started/auth0-overview>
- Aven, T. (2015). *Risk Analysis*. Wiley.
- Azure. (n.d.). *Hva er DevOps? DevOps forklart*. Microsoft Azure. Retrieved May 15, 2022, from <https://azure.microsoft.com/nb-no/overview/what-is-devops/#devops-overview>
- Azure DevOps. (2022, February 9). Microsoft Docs. Retrieved May 14, 2022, from <https://docs.microsoft.com/en-us/azure/devops/user-guide/what-is-azure-devops?view=azure-devops>
- Azure DevOps. (2022, May 9). *Azure Artifacts overview - Azure Artifacts*. Microsoft Docs. Retrieved May 15, 2022, from <https://docs.microsoft.com/en-us/azure/devops/artifacts/start-using-azure-artifacts?view=azure-devops>
- Azure DevOps Boards. (2022, March 7). Microsoft Docs. Retrieved May 14, 2022, from <https://docs.microsoft.com/en-us/azure/devops/boards/get-started/what-is-azure-boards?view=azure-devops>

Azure DevOps Pipelines. (2022, April 1). Microsoft Docs. Retrieved May 14, 2022, from <https://docs.microsoft.com/en-us/azure/devops/pipelines/get-started/what-is-azure-pipelines?view=azure-devops>

Azure DevOps Repos. (2022, February 10). Microsoft Docs. Retrieved May 14, 2022, from <https://docs.microsoft.com/en-us/azure/devops/repos/get-started/what-is-repos?view=azure-devops>

Azure DevOps Wiki. (2022, May 6). Microsoft Docs. Retrieved May 14, 2022, from <https://docs.microsoft.com/en-us/azure/devops/project/wiki/wiki-create-repo?view=azure-devops&tabs=browser>

Benyon, D. (2019). *Designing User Experience: A Guide to HCI, UX and Interaction Design*. Pearson Education Limited.

Code standards. (n.d.). University of St Andrews. Retrieved May 15, 2022, from <https://www.st-andrews.ac.uk/digital-standards/code-standards/>

Eco Trawl. (n.d.). Eco Trawl. Retrieved May 13, 2022, from <https://ecotrawl.no/>

Evans, Z. (2020, November 6). *Why Should You Use ClickUp?* ClickUp. Retrieved May 14, 2022, from <https://clickup.com/blog/why-clickup/>

Features. (n.d.). Clockify. Retrieved May 14, 2022, from <https://clockify.me/feature-list>

Fished. (n.d.). Fished. Retrieved May 13, 2022, from <https://fished.com/>

Flutter. (n.d.). *FAQ | Flutter*. Flutter documentation. Retrieved May 15, 2022, from <https://docs.flutter.dev/resources/faq>

Gillis, A. S. (2019, 07). *What is Quality Assurance?* TechTarget. Retrieved May 15, 2022, from <https://www.techtarget.com/searchsoftwarequality/definition/quality-assurance>

Google. (n.d.). *FINN.no – Apper på Google Play*. Google Play. Retrieved May 16, 2022, from <https://play.google.com/store/apps/details?id=no.finn.android&hl=no&gl=US>

Google. (n.d.). *Google Trends*. Google Trends. Retrieved May 16, 2022, from <https://trends.google.com>

Google Drive. (n.d.). Google. Retrieved May 14, 2022, from <https://www.google.com/drive/>

IBM. (n.d.). *What is Software Testing and How Does it Work?* IBM. Retrieved May 15, 2022, from <https://www.ibm.com/topics/software-testing>

Indeed Editorial Team. (2021, September 27). *8 Types of Estimation Techniques for Projects*. Indeed. Retrieved May 14, 2022, from <https://www.indeed.com/career-advice/career-development/types-of-estimation>

Intoto. (n.d.). Vann som vær - Intoto. Retrieved May 13, 2022, from <https://intoto.io/>

Joubert, S. (2020, March 31). *The Critical Role of Communication in Project Management*. Northeastern University. Retrieved May 15, 2022, from <https://www.northeastern.edu/graduate/blog/communication-in-project-management/>

KnowIt. (n.d.). *UX og UI-design*. Knowit. Retrieved May 16, 2022, from <https://www.knowit.no/tjenester/experience/design-og-merkevare/ux/>

Lucidchart. (n.d.). *Why Is Project Management Important?* Lucidchart. Retrieved May 16, 2022, from <https://www.lucidchart.com/blog/why-is-project-management-important>

Madan, S. (2019, December 16). *DONE Understanding Of The Definition Of "Done"*. Scrum.org. Retrieved May 15, 2022, from <https://www.scrum.org/resources/blog/done-understanding-definition-done>

Martin, R. C. (n.d.). *The Three Rules Of TDD*. ButUncleBob. Retrieved May 15, 2022, from <http://butunclebob.com/ArticleS.UncleBob.TheThreeRulesOfTdd>

Mathiassen, L., Munk-Madsen, A., Nielsen, P. A., & Stage, J. (2018). *Object-oriented Analysis & Design*. Metodica.

Microsoft. (2021, July 8). *The Model-View-ViewModel Pattern - Xamarin*. Microsoft Docs. Retrieved May 16, 2022, from

<https://docs.microsoft.com/en-us/xamarin/xamarin-forms/enterprise-application-patterns/mvvm>

Myers, G. J., Badgett, T., & Sandler, C. (2011). *The Art of Software Testing*. Wiley.

Nielsen, J. (1993). *Usability Engineering* (1st Edition ed.). Morgan Kaufmann.

The Open University. (n.d.). *Self reflection*. The Open University. Retrieved May 16, 2022, from <https://www.open.ac.uk/choose/unison/develop/my-skills/self-reflection>

O'Reilly. (n.d.). *Pros and Cons about MVVM*. O'Reilly. Retrieved May 16, 2022, from <https://www.oreilly.com/library/view/learning-javascript-design/9781449334840/ch10s07.html>

Oxidane Venture. (n.d.). Oxidane Venture. Retrieved May 13, 2022, from <https://no.oxidane.vc/>

Proff. (n.d.). Proff.no. <https://www.proff.no/aksjon%C3%A6rer/-/fished-as/924507748>

Radigan, D. (n.d.). *Agile vs. waterfall project management*. Atlassian. Retrieved May 14, 2022, from

<https://www.atlassian.com/agile/project-management/project-management-intro>

Ramel, D. (2021, July 27). *Microsoft Replaces Xamarin Toolkits with New .NET MAUI Alternatives*. Visual Studio Magazine. Retrieved May 16, 2022, from

<https://visualstudiomagazine.com/articles/2021/07/27/net-maui-kits.aspx>

Rehkopf, M. (n.d.). *Kanban vs Scrum*. Atlassian. Retrieved May 14, 2022, from <https://www.atlassian.com/agile/kanban/kanban-vs-scrum>

Reid, R. D., & Sanders, N. R. (2012). *Operations Management*. Wiley.

Ridland, M. (2021, 05 12). *The Evolution of Xamarin.Forms*. XAM Consulting. Retrieved May 16, 2022, from <https://xam.com.au/the-evolution-of-xamarin/>

Schwaber, K., & Sutherland, J. (2020, 11). *The Scrum Guide*. The Scrum Guide. Retrieved May 14, 2022, from

[https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf#zoom=1](https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf#zoom=100)

00

Sommerville, I. (2011). *Software Engineering*. Pearson.

Use Cases. (2013, 10 09). Usability.gov. Retrieved May 15, 2022, from

<https://www.usability.gov/how-to-and-tools/methods/use-cases.html>

Wrike. (n.d.). *What is Time Management in Project Management?* Wrike. Retrieved May 14,

2022, from

<https://www.wrike.com/project-management-guide/faq/what-is-time-management-in-project-management/>

Zettler, K. (n.d.). *Trunk-based Development*. Atlassian. Retrieved May 15, 2022, from

<https://www.atlassian.com/continuous-delivery/continuous-integration/trunk-based-development>

Appendix

Appendix 1: Statement from Oxidane Venture



Statement from Oxidane / Fished

Oxidane Venture is the founder (and majority owner) of Fished AS. Fished is a marketplace for seafood making it easier to trade. Fished.com tries to open up the market, automate logistics, makes payment easier and secure and automate customs handling.

Fished.com is created as a responsive web-application, running on top of a set of microservices.

Our challenge to the students was: "Create a native app for IOS/Android for fished.com". This challenge included tasks like:

- Setup CI in Azure Devops for an app
- Deal with the security-constraints from fished.com (Auth0, tokens etc)
- Make use of existing backend rest api
- Programming and unit-testing in chosen framework
- Adoption and implementation of predefined design elements

The project

Our expectations when starting the project was that the group should learn (and use) a good development process. We also told the students a well-documented process and clean and well tested code was more important than the finished product.

As we see it the students excel on the following:

Oxidane Venture AS
Dronningens gate 2
4610
oxidane.no

Dokument ID:
Version:
Dokumenttype:
Infoklasse: Offentlig

Forfatter: Tor Oskar Willhelmsen
Dato: 10.05.2022

1

- They were self-organized, seemed to cooperate very well and managed to persistently make good progress throughout the project.
- They followed the stated process in a good way, have a proper branching strategy, doing pull request and making sure the code quality was good by writing proper unit tests.
- The technology used (Dart/Flutter) was new to the students and they made very good progress on the programming skills during the project. The process of choosing framework was also done in a good way as they tried out a couple of frameworks before deciding.
- Have used supporting tools (Azure devops) in a good way to support the development process.
- They have managed to deliver a multiplatform (IOS/Android) app with basic functionality, adopting our design principles and our security constraints.
- They have cooperated with Oxidane (customer) in a good way. Regular sprint demos and discussions were well received.

On the negative side we found one small issue:

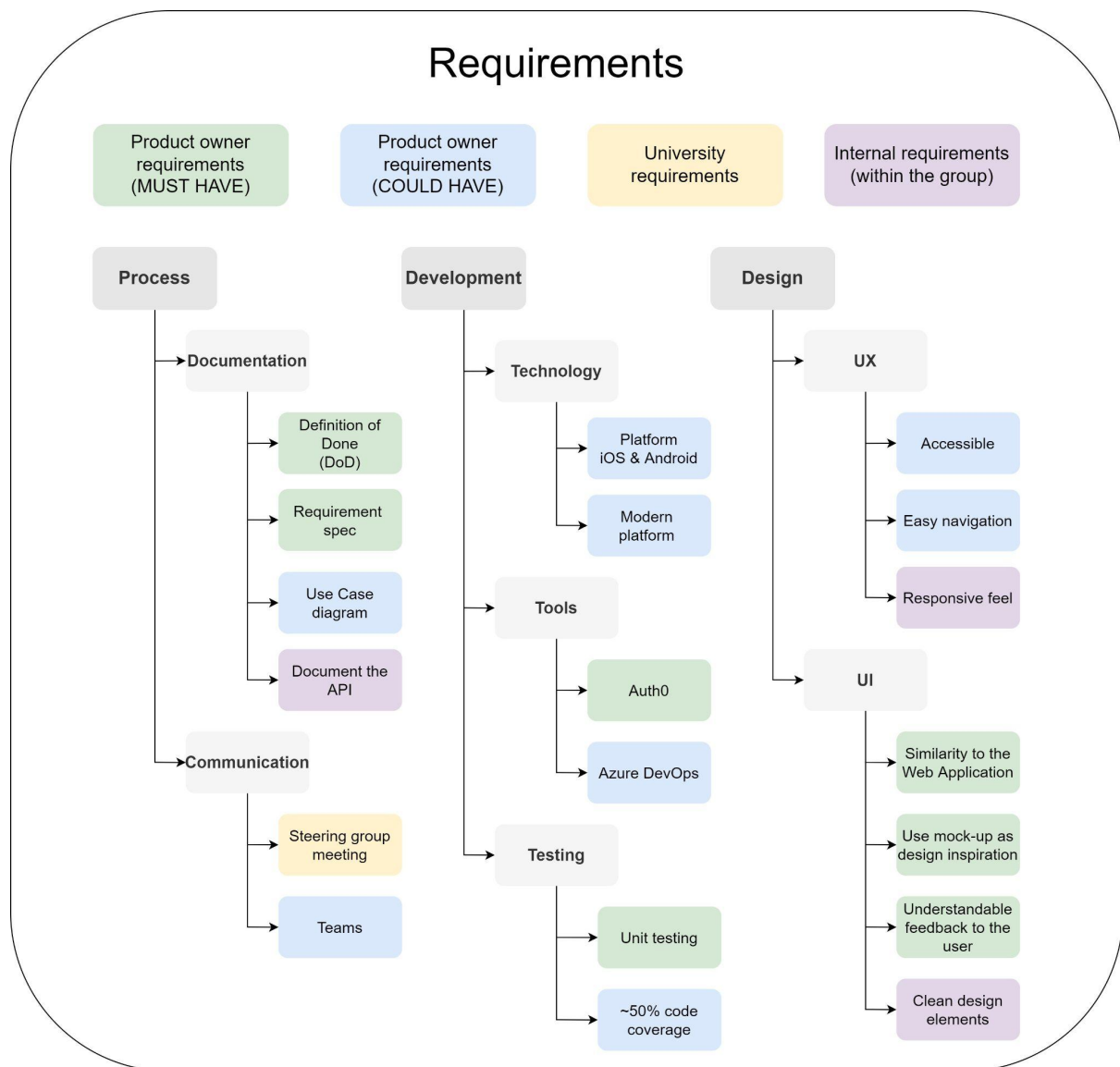
- A project always consists of a lot of activities. From the outside it seems like they allocated too many resources to other tasks than programming. In a project like this I would have assumed at least 3-4 persons fulltime on programming (and programming related tasks).

Conclusion

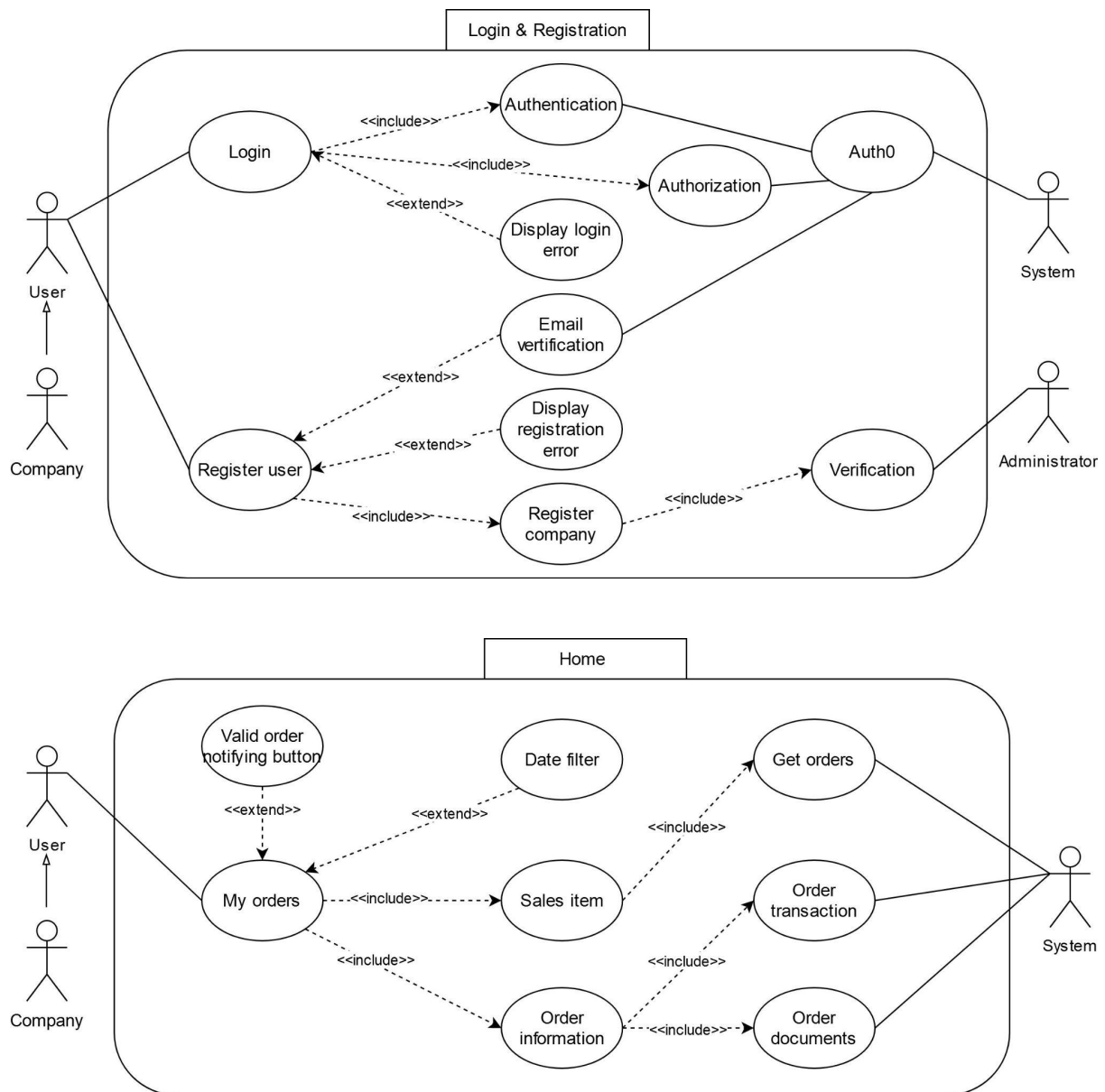
The group has shown ability to create an app in a new (for them) framework with build pipelines in place. They have also proved that they are ready for even more challenging projects. Overall we are very satisfied with the performance of the group which was better than expected when starting the project.

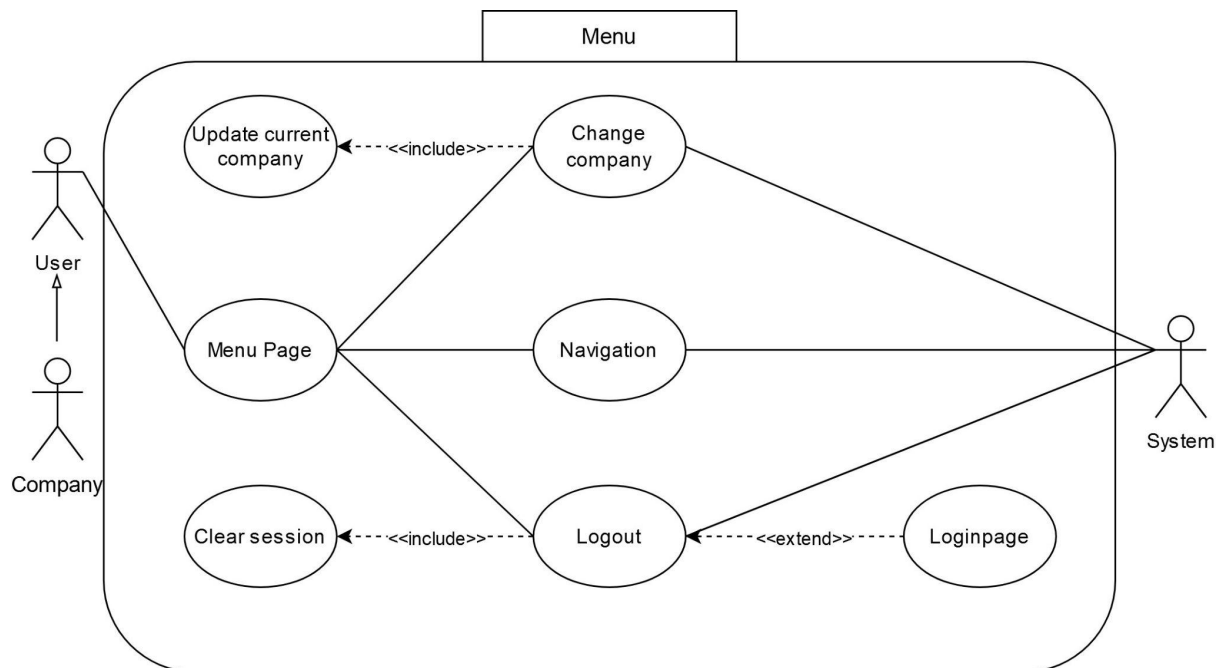
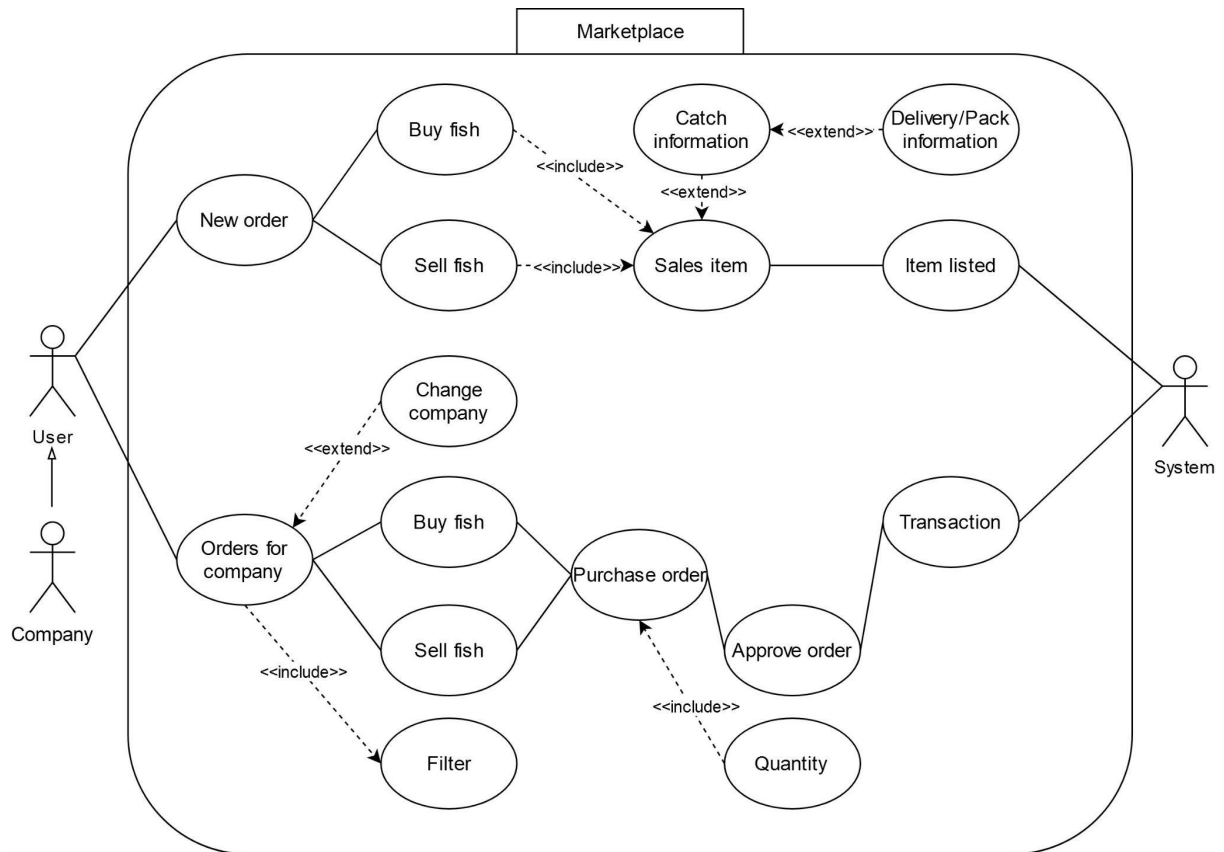
Tor Oskar Wilhelmsen
CTO Oxidane Venture
CTO Fished

Appendix 2: Map of the Requirements



Appendix 3: Use Cases





Definition of Done (DOD)

Last updated by | Michael Herland Valen | 15 May 2022 at 21:52 CEST

Table of Contents

1. [Definition of DOD](#)
2. [Administrative](#)
 - 2-1. [Before the task](#)
 - 2-2. [During the task](#)
 - 2-3. [After the task](#)
3. [Code \(DOD\)](#)
4. [Report & Assignment \(DOD\)](#)
5. [Formating](#)
6. [Common Pitfalls](#)
7. [Sources](#)

Definition of DOD

The Definition of Done is an agreed-upon set of items that must be completed before a task can be considered complete. It is applied consistently and serves as an official gate separating things from being "in progress" to "done."

Administrative

It takes into account the administrative process of how the Sprint Board should work.

Before the task

- Create the task if there's no task in the backlog
- Choose a task with high/est **priority** and **business value** within the scope of that Sprint
- Add **priority**, **business value**, **state** or **effort** to the task
- Agree with the group about what you are going to do that day (Daily scrum)
- Assign the task to the one completing it
- The timer should be started whenever a member goes for the next phase

During the task

- Set the task to **In progress**
- Ask for help if you use more time than the estimation
- Write down the challenges you've faced and how to solve them

After the task

- Set the task to **Done** after you complete it
- Ask for a review after the task is completed
- Task is done when two members or more have agreed that the task is done. It can be done at a daily scrum meeting

Code (DOD)

- A Sprint is only completed when the group has agreed on when to end the Sprint
- The code has to be reviewed by at least one other group member before merging
- At least three group members should review the code before merging into the main
- The code needs to pass the automated build and tests
- The code needs to have a test included before merging
- Data objects should not be tested
- Merge conflicts should be handled by the author and another group member
- When debugging, try to create a diary on what you did to fix the problem
- Ensure that there are no secrets inside the codebase when pushing up to the repository
- Follow the code & git standard that is written in the Wiki

Report & Assignment (DOD)

Note - Report and assignment is used interchangeably

- Every member should review the report before submission. Furthermore, it includes proofreading and giving feedback
- The report should follow APA 7 and contain a source for every statement that needs it
- Figures should contain numbers and a name
- Sources shouldn't be older than five years when talking about recent activities
- Assignment should contain the work and a reference page
- Everything in the report should be written in English
- The report must be rewritten once after giving feedback teacher
- The report or assignment needs an agreement within the group before it qualifies as "Done."

Formating

- 11 font size
- 1.15 linespace
- Times New Roman
- Use predefined titles

Common Pitfalls

- Obsessing over the list of criteria can be counter-productive; the list needs to define the minimum work generally required to get a product increment to the "done" state
- Individual features or user stories may have specific "done" criteria in addition to the ones that apply to work in general
- If the definition of done is merely a shared understanding, rather than spelled out and displayed on a wall, it may lose much of its effectiveness; a good part of its value lies in being an explicit contract known to all members of the team

Sources

<https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf#zoom=100> ¹⁷

<https://www.scrum.org/resources/blog/done-understanding-definition-done> ¹⁷

<https://www.agilealliance.org/glossary/definition-of-done/> ¹⁷

Best practice Flutter & Dart

Last updated by | Ole Bjørnar Granås | 15 May 2022 at 20:02 CEST

Naming convention

General naming convention

```
class MainScreen {...}
enum MainItem {...}
typedef Predicate<T> = bool Function(T value);
extension MyList<T> on List<T> {...}
```

Naming convention for library & imports

```
library firebase_dynamic_links;
import 'package:demo/src/utils/dialog_utils.dart';
```

Naming convention for declaring variables.

```
int item = 10;
final Car car = Car();
String name = 'John';
const int timeOut = 20;
```

Exception handling

```
try {
  await foo();
} on Exception catch (e) {
  print(e); // Only catches an exception of type `Exception`.
} catch (e) {
  print(e); // Catches all types of `Exception` and `Error`.
}
```

Adding values to already existing object/variables

```
var y = [4,5,6];
var x = [1,2];
x.addAll(y);
```

Cascades Operator

```
var path = Path()
  ..lineTo(0, size.height)
  ..lineTo(size.width, size.height)
  ..lineTo(size.width, 0)
  ..close();
```

Raw Strings instead of escaping inside strings.

```
var s = r'This is demo string \ and $';
```


Return variables & functions

```
get width => right - left;
Widget getProgressBar() => CircularProgressIndicator(
  valueColor: AlwaysStoppedAnimation<Color>(Colors.blue),
);
```

Avoid `print` method when debugging, use:

```
debugPrint()
```

Use `$variable` instead of `variable + string + variable`

```
var description = 'Hello, $name! You are ${year - birth} years old.';
```

Less code is good

```
List<String> names=[]

// Don't
names.forEach((name) {
  print(name);
});

// Do
names.forEach(print);
```

Async/await example

```
Future<int> countActiveUser() async {
  try {
    var users = await getActiveUser();

    return users?.length ?? 0;
  } catch (e) {
    log.error(e);
    return 0;
  }
}
```

Sub Widgets (unsure)

```
Scaffold(
  appBar: CustomAppBar(title: "Verify Code"), // Sub Widget
  body: Container(
    child: Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: <Widget>[
        TimerView( // Sub Widget
          key: _timerKey,
          resendClick: () {}
        ),
      ],
    ),
  ),
)
```

Use Const for values that are not supposed to change

```
Container(  
  padding: const EdgeInsets.only(top: 10),  
  color: Colors.black,  
  child: const Center(  
    child: const Text(  
      "No Data found",  
      style: const TextStyle(fontSize: 30, fontWeight: FontWeight.w800),  
    ),  
  ),  
);
```

Pro tips

- Use `ListView.builder` instead of `ListView` for longer list.

Git Cheatsheet

Last updated by | Michael Herland Valen | 25 Jan 2022 at 11:47 CET

Basics

```
git add , OR git add -A
```

```
git commit -m "Message"
```

```
git checkout <branch_name>
```

```
origin/<branch_name> -> Remote branch
```

Git tree

```
git log --graph --online --all
```

Reset branch

```
git reset --hard <last_working_commit_id> -> git push --force
```

Git Rebase

1. `git add , OR git add -A`
2. `git commit -m "Message"`
3. `git rebase -i OR git rebase -i HEAD~(x commits)`
4. `Pick 00000 Commit message OR Squash 00000 Commit message` (Alternative P = Pick / S = Squash)
5. Save & Quit out of VIM
6. `#Commented out commit message OR Non commented commit message`
7. Save & Quit out of VIM

VIM

`i` = INSERT MODE

`ESC` = CODE / Quit INSERT MODE

`:wq` OR `:x` = Save & Quit

`:q!` = Force quit