



Forside

IS-304: 2022

Tittel: Development of planning system

Emnekode	IS-304
Emnenavn	Bacheloroppgave i informasjonssystemer
Emneansvarlig:	Hallgeir Nilsen
Veileder	Niels Frederik Garmann-Johnsen
Oppdragsgiver:	Lars Lohne, Red Rock AS

Studenter:

Etternavn	Fornavn
Hansen	Denni
Ismailov	Ramzan
Gayrbekov	Israil
Minzat	Tiberiu
Dørdal	Ulrik
Krath	Julie

Jeg/vi bekrefter at vi ikke siterer eller på annen måte bruker andres arbeider uten at dette er oppgitt, og at alle referanser er oppgitt i litteraturlisten.	JA x	NEI
Kan besvarelsen brukes til undervisningsformål?	JA x	NEI
Vi bekrefter at alle i gruppa har bidratt til besvarelsen	JA x	NEI

Preface

This is a report detailing our experience of our last semester working together as a team on our bachelor's project. It will include how we as a team have worked together to bring about a quality product within a methodological framework, and the experience and knowledge gained during the project duration. Before that, we feel obligated to give a round of thanks and gratitude to those involved in the project.

First and foremost, we would like to give thanks to Red Rock AS for their generosity to have granted us the opportunity to work under their excellent supervision and guidance, and to have commissioned us to develop a product which would not only be to their benefit but also to us with regards to the knowledge and experience gained working in an environment which more resembles the professional life that is hopefully to come.

Secondly, we also want to express our gratitude to Red Rock AS' representative to us, Lars Lohne, Chief Project Manager. Lohne has been critical in pulling us through tough roadblocks throughout the project and we credit the project's success to his supervision and earnest ambition for us to accomplish our goals. We must also not forget to credit Andreas Kvalbein Fjetland, a Senior R&D Engineer at RedRock.AI, for his assistance in the choice of technologies and advice on suitable platforms for the purpose of organizing the project work.

Thirdly, we want to acknowledge Hallgeir Nilsen, who is a senior lecturer and the course coordinator, representing the faculty staff of University of Agder. Nilsen has provided us students with every opportunity to work in a professional setting, and we have experienced him as someone with the students' well-being as primary interest. It is hard to believe that we would land ourselves a project with a professional company at all without his helping hand and encouragement.

Finally, we recognize Niels Frederik Garmann-Johnsen, our supervisor and representing the faculty staff of University of Agder, for his counselling to have directed this project in a positive direction. Garmann-Johnsen would ask the right questions which gave us new perspectives on how to approach problems.

Table of contents

1. Introduction	7
1.1 Overview	7
1.2 Purpose of the project.....	8
1.3 About Red Rock	8
1.4 User group	9
2. Project approach.....	9
2.1 Requirements from Red Rock AS	9
2.2 Balancing features, time and costs	12
2.3 Project management requirements	13
2.4 Technological requirements	13
2.4.1 Micro-service architecture.....	14
2.4.2 Docker and containers	15
2.5 Quality requirements	15
2.6 Project management choices	16
2.6.1 Jira vs Trello.....	16
2.6.2 Scrum	17
2.6.3 Risk management	18
2.6.4 Estimating and prioritizing Backlog issues	18
2.7 Communication	19
2.8 Project roles.....	20
2.9 Extent of involvement in the development process	21
3. Planning and execution	22
3.1 Initial stages.....	22
3.2 Requirements.....	22
3.3 Dataset and Database design	24
3.4 UX-design	25
3.4.1 Personas.....	25
3.4.2 Usability	26
3.4.3 Wireframes and mockups.....	26
3.4.4 Benyon's design principles	29
3.5 Sprint retrospective	33
3.6 Development process	34
3.6.1 Project qualifications.....	34

3.6.2 Initial planning	34
3.6.3 Technology used	35
3.6.5 Programming process	37
4.0 The project.....	40
5.0 Reflections and challenges	40
5.1 Project management	40
5.1.1 Scrum in practice.....	41
5.1.2 Communication	42
5.1.3 Task allocation	43
5.2 Quality of end product.....	43
5.3 COVID-19	44
5.4 Time management	44
5.5 Change of product owner	45
5.6 Summary	45
6.0 Statement from product owner	45
7.0 Self evaluation	46
References	49
Appendix	51

List of Figures

Figure 1. 1 System Overview	8
Figure 1. 2 Products by Red Rock AS	9
Figure 2. 1 Qualities	10
Figure 2. 2 GitHub	12
Figure 2. 3 Screenshot of backlog	19
Figure 2. 4 Screenshot of Discord channel and chat	20
Figure 3.1 Database (ER diagram)	25
Figure 3.2 Persona.....	26
Figure 3.3 Initial wireframe created together with the product owner.....	27
Figure 3.4 Final wireframe.....	28
Figure 3.5 Mock-up of the system.	28
Figure 3.6 Example of visibility.....	29
Figure 3.7 Example of familiarity	30
Figure 3.8 Example of constraints.....	32
Figure 3.9 Example of flexibility	33
Figure 3.10 Display of the Subtask class from models.py file.....	36

List of Tables

Table 3.1 System Requirements	23
-------------------------------------	----

1. Introduction

We students of the IT and Information Systems Bachelor's Program have been commissioned by the product owner Lars Lohne to develop a software system, which will serve as a planning and management platform for loading and off-loading operations between ships and docks. The software system, intended for use by business clients such as the likes of ASKO, improving upon a business process which saw a lack of a system like it and thereby inefficiencies.

Further in this chapter, we will introduce a general overview of the project at hand, its stated purpose, what it intends to deliver and the user group of the system in order to establish a context. We will also present who the stakeholder Red Rock AS is and what they are involved in.

1.1 Overview

Red Rock AS has assigned us a task to develop a software platform for autonomous load management and planning of the operations and performing functions, and this software platform is intended to be integrated into Red Rock AS' larger complex of system. As seen in the figure below. This is only a part of a more complex system. The planning system's immediate purpose is to feed data to autonomous robots which takes care of the physical work of moving goods around. The software platform itself focuses on a planning system for autonomous operations. Specifically, there is a need to take a closer look at the scheduling function, which structures the operations together with their subtasks to be performed by the autonomous machines. The plan will be based on orders reported from external sources.

The planning function should retrieve information about orders and a corresponding set of subtasks through an API, in our case the information will be retrieved from a database. The order is typically a picking list or sequence of transactions the machines will perform. The plan itself is called an operation and each of the order lines is a Subtask. It should be possible to sort the jobs based on the different values they have. The plan should be able to be adjusted by changing the order of the jobs, adding jobs in addition to those that are based on the order lines and deleting jobs. The manual lines should be easily identifiable in the plan.

A plan should be able to have different statuses based on how long it has come. These statuses must be defined later. The corresponding order line should also be displayed on the jobs as

information. Based on available data about sequencing the order lines from the order, the plan should reflect this.

To summarize, the task was to create a partly front and backend-based planning system. The data received from external sources needed to go through a planning system before the data was forwarded to robotics for further integration and finally the last step of execution. This step of planning which arguably is the most crucial step of the process, is the one we had our main focus on.

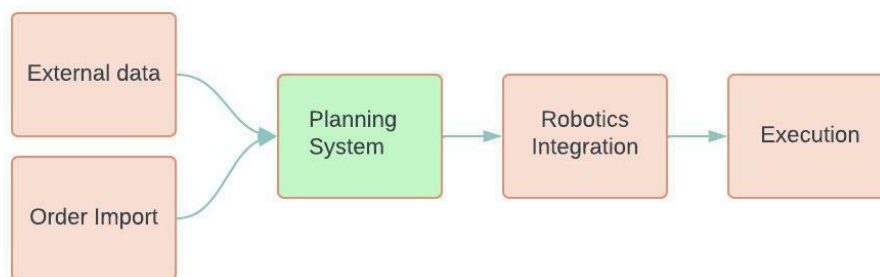


Figure 1. 1 System Overview

1.2 Purpose of the project

The purpose of the project has mainly revolved around the learning objectives of IS-304 which we had set out to fulfil. Not only would it be a substantial learning experience for us students, but it could potentially return value to the stakeholder Red Rock AS, a case where both parties will benefit. Our knowledge and experience from the past three years of our studies have culminated up to this point, we therefore hoped it would all come to benefit us as much as possible during the project period and to conclude it with a potentially shippable product. Beyond that, it would also give us the experience of working on system development in a more corporate, professional setting, preparing us for the professional life ahead.

1.3 About Red Rock

Red Rock AS, recently acquired by Ocean Infinity (Shevchenko, 2022), is a leading organization involved in several fields within marine engineering and information technology. The organization employs highly technically proficient personnel with their own field of expertise, who all co-operate together to deliver innovative products, services, and ultimately value, while also offering substantial support to its clients, whether local or international. Beyond delivering marine engineering solutions, Red Rock AS also has its hands in more

traditional information technology solutions, such as delivering tailor-made cloud-based ticketing platforms for public transportation systems. Lars Lohne, a representative of Red Rock AS and Chief Project Manager, will be our main supervisor for the project and the product owner of the commissioned system.

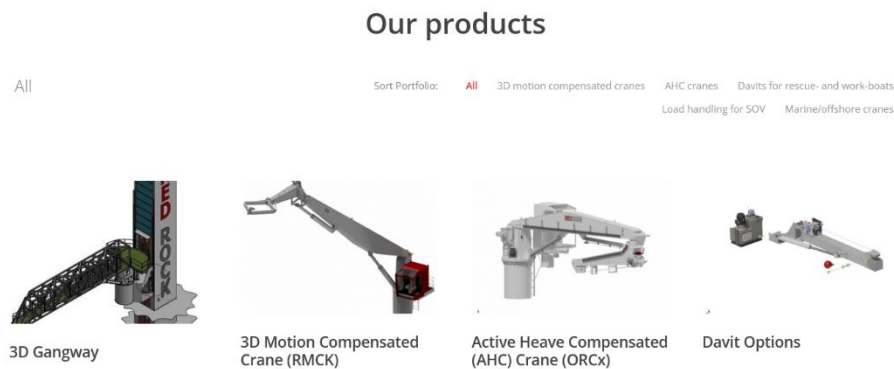


Figure 1. 2 Products by Red Rock AS

1.4 User group

Red Rock AS intends to offer the commissioned system to business clients involved in shipping logistics, an example of which is ASKO, Norway's largest grocery wholesaler (ASKO, 2022). This system will then be in the hands of staff managing and overseeing operations between ship and dock. Beyond that, we do not have explicit demographical data and neither do we consider them ultimately necessary for us to deliver a quality product. However, given the field of work the system would be deployed in, we can expect our user group to have a large, majority composition of men ranging in age from the middle 20s up to middle age who are responsible for logistical co-ordination.

2. Project approach

2.1 Requirements from Red Rock AS

Red Rock had given us several requirements that impacted our central decisions on methodology, project management, technology, standards, quality, and quality assurance.

Internal and external quality

Beyond just quality, there are two additional dimensions, these are external and internal qualities (Long, 2010). Striking a balance between these is what developers should aim for, as neither should be neglected for the other (Long, 2010). When it comes to external quality,

essentially it is to which degree the product conforms with the set expectations and requirements of the product owner. Of which there are numerous factors, such as reliability, ease of use, adaptability and so on. It may be summarised as the immediate, subjective experience of the end user. Internal quality on the other hand may be described as the structure of the product, which concerns characteristics such as cohesion, low coupling, low duplicity, simplicity and so forth. These are not directly experienced by the end user (Long, 2010).

We find that our case is somewhat special, as our product owner and stakeholders are for the most part experienced developers themselves, and therefore the requirements and expectations they have put forth may be argued as overlapping with external and internal qualities, which we have illustrated with a Venn diagram in the figure below.

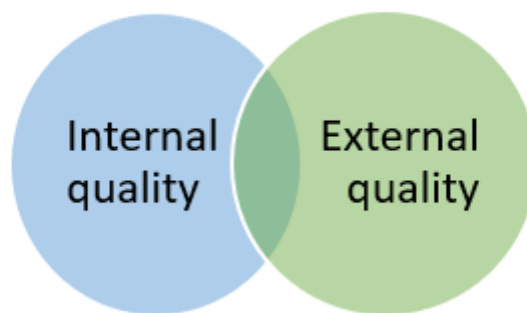


Figure 2. 1 Qualities

For example, Red Rock AS' requirement for a micro-service architecture would not be typical of, for example, a product owner with a background in finance commissioning a product for a financial enterprise. Such a product owner would know what external qualities to ask for, such as an interface to transfer funds, but hardly likely would know the technical requirements to support such an element. We argue that their specifications such as micro-service architecture and Docker are not just demands to internal quality but also external quality, and therefore the pressure on us to not compromise on either is greater.

Quality Assurance

Quality Assurance plays a critical role in every product under development. Our goal for the development process is to ensure high internal and external quality. Which means the quality of a product or service provided to customers should be of outmost quality both internal and

externally (Hamilton, 2022). Its process is mainly based on identifying bugs or poor-quality code. In our development process which we will come back to later, we proactively ran tests of the application, reviewed the code submitted by peers, implemented a coding standard, and had a version control system in place to track changes.

Coding standard

Having a set of rules and guidelines on how the source code should be written in an application has its benefits. Your working peers will have minimal issues understanding the source code and further development (PVS.Studio, 2013). Coding standards used in developing RedRock AS's planning system consist of commonly used naming conventions, for example; instead of "ISDISPLAYING()", we would use "isDisplaying". Coding standard currently in use for this project are on [appendix 12](#).

Tracking changes to the source code is an important part of the development process. To ensure good workflow and prevent conflicts in the source code, we decided to use GitHub along with its GUI GitHub Desktop. The source code for the application is stored in the "main" branch of the project repository on GitHub, and every member of the group also has their branches. These branches enable every member to work on the source code in their respective branches without affecting the "main" branch code. They also can create a pull request, which will propose a merge between their branches.

Design Pattern

Since the applications backend is developed using the Django framework, its entire structure is heavily based on the Model-View-Template (MVT) architecture (GeeksForGeeks, 2021). MVT is a design pattern that utilizes the model, view, and template components of an application. MVT assists in keeping the application organized by having each component execute its objectives. The model is linked to the database; its objective is to ready data from the database for the view component to fetch. The view takes a web request and returns a web response.

This response can be data fetched from the model or normal HTML or JavaScript content of a webpage (GeeksForGeeks, 2021). The template is the front-end component of Django, and it consists of the HTML output of every page in the application, for example, a page requested in view (Shadhin, 2021). For the front end, we didn't go with a specific design pattern.

Testing of application

Ensuring that every component of the application is running without any occurring exceptions or runtime errors. We decided to utilize the Django framework's unit test for exactly executing this part. These unit tests are based on Python class, Testcase, and both the frontend and backend of the application are tested by iterating and checking the URLs, Views, Forms, and Models (Django, n.d.). For example, by testing URLs it would check if the URL matched the View, this process continues throughout the entire testing phase.

Code review

Reviewing peer code is an important part of a development team, this was not an exception for our team. Before the start of the development, we planned on how the code of each member should be reviewed. The conclusion was one member would be assigned in charge of reviewing the code each member submitted by creating pull requests from their branch to the main branch. Since the group consists of six members, it would be more suitable to have one assigned to review the code while being part of the development team. These pull requests would contain their own headings and information about which new changes and functions were implemented. The reason for going with detailed pull requests is that they would ensure a more organized and well-logged history of every pull request.

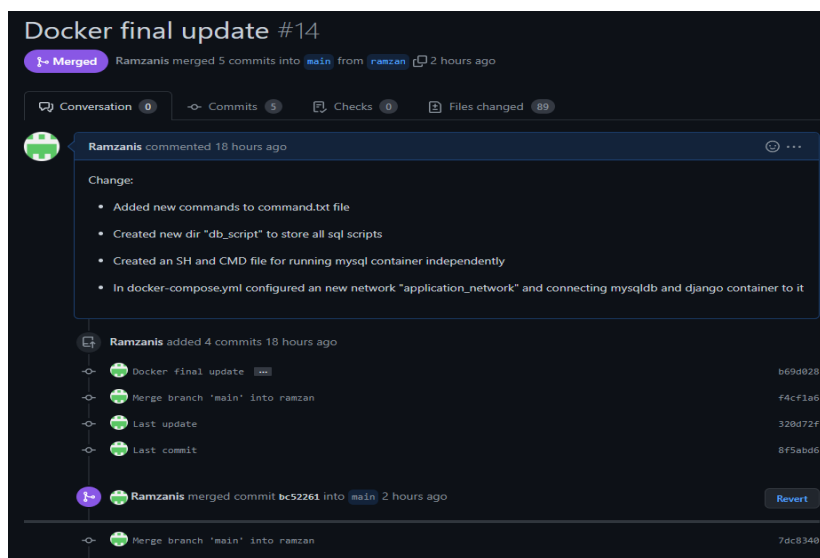


Figure 2. 2 GitHub

2.2 Balancing features, time and costs

This project has presented us with several constraints, constraints which had forced us to make trade-offs between objectives, the objectives being time, scope (or features), and budget (or costs). An example of this is where we were made to balance the objectives of features

and time, was with regards to the scope of the product. Red Rock AS stated several wishes as to what features should be included beyond the most important. However due to our own constraints on resources. We made the decision to do away with a number of them with the understanding of Red Rock AS. Despite the potential of a more advanced system, given the time constraint of expected delivery, and our lack of experience in the matter, we believed it would not be up to par to deliver in a reasonable amount of time.

Had we made the decision to implement the features which were discarded, it may have in worsened the quality of the final product, falling short of the product owner's requirements and expectations. With this in mind, we justified our decision to cut features by the fundamental goal of delivering the right quality, that is quality with a healthy balance between the aforementioned objectives, and actually meeting the product owner's expectations.

2.3 Project management requirements

For project management, Red Rock AS suggested that we adopt an agile work management solution. Agile methodologies are frameworks that teams and organizations use to put the Agile mindset into practice (Agile Methodologies, 2021). After discussing and listening to all parties, we concluded to use Jira. Jira Software is built for every member of your software team to plan, track, and release great software (Jira, 2022). Jira is an issue and project tracking software used by teams across software, IT and others. It's an agile task management solution that allows users to organize tasks and track them more accurately with predefined yet customizable workflows. Because of this and combined with the fact that Red Rock AS itself had Jira as a standard in its daily and ongoing projects. We decided to go with Jira as our main framework.

2.4 Technological requirements

Deciding which tools would be used for this project was the first of many decisions we took. After discussing this topic with the product owner, we concluded to use the Python programming language which they already had experience working with along with its web framework Django. Python is an interpreted, object-oriented, high-level programming language (Python, n.d.). Its syntax is fairly similar to languages such as Java. While Django is a high-level Python web framework used by corporations such as Instagram and Spotify (Django, 2022). Due to its availability of packages used to build projects, it is most suitable for rapid development (Django Packages, 2022). Since none of the group members had any former experience with Python or Django, learning to use these new tools would be done in a form of self-study by watching online tutorials on platforms such as YouTube and Udemy.

For storing data, we chose to go with MySQL database because everyone in the group has former experience working with it. MySQL is a relational database management system based on structured query language (SQL) (Talend, n.d.). We used DataGrip to work with and manage the database, which is a cross-platform IDE for databases. It is designed to query, create, and manage databases. It is software that may be deployed locally, on a server, or on the cloud (JetBrains, 2022). DataGrip has a similar user interface to IntelliJ IDEA and Visual Studio Code which is used by everyone in the project team, hence why DataGrip was chosen as opposed to f.ex. MySQL Workbench, not to mention.

2.4.1 Micro-service architecture

A basic requirement put forth by Red Rock AS was that the system should be developed with a micro-service architecture in mind. In a micro-service architecture, an application is broken down into constituents which do not depend on each other to function, as opposed to a monolithic structure where its components are too dependent on each other to operate autonomously by themselves, where a single modification of a line of code may result in disastrous consequences for the entire system, an extreme example of which being a complete shutdown.

Although a micro-service architecture is additionally more complex than a system of monolithic architecture (Johnson & Shiff, 2021), its benefits are numerous. A system constructed as a collection of modular and independent constituents better facilitates Agile development, it is easier to maintain and test, and the isolation of components increases resilience in times of failure or downtime of individual parts. A micro-service architecture will also make the lives of later developers easier should there be a need to develop additional features.

However, despite this such an architecture also comes with some drawbacks, notably the concern for security (Johnson & Shiff, 2021). As a micro-service architecture introduces a multitude of points of contact for a malicious attacker to potentially exploit, and the increased complexity may demand higher costs and initial development time. Despite these, its benefits are too great to set aside, and the increase in adoption of Agile practices (McLarty, 2016) means that micro-services have taken a prominent role in today's software development landscape.

Knowing that Red Rock AS is continuously working towards being at the forefront of IT innovation and adopting the best practices, we understand the demand for a micro-service architecture to be a substantial quality requirement and wish to fulfil it to satisfyingly meet the product owner's expectations.

2.4.2 Docker and containers

Docker is a “popular virtualization software that helps its users in developing, deploying, monitoring, and running applications in a Docker Container with all their dependencies” (Lilly021 Team, 2022) and has been posited by Red Rock AS as a requirement that the commissioned product should be able to be deployed in such a manner.

Using Docker would permit Red Rock AS to deploy the system across a vast range of machines running different operating systems. Thereby making DevOps a more streamlined process. Removing the time-consuming process of ensuring that software works on all computers and having to alter settings and configurations. In short, Docker may be summarized as eliminating the problem of “it works on my machine” and opening the door for deploying software as “encapsulated, ready-to-run applications” (Palmer S., 2022) (Palmer, 2022)

2.5 Quality requirements

Red Rock AS did not make explicit quality requirements beyond that the commissioned system must support logistical operations with a set of features. Which before were lacking in the business process A good definition of business process being given as “a series of steps performed by a group of stakeholders to achieve a concrete goal.” (Kissflow, 2022). It was then left to us to discover the best means of ensuring quality in developing said features. On the onset, we saw it necessary to settle for a development process as the means of assuring quality.

Red Rock AS did not make explicit quality requirements beyond that the commissioned system must support logistical operations with a set of features. Which before were lacking in the business process A good definition of business process being given as “a series of steps performed by a group of stakeholders to achieve a concrete goal.” (Kissflow, 2022). It was then left to us to discover the best means of ensuring quality in developing said features. On the onset, we saw it necessary to settle for a development process as the means of assuring quality.

2.6 Project management choices

Proper project management can be a vital aspect of any development process. Therefore, it is a major concern to carefully approach this topic when choosing how to organize a project. With that in mind, we have documented the choices made with regards to project management and the arguments behind them in the remaining sections of this chapter.

2.6.1 Jira vs Trello

In the project's initial stages of choosing a suitable project management tool, it was internally suggested as a team that we should utilize Trello for our project management needs, the reason being that we had experience with the platform from former projects. This was then brought up in a discussion with representatives from Red Rock AS, and the suggestion from then was that we should rather use Jira as our choice of a project management platform as they spoke positively of it and they themselves co-ordinate projects with Jira, to which we agreed.

Furthermore, Jira appeals more to the software developer by better integrating features intended for software development such as GitHub. Something we refrained from using, as we felt that the training would consume too much time and instead worked in a more traditional, isolated side-by-side fashion. In comparison with Trello, which is more appropriate for general projects such as marketing campaigns, product development and so on (Piras, M, 2022). Another benefit over Trello would be that our product owner would have an easier time having an overview and supervising the project if it was hosted on Jira given that they were far more familiar with the platform. Other than what was mentioned, the differences between the two platforms are rather negligible, coming down to subjective experiences, and therefore we do not see them important to raise.

There are 3 main features of Jira to facilitate project management, and they are Roadmap, Backlog and Board. The Roadmap feature neatly presents former and current sprints in a linear fashion. Which can be displayed in the scales of weeks, months or yearly quarters, which comes in handy for a quick overview and giving the scrum master an idea of where the project is currently standing from a bigger perspective. Backlogs are where issues (further represented into Stories, Tasks and Bugs) are added, prioritized, assigned to team members, and estimated in terms of difficulty, known as Story Points. According to Atlassian, issues “typically represent things like big features, user requirements, and software bugs “ (Atlassian, 2022), and as convention, we did not add issues that were not directly related to

the objective of delivering functional software at the culmination of every Sprint, such as administrative work. With regards to estimating difficulty of issues, there really is no absolute standard scale to abide by. In our case we settled for a scale which goes from 1 to 10, 1 being very easy and 10 being the most difficult, with the benefit of easily communicating the meaning of the scale. New issues in the Backlog may then be dragged and dropped onto a new Sprint session as desired, which brings us to the Board feature.

As its name might suggest, the Board is where issues are presented in an orderly manner, which are then further categorized into the columns To Do, In Progress and Done, their names being self-explanatory. An issue may then be dragged and dropped into its appropriate column according to its progress. Issues that are still in the stage of To Do or In Progress may also be transferred over to the next Sprint session if needed. All of these aforementioned, convenient features will be of great support in effectively managing the project work.

2.6.2 Scrum

Even at the onset of the project, we had decided to use Scrum as our framework to facilitate the project as opposed to Kanban or Waterfall. This is due to our previous experience in former project work using Scrum, which has consistently delivered positive results. Regardless, to reinforce our decision, we will further argue for it by comparing Scrum with Waterfall, as Kanban and Scrum is a part of the Agile family of methodologies.

Our first major point is that Scrum permits changes in all of its stages, as opposed to Waterfall, where after the initial stage of documenting requirements, a change in requirements is off-limits (Gaille, 2020). This inflexibility of Waterfall is unacceptable to us, especially as developers. In a software development project commissioned by a customer/client/product owner, one must always expect constantly dynamic customer requirements and priorities as the project progresses (Alfonso, 2018) for example in the form of a new feature.

We had also settled on two-week Sprint periods. We reasoned, in our case, that a Sprint duration of one week would place too much of a time pressure on us, creating an uncomfortable work environment and potentially causing burnout, consequently resulting in less than desirable results. A Sprint duration of three or more weeks could have possibly incited complacency and sense of postponement. Therefore, we concluded that two-week Sprint periods was the most optimal balance between urgency to deliver and postponement, and subsequently would return the best quality. This choice was also reaffirmed by the Red Rock AS representatives, as their practice were also Sprint periods of two weeks.

2.6.3 Risk management

It is imperative that we understand not everything will take place as planned. Risks, whether they manifest during the project work or not, can present themselves as major roadblocks in the period of project work- negatively affecting the quality of delivered products.

Therefore, we have speculated and identified what risks could reasonably take place during the project period and organized them in the format of a risk matrix according to the extent of their consequences; their likelihood of occurring; proactive measures to reduce the likelihood of said risk, as risk management is an active process for the entirety of the project duration. These reactive measures to reduce the harm in the occurrence of said risk. The scope of the risk matrix must be so that it includes potential risks during the whole project period, and not only the immediate present.

With the risk matrix in place, there is now an orderly mitigation strategy as a reference for the purpose of active risk management and in the event of a risk occurrence, which will certainly save time and resources in resuming optimal project work. The risk matrix may be found in [appendix 15](#).

2.6.4 Estimating and prioritizing Backlog issues

We would like to further flesh out the rationale in how we have selected, estimated, and assigned issues in the Backlog, and how Stories and tasks relate to those issues. As was previously mentioned with regards to estimation, we settled for a scale which goes from 1 to 10. Assigning the correct value to issues was a matter of speculating on the approximate time it would take to deliver, as we reasoned that the more time an issue will take, the more difficult it would be to complete. As an example, if we estimate an issue will take more time to reach completion relative to other issues in the span of the Sprint, we would say that its place in the scale of difficulty nears 10, and vice-versa for issues which we estimate to take less time for completion.

Issues are then further categorized into Stories and Tasks. Bugs, as mentioned, are also available as an issue category, but we never saw the need to categorize any issue into Bugs, therefore we will only further describe the two categories of Stories and Tasks. Stories, also known as User Stories, relate to features from a user's perspective. They are often expressed in non-technical terms so that everyone in the Scrum team, regardless of their technical knowledge, can comprehend. It can also help to visualize what frontend elements are needed to support the user task by putting oneself in the shoes of the end-user.

Tasks are issues that do not necessarily relate to the immediate user experience but support the system as a whole. Tasks are also lesser in scale as compared to User Stories, the smallest unit of work. One can consider Tasks to be a decomposition of Stories, meaning that a Story may have one or several Tasks related to it for its functionality, whether they are elements in the front- or backend. Task may also be completely unrelated to a User Story, and rather support background processes which a user would not directly experience (Nelson, 2020).

Regarding assignment of issues, we mostly left this is a matter of one's own initiative. Issues which involved several group members were left unassigned as Jira as of now has no means of assigning multiple group members to one issue, so we improvised in order to communicate that meaning.

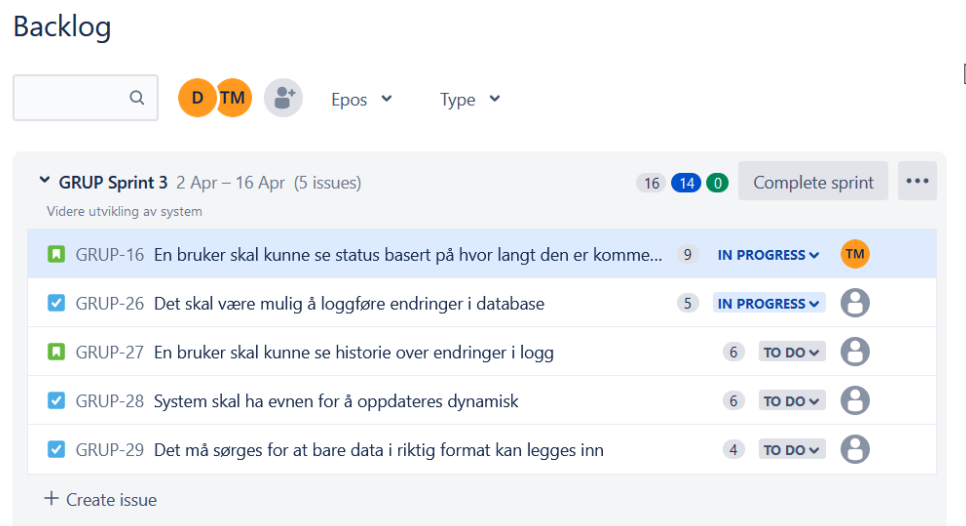


Figure 2. 3 Screenshot of backlog

2.7 Communication

In order for the parties involved in the project work to effectively communicate ideas it was important to decide on the most optimal channels of communication, not only between ourselves as the Scrum team but also with the product owner.

As for communication between the Scrum team and with the product owner, Discord was the choice of platform. Initially, dialogue with the product owner was done over e-mail, but it was quickly experienced that such a mode of communication was too slow for exchanging ideas and arranging meetings, considering the desired pace of the project work. Meetings with the product owner were also hosted on the Zoom, Microsoft Teams and Google meet platforms.

Conveniently, Red Rock AS were already present on Discord together with other student groups, so communication was resumed there, including meetings, as Discord has voice, video and screen-sharing features. Discord is also a platform that we students are quite familiar with and have a constant presence on, so missing out on important updates and messages were unlikely as compared to e-mail.

Our supervisors from the faculty staff of University of Agder, did not have much of a presence on Discord, so communication was still through e-mail and meetings were hosted on Zoom.

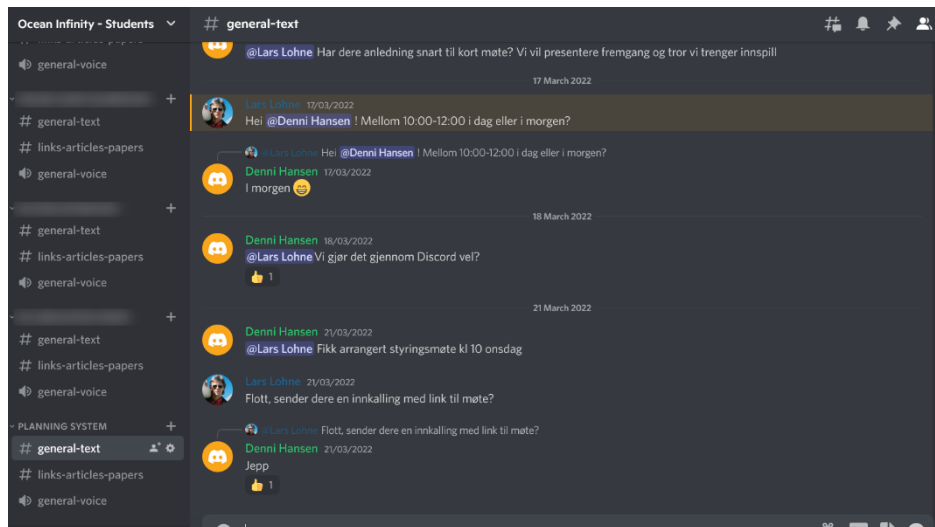


Figure 2. 4 Screenshot of Discord channel and chat

2.8 Project roles

For the purpose of assigning roles in the Scrum team, we had resolved to adhere to Scrum's three central roles so that a base definition of responsibilities and liability would be in place (West, n.d.). In return we would see an increase of productivity in comparison to if assigning roles had not taken place. The three central roles are as follows: (1), the developer. The developer is the driving force behind the productive work to deliver a working product at the end of a Sprint. It should be noted that the role of developer is not exclusive to a software developer, but it may also be attributed to Scrum members of different abilities and expertise, such as a UX designer, a database modeler, and so on (West, n.d.).

(2), the product owner. The product owner has a responsibility to communicate their needs, specifications and value. That is so that the developer team can arrive at a quality product which fulfils the product owner's vision, including the stakeholders, of their commissioned product. The product owner has a responsibility that the Backlog of issues are in place for the

development team to start working on, but this process may also involve other Scrum roles. The product owner is also involved in Sprint reviews, together with the rest of the Scrum team, with the feedback of the product owner being the most sought after (West, n.d.).

(3), the Scrum master. The Scrum master is the glue that holds all the involved parties of Scrum together, ensuring the Scrum process runs smoothly and its principles are adhered to. The Scrum master is also responsible for administrative tasks, such as arranging meetings, and in these meetings the Scrum master will usually be the most prominent voice of the Scrum team. The Scrum master assists the product owner in communicating value, which are then relayed to the development team so that a firm understanding of the product owner's vision of a quality product is achieved. It is fitting to describe the Scrum master as the bridge between the development team and the product owner to arrive at efficient co-operation and understanding (West, n.d.).

2.9 Extent of involvement in the development process

In this short section, we will further elaborate on the rationale behind how we have assigned specific roles to the team members. The team members had expressed a wish for everyone to be involved in the development work for the purpose of a quality learning outcome, while also considering a team member's strengths and weaknesses.

We had therefore decided that a team member may work wherever their strengths may return the most productivity, but there must be a minimum involvement in all aspects of the development work. This is so everyone involved has a base understanding of the system as a whole. Therefore, every team member was a part of the development team, including the Scrum master, the Scrum methodology will not hold back a Scrum master also assuming the role of a developer. It did not compromise on the Scrum master's commitment to their primary role, and neither did we see it as a violation of Scrum's principle of commitment (Konduru, n.d.). The product owner's primary duties, as has been described, and involvement in other projects would make them an inappropriate participant of the development process, ergo their exclusion from it (Nettleton, 2022).

3. Planning and execution

In this chapter, we will describe the initial experiences of the project, the stages following it and how it was all conducted with regards to project management, the agile methodology, and Design process.

3.1 Initial stages

The project was initiated in the start of January of 2022. Although by late January we had received the documentation of central requirements from the product owner. We were still left in the dark with regards to details and other aspects, such as desired design of the system and other elements. Consequently, we did not feel we were in the right place to officially initiate the Scrum process, as we would not know what items would be appropriate to add in the Backlog. Therefore, the initial weeks were occupied by dialogue with the product owner to further explain details. Thereby further delaying the initial start of the project. With these issues mostly resolved, the Sprint cycle was inaugurated with the beginning of the first Sprint on the 19th of February 2022.

3.2 Requirements

In order to develop a system, it is required to have both functional and non-functional requirements. Functional requirements define how the system needs to work, and non-functional requirements describe how the user experience of the system should be like. (Pavel Gorbachenko, 2021). Without functional requirements, the system will not work. Users might not be satisfied if at least some non-functional requirements are not met to some extent.

In order to create an overview of the desired system. We sat down together with our product owner present and discussed both the functional and nonfunctional requirements needed for this system. It is important to document clearly the functional and non-functional requirements for the following reasons.

Firstly, to avoid misunderstandings between the parties. Therefore, we made sure Red Rock AS was present when we documented and determined the requirements of the system. If the requirements are not documented well, it could cause large problems for the project down the line. The developers may not have a clear understanding of what is expected of them, and poorly documented requirements will most likely cause the project to take much longer than expected to complete. In the worst case, this can lead to the project needing to be restarted and scrapped entirely. Further are some of the important functional and non-functional requirements we arrived at.

Functional	Non-Functional
Retrieve Data	Security
Edit Data	Modifiability
Add new Jobs	Usability
Sort Jobs	Visibility
Change order off Jobs	Affordance

Table 3. 1 System Requirements

Retrieve Data

One of the functional requirements we decided upon was to be able to retrieve data from the external source into our planning system. The data which was provided to us by Red Rock is now being stored in a local database set up by ourselves. In order to retrieve it we used API (Application programming interface) which is a software interface that allows two applications to interact with each other without any user intervention (Walker, 2022)

Edit Data

Being able to edit and manipulate the plan orders was also a crucial requirement for this system. If something occurs unexpectedly the user must have the ability to make edits and change the plan orders. In order to meet these requirements, we created a button called “edit” on each plan order, which gives the user the possibility to make edits according to your needs.

Add new Jobs

Another requirement given by Red Rock AS was the possibility to add new jobs in addition to the already existing ones. We not only created an option for adding new jobs, but deleting existing ones as well, the reason being if a specific container was for some reason to be held back. The system should have a function allowing the user to delete that specific instance from the plan order. And when this spot is freed up, you would also have the possibility to add a new one instead. As shown in [figure 3.8](#).

Sort Job

The System should be able to sort the jobs based on the different values they have. An important requirement which amplifies the usability of the system. This requirement was met by adding “sort” function on the front end of the system. As shown in [figure 3.6](#).

Change order of Jobs

It must be possible to adjust the plan by changing the order of the jobs. If one job was delayed and needed to be rescheduled, we needed to create a function letting the user adjust the order of the jobs. In order to fulfil this requirement, we created an edit button in the order section allowing the user to change the order and values of the job itself.

3.3 Dataset and Database design

For this project we received a small sample dataset from the product owner. The dataset was a fictional dataset representing how a real dataset roughly would look like. The dataset contained all the attributes which were necessary, such as *ContainerID*, *MoveTo*, *Stow* etc. During the project this was the only dataset we had to work with, which made it somewhat easy for us to comprehend considering the size of the dataset. We also saved valuable time by not having the necessity to gather the data ourselves for this project.

Based on the data from our dataset, we now needed to create and form a structure for our database. We wanted to generate a visual representation of the database. This would help us to identify the systems' elements and attributes as well as their relationships with each other. The tools we utilized to create an overview of our database was an ER diagram. Entity relationship diagrams provide a visual starting point for database design that can also be used to help determine information system requirements throughout an organization (Biscobing, 2019)

During the process we actively had to consult with our product owner. Due to this being a crucial step in the system's development, we wanted to receive as much feedback as possible from the product owner, and as such we saved ourselves from unnecessary waste of time going back and correcting elements of the system. After important dialogues and advising with our product owner, we arrived at a mutual understanding on how the database should be structured.

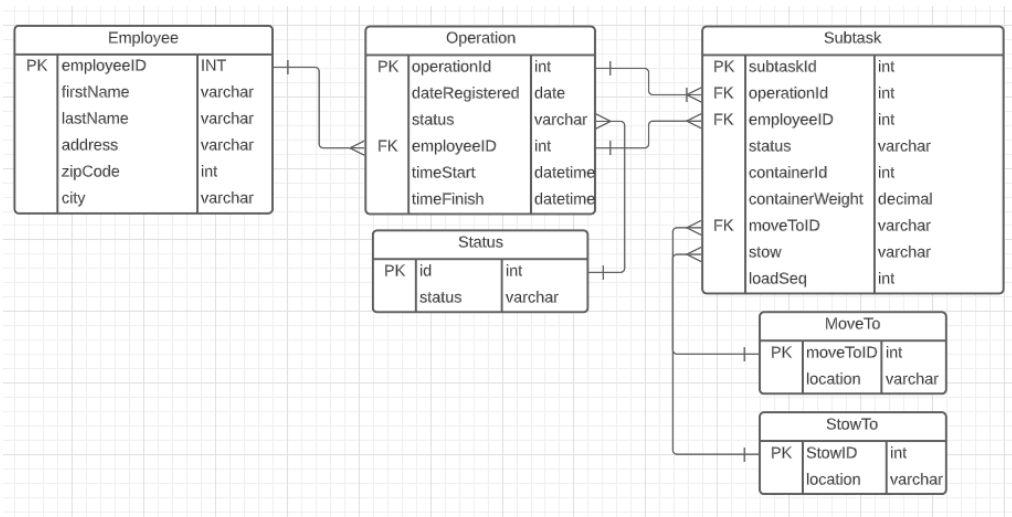


Figure 3. 1 Database (ER diagram)

3.4 UX-design

A system interface is what the user uses to interact with a system. It is therefore important that the interface is comprehensible to the user, and the designer needs to keep the end user in mind when designing a system (Benyon, 2019, p. 12). In the task given to us by the product owner, the only requirements regarding the design of the system were that it needed to be functional and display the correct amount of data and metadata in an organized way. Critical design decisions, such as the layout of the system, were discussed, and decided upon together with the product owner. Other than that, we were free to design the system as we wanted.

For the colour scheme of the design, we took inspiration from Red Rock's logo and their website. Mainly using red, white, grey and black to match their logo and have a recurring theme in the design. Keeping the design consistent was something we viewed as important to achieve an overall professional look for the system.

3.4.1 Personas

Personas are concrete representations of users for a system that is being developed. Designers create and use personas to get a better understanding of the people who are going to use and interact with the system (Benyon, 2019, p. 55). When creating personas it is important to keep in mind who the end user is. Contrary to target groups that mostly focus on age, location or role of the user, personas aim at understanding the thoughts and needs of the user (Bertelsen, 2021). For this project we imagine the target user of the system to be a dock worker, using the system to look at or edit the load sequence of containers. We imagine that the computer skill of users wearies from basic to high. We have created the following persona to represent a potential user of the system.

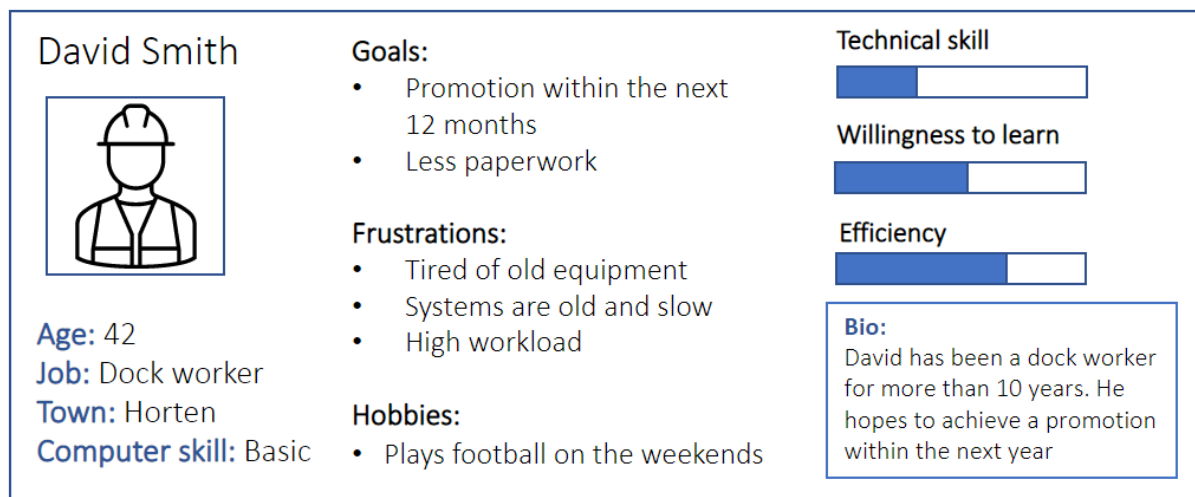


Figure 3. 2 Persona

3.4.2 Usability

Usability in design aims at making a human-centred approach, and has the following characteristics: efficiency, easy to learn, safe to operate and have high utility (Benyon, 2019, p. 108). If the user is not able to interact or achieve their goals with the system in an efficient and satisfactory way, then they might look for other alternatives (Komninos, 2021). When designing this system our main priority was achieving a high level of usability. This is important because the system is meant to be used by people with a warring degree of computer knowledge.

A large and important part of usability is efficiency. It concerns the user's ability to complete their goals quickly with a prominent level of accuracy. A substantial portion of efficiency in a system comes from the support provided to the user. Having all available functions easily visible and accessible for the user to interact with may add to the system's level of efficiency, and therefore increase usability (Komninos, 2021). An example of efficiency in design is the auto fill function when the user is filling out a form. For this project, our system needed to have an efficient way to create and add a new subtask to the jobs list because this is one of the main functions of the system. The user simply clicks on the add button, fills out the needed and lastly clicks submit.

3.4.3 Wireframes and mockups

Wireframes are a key technique and help designer plan and test out the outline or structure of a software system. They focus on the interaction design and the information architecture of a product (Benyon, 2019, p. 194). Information architecture concerns how content is classified and organized (Benyon, 2019, p. 348). For this project wireframes were used at the beginning of the design phase. They were created together with the product owner during one of the

early meetings. Using a whiteboard, we drew up wireframes for the systems. The product owner also gave examples of designs that would work well with our system. These examples were of other systems with similar design and functions.



Figure 3. 3 Initial wireframe created together with the product owner

The initial wireframes were seen as more of a guide to help us get ideas of how the layout of the system would look like. After the session with the product owner, the group came together and reworked and improved upon the initial wireframes. This enabled us to have a better framework to base our design on.

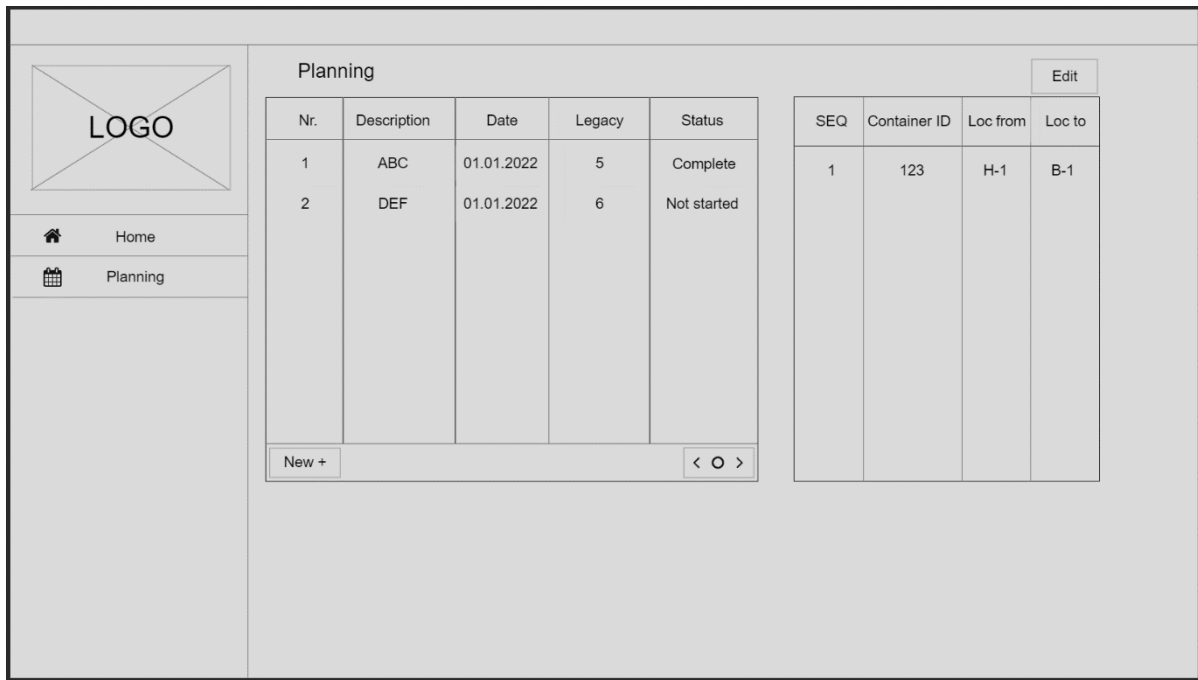


Figure 3. 4 Earliest wireframe

After completing the wireframes, the group created mock-ups of the system using a program called Miro. During this design session central design decisions were made, and the group got an overall understanding of what would be used in the final product. Mock-ups are static, high-fidelity representation of the final product. This includes typography, icons, colours, and overall style. Where a prototype focuses on the interaction design, mock-ups are more concerned with how the user will react to the visual identity and overall style of the system (Hannah, 2021). Some minor changes were made from mock-ups to the final design, however these were small changes.

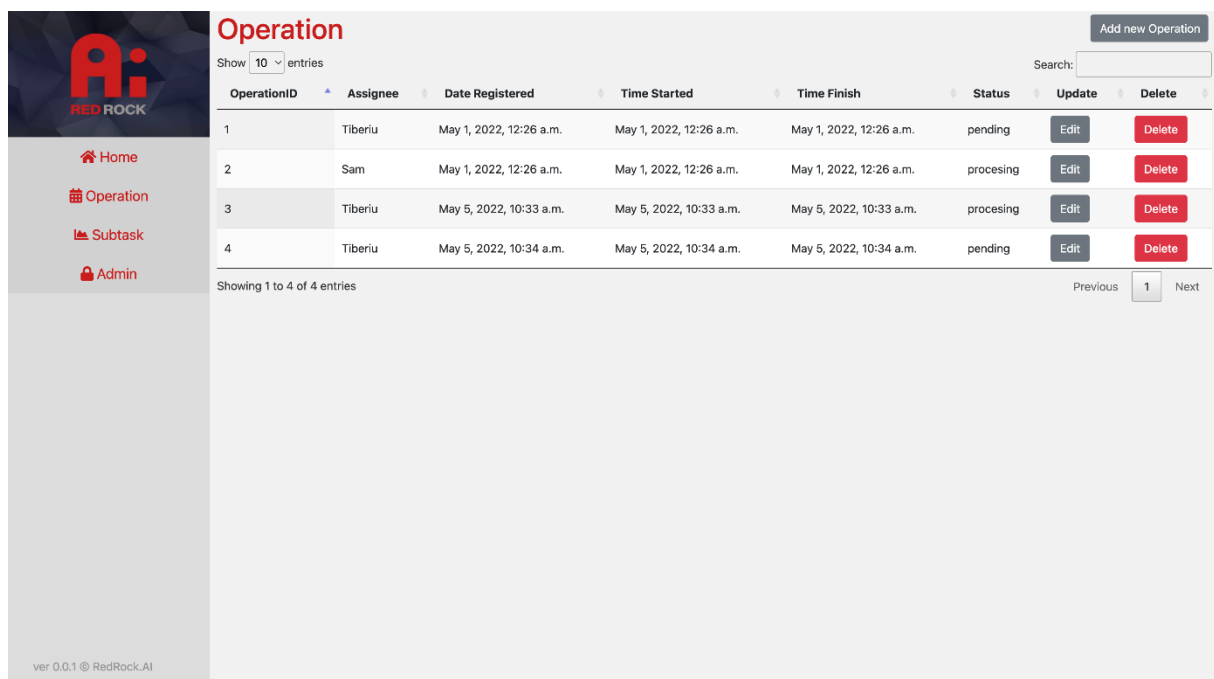


Figure 3. 5 Mock-up of the system.

3.4.4 Benyon's design principles

Benyon's design principles can be used by designers to guide them in the design process and to evaluate and critique prototype design ideas. The 12 design principles are: Visibility, consistency, familiarity, affordance, navigation, control, feedback, recovery, constraints, flexibility, style, and conviviality (Benyon, 2019, s. 116). For this project we have made an effort to consider all of the design principles mentioned. As this system is foremost meant to be functional and professional, some of the principles were more relevant to the system than others.

The visibility principle concerns the visibility of functions in a system. It focuses on ensuring that people can see and are aware of all the available functions in the system they are using (Benyon, 2019, s. 117). For this project we have prioritized designing the system in a way that makes it easy for the end user to understand and interact with. We felt it was important to have all the available functions visible for the user to see and easily interact with. On the operation page the user can choose how many entities are displayed at a time. Giving the user the ability to choose the level of visibility of jobs. The picture below shows how the different functions are visible for the user.



The screenshot displays the 'Operation' management page. On the left is a sidebar with a 'RED ROCK' logo and navigation links: Home, Operation (selected), Subtask, and Admin. The main content area features a table of operations. Above the table, there is a search bar, a dropdown for 'Show 10 entries', and an 'Add new Operation' button. The table has columns for OperationID, Assignee, Date Registered, Time Started, Time Finish, Status, and actions (Update, Delete). The data shows four operations with statuses 'pending' and 'procesing'.

OperationID	Assignee	Date Registered	Time Started	Time Finish	Status	Update	Delete
1	Tiberiu	May 1, 2022, 12:26 a.m.	May 1, 2022, 12:26 a.m.	May 1, 2022, 12:26 a.m.	pending	Edit	Delete
2	Sam	May 1, 2022, 12:26 a.m.	May 1, 2022, 12:26 a.m.	May 1, 2022, 12:26 a.m.	procesing	Edit	Delete
3	Tiberiu	May 5, 2022, 10:33 a.m.	May 5, 2022, 10:33 a.m.	May 5, 2022, 10:33 a.m.	procesing	Edit	Delete
4	Tiberiu	May 5, 2022, 10:34 a.m.	May 5, 2022, 10:34 a.m.	May 5, 2022, 10:34 a.m.	pending	Edit	Delete

Showing 1 to 4 of 4 entries

Previous 1 Next

ver 0.0.1 © RedRock.AI

Figure 3. 6 Example of visibility

Consistency in design means having uniformity in both language and the design features. Having a set standard for working and doing tasks in the system can improve consistency (Benyon, 2019, s. 117). Having consistency in design helps to improve learnability in the

design. This means people can transfer knowledge to different elements in the system. And therefore, learn new functions faster (Nikolov, 2017). For this project having a consistent design is something we view as important for the users' experience when interacting with the system. To achieve this in the system, all the pages have a similar structure and layout, as well as a consistent colour scheme.

Familiarity means using language and symbols that are familiar to the user. This might make navigating and interacting with the system easier and more efficient for the user (Benyon, 2019, s. 117). For this system we have chosen to use icons and functions that are commonly used in other systems. For example, on the navigation bar on the left side of the page there is an icon of a house to represent the home page. Clicking on this will take the user to the home page. One example of a common function used in the system is the dropdown function. When clicking on this function the user will be presented with a dropdown bar, where they can choose the desired input. In the picture below both the dropdown function and the navigation bar on the left side are visible.

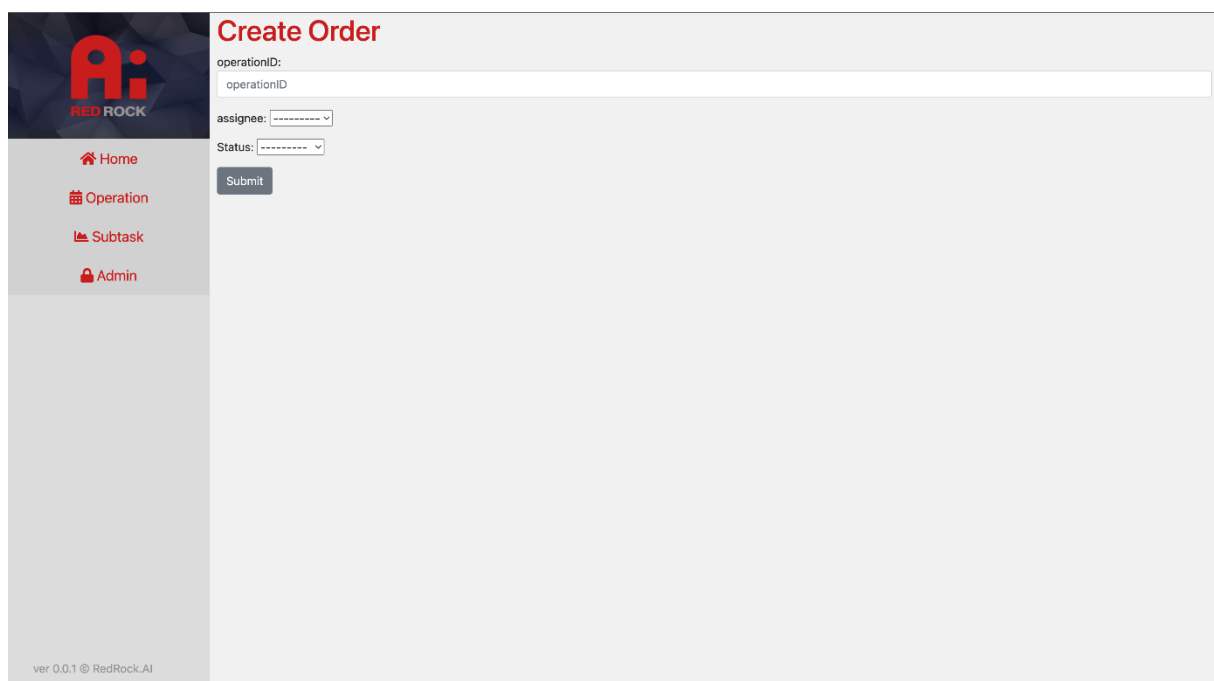
The image shows a web application interface for 'RED ROCK AI'. On the left is a dark sidebar with a logo at the top and four navigation links: 'Home' (house icon), 'Operation' (calendar icon), 'Subtask' (list icon), and 'Admin' (lock icon). The main content area is titled 'Create Order' in red. It contains three input fields: 'operationID:' with a text box, 'assignee:' with a dropdown menu, and 'Status:' with a dropdown menu. Below these is a grey 'Submit' button. At the bottom left of the sidebar, it says 'ver 0.0.1 © RedRock.AI'.

Figure 3. 7 Example of familiarity

Affordance refers to designing objects clearly for the user, so they understand what the thing does and its purpose. For example, designing a button to look like a button (Benyon, 2019, s. 117). In this project we have made an effort to design functions as clearly as possible. One example of this is the edit and delete buttons on the subtask site. They are rectangular buttons

with the name of their functions written on them. We believe this gives clarity to the user and therefore increases the degree of affordance in the systems design.

Navigation entails providing sufficient support to the user to enable them to navigate the system freely. This can be done by adding maps, directional signs, and information signs (Benyon, 2019, s. 117). As mentioned earlier the system has a navigation bar on the left side of the page. This bar enables the user to flawlessly navigate to different pages of the system with just a click of a button. For example, the user can navigate from the home page to the planning page by clicking on the planning button.

The control principle aims at allowing the user to take control. However, it should be clear to the user what they can and cannot control. It should also be easily understandable to the user what the relationship between what the system does and what will happen outside of the system (Benyon, 2019, s. 117) For this system, admin users can import and export larger amounts of data, than a normal user can. A normal user can only add one “job” at a time.

The feedback principle refers to giving the user feedback information from the system to let them know the effects of their actions. Rapid, constant, and consistent feedback will enhance the users feeling of control (Benyon, 2019, s. 117). Due to lack of time this principle was not implemented into the system to the degree we would have wanted. If given more time we would have liked to have added a feedback message when the user made a change to the jobs list.

Recovery allows the user to quickly and efficiently recover from mistakes and errors made. Providing warnings for drastic actions, such as “Are you sure you want to delete this item?.” may prevent the user from making unwanted mistakes. If a user has already made a mistake, an undo button can provide the user with quick and easy recovery (Benyon, 2019, ss. 117-120). For this system there is no direct undo button, however the user can always edit or delete items in the list. If given more time adding an undo button is something that would give the system a higher level of recovery and may therefore enhance user experience.

Providing constraints are so that users are unable to do things that are inappropriate and may potentially harm the system. Preventing people from making serious mistakes through proper constraints will enhance the system's level of accessibility and security (Benyon, 2019, s. 117). In this system we have provided constraints in the form of only allowing the user to input the correct type of data. One example of this is when the user is adding a new subtask to

the list, they must input where the container is to be moved to, and the user is then faced with a dropdown menu which only allows them to input from a select set of data.

Add new Subtask

subtaskID:

operation ID:

containerID:

containerWeightT:

loadSeq:

moveTo:

stow: ☒

status:

HT-A1
HT-A2
HT-A3
HT-A4
HT-B1
HT-B2
HT-B3
HT-B4
HT-C1
HT-C2
HT-C3
HT-C4
HT-D1
HT-D2
HT-D3
HT-D4

ver 0.0.1 © RedRock.AI

Figure 3. 8 Example of constraints

Flexibility means allowing the user to have multiple ways of doing things. Accommodating people with various levels of experience with the system. Providing the user with the ability to change the way things look or behave will enhance the user's experience when interacting with the system (Benyon, 2019, s. 117). One way flexibility is implemented in the system is with the search bar as seen in the picture below. The user can choose to scroll through the list of jobs or search for the specific job wanted. As this is a system with a straightforward set of functions that are available to the user, the need for numerous ways of completing tasks was something we viewed as less important. However, if we were given more time to develop the system, we would like to develop additional ways to complete tasks. And thereby adding to flexibility.

LoadSeq	Status	Stow	MoveTo	Assignee	ID	ContainerID	Weight
1	finished	H1-D1	B1-D4	Tiberiu	11	CONTAINER1124	25.00
5	finished	H1-A3	B1-A3	Tiberiu	15	CONTAINER1128	26.30
7	finished	H1-B1	B1-B1	Tiberiu	17	CONTAINER1130	20.00
8	finished	H1-D3	B1-D3	Tiberiu	18	CONTAINER1131	39.00
9	finished	H1-D2	B1-D2	Tiberiu	19	CONTAINER1132	26.30
10	finished	H1-A1	B1-A1	Tiberiu	20	CONTAINER1134	26.30

Figure 3. 9 Example of flexibility

The style principle refers to designing a system to look stylish and attractive to the user (Benyon, 2019, s. 117). There were no requirements towards the style of the system from the product owner. We made an attempted to design a system that was first of all functional, but also nice to look at. The system has a consistent colour scheme and a simplistic, yet professional look.

Conviviality aims at creating a system that is polite, friendly, and generally pleasant. Operating within the standard design guidelines may elevate conviviality in the system design to some degree. However, one area where designers are freer to influence conviviality is error messages. These types of messages are meant to be visible and provide the user with information. Therefore, the language used should be polite and to the point (Benyon, 2019, ss. 118-122). As there is limited communication between the user and the system. Conviviality was therefore implemented in this system to a lesser degree than some other principles.

3.5 Sprint retrospective

As for the Sprint retrospectives, we used the Four Ls retrospective idea. It is a straightforward technique to highlight the ups and downs of a former Sprint, from an objective and emotional angle of approach (TeamRetro, n.d.). The format is as such; Liked; this is where we recall the positive experiences of the former Sprint. Learned; as the name says, this is where we detail what we have learned. Lacked; here we reiterate what we feel was missing from the former

Sprint, which could be for example lacking Scrum implementation. And finally, Longed for; what we longed for in the former Sprint so that we could be more successful in our project endeavours, an example of this could be additional learning resources. For an overview of all Sprint retrospectives, refer to [appendix 16](#).

3.6 Development process

In this section we will be looking at how we managed the project during the development process. Covering the vital possesses needed to develop our project. Largely consisting of the planning process, the problem analysis, and the approach we took regarding building the groundwork for the project. In addition, we will be covering how we managed the project, what methods we used, and the scope of the development process.

3.6.1 Project qualifications

When we first started the project, we were uncertain about how to approach the project. Due to a lack of specification, we had a bit of a hard time understanding exactly what the project was supposed to look like. However, after some communication with the project owners we had a good idea of what the project was going to look like. At this point we were brainstorming various ideas of how we wanted to approach the project. We were considering what elements we wanted to include and created some estimates of how long it would take to realize it.

3.6.2 Initial planning

Once we had a good understanding of the qualifications and scope of the project, one of the first things we decided on was what kind of project management structure we were going to use. As previously mentioned, we decided to go with scrum, seeing how we have previously used this method in other projects. In accordance with the scrum framework, we decided to write down what we believed to be all the vital elements of the project. After this we split them up and put them into sprints, each having a certain number of tasks we wanted done by the set deadline.

We tracked the scrum process using Jira, which is an online project management program. Here everyone had access to see what tasks were currently being worked on, add new tasks, and change details of the various sprints and tasks.

Another tool used for planning was Miro, we used this to efficiently do work breakdown structure. Here we laid out all the various aspects of the development process that we wanted

to focus on as seen in the picture below. We chose to focus on five aspects: iteration, planning, execution, control, and closure.

3.6.3 Technology used

Aside from using Jira as the planning system, we used other technologies to produce the final product for Red Rock AS. Working with the frontend part of the project, we went with the standard frontend stack of HTML, Bootstrap/CSS, and JavaScript. Using this stack benefited us immensely in terms of constructing a lightweight front end with interactable buttons and typable fields in the tables. Creating mock-ups and wireframes was crucial in terms of visualizing how the application would look from a frontend perspective. To create these, we used UXPin and Miro, which serve as designing tools that can be used for developing and testing UI design. Additionally for the backend of the application we used the Python web framework Django, and its packages of reusable apps. For the database we chose MySQL database due to its convenience of use and almost unlimited resources about it on the web. To implement micro-service architecture for the application, Docker along with its Compose tool was used to serve this purpose.

3.6.4 Django and Docker

Django

Working with Django was a new experience for all of us since no one had prior experience with it. To get started with this framework, we occupied ourselves learning and studying how Django works. Udemy and YouTube benefited us greatly in the learning process. Django immensely leverages the possibility of importing and utilizing apps from its project's library. The entire development process gets uncomplicated due to its ease of for example importing an already made table or admin page. Django also reduces the process of executing queries in the database by utilizing the database-abstraction API which allows for to creation, retrieve, update and delete objects (Django, n.d). This API is instantiated when the data models or models.py file is created. Working with the application, we first managed to create a subtask page containing the subtask table from the database.

The data came from the models.py subtask class along with the views.py file which was building up the database by fetching the entire database. By using these two files we were able at the same time to save data, provide it to the tables or different functions throughout the application needing it and increase cohesion by having each python file assigned to one role instead of many causing reductions in cohesion. This process was repeated for all data

needing to be displayed in a table on the browser. Configuration of a database was already done in the settings.py file, which serves as an admin to all files in the application directory. Settings.py file dictates which apps are currently installed, and what database is connected and ready to be used. Django also has HTML prepared for the frontend part through the template folder. In the same folder, all HTML outputs are stored and visible on the browser when the application is started.

```
class Subtask(models.Model): # Subtask table
    subtaskID = models.IntegerField(primary_key=True, auto_created=True)
    assignee = models.ForeignKey('Employee', on_delete=models.CASCADE)
    containerID = models.CharField(max_length=255)
    containerWeightT = models.DecimalField(max_digits=6, decimal_places=2)
    loadSeq = models.IntegerField()
    moveTo = models.ForeignKey('MoveTo', on_delete=models.CASCADE)
    stow = models.ForeignKey('Stow', on_delete=models.CASCADE)
    status = models.ForeignKey(
        'Status', on_delete=models.CASCADE)
```

Figure 3. 10 Display of the Subtask class from models.py file

Docker

We had prior experience working with Docker in another course. Even though we had worked with Docker before, our knowledge of it was not of the standard needed to establish a micro-service architecture, which essentially is breaking down an application into smaller parts with their responsibilities (Google, n.d). This can be attained by using Docker containers when multiple of them are run together. After some research on how we could implement a way to run multiple containers simultaneously, we came upon Compose, which is a Docker tool allowing you to do the exact, run of multiple containers at the same time. Therefore, it was decided between us to utilize this tool with Docker. Next would be to implement the architecture Red Rock AS desired. Which would be most optimal separating the backend and frontend into their own containers on a common network.

This is achievable by creating a web app and database service and a network configuration in the docker-compose.yml file. Before implementing a docker-compose.yml file, we first needed to build docker container images. These docker images are lightweight executable packages of software including everything needed to run an application, this ranges from the source code to the libraries or apps used in an application (Docker, n.d.). In our docker-compose.yml file as seen in [appendix 6](#), we have two services, Django which is the web app with container name, Django, and db standing for the database with container name, mysqldb. Lastly, we have configured a common network named, application. For the Django container,

we assigned docker to create an image of the entire application including its source code and apps in use. For the mysqldb container, we pulled an online docker image called “mysql”. Our mysqldb container would run its own MySQL server, meaning it can be accessed from the terminal and do changes with the database if wanted. We then managed to get the Django and mysqldb container dependent on each other, meaning if mysqldb container is not running and Django is, then the entire application will not work either. Lastly, we assigned the application network to both. Instead of running each container independently, we only need to enter “docker-compose up” if it is running for the first time, it will install all images, containers and run pip requirements.txt file, which contains all apps and versions needed to run the application. After everything is installed, it will then run all the containers simultaneously and achieve the goal of establishing a microservice architecture of the application Red Rock AS gave us. As illustrated in [appendix 7](#)

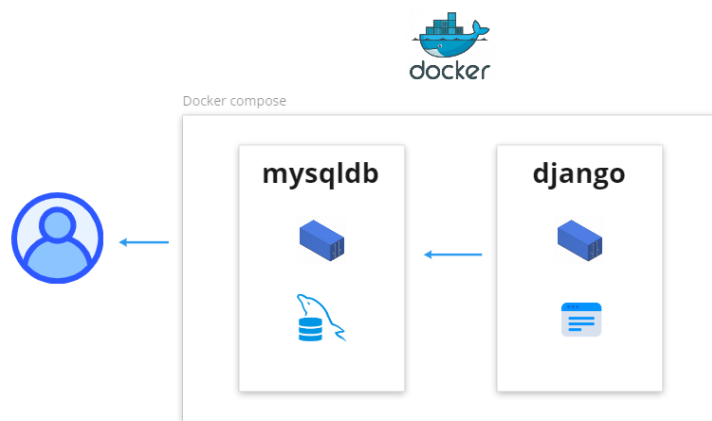


Figure 3. 11 Simplified illustration on how application runs in docker compose

3.6.5 Programming process

Review of code

In order to have an organized and structured source code of the application when changes are being implemented or requested, we decided to create a procedure on how the review of these code changes would be conducted. On every pull request, the developer must include a heading and detailed description of what changes are waiting to be merged into the source code on the main branch. One developer was in charge of reviewing the code from each pull request a peer created.

We used Github Desktop to work with the branches and create pull requests. When the assigned developer was reviewing the code from the pull request, he first checked if there would be any conflicts of files if a merge was to occur, and secondly have the coding standards we set as a group had been followed? If these problems were not solved, the developer would give feedback on what's missing and needs to be corrected.

Design pattern

Since the application is built with the Django framework, it is mainly based on the Model-View-Template software design pattern. Model-View-Template or also known as MVT compromises three important components Model, View, and Template (Javapoint, 2022).

Working with the application, our model component is represented in the models.py file in the project folder. We have different classes in the models.py file linked to tables in the database, these are Subtask(models.model), Operation(models.model), Employee(models.Model) and Status(models.Model) classes as shown in [appendix 4 and 5](#). Its objective is to map every single table in the database and allow the view component to fetch, apply the business logic and send it further to the various HTML outputs in the template component. With the Django framework comes the database API to which the classes in models.py are connected, and this provides a pathway when it comes to retrieving, deleting, updating, and deleting a record from a table. Since our goal in the development has been good internal and external quality, following this design pattern reduces the coupling capabilities as well as benefits the organizational and structural architecture of the entire application. Instead of creating for example a file with classes where both query and business logic is present, it has its file with classes doing the exact process more structured, and in turn, it increases the cohesion of the class and overall application.

The view component in our application is present in the views.py file. The overall business logic of our backend is applied in the views.py file as seen on [appendix 13 and 14](#). Output the template receives comes from the views.py file in our application. As an example, we have a table on the home page displaying an ordinary table revealing which subtask has not yet been started, this data comes viewable due to it being applied in a code snippet in the views.py file. Instead of being a class, it's rather a function executing the request, displayIndex(request). The template component can be recognized as our front end of the application. We have a template directory where all dynamic HTML outputs are stored and used when a user requests them. We write these files in plain HTML, and to display data, we use python in curly

brackets as shown in [appendix 3](#). These files include those containing subtask and operation tables.

Version control

We used Github along with its Desktop app as our version control in the development process. Our experiences with Github as an efficient tool for tracking and updating changes have always been positive, hence why we chose it as version control. GitHub desktop also reduces the need to enter git commands in the terminal. In the group we have those with both laptop and desktop computers, for them, Github Desktop was optimal due to its convenience of creating branches with one click for each member and being easy to use across different computers. To showcase how this process is done, you commit a change and push it to your branch on a desktop computer, after that you pull that update to the repository folder on your laptop. [Appendix 8](#) displays a brief view of this process. This ease and simplicity made the entire development process much easier as well as tracking and updating changes. When a pull request from a member branch to the main branch is created, as mentioned earlier it must be detailed or it will be discarded. This track of record procedure enabled us to efficiently use this tool.

Testing of application

For testing, we used Django frameworks own unit test for both the front end and the back end. The framework is a Python subclass called unit test. For running tests in Django, we usually have a “test.py” file where we write the code to test different parts of the system. To create a much cleaner look and tidy test files we decided to make a test folder that contains all the test files.

We ran tests on views, URLs, models, and forms, and for each one, we generated a test file containing the code for the individual tests we wanted to run. Then we used the command "python manage.py test myproject" to run the test in a pip environment that is part of Django. Then Django runs the test code, and if an error occurs, you'll see an error message stating the number of errors and the location of the error. However, if everything is fine, you will know the number and time of tests Django ran, and that the test was successful.

As mentioned earlier, we ran four tests, but now we'll better explain the functionality of each test. "URLS" is a backend test that tests if the connection between the URL and the view is established and if there is a problem with the connection. To make the test work, create a class that defines each URL in your project, and in each definition select the URL and the view

associated with the URL test and the Django test. The view test iterates through different views and definitions to create a client that checks the status code of different views. As an example, a page may receive data in a table. The Django test `status_code` is 200, this means that the page loads successfully and is okay. This is both a front-end test and a back-end test because both parts are tested.

For the model, we test the backend of your project. Here we will test all the functions and labels in the `models.py` file. As an example, if you have a function that creates a name, you can write code to test that function and display an error message if something goes wrong. The form test also has a front-end test and a back-end test to test how Django retrieves the data. When writing code, put the data in the code to see how the code handles incorrect data and empty fields, and whether the user sees the error needed to complete the task.

4.0 The project

The following video is a practical presentation of our finished project:

<https://www.youtube.com/watch?v=jgkjNvX-X6I>

5.0 Reflections and challenges

In this chapter, we will reflect on the overall process of the project. We will go into detail of what went well and what turned out to be more of a challenge, from having to cancel physical meetings due to COVID-19 to struggles with time management. However, these challenges were something we as a group had been able to overcome to some degree, or we were able to work around them and learned from the experiences, as we will detail further in this chapter.

5.1 Project management

As mentioned previously, at the beginning of the semester we ran into some problems with the specifications of the project. Largely due to a misunderstanding regarding the project qualifications. This caused us to spend some time attempting to figure out what the project owner wanted, however because the specification used a lot of terminology, we were far from familiar with. It caused us to be unable to get much work done during the initial few weeks, we resolved this however by consulting with our company representative and getting a more refined specification of what the project was supposed to look like, although it did not take little time to have arranged, further adding onto delays.

Once we had a good understanding of what the product was supposed to look like, we began to approach how we were going to go about doing so. When we first started out judging the

scope of the project. We were throwing around a lot of ideas about how we were going to approach the project. What had to be in the product and what was potentially excludable. After a few days of considering this, we finally ended up with the product backlog, which we later on would implement into the scrum framework we intended on using.

5.1.1 Scrum in practice

Throughout this project we have aspired to follow the Scrum framework. Overall, this has been a good boon for the development of the project. We have previously used Scrum to various effects in the past, and so we were at least familiar with the core concepts of the framework. However, every project is different, and despite using the same framework as before, new issues and problems will arise.

In the case of this project, it came in the form of the scope of the project. In the past we have been using Scrum while working on smaller projects. This has come with its own share of problems, however due to the lower scope of the projects, the problems didn't cause too much disruption overall. In this project it was a lot more noticeable whenever things that were scheduled fell through, such as not finishing certain Backlog items of the Sprints on time, or not being able to hold important meetings due to scheduling errors. We would in the future strongly recommend that scheduling must take place as early as possible without delay, so that all involved parties may have the opportunity to plan accordingly.

Something else we learned along the way was how important proper Sprint planning was, initially we spent little time on planning the Sprints, which was especially noticeable in the first Sprint. However, as the project went on, we realized that we had to create a more refined product backlog, seeing how we were sometimes unable to finish the given tasks or would have little idea on how to proceed with development. This was alleviated by consulting Red Rock AS for advice on how to further flesh out a detailed Backlog, seeing as they have a great deal of experience.

In summary, enforcement of Scrum has not been a breeze, and it proved to be difficult even for the Scrum master. We have experienced that a Scrum master alone, even if they should be the most knowledgeable on Scrum, cannot guarantee or force a good Scrum performance if there is little initiative in the team. There may be several factors at play here, and we believe the most substantial to be incompatible personalities, and by that we mean personalities that are to a lesser degree compatible with a methodological and organised approach to work, if that should be Scrum, Kanban, Waterfall, and so on. Another factor may be the sentiment of

involuntary or forced participation, which is experienced more as “this must be done” just for the sake of doing it rather than “this should be done” for the sake of excellent quality, which we experienced as not being conducive to passionate and motivated work. This has been expressed more or less by all team members, and as reiterated, a pessimist mindset does not couple well with optimally productive work with a concern for quality assurance. Regrettably, this is to a degree reflected in the quality of the final product.

We believe that despite many of the problems we encountered for the duration of the project, the Scum methodology was reasonably helpful in keeping the project on track during our time working on it. While we did without a doubt come across numerous issues, some greater than others, and acknowledge that it has impacted the quality of the final product, they were never insurmountable to such a degree that we were unable to overcome them.

5.1.2 Communication

At the start of the project there were still a lot of restrictions as a result of the pandemic. Because of this most of our meetings as well as work were done digitally from home. This had somewhat of a negative effect on communication and workflow at times. Not being able to ask direct questions or have a face-to-face conversation with group members or others connected to the project put a damper on communication at times. One of the major issues we came across was having to wait for responses to questions regarding the system. These questions were sent either by e-mail or through Discord. Having to wait for responses to crucial system questions meant that we had to focus our attention on other issues before being able to work on the system.

With the group itself we also had a fair share of communication problems. Seeing how meeting in person was not optimal, both with restrictions being tightened again at the time, and the fact that many of the group members lived far away from the university. We made the choice to mostly hold digital meetings over discord. Or generally just communicate through chatting features. This worked fine for the most part, however some problems did arise when it came to synchronizing our work. At some points we were working on the same tasks unknowing of the fact that another person was already doing it or had already done it. This happened a few times throughout the project. However, this is largely a management and work allocation problem, rather than a problem caused by having to communicate digitally.

We got a lot better at arranging meetings as the project went along. In the beginning we would usually not meet too often. And when we were meeting it was often a bit unfocused

and unclear where we were going and what needed to be done. Naturally this is not unusual for the start of a project like this. However, the structure of the meetings we were holding later on in the project were remarkably better and would often be shorter seeing how we were more focused on what needed to be done.

5.1.3 Task allocation

As a result of being six people working on one bachelor project, we had a somewhat easy time dividing up what tasks needed to be done, and who should do them. Seeing how some group members were more experienced at certain aspects like programming, we decided to split up the tasks accordingly. Although initially there was the motivation to further increase in our programming competence, the allocation of work gradually assumed a different path than intended to the point where some of us were exclusively managing the project, doing the documentation of the project, while others were developing, programming, testing and so on. In retrospect, it would probably have been beneficial to have been more involved in the development process, seeing as how it towards the end, there was a lack of insight and knowledge with regards to the other parts of the project, despite this working fine for the most part. It did lead to some group members not having any work to do at certain points during the work process, such as when we were finished with a writing assignment, and there was of course always developing which needed to be done, however, since those who assumed the role of managing and documenting were not updated on beyond surface-level details where the development currently was and where it was going, it proved to be difficult to catch up and be able to contribute positively, especially at such a late stage in the project.

While it might have been beneficial for the timely completion of the project. Overall, if we could do it again, we would attempt to have everyone be involved directly in the programming during the entire stage of the project. We would make sure that every member of the group had some tasks at every stage of the development process of the project. And only once everyone had caught up, we would proceed to the next stage.

5.2 Quality of end product

Within the scope of the project, we believe that the end product is one of sufficient quality. As was determined from the start, we intended on making a high-fidelity prototype of what could later be developed into a fully-realized planning system. This is largely because making an actual functional planning system, that can communicate with a larger technological infrastructure is simply out of our reach, both in time, resources, and would likely require

more competence than we our current skillset was capable of. As a result, the system does not feature every element it would have, given we had more time and resources.

However, within the boundaries of the project we decided on at the beginning, we feel the end result is quite good. It has all the functions we scheduled for it to include, and after showcasing it for Red Rock AS several times along the development process, we have progressively made more iterations on the project after receiving their feedback together with positive comments. One of the other aspects we considered was Inclusivity, seeing how the system is likely going to be used by several different people. It was important to us that we made sure the system fell in line with current Web Accessibility Guidelines, also known as WCAG. We tried to fit within these guidelines as applicably as we could, however in certain areas there is room for improvement, which we would have liked to change, given we had more time.

Overall, we do believe the product we have created is of sufficient quality. While the implementation might not be ready, all the functionality is there, and the system works the way we have intended it to, although with rough edges here and there. As this has been a project of much higher scope and complexity than anything we have ever worked on in the past, we find ourselves satisfied with the product the project has put forth.

5.3 COVID-19

The COVID-19 pandemic has brought with it several challenges that have at times affected the workflow of this project. One of the major impacts it has had was when physical meetings had to be cancelled due to fear of contracting the virus, which resulted in the group having to work mostly from home. Working from home presented its own challenges, it affected the motivation to do productive work by not being surrounded by other group members working. Concentration was another thing affected by having to work from home, being bombarded by distractions and other stimulates, and not being in a proper work environment had a negative effect on concentration and therefore productivity.

5.4 Time management

Time management turned out to be a challenge as some aspects of the project took longer to do than expected. Estimating how long it would take to learn a new programming language was one of the main issues we had with time management. At the beginning of the semester, we spent longer than anticipated to learn and understand Python and Django. There were also

some minor issues with completing some of the elements in the Sprint backlog within the given timeframe, and these backlog items were then moved over to the next Sprint.

5.5 Change of product owner

Close to the end of the semester, our product owner at Red Rock Lars Lohne had left the company to pursue other work ventures. This came as a surprise to us, and it meant that we were assigned a new product owner, Andreas Fjetland, to guide and help us through the rest of the project. He was someone we knew beforehand and had previous dialogue with regarding the project. This change did not have a great impact on the project, as the development was on the way to completion.

5.6 Summary

This semester has posed numerous challenges such as COVID-19 restrictions, project management and. However, this has led to unique opportunities to learn from and gain new experiences, as working on this project as a group has taught us more about what it takes to create a successful product when working as a team, which we do not believe to be a prime example of. Furthermore, we have learned how important good communication with the product owner is and how impactful it can be on the final product. By working closely as a team and with the help of our project supervisor and product owner, we have managed to create a potentially shippable product of a planning system that meets the requirements set by the product owner to a satisfying degree.

6.0 Statement from product owner

‘I forbindelse med utviklingen av autonome logistikk-løsninger har Ocean Infinity et behov for ett planleggingssystem for autonome operasjoner. Denne bacheloroppgaven har tatt for seg omfanget med å strukturere og presentere jobbene som kommer inn fra tredjepartssystemer og blir gitt til maskinene. Målet med oppgaven fra Ocean Infinity sin side har vært å få utforsket hvordan en slik løsning kan se ut gitt krav til funksjonalitet og brukervennlighet.

For Ocean Infinity har denne oppgaven vært en nyttig forberedelse i hvordan vi må gå frem når vi starter jobben med planleggingssystemet hos oss for fullt. Løsningen som er tatt frem i bacheloroppgaven har gitt oss bedre innsikt i hvordan front-end for vårt planleggingssystem bør være og hva som bør utvikles videre.

Samarbeidet med studentene har vært bra. Til tross for pandemien har vi fått mulighet til å møtes fysisk et par ganger og hatt flere oppfølginger digitalt. Studentene har vært interessert i faget og benyttet anledning til å utforske nye teorier og forslag underveis.

Ocean Infinity takker for samarbeidet og ønsker gruppen lykke til videre.

Hilsen

Lars og Andreas'

7.0 Self evaluation

Israil Gayrbekov

During this project I have gained a lot of new experience and learned a lot. Especially when it comes to system development, project management, working in a team, and getting acquainted with new tools methods and technologies. In the initial stages of this project, I contributed to planning and analysing. I participated on designing our database ER model and documenting the process. I also took part in the development of the front end of our system, documenting and reviewing the code written. Learning a new programming language was a difficult challenge, but I gained a great deal of experience from it, which I will take with me going forward in my career. I also contributed to writing parts of the document such as system requirements, database design, project management etc. Working with a real IT project was an experience I think I will benefit from a great deal. I'm very pleased with how our system ended up, and how we as a team managed to work towards a common goal together.

Julie Krath

During the course of this I have gained a great deal of new and valuable knowledge. I have learned so much about system design and development. Having mainly contributed to the design element of the system, as well as writing large parts of chapter three and five. Learning a new programming language was a challenge, but through persistent work I was able to overcome this challenge. I am grateful for all the help we received from both the product owner as well as the project supervisor. The project would be far more challenging without their help. I am proud of how the end product turned out and the effort every group member

put in to create it. The experience of working on a real IT project is something I will take with me to my future work endeavours.

Ulrik Dørdal

Throughout this semester I have been a part of one of the largest projects I have ever had a part in working on. As a result, I faced a lot of new challenges which I also learned a lot from. Early on I learned a lot about how to structure a project of this stature, while it was new to everyone, I feel like we were able to do it well. As the project went along, later I worked on the project document writing a sizeable amount of the document. I had to do a lot of research into how to make a proper bachelor document seeing how it was also the first time we wrote a document of this scale. Overall, I feel the project was a challenging, yet very valuable experience.

Tiberiu Minzat

During this project, I acquired many new skills and knowledge. When the project started, I decided to use Python to create the project. This was a challenge for me as I had to learn Python because I had never used Python before and was mainly working on the programming part. But this semester, I learned a lot about how to start a new project and what it means to have clients with different expectations. Some of the other challenges I faced were helping with ER diagram design and website design. It also provides feedback to the client and corrects what the client asked while working on this project. The experience I gained in this project helps me because I want to continue working as a developer, and all the skills I gained help me.

Ramzan Ismailov

Working on this project has increased my knowledge in different fields of information technology and systems. Learning Python and Django was a valuable experience that I will greatly benefit from. Working on the project I tried to be flexible as possible by working on both the front end and back end of the application and writing the bachelor's document. This has demanded a lot from me in terms of being flexible with my daily schedule.

Denni Hansen

Of all projects I have been involved in, none have demanded the most of my time and effort like this one, and I do not doubt that I have gained a dozen grey hairs as a result of that. I naturally assumed the role of Scrum master and it has involved a multitude of responsibilities,

ranging from managing the project, correspondence with the product owner, documenting in the report, design of the system, and communicating requirements to the rest of the Scrum team. As the project neared its end, I began appreciating how demanding project management can be on a person. I now know what is expected of a project manager in future project endeavours.

References

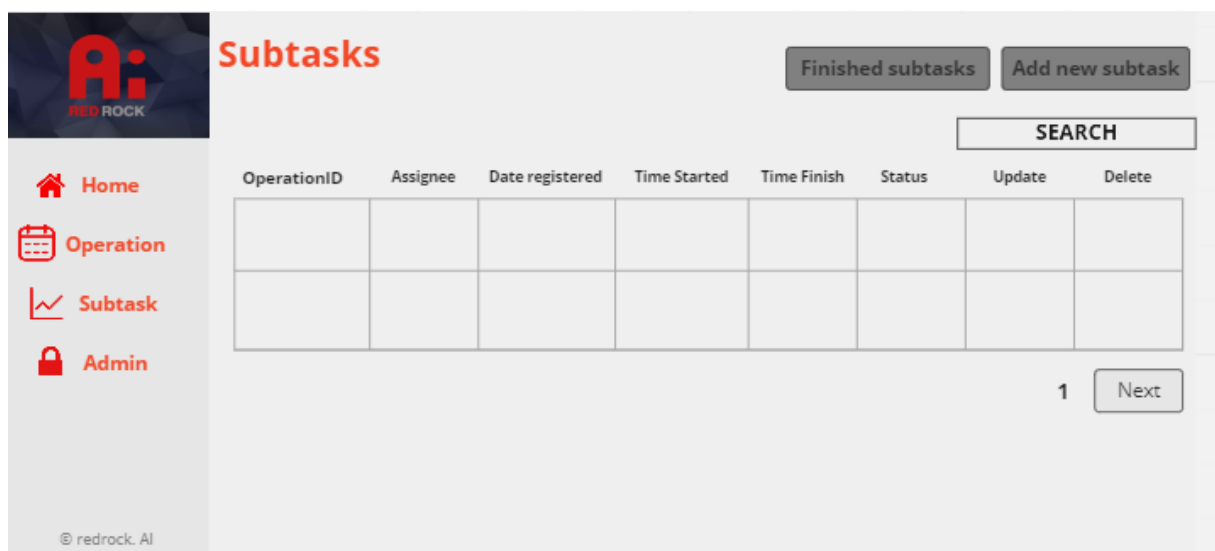
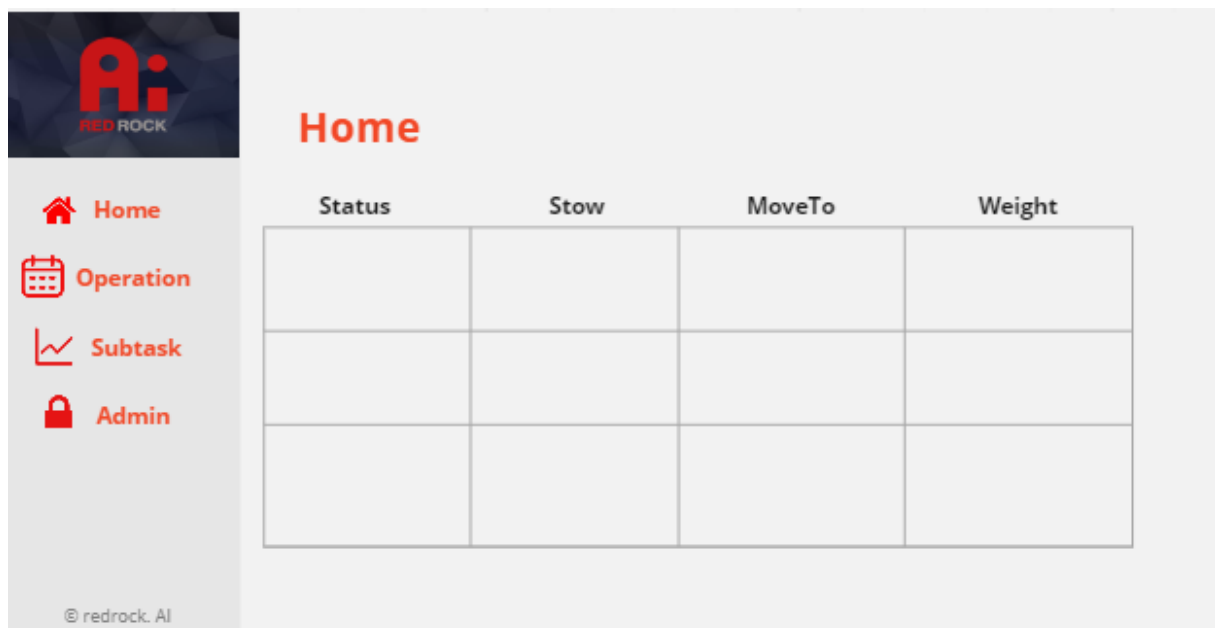
- (2019). In Benyon, *Designing user experience* (pp. 12-13). Harlow: Pearson.
- Alfonso. (2018, May 06). *Alfonso*. Retrieved from How we manage clients and their expectations: <https://medium.com/ideas-by-crema/what-to-expect-as-a-client-a88ee5677313>
- Appelbaum, B. (2021, August 09). *Agile Methodologies*. Retrieved April 15, 2022, from <https://www.planview.com/resources/guide/agile-methodologies-a-beginners-guide/?msclkid=e689bfc9bd1811ec97a024349549760c>
- ASKO. (2022, May 06). *Asko.no*. Retrieved from We deliver food all over Norway: <http://asko.no/en/>
- Atlassian. (2022, April 16). *Jira*. Retrieved from Atlassian: https://www.atlassian.com/software/jira?gclid=8a3f1af1cf4b1f3b3e1178abcfe15aee&gclidsrc=3p.ds&&adgroup=1306220045969962&campaign=380755336&creative=81638805017697&device=c&keyword=jira&ds_k=jira&matchtype=e&network=o&ds_kids=p54408781851&ds_e=MICROSOFT&ds_e
- Benyon, D. (2019). *Benyon*. Pearson.
- Bertelsen. (2021, June 13). *Inbound*. Retrieved from Hva er personas?: <https://www.inbound.no/blogg/hva-er-personas>
- Biscobing. (2019, September). *TechTarget*. Retrieved from Entity Relationship Diagram: <https://www.techtarget.com/searchdatamanagement/definition/entity-relationship-diagram-ERD>
- Django. (2022, April). *DjangoProject*. Retrieved from Meet Django: <https://www.djangoproject.com/>
- Django. (n.d.). *Django*. Retrieved from Writing and running tests: <https://docs.djangoproject.com/en/4.0/topics/testing/overview/>
- Django. (n.d.). *Django*. Retrieved from Making queries: <https://docs.djangoproject.com/en/4.0/topics/db/queries/>
- Django. (n.d.). *Django*. Retrieved from Models: <https://docs.djangoproject.com/en/4.0/topics/db/models/>
- Django Packages*. (2022). Retrieved from Django: <https://djangopackages.org/>
- Docker. (n.d.). *Docker*. Retrieved from About us: <https://www.docker.com/resources/what-container/>
- Docker. (n.d.). *Use containers to build, share and run your applications*. Retrieved from <https://www.docker.com/resources/what-container/>
- Gaille, L. (2020, March 19). *15 Advantages and Disadvantages of a Waterfall Model*. Retrieved from Vittana Blog: <https://vittana.org/15-advantages-and-disadvantages-of-a-waterfall-model>
- GeeksForGeeks. (2021, May 27). *GeeksForGeeks*. Retrieved from Django Request and Response cycle – HttpRequest and HttpResponse Objects: <https://www.geeksforgeeks.org/django-request-and-response-cycle-httprequest-and-httpresponse-objects/>

- GeeksForGeeks. (2021, August 16). *GeeksForGeeks*. Retrieved from Django Project MVT Structure: <https://www.geeksforgeeks.org/django-project-mvt-structure/>
- Google. (n.d). Retrieved from What is microservices architecture?: <https://cloud.google.com/learn/what-is-microservices-architecture>
- Hamilton, T. (2022, April 30). *Guru99*. Retrieved from What is Quality Assurance(QA)? Process, Methods, Examples: <https://www.guru99.com/all-about-quality-assurance.html>
- Hannah. (2021, August 05). *Career foundry*. Retrieved from What's the difference between a wireframe, a prototype, and a mockup?: <https://careerfoundry.com/en/blog/ux-design/difference-between-wireframes-prototypes-mockups/#what-is-a-mockup>
- inc., K. (2022, April 03). *Kissflow*. Retrieved from The extensive guide to business processes: <https://kissflow.com/workflow/bpm/business-process/>
- Javapoint. (2022). *Javapoint*. Retrieved from Django MVT: <https://www.javatpoint.com/django-mvt>
- JetBrains. (2022, April 7). *JetBrains*. Retrieved from Introduction: <https://www.jetbrains.com/help/datagrip/meet-the-product.html>
- Johnson, J., & Shiff, L. (2021, March 8). *What Is Microservice Architecture? Microservices Explained*. Retrieved from BMC Blogs: <https://www.bmc.com/blogs/microservices-architecture/#>
- Komninos. (2021). *Interaction design foundation*. Retrieved from An introduction to usability: <https://www.interaction-design.org/literature/article/an-introduction-to-usability>
- Konduru, S. (n.d.). *Scrum Values – Commitment: What it means to the Scrum Roles?* Retrieved from PremierAgile: <https://premieragile.com/scrum-values-commitment-what-it-means-to-the-scrum-roles/>
- Lilly021 Team. (2022, April 11). *Lilly021*. Retrieved from Docker vs. VM (Virtual Machine): <https://lilly021.com/docker-vs-vm-virtual-machine/>
- Long, M. (2010, October 31). *Internal vs. External Software Quality*. Retrieved from Mike Long's Blog: <https://meekrosoft.wordpress.com/2010/10/31/internal-and-external-software-quality/>
- M, P. (2022, April 23). *Piras, M*. Retrieved from Trello vs Jira: <https://nira.com/trello-vs-jira/>
- McLarty, M. (2016, May 30). *Microservice architecture is agile software architecture*. Retrieved from InfoWorld: <https://www.infoworld.com/article/3075880/microservice-architecture-is-agile-software-architecture.html>
- Nelson. (2020, April 20). *Nelson, C*. Retrieved from Understanding issue types in jira: https://community.atlassian.com/t5/Jira-articles/Understanding-issue-types-in-jira/ba-p/1497237?jira_cloud_basics_1=&link_1=&ref=community_jira&showcase=
- Nettleton, C. (2022, January 20). *Nettleton, C*. Retrieved from A developer can't be the product owner: <https://appliedframeworks.com/a-developer-cant-be-the-product-owner/>
- Nikolov. (2017, April 08). *UX collective*. Retrieved from Design principle: Consistency: <https://uxdesign.cc/design-principle-consistency-6b0cf7e7339f>
- Palmer, S. (2022, May 09). *Palmer S*. Retrieved from DevTeamSpace: <https://www.devteam.space/blog/how-to-deploy-a-web-app-with-docker-containers/>

- Pavel Gorbachenko. (2021, April 09). *Enkonix*. Retrieved from <https://enkonix.com/blog/functional-requirements-vs-non-functional/>
- PVS.Studio. (2013, April 06). *PVS-Studio*. Retrieved from Coding standard: <https://pvs-studio.com/en/blog/terms/0008/>
- PVS.Studio. (2013, April 6). *PVS-Studio*. Retrieved from Coding standard: <https://pvs-studio.com/en/blog/terms/0008/>
- Python. (n.d.). *Python*. Retrieved from What is Python? Executive Summary: <https://www.python.org/doc/essays/blurb/>
- Shadhin, F. (2021, May 4). *Python in Plain English*. Retrieved from The MVT Design Pattern of Django: <https://python.plainenglish.io/the-mvt-design-pattern-of-django-8fd47c61f582>
- Shevchenko, N. (2022, May 05). *redrock.no*. Retrieved from Ocean Infinity acquires Red Rock AS: <https://www.redrock.no/2021/10/12/ocean-infinity-acquires-red-rock-as/>
- Talend. (n.d.). *Talend*. Retrieved from What is MySQL? Everything You Need to Know: <https://www.talend.com/resources/what-is-mysql/>
- TeamRetro. (n.d.). *The Four Ls retrospective*. Retrieved from TeamRetro: <https://www.teamretro.com/retrospectives/4ls-retrospective>
- Walker. (2022, April 30). *Guru99*. Retrieved from What is an API?: <https://www.guru99.com/what-is-api.html>
- West, D. (n.d.). *Atlassian Agile Coach*. Retrieved from Agile scrum roles: <https://www.atlassian.com/agile/scrum/roles>

Appendix

Appendix 1: Mock-ups



Appendix 2: Project description from RedRock

Red Rock – Bacheloroppgave for planleggingsfunksjon

Red Rock er leverandør av løsninger innen autonom lasthåndtering og har to software plattformer som sammen muliggjør planlegging av de autonome operasjonene og utførende funksjoner, inkludert kunstig intelligens, for maskinene i seg selv.

Denne oppgaven fokuserer på planleggingssystemet for autonome operasjoner. Spesifikt er det behov for å se nærmere på planleggingsfunksjonen som strukturerer opp jobbene som skal utføres av maskinene. Planen vil ta utgangspunkt i ordre som meldes inn fra eksterne kilder.

Planleggingsfunksjonen skal hente informasjon om ordrer og et tilhørende sett med ordrelinjer gjennom et API. Ordren er typisk en plukklister eller en sekvens av transaksjoner maskinene skal utføre. Planen i seg selv kalles en «Mission Plan» og hver av ordrelinjene eller transaksjon er en «Job». Se vedlegg med et eksempel på hvordan en ordre ser ut med de forskjellige ordrelinjene.

I planfunksjonen skal det opprettes en «Mission Plan» basert på en ordre. Metadata om ordren skal vises og det skal lages en header for selve planen. Den skal også få sitt eget løpenummer uavhengig av ordren. Ordrelinjene legges ut som underliggende «Jobs». Data tilhørende ordrelinjen skal også vises på jobbene som informasjon. Basert på tilgjengelig data om sekvensering av ordrelinjene fra ordren, skal planen reflektere dette. Det skal være mulig å sortere jobbene basert på de forskjellige verdiene de har.

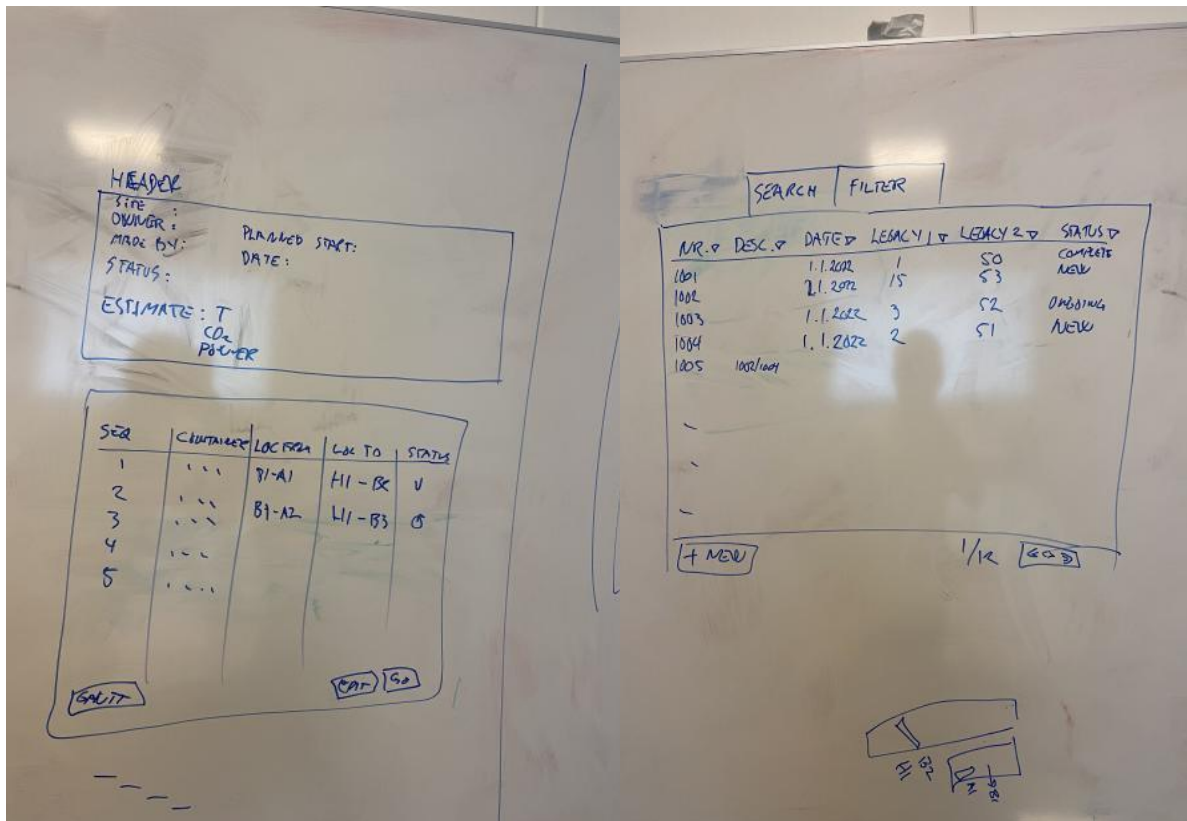
Det skal være mulig å justere planen ved å endre rekkefølgen på jobbene, legge inn jobber i tillegg til de som er basert på ordrelinjene. De manuelle linjene skal være lette å identifisere i planen.

En plan skal kunne ha forskjellige statuser basert på hvor lang den er kommet planleggingen. Disse statusene må defineres senere.

Når det kommer til hvordan planen presenteres og brukeropplevelser knyttet til å vurdere planen og eventuelt manipulere denne i front-end så står studentene helt fritt. Det kan hentes inspirasjon fra andre typer planleggingssystemer og ordrehåndtering samt tilnærminger fra gaming og andre kreative løsninger.

I forhold til arkitektur så stilles det krav til at det fokuseres på at tjenestene skal stå for seg selv og kommunisere gjennom grensesnitt, ofte referert til som en micro-service arkitektur. Derfor bør back-end og front-end ikke dele kodebase og kunne ansees som to separate funksjoner som kommuniserer gjennom et API. Funksjonene vil også kommunisere gjennom et API for å hente ordre og et annet for å overføre data til maskinene.

Skulle omfanget av oppgaven ikke være tilstrekkelig så er det ytterligere elementer rundt planlegging på roadmap som kan inkluderes.



Appendix 4: Function used to get data for table located on home page

```
def displayindex(request): # View for Home page
    finish = Subtask.objects.prefetch_related(
        'moveTo', 'assignee', 'stow', 'status').filter(status_id=1)
    return render(request, 'index.html', {'Subtask': finish})
```

Appendix 5: Python code used to get data in HTML page

```
{% for Subtask in Subtask %} <!-- Getting data from views.py -->
<tr>
  <td class="">{{Subtask.loadSeq}}</td> <!-- Rows with valeus from Database -->
  <td class="">{{Subtask.status}}</td> <!-- Rows with valeus from Database -->
  <td class="">{{Subtask.stow}}</td> <!-- Rows with valeus from Database -->
  <td class="">{{Subtask.moveTo}}</td> <!-- Rows with valeus from Database -->
  <td class="">{{Subtask.assignee}}</td> <!-- Rows with valeus from Database -->
  <td class="">{{Subtask.subtaskID}}</td> <!-- Rows with valeus from Database -->
  <td class="">{{Subtask.containerID}}</td> <!-- Rows with valeus from Database -->
  <td class="">{{Subtask.containerWeightT}}</td> <!-- Rows with valeus from Database -->
  <td><a href="{% url 'updatesubtask' Subtask.subtaskID %}" type="button"
    class="btn btn-secondary">Edit</a></td> <!-- Button for editing -->
  <td><a href="{% url 'deletesubtask' Subtask.subtaskID %}" type="button"
    class="btn btn-danger">Delete</a></td> <!-- Button for deleting -->
</tr>
{% endfor %} <!-- Ending for loop -->
```

Appendix 6: The models.py file and its classes

```

models.py X
src > redrock > models.py > MoveTo
1 from auditlog.registry import auditlog
2 from datetime import date
3 from email.headerregistry import Address
4 from re import M
5 from tkinter import CASCADE
6 from django.db import models
7
8 class Employee(models.Model): # Employee table
9     employeeID = models.IntegerField(primary_key=True)
10    firstName = models.CharField(max_length=40)
11    lastName = models.CharField(max_length=40)
12    address = models.CharField(max_length=40)
13    zipCode = models.IntegerField()
14    city = models.CharField(max_length=40)
15
16    def __str__(self) -> str: # definition to name the Foreign Key
17        return self.firstName
18
19 class Status(models.Model): # Status table
20    statusID = models.IntegerField(primary_key=True)
21    status = models.CharField(max_length=20)
22
23    def __str__(self) -> str: # definition to name the Foreign Key
24        return self.status
25
26 class MoveTo(models.Model): # MoveTo table
27    moveToID = models.IntegerField(primary_key=True)
28    location = models.CharField(max_length=20)
29
30    def __str__(self) -> str: # definition to name the Foreign Key
31        return self.location
32
33 class Stow(models.Model): # Stow table
34    stowID = models.IntegerField(primary_key=True)
35    locationStow = models.CharField(max_length=20)
36
37    def __str__(self) -> str: # definition to name the Foreign Key
38        return self.locationStow
39
40 class Operation(models.Model): # Operation table
41    operationID = models.IntegerField(primary_key=True, auto_created=True)
42    assignee = models.ForeignKey('Employee', on_delete=models.CASCADE)
43    dateRegistered = models.DateTimeField(auto_now_add=True)
44    timeStart = models.DateTimeField(auto_now_add=True, null=True)
45    timeFinish = models.DateTimeField(auto_now=True)
46    status = models.ForeignKey(
47        'Status', on_delete=models.CASCADE)

```

Appendix 7: The models.py file and its classes

```

models.py X
src > redrock > models.py > MoveTo
38        return self.locationStow
39
40 class Operation(models.Model): # Operation table
41    operationID = models.IntegerField(primary_key=True, auto_created=True)
42    assignee = models.ForeignKey('Employee', on_delete=models.CASCADE)
43    dateRegistered = models.DateTimeField(auto_now_add=True)
44    timeStart = models.DateTimeField(auto_now_add=True, null=True)
45    timeFinish = models.DateTimeField(auto_now=True)
46    status = models.ForeignKey(
47        'Status', on_delete=models.CASCADE)
48
49 class Subtask(models.Model): # Subtask table
50    subtaskID = models.IntegerField(primary_key=True, auto_created=True)
51    assignee = models.ForeignKey('Employee', on_delete=models.CASCADE)
52    containerID = models.CharField(max_length=255)
53    containerWeightT = models.DecimalField(max_digits=6, decimal_places=2)
54    loadSeq = models.IntegerField()
55    moveTo = models.ForeignKey('MoveTo', on_delete=models.CASCADE)
56    stow = models.ForeignKey('Stow', on_delete=models.CASCADE)
57    status = models.ForeignKey(
58        'Status', on_delete=models.CASCADE)
59
60 auditlog.register(Subtask)
61 auditlog.register(Operation)
62
63

```

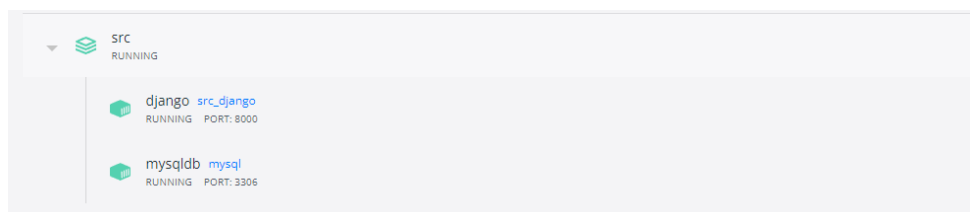
Appendix 8: Docker-compose.yml file

```

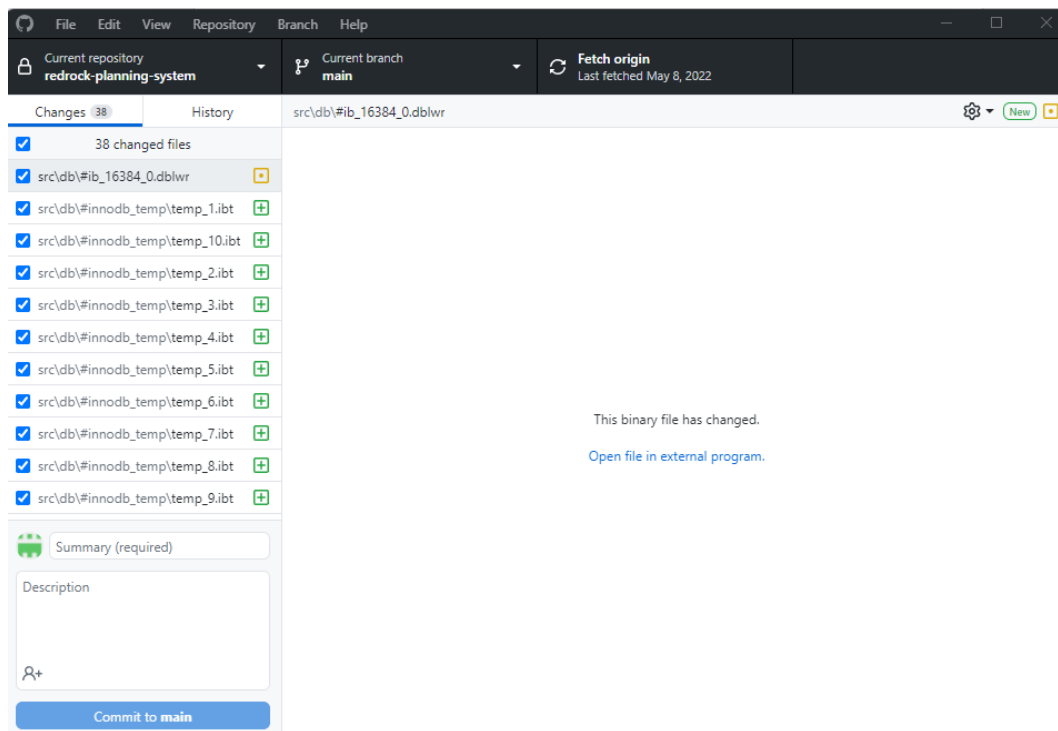
docker-compose.yml x
src > docker-compose.yml
1 version: '3'
2
3 services:
4
5   django:
6     build: .
7     container_name: django
8     restart: always
9     command: python manage.py runserver 0.0.0.0:8000
10    ports:
11      - "8000:8000"
12    volumes:
13      - ./usr/src/app/
14      - /tmp/app/mysql:/run/mysql
15    networks:
16      - application_network
17
18    depends_on:
19      - db
20
21   db:
22     image: mysql
23     container_name: mysqldb
24     restart: always
25     environment:
26       - MYSQL_DATABASE=redrock
27       - MYSQL_ROOT_PASSWORD=root
28     ports:
29       - "3306:3306"
30     volumes:
31       - ./db:/var/lib/mysql
32     networks:
33       - application_network
34
35   networks:
36     application_network:
37       name: application
38       driver: bridge
39       enable_ipv6: true
40       ipam:
41         driver: default
42         config:
43           - subnet: 172.16.238.0/24
44             gateway: 172.16.238.1
45           - subnet: 2001:3984:3989::/64
46             gateway: 2001:3984:3989::1
47
48
49

```

Appendix 9: Docker containers running simultaneously



Appendix 10: Display of GitHub Desktop



Appendix 11: Group contract

Avtale om gjennomføring av IS-304 Bacheloroppgave i IT og informasjonssystemer

Formålet med avtalen er å sikre gjensidig utbytte av samarbeidet, både for virksomheten og studenten. Avtalen inngås når studentene har fått godkjent skisse til tema og arbeidsmåte for bacheloroppgaven.

Virksomheten ansvar

- Virksomheten oppnevner en kontaktperson.
- Virksomheten fastsetter bacheloroppgaven i samarbeid med studenten.
- Virksomheten skal orientere studenten om de regler og retningslinjer som gjelder for virksomhetens ansatte, herunder eventuelt regelverk knyttet til personvern og taushetsplikt, og gi studenten tilgang til nødvendige systemer.
- Virksomheten skal så raskt som mulig ta opp eventuelle problemer i forbindelse med samarbeidet med studenten og universitetets representant.

Kontaktperson (Lars Lohne, lars.lohne@redrock.no, mobil +4791380341)

Universitetets ansvar

- Universitetet oppnevner en representant (normalt veileder eventuelt emneansvarlig) som virksomhet og student kan forholde seg til i forbindelse med gjennomføring av samarbeidet.
- Universitetet forbereder studenten på samarbeidet og de kriterier som gjelder for gjennomføring.
- Representant fra universitetet følger opp virksomheten og studenten under prosjektperioden.
- Representant fra universitetet skal følge opp eventuelle meldinger om problemer i forbindelse med samarbeidet fra student eller virksomhet.

Universitetets representant (Hallgeir Nilsen, hallgeir.nilsen@uia.no, mobil +4741565684)

Studentens ansvar

- Studenten skal møte til avtalt tid i virksomheten.
- Dersom studenten pga. sykdom eller annen årsak ikke kan møte som avtalt, skal studenten gi beskjed til virksomheten så snart som mulig.
- Studenten forplikter seg til å følge de regler og retningslinjer som gjelder for virksomhetens ansatte.
- Studenten skal så raskt som mulig ta opp eventuelle problemer i forbindelse med samarbeidet med virksomhet og universitetets representant.
- Studenten skal ikke motta lønn for arbeidet.
- Studenten dekker reisekostnader til/fra virksomheten (og eventuelle oppholdskostnader) selv med mindre annet er avtalt med universitetet.

Rettigheter

Dersom annet ikke er avtalt så er det virksomheten som har alle rettigheter til produktet.

Tillegg til avtale

Annet som avtalepartnere har blitt enige om, fyll inn (for eksempel hvis virksomheten ønsker at arbeidet inkludert bacheloroppgaven skal holdes konfidensiell).

Sted/dato
Kristiansand
22.01.2022
For Universitetet i Agder

Emneansvarlig

Hallgrin Nihon

Lars Eidetveit
For virksomhet

Israil G
Student

Julie Korth

Denni H

Ragnar
W. I. D

Appendix 12: Coding standard

Code standard for application

This coding standard is common for both front end and back end. Although many of these practises are common among developers, they are hard to maintain. We as a group agreed upon following these standards.

Benefits of following a coding standard:

- Easier to detect errors in the code
- Improves the developer's knowledge
- Easier to further develop the application
- Increases efficiency

Naming conventions:

- Use camel case (example: ~~subtaskRemove~~)
- Classes, functions and variables should have meaningful names
- Avoid digits in variable names
- Don't have very long names, keep it simple and easy

Code quality:

- Every code function should be well documented
- Avoid repeating the same code over-again
- Avoid deep nesting, it makes the code more disorganized
- Try to keep the source code, as simple as possible

Appendix 13: Views.py file and its functions and classes

```
src > redrock > views.py @ displayindex
1 from lib2to3.pygen2.pygen import pygen2
2 from pyexpat.errors import messages
3 from re import sub
4 from django.db.models.aggregates import Count
5 from django.http import HttpResponseRedirect
6 from django.shortcuts import get_object_or_404, redirect, render
7 from rest_framework.viewsets import ModelViewSet
8 from forms import SubtaskForm, OperationForm
9 from django.db.models import Q
10 from .models import Subtask, Operation, Status
11 from .serializers import StatusSerializer, OperationSerializer, SubtaskSerializer
12
13
14 class StatusViewSet(ModelViewSet): # ViewSets
15     queryset = Subtask.objects.prefetch_related(
16         'moveTo', 'assignee', 'stow', 'status').all()
17     serializer_class = StatusSerializer
18
19
20 class OperationViewSet(ModelViewSet): # ViewSets
21     serializer_class = OperationSerializer
22     queryset = Operation.objects.prefetch_related('assignee', 'status').all()
23
24
25 class SubtaskViewSet(ModelViewSet): # ViewSets
26     queryset = Subtask.objects.prefetch_related(
27         'moveTo', 'assignee', 'stow', 'status').all()
28     serializer_class = SubtaskSerializer
29
30
31 def displayindex(request): # View for Home page
32     finish = Subtask.objects.prefetch_related(
33         'moveTo', 'assignee', 'stow', 'status').filter(status_id=1)
34     return render(request, 'index.html', {'Subtask': finish})
35
36
37 def displayorder(request): # View for operations
38     results = Operation.objects.prefetch_related('assignee', 'status').all()
39     return render(request, 'order.html', {'Operation': results})
40
41
42 def displaydata(request):
43     results1 = Subtask.objects.prefetch_related(
44         'moveTo', 'operationID', 'stow', 'status').filter(status_id=1)
45     return render(request, 'index.html', {'Subtask': results1})
46
47
48 def displaydata(request): # View for Subtask
49     results1 = Subtask.objects.prefetch_related(
50         'moveTo', 'assignee', 'stow', 'status').filter(~Q(status_id=3))
51     return render(request, 'subtask.html', {'Subtask': results1})
52
53
54 def displayfinish(request): # View for finished Subtasks
55     finish = Subtask.objects.prefetch_related(
56         'moveTo', 'assignee', 'stow', 'status').filter(status_id=3)
57     return render(request, 'finish.html', {'Subtask': finish})
58
```

Appendix 14: Views.py file and its functions and classes

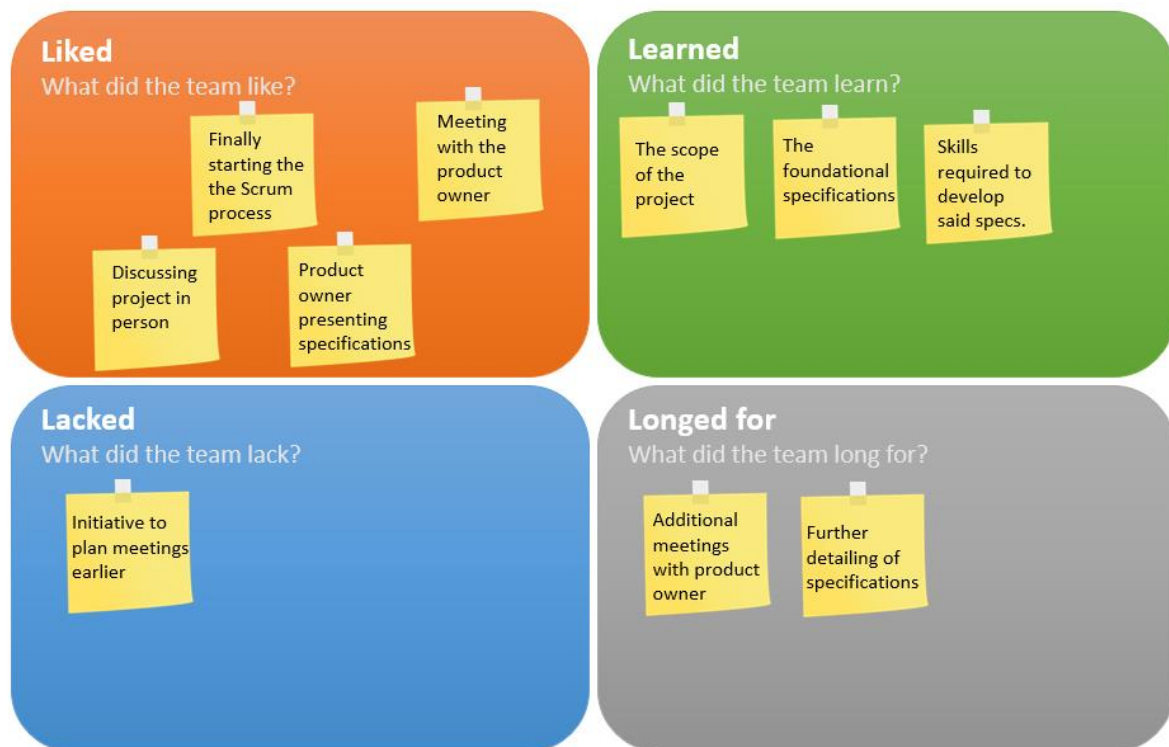
```
src > redrock > views.py @ displayindex
59 def add_subtask(request): # definition for adding a subtask
60     submitted = False
61     if request.method == "POST":
62         form = SubtaskForm(request.POST)
63         if form.is_valid(): # if the form is valid save data
64             form.save()
65             return redirect('displaydata')
66     else:
67         form = SubtaskForm
68         if 'submitted' in request.GET:
69             submitted = True
70     form = SubtaskForm
71     return render(request, 'addsubtask.html', {'form': form, 'submitted': submitted})
72
73
74 def update_subtask(request, subtaskID): # definition for updating a subtask
75     subtask = Subtask.objects.prefetch_related(
76         'moveTo', 'assignee', 'stow', 'status').get(subtaskID=subtaskID)
77     form = SubtaskForm(request.POST or None, instance=subtask)
78     if form.is_valid(): # if the form is valid save data
79         form.save()
80         return redirect('displaydata')
81     return render(request, 'editsubtask.html', {'subtask': subtask, 'form': form})
82
83
84 def update_operation(request, operationID): # definition for updating a Operation
85     operation = Operation.objects.prefetch_related(
86         'assignee', 'status').get(operationID=operationID)
87     form = OperationForm(request.POST or None, instance=operation)
88     if form.is_valid(): # if the form is valid save data
89         form.save()
90         return redirect('displayorder')
91     return render(request, 'editoperation.html', {'operation': operation, 'form': form})
92
93
94 def delete_subtask(request, subtaskID): # definition for deleting subtasks
95     subtask = Subtask.objects.prefetch_related(
96         'moveTo', 'assignee', 'stow', 'status').get(subtaskID=subtaskID)
97     subtask.delete()
98     return redirect('displaydata')
99
100
101 def delete_operation(request, operationID): # definition for deleting operations
102     operation = Operation.objects.prefetch_related(
103         'assignee', 'status').get(operationID=operationID)
104     operation.delete()
105     return redirect('displayorder')
106
107
108 def add_operation(request): # definition for adding a operations
109     submitted = False
110     if request.method == "POST":
111         form = OperationForm(request.POST)
112         if form.is_valid(): # if the form is valid save data
113             form.save()
114             return redirect('displaydata')
115     else:
116         form = OperationForm
117         if 'submitted' in request.GET:
```

Appendix 15: Risk matrix

Risk	1 to 5 Consequences	1 to 5 Likelihood of occurring	Risk	Preventative measures	Measures to reduce damage
Remote work will reduce work effectiveness	3	5	15	Attend physical meetings as often as possible	Make the best of it by referring to best remote work practices
Members not finishing backlog items in time	3.5	4	14	Stress the importance of delivery on time	Caution member or members to deliver on time next time
Miscommunication , misinterpretation of product specifications	3.5	3.5	12.25	Constantly revisit specifications and check if current state of development is in line	Communicate with product owner to review and further define specifications
Members not able to attend group meetings or remote sessions	2.5	5	12	Check if planned sessions are appropriate for everyone	Member or members later checks up on what was missed
Expected quality of work is not met such as coding standard	3	3.5	10.5	Reflect if quality of work is of appropriate standard	Inform oneself of current standards
One or several group members contracting COVID-19	2.5	4	10	Avoid unnecessary close contact	Quarantine oneself and avoid contact to prevent further spreading in group
Hard to troubleshoot and resolve glitch/bug	4	2	8	Keep the system design simple, code with low cohesion and coupling in mind	Inform and involve multiple group members to combine knowledge and experience

					in order to resolve
Conflicts and disputes occurring between group members	2	2	4	Foster a culture of accepting different opinions, minority accepts majority wish	Discuss to settle disputes
One or several group members dropping out of the course	3	1	3	Encourage perseverance throughout the duration of the course	Group members assume additional responsibilities

Appendix 16: Sprint retrospectives



Sprint 1, date 05/03/2022



Sprint 2, date 21/03/2022

Liked

What did the team like?

Backend elements working together

Frontend design finalizing

Further detailed specifications

Learned

What did the team learn?

Additional Django and Python

Resolving bugs

Lacked

What did the team lack?

Lacking Scrum implementation

Meetings, standups

Longed for

What did the team long for?

Faster input from product owner

To use time wisely

Sprint 3, date 16/04/2022

Liked

What did the team like?

System taking its final shape

Took Scrum more seriously

A lot of input from product owner

Learned

What did the team learn?

Importance of planning earlier

System taking its final shape

Lacked

What did the team lack?

System taking its final shape

Initiative, self-responsibility

Longed for

What did the team long for?

Use time accordingly

Sprint 4, date 30/04/2022

