Prosjektplan Cryogenetics

Forfattere:
Lars Lahlum Ruud
Axel Elias Wollebekk Jacobsen
Matthias David Greeven
Håvard Bø

1 Background and Scope

1.1 Background

Cryogenetics AS, henceforth known as the client, is a Norwegian biotech company that provides services and products for cryopreservation of milt and the fertilization of fish. The company is based in Hamar, but maintains an international presence with labs in Chile, Canada, the United States and Scotland.
Steffen Wolla, in his role as Production Manager and Business Developer for the client, has requested the development of a logistical system to register the movement of their liquid nitrogen containers. The ambition of this new system is that employees of the company would have an easier time managing the containers.

1.2 Subject area

The main subject area for the project is logistics. Logistics deals with the movement of materials and products towards facilities, in order to sell or produce materials and services. Logistics are a part of the company's operational costs. As companies grow and expand, it gets increasingly more complex to acquire, store and transport resources.

The digitalization of logistics has made many logistics processes more precise and manageable. By using specialized tools, companies are able to handle larger amounts of transactions, directly increasing profits and efficiency. Features like tracking and estimated delivery times allow companies to acquire more precise information surrounding their products, improving the experience for employees and customers.

1.3 Delimitation

We’re going to focus on the logistics of liquid nitrogen containers, the contents of said containers will not be registered in our system. These containers are designed to keep its contents below -140°C for extended durations, which removes the biological decay from its content. To ensure this they have to be regularly refilled with liquid nitrogen. The frequency of refilling depends on the size and model of the container, where larger containers often require recurrent refilling. Therefore it is crucial to document when refilling happens, as temperature increases could lead to the biological content to contaminate or expire.

1.4 Task description

The task is to develop an application to track and register the movement and status of their liquid nitrogen tanks. The project requires user interfaces, a server and a database. Operators and administrators will have two different user interfaces, which will communicate with a common server and database.

The tablet application must be able to do the following:

- Provide authentication through a 3 digit code.
- Designed with the principles of user centered design to limit the need for training.
- Scan QR codes on containers to identify the container.
- Sort transactions based on date, location, employee, container type and customer.
- Show transaction logs for the scanned container.
- Scan multiple containers in a filling menu", and update the server with all newly filled containers.
- Display when specific tanks were last filled with nitrogen, and give an approximation of when they should be refilled, based on tank size and model.

The administration application is requested to be designed for desktop environment, and must do the following:

- Deliver a report of change logs in between two desired time points.
- Authenticate users through email, password and two factor authentication.
- Sort on transactions based on date, location, employee, container and customer.
- Provide detailed logs over the history of a specific tank, filtered by locations.
- Administrate client locations in the database.
- Administrate operators authentication codes.
- Administrate container models in the database.
- Register new customers and update existing ones.
- Retrieve a detailed change log of all desired containers.

2 Goals and frames

2.1 Project goals

These are the goals we want to achieve during this project.

2.1.1 Result goals

Our project should produce a logistics solution which can be used to keep track of liquid nitrogen containers location and internal processes used for cryopreservation. The logistics solution will be custom designed solution for the clients needs, it will include:

- A mobile tool for operators.
- A desktop tool for administrators.
- A server and database.
- Deployment on Microsoft Azure.

2.1.2 Effect goals

By the projects end, cryogenetics will have:

- Better control and traceability over their liquid nitrogen containers.
- Digitized and eased their workflow.
- Warnings for when containers require immediate or scheduled maintenance.

2.1.3 Learning goals

During the development of our bachelor thesis we will be exploring and learning about our clients subject area. The knowledge we will focus on learning include, but is not limited to:

- Learning about the practices of the logistics business.
- Designing an efficient and user friendly logistics system.
- Creating a product with sound fundations for use in a real work environment.
- Establishing a server-based database that is internationally available and secure for company wide use.
- Using User Centered Design principles and techniques to develop a design which is intuitive to use by staff of different ages.
- Working efficiently and professionally in a group together with a third-party client
- Improving our mobile programming skills by developing a professional application.
- Creating an efficient database using data modeling.
- Developing a web application with the use of ReactJS.
- Creating a modern and fast back-end using Golang.

2.2 Constraints

2.2.1 Time Constraints:

- The project's final delivery deadline is 22.05.2023 12:00 PM, at this point the product, final report and documentation needs to be complete.

2.2.2 Technological constraints

- User friendly solution, which can be used by employees with different prerequisites.
- Application for a mobile device.
- Register and monitor movement and maintenance of nitrogen containers.
- Use of QR- or bar-code for swiftly registration of movement and maintenance.
- A web application for administrative tasks.

3 Project organization

The group consists of four BPROG students, each with roughly the same academic experience. However, in the course PROG2052 - Integrasjonsprosjekt, we worked on two different groups,

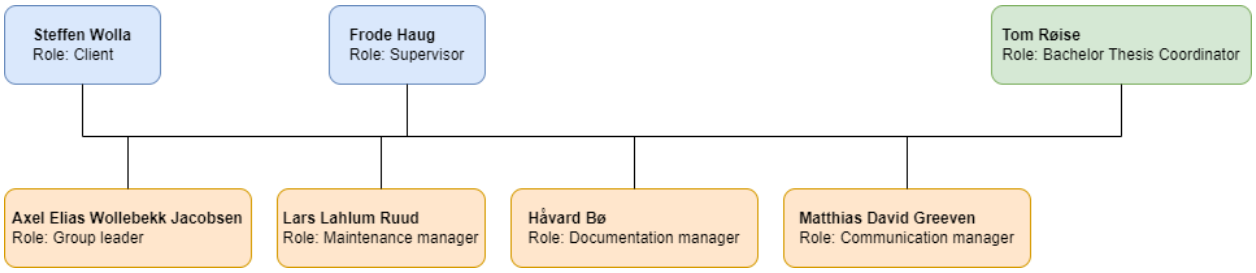
gaining expertise in different fields. Håvard and Matthias developed an android application in kotlin, while Lars and Axel made a GoLang backend.

3.1 Responsibilities and roles

Each group member has an area of responsibility based on their previous experience,

- Håvard is responsible for the Android application.
- Matthias is responsible for the website.
- Axel is responsible for the Golang backend.
- Lars is responsible for the SQL database and its deployment.

In addition each group member has a role, as shown in the image below.



3.2 Routines and group rules - set up a contract

Members: Axel Elias Wollebekk Jacobsen, Matthias David Greeven, Lars Lahlum Ruud, Håvard Bø

Procedures for the teamwork

A. *Meetings*

Meetings are held according to the Scrumban agile framework. Each Monday and Thursday 12:00 a Scrumban meeting is held physically or over Discord. The scrumban leader is responsible for incorporating proper scrumban technique, such as the use of the Github Project issueboard. Meetings with the client are held on mondays at 15:00 over Teams, if necessary. These are arranged at least 96 hours prior by the Communication manager and may be moved to a more suitable time if needed.

B. *Notification in case of absence or other incidents*

If you are late to- or cannot attend a meeting with the client, this should be communicated as soon as possible so that the meeting can be moved. If the meeting cannot be moved, it is held without you.

C. *Expected effort*

It is expected that you spend about 30 hours a week on this project, but as long as you complete your tasks and responsibilities it is not strictly enforced. In addition, you should be available from 12:00 to 15:00 each day a Kanban meeting has been held. If you cannot be available during this time, you should notify the group at least 12 hours prior.

D. *Disagreement*

If there is an academical disagreement the group should try to find a solution which the majority agrees on. If exactly half the group disagrees with the other, the Group leader decides. After the solution has been set, everyone must be loyal to it.

E. *Documents*

Discord and GIT are used to share files, with Google Docs as a collaborative writing tool. The Documentation Manager is responsible for meeting reports from meetings, these reports give at least a short description of which decisions have been made and what has been done during the meeting.

F. *Policy for monitoring tasks*

The Group Leader is responsible for monitoring tasks and ensuring that everyone has something to do. If you are struggling with completing your tasks, you should notify the group leader so that your tasks can be more evenly distributed.

G. *Submission of teamwork*

The Group Leader is responsible for ensuring that deadlines are kept and files are submitted.

H. Maintenance / Services

The Maintenance Manager is responsible for ensuring that the necessary services are available, such as cloud hosting services for backend. This is done in cooperation with the client, who has agreed to provide such services.

Breach of contract

If a group member does not follow the agreed upon procedures, a meeting can be held with every group member present about the breach of contract. If this does not solve the issue, a written warning is created by the other group members. The written warning should contain: An explanation of how the contract was breached, requirements for solving the issue, and what action will be taken if the group member does not fulfill the requirements. If the breach is severe enough, this action should be holding a meeting with the group and advisor about solving the issue. If none of this solves the issue, the group member who breached the contract will be expelled from the group.

4 Planning, follow-up & reporting

4.1 Main division of the project

For our bachelor thesis we decided to utilize the scrumban SDLC framework. Initially, our plan was to just use scrum. By having daily meetings we could ensure that everyone had work to do, and that they got it done. However, an issue we found with this was that the daily meetings ended up consuming a significant portion of extra time, even if the meeting itself only lasted 15-30 minutes. After consulting our advisor who suggested we tried an alternative approach, we decided to try using both kanban and scrum. Kanban would offer some flexibility in the beginning of the project where members can work on their parts when they have time, while scrum would ensure efficient progress later when we are mostly coding. After some discussion however, we realized that moving from one framework, to another mid-project could pose unwanted delays. Due to this we found that scrumban would suit our needs. By having two scrumban meetings a week with specific tasks distributed from a backlog we can ensure both progress and flexibility. Additionally, altering the length of sprints would permit us to optimize the framework to our needs as we progress in the project, while only utilizing one development framework.

4.2 Plan for status meetings and milestones during the project

During the initial design phase of the project we will have weekly meetings with our Cryogenetics representative, and NTNU Advisor, Frode Haug. This will help us ensure that the project's design models are correctly designed and up to the desired standards of both the school and our client. When the design gets accepted, the group will shift its focus from designing to development. Since the design is already decided on, we will reduce the amount of meetings with our representative and councilor. This will permit us to work faster since we can divide larger tasks between meetings. The meetings will also transition into a presentation format where we present our work so far, and receive feedback for improvement or alterations.

To improve progress tracking through the project, we have placed progress milestones. Setting dates for major progress checkpoints will help us change our focus from one work pattern to another. Since we do not have perfect foresight these dates will be subject to change as we progress in the project. However, we initially have chosen to place our milestones at these dates:

- 25.01 : Project plan
- 02.02 : Physical database set up
- 13.02 : First backend application draft
- 16.02 : First web application draft
- 20.02 : First status report
- 25.02 : First mobile application draft
- 01.03 : Main user testing round

- 13.03 : Web application mostly finished
- 17.03 : Mobile application mostly finished
- 22.03 : Backend completed
- 27.03 : Thesis outline completed
- 03.04 : Second status report
- 21.04 : Mobile & web application completed
- 28.04 : First rough draft of thesis
- 01.05 : Final status report
- 12.05 : Thesis mostly done
- 17.05 : Thesis completed

5 Organization of quality assurance

5.1 Documentation and standards

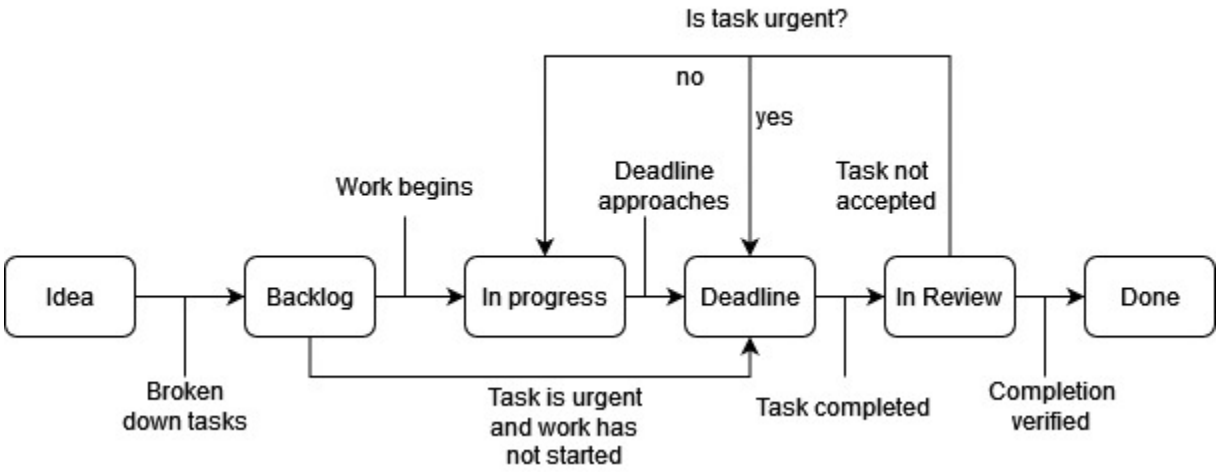
During the development process we will put an emphasis on client-friendly development practices. The intent behind this is to make it easier for our client to further develop our project after its completion without it demanding significant work with code and structural interpretation. For this reason we will be using the commenting and documentation practices each individual coding language uses respectively. Additionally we will include links to documentation practices in our GitHub repository to further simplify the process. To further ensure consistency we will be developing and documenting exclusively in english.

5.2 Standardized workflow

To simplify the workflow of our project, we have decided that all tasks will go through the same development steps. This ensures that all work is tracked properly, and follows a clear step-by-step path. A task begins its lifecycle as a feature desired by our client. We will then discuss specifications around how this feature operates before we break it down into smaller, more manageable tasks. These tasks are then added to the backlog of our GitHub-project issue board. Our issue board consists of five sections, “Backlog”, “In progress”, “Deadline [XX.XX.XXXX]”, “In review” and “Done”.

As mentioned, a new task is initially added to the backlog. When a group member starts work on a task they move it into “In progress”. Usually this is done during a scrumban meeting, but if someone decides they want to work more, then they can move it themselves to signify that it's being worked on to the rest of the group. At the same time as moving it, a task is assigned to whoever is working on it as well as being given a size and priority tag. More often than not we will move a task straight from the backlog into the “Deadline” category. The reason for this is that “Deadline [XX.XX.XXXX]” signifies when the next meeting, and delivery date is. We update the x’es to display the date of our next meeting. Thus a task in this category is expected to

be completed by that date. When a task has been completed it will be moved into “In review”. In our scrumban meetings we go through all tasks in “In review” and decide whether or not they have been completed up to the groups standards, if yes, the task is closed and moved to “Done”. However, if a task needs more work it will be moved back into either “In review” or “Deadline”, depending on the priority and size of the task.



Visualized progression of a feature’s life cycle

5.3 Tools

Name	Usage level	Reason
Github	High	We are familiar with using github for version control as well as utilizing trunk based development offers code insurance.
Golang	Medium	Golang is an efficient and fast language with great external libraries. Due to it being a young language with tremendous support it is highly unlikely that it will become deprecated any time soon.
Visual Studio Code	High	Our main development IDE will be VS code due to its simplicity and large library integration. This will be used to develop the backend and the web application.
Android studio	Medium	Android studio will be used for developing the mobile application aspect of our project. This is mostly due to previous experience with the IDE as well as its capabilities for quickly testing code with its built in android emulator.
Discord	High	For day to day communication and meetings within the group we will be using discord. We have chosen to work mostly online due to living far apart and most of us developing on a stationary computer. **NOTE:** We make sure to not share any sensitive client information via normal communication channels to ensure confidentiality of the data.
Microsoft Azure	Low	For hosting our backend and database we will be using Microsoft Azure in accordance with our clients wishes.
Toggl	High	To track time spent working we will be using Toggl. Toggl makes it easy to either track time spent live while working or insert worked hours afterwards.

5.4 Plan for Inspections and Testing

We've been granted permission to conduct testing with some staff members from Cryogenetics. This is a great help for our user tests, as the employees are our main target for users in the finished product (Adults ranging between the ages of 25-60). We will conduct these tests on both our low-fidelity mobile application model as well as our first mobile- and web application draft. With the data we receive from the tests we can add, remove or alter functionality to better serve our client's needs.

During development we'll utilize Postman to test our API. With postman we can send specific requests to a server endpoint with both SQL and JSON. Postman will then display the resulting response data and code in a structured fashion which will help identify bugs more efficiently. We will also write some simple tests that can be run to check that the endpoints are functioning as expected during development.

5.5 Risk analysis at project level (identify, analyze, measures, follow-up)

We have performed an overarching risk analysis for the entire project where we have identified how likely an event is, as well as the impact of each event.

1. **Overestimated development speed**
   As a group we overestimated our own skill and development speed, causing us to fall behind on the development plan.
2. **Late and sudden changes in desired product or client wishes**
   The client realizes late into development that there has been significant miscommunication leading us to develop a product that doesn't suit their needs.
3. **Loss of documentation or source code**
   Somehow all code or documentation is lost or becomes inaccessible for a longer period of time, causing a halt in development.
4. **Leaked customer data**
   An insufficient security protocol causes client data to be accessible to anyone.
5. **Loss of group members**
   A group member becomes inaccessible or decides that they do not wish to work together, causing them to leave the group.
6. **Conflicts within the group**
   A tear in the group causes work to slow down or grind to a halt.
7. **Lack of competence**
   Insufficient competence leading to an unfinished or inferior product.
8. **Server crash**
   Temporarily stored data gets erased or server access is lost.
9. **Local data is lost**
   Local progress is lost causing a setback to previous Git commit.

		Probability		
		Low	Medium	High
Impact	Low	5 & 6		9
	Medium	1 & 4	7	
	High	2 & 3	8	

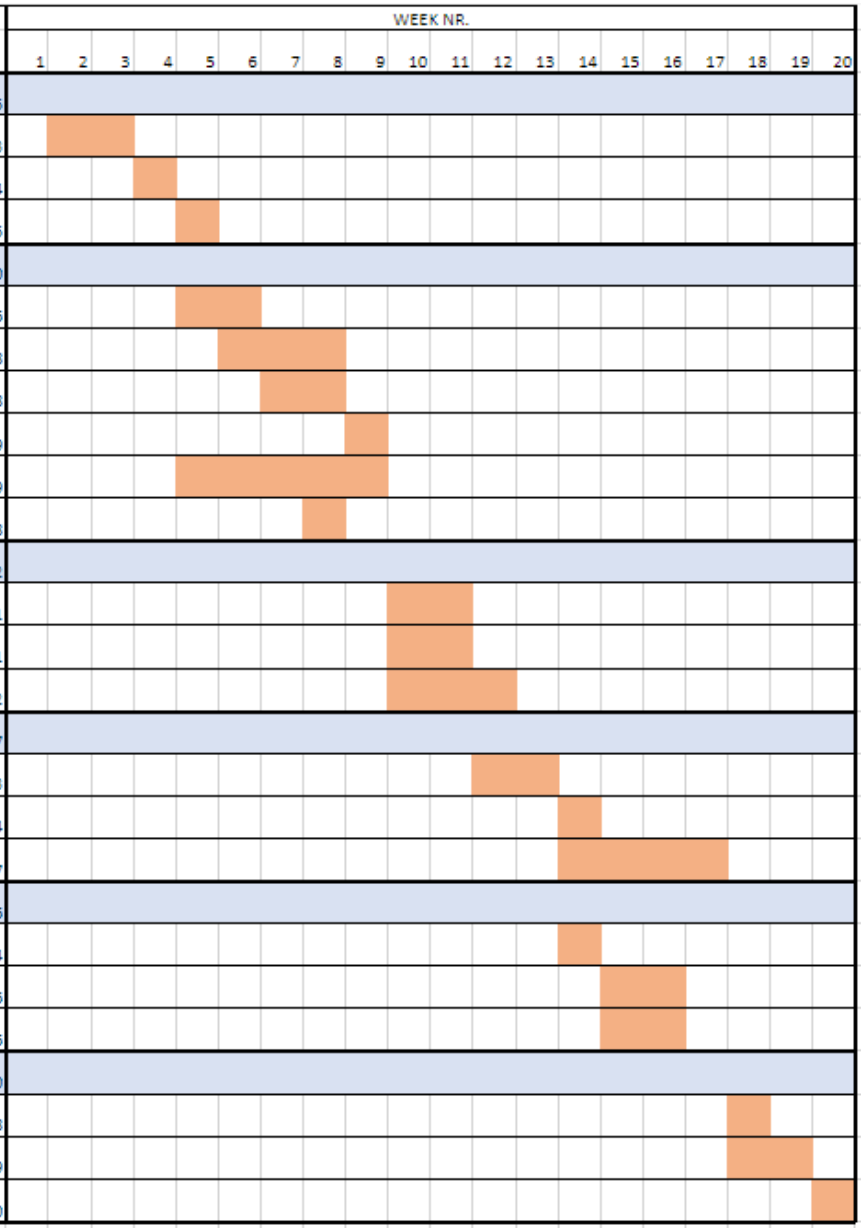
5.6 Plan for risk handling through preemptive and corrective measures

Nr	Preemptive	Corrective
1	By estimating progress based on our milestone estimates we can quickly see if we are falling behind.	Working longer hours or assigning more issues per sprint should help catch up to our desired schedule. Worst case, discuss with the client about decreasing the project's scope.
2	Through early user testing and prototyping we can clearly convey our intentions with our client to avoid miscommunication.	Initiate an emergency meeting with our client to correct our misinterpretations, and specify customer desires. Then salvaging what we can from the product and attempt to meet the client's wishes.
3	By storing backups of the source code locally on each developer's computer, we ensure that there is always an additional copy either locally or in Github's commit history.	Utilize a previous backup of the project locally stored on a group member's computer. Worst case, if all is lost contact supervisor and client in a joint meeting to discuss a solution.
4	By utilizing hashed passwords and individual ID's to access the sensitive endpoints we prevent unauthorized access. Additionally, perform regular tests to see if we can breach our own security.	First contact our client to alert them. Then, identify and fix the security flaw. Afterwards, host a meeting with our client to resolve the situation diplomatically.
5	By dividing work equally and ensuring that work is done cooperatively, there will always be someone who knows what the missing person was doing, and can	Communicate with the missing member to see if they will be gone permanently or temporarily. Worst case, reconfigure our work schedule and increase the workload of each individual member.

	thus fill their position.	
6	Regular and clear communication about issues ensures that there is no festering disagreements among group members that could escalate further.	In the case of a full blown conflict among group members, we will host a special meeting to discuss the problem and clear the air between the offending group members.
7	Tight knit cooperation and development with other group members will ensure that knowledge and expertise is shared in the case of a group member struggling with progressing their appointed task.	Lack of competence will hopefully be noticed during a weekly scrumban meeting when a member's work is not on par with expectations. The task will then be made cooperative so that knowledge can be shared. Worst case, a member will be given more time to research and acquire the needed expertise
8	Microsoft Azure permits saving temporary database backups on their server. Enabling this will grant us previous versions of the server available for rollback scenarios.	If the server crashes we should be able to restore the database using a backup file. Worst case, due to the clients wishes of not having a regular database backup system, we would have to rely on the client themselves having saved a backup.
9	Frequent and atomic sized git commits will ensure that even if local data is lost, the time since the last commit is minimal.	Restoring from the previous git commit, or previous git push should not cause a huge set back. Worst case, a group member has to start working from the beginning of their work session.

6 Gantt diagram & Implementation plan

ACTIVITY	START	DURATION	END
Planning	2	4	5
Project plan draft	2	2	3
Project plan review	4	1	4
Project plan deadline	5	1	5
Development start	5	6	10
Physical database	5	2	6
Backend draft	6	3	8
Website draft	7	2	8
First status report	9	1	9
App draft	5	5	9
User testing	8	1	8
User test revision	10	3	12
Website revision	10	2	11
App revision	10	2	11
Backend complete	10	3	12
Thesis start	12	6	17
Thesis outline	12	2	13
Second status report	14	1	14
Thesis draft	14	4	17
Development end	15	2	16
User test final round	14	1	14
Website complete	15	2	16
App complete	15	2	16
Thesis end	18	3	20
Third status report	18	1	18
Thesis review	18	2	19
Thesis complete	20	1	20



GANTT Chart. Blue, bold lines indicate the start of a new development phase.

i	Activity	Start Date	End Date
1	**Planning**	09.jan	31.jan
1.1	Project plan draft	09.jan	22.jan
1.2	Project plan review	23.jan	25.jan
1.3	Project plan deadline		31.jan
2	**Development start**	26.jan	28.feb
2.1	Physical database	26.jan	02.feb
2.2	Backend draft	02.feb	13.feb
2.3	Website draft	06.feb	16.feb
2.4	First status report	17.feb	20.feb
2.5	App draft	26.jan	25.feb
2.6	User testing	20.feb	20.feb
3	**User test revision**	01.mar	22.mar
3.1	Website revision	01.mar	13.mar
3.2	App revision	01.mar	13.mar
3.3	Backend complete	01.mar	22.mar
4	**Thesis start**	14.mar	12.mai
4.1	Thesis outline	14.mar	27.mar
4.2	Second status report	28.mar	03.apr
4.3	Thesis draft	28.mar	12.mai
5	**Development end**	10.apr	21.apr
5.1	User test final round	03.apr	09.apr
5.2	Website complete	10.apr	21.apr
5.3	App complete	10.apr	21.apr
6	**Thesis end**	22.apr	17.mai
6.1	Third status report	22.apr	01.mai
6.2	Thesis review	22.apr	12.mai
6.3	Thesis complete	13.mai	17.mai

GANTT Chart as text, with more detailed start- and end dates.

7 Conceptual model