Employment contract for grupe 21

Members: Ari Maman, Lars Mikkel L. Nilsen, Trygve Jørgensen, Ingar Aasheim.

…….

The contract aims to confirm a joint duty to maintain efficient and good work. The ultimate goal is to present competent solutions to various tasks that are given to the group.

Goals

Efficiency goal

1. *Can collaborate efficiently*

   To achieve this, we will have good dialog between group members and accept communication through several channels.

2. *Practice flexibility and solution-oriented behavior*
   To achieve this, we will have proper follow-up and a status report on the work every week.

Result goals

1. Deliver all tasks on time
   To make it happen, we will carefully assign tasks to each person and make sure that they start working on them right away.

2. Achieve a certain grade
   To achieve this, we will discuss in the group how to proceed with achieving the desired grade.

3. Complete the study
   To achieve this, we will be sociable and helpful.

Roles and distribution of tasks

A. *Team leader*
   Trygve

B. *Meeing orgenizer (sending meeting notice*, preparations, management
   Lars Mikkel

C. Archivist/document manager
   Ingar

D. *Referent*
   Ari

E. Delivery manager, quality assurance of what is delivered
   Ari

Procedures

A. Meeting notices are sent via e-mail. Physical and digital meetings are possible.

B. If someone cannot come to a meeting or something bad happens, they should tell someone who will be there. If someone misses a lot of meetings without a good reason, they might be kicked out of the group, but they will be warned first.

C. Documents are shared through OneDrive, Ingar is currently responsible for formatting and processing documents. Other documents are shared.

D. All members are obliged to comply with deadlines.

Interaction

A. The use of a PC and mobile phone is acceptable, as long as it is possible to maintain a dialogue.

B. Among the group, an attempt is made to maintain a good study environment, and with joint efforts, this will lead to a good work ethic.

C. If things should not work between members of the group, or the group as a whole does not work, the members must go through and discuss what measures should be taken to change the bad patterns.

Signatur

Trygve Jørgensen 

Ari Maman 

Ingar Aasheim 

Lars Mikkel Lødeng Nilsen 

Tasks	Start	Finish	Deadlines	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13	Week 14	Week 15	Week 16	Week 17	Week 18
Set application requirements with customer	09.01.2023	06.02.2023																		
Create and sign employment contract	09.01.2023	06.02.2023	**10.02.2023**																	
Set up a new project in GitLab	09.01.2023	06.02.2023																		
Draw Wireframe Prototypes	06.02.2023	26.02.2023	**03.03.2023**																	
Write Vision Document	06.02.2023	26.02.2023	**03.03.2023**																	
Draw Domain-model	06.02.2023	26.02.2023	**03.03.2023**																	
Code Minimal Viable Product (MVP)	06.02.2023	19.03.2023	**24.03.2023**																	
Prepare & Conduct User Test (MVP)	27.02.2023	19.03.2023																		
Prepare & Conduct User Test (Application)	08.03.2023	10.04.2023																		
Write Wiki	06.02.2023	24.04.2023	**28.04.2023**																	
Code Final Product	20.03.2023	24.04.2023	**28.04.2023**																	
Documentation	09.01.2023	24.04.2023	**28.04.2023**																	
Prepare and practice presentations	29.04.2023	30.04.2023	**01.05.2023**																	

Legend	Description
	External deadline
	Internal deadline
	Planned work time
	Planned extra time
Start	Actual date of starting a task
Finsh	Actual date of finishing a task
***Each sqaure in the diagram represents a day in the week**	

Summary of timesheets for project:

Week no	Ari	Ingar	Lars	Trygve	Sum hours week
Week 5	3	3	3	3	**12**
Week 6	2	2	2	3	**9**
Week 7	4	4	4	4	**16**
Week 8	2.5	4	4	4	**14.5**
Week 9	3	8	8.5	8.5	**28**
Week 10	6	6	6	7	**25**
Week 11	6.5	5.5	4.5	5.5	**22**
Week 12	10.5	10	9	10.5	**40**
Week 13	3.5	9	0.5	0	**13**
Week 14	14	4	0	0	**18**
Week 15	8	12.5	3.5	4	**28**
Week 16	13	20.5	22.5	18	**74**
Week 17	33	20	44	46	**143**
Total sum hours pr person	**109**	**108.5**	**111.5**	**113.5**	**442.5**

Summary of hours by activity

Activity	Ari	Ingar	Lars	Trygve	Total sum hours pr activity
User testing	2	0	0	0	**2**
Coding	72	31	58	64	**225**
Testing of code	0	0	0	0	**0**
Project reporting	22	64.5	28.5	28	**143**
Diagrams and Models	0	0	14	0	**14**
Team meetings	12	12	9.5	11.5	**45**
Wireframe	0	0	0	8	**8**
Team meetings with supervisor	1	1	1.5	2	**5.5**
Presentation including preparation	0	0	0	0	**0**
Total sum hours	**109**	**108.5**	**111.5**	**113.5**	**442.5**

Summary of hours by category

Category	Ari	Ingar	Lars	Trygve	Total sum hours pr category
Documentation	5	53	38	28	**124**
Administration	12	9	10.5	11	**42.5**
Prototyping	0	0	0	8	**8**
Product development	92	46.5	63	66.5	**268**
Illness	0	0	0	0	**0**
Total sum hours	**109**	**108.5**	**111.5**	**113.5**	**442.5**

Team - data *Fill inn names for each team-member and adjust activities and categories according to your team needs in the project*

Name - team-members
Ari
Ingar
Lars
Trygve

Activities
User testing
Coding
Testing of code
Project reporting
Diagrams and Models
Team meetings
Wireframe
Team meetings with supervisor
Presentation including preparation

Categories
Documentation
Administration
Prototyping
Product development
Illness

Week 5

Timesheet		Ari
Activity	**Category**	**Duration (hours)**
Team meetings	Adminstration	3.0
Week 5		**3.0**

Timesheet		Ingar
Activity	**Category**	**Duration (hours)**
Team meetings	Adminstration	3.0
Week 5		**3.0**

Timesheet		**Lars**
Activity	**Category**	**Duration (hours)**
Team meetings	Adminstration	3.0
Week 5		**3.0**

Timesheet		**Trygve**
Activity	**Category**	**Duration (hours)**
Team meetings	Adminstration	3.0
Week 5		**3.0**

Timesheet		0
Activity	Category	Duration (hours)
Week 5		0.0

Timesheet		0
Activity	Category	Duration (hours)
Week 5		0.0

Weekly status report

Week 5 was the groups first working week, with the project assignement being handed out on Monday. The work agreement contract and team roles were resolved in this week.

Team infrastructure was set up by creating groups for communication, documentation and coding.

The chosen platforms were messenger, onedrive and gitlab.

Week 6

Timesheet		Ari
Activity	**Category**	**Duration (hours)**
Team meetings	Adminstration	2.0
Week 6		**2.0**

Timesheet		Ingar
Activity	**Category**	**Duration (hours)**
Team meetings	Adminstration	2.0
Week 6		**2.0**

Timesheet		**Lars**
Activity	**Category**	**Duration (hours)**
Team meetings	Adminstration	2.0
Week 6		**2.0**

Timesheet		**Trygve**
Activity	**Category**	**Duration (hours)**
Team meetings	Adminstration	3.0
Week 6		**3.0**

Timesheet		0
Activity	Category	Duration (hours)
Week 6		0.0

Timesheet		0
Activity	Category	Duration (hours)
Week 6		0.0

Weekly status report

We defined the scrum sprint times, scrum roles, and contacted the student assistant for a meeting.
Forked gitlab template project and imported issues from original project.

Week 7

Timesheet		Ari
Activity	**Category**	**Duration (hours)**
Team meetings	Adminstration	3.0
Team meetings	Product development	1.0
Week 7		**4.0**

Timesheet		Ingar
Activity	**Category**	**Duration (hours)**
Team meetings	Adminstration	3.0
Team meetings	Product development	1.0
Week 7		**4.0**

Timesheet		**Lars**
Activity	**Category**	**Duration (hours)**
Team meetings	Adminstration	3.0
Team meetings	Product development	1.0
Week 7		**4.0**

Timesheet		**Trygve**
Activity	**Category**	**Duration (hours)**
Team meetings	Adminstration	3.0
Team meetings	Product development	1.0
Week 7		**4.0**

Timesheet		0
Activity	Category	Duration (hours)
Week 7		0.0

Timesheet		0
Activity	Category	Duration (hours)
Week 7		0.0

Weekly status report

Had our first meeting with the client. This was done through discord, which allows for both text and live meeting communication.
Got information on client needs, financial situiation, job etc. in addtition to creating a list of all the product features the client
Also started formatting the vision document to fit

Week 8

Timesheet		Ari
Activity	**Category**	**Duration (hours)**
Project reporting	Product development	2.0
Team meetings	Adminstration	0.5
		III
Week 8		**2.5**

Timesheet		Ingar
Activity	**Category**	**Duration (hours)**
Project reporting	Product development	4.0
Week 8		**4.0**

Timesheet		**Lars**
Activity	**Category**	**Duration (hours)**
Project reporting	Product development	1.0
Project reporting	Product development	3.0
Week 8		**4.0**

Timesheet		**Trygve**
Activity	**Category**	**Duration (hours)**
Wireframe	Prototyping	4.0
Week 8		**4.0**

Timesheet		0
Activity	Category	Duration (hours)
Week 8		0.0

Timesheet		0
Activity	Category	Duration (hours)
Week 8		0.0

Weekly status report

Begun work on product wireframe, domain model and vision documtent. Setup meeting with studen assistant for the 27th.

Week 9

Timesheet		Ari
Activity	**Category**	**Duration (hours)**
Project reporting	Product development	3.0
Week 9		**3.0**

Timesheet		Ingar
Activity	**Category**	**Duration (hours)**
Project reporting	Product development	7.5
Team meetings	Product development	0.5
Week 9		**8.0**

Timesheet		**Lars**
Activity	**Category**	**Duration (hours)**
Diagrams and Models	Documentation	4.0
Team meetings with supervisor	Adminstration	0.5
Diagrams and Models	Documentation	4.0
Week 9		**8.5**

Timesheet		**Trygve**
Activity	**Category**	**Duration (hours)**
Wireframe	Prototyping	4.0
Team meetings with supervisor	Adminstration	0.5
Project reporting	Documentation	4.0
Week 9		**8.5**

Timesheet		0
Activity	Category	Duration (hours)
Week 9		0.0

Timesheet		0
Activity	Category	Duration (hours)
Week 9		0.0

Weekly status report

Finished part 1, meaing wireframe, domain model and vision document was delivered on Friday. Also had a meeting with our student assistand

Week 10

Timesheet		Ari
Activity	**Category**	**Duration (hours)**
Coding	Product development	6.0
Week 10		**6.0**

Timesheet		Ingar
Activity	**Category**	**Duration (hours)**
Coding	Product development	6.0
Week 10		**6.0**

Timesheet		**Lars**
Activity	**Category**	**Duration (hours)**
Coding	Product development	6.0
Week 10		**6.0**

Timesheet		**Trygve**
Activity	**Category**	**Duration (hours)**
Coding	Product development	7.0
Week 10		**7.0**

Timesheet		0
Activity	**Category**	**Duration (hours)**
Week 10		**0.0**

Timesheet		0
Activity	**Category**	**Duration (hours)**
Week 10		**0.0**

Weekly status report

Started on proper product by creating the propgrams class structure according to the domain model

Week 11

Timesheet		Ari
Activity	**Category**	**Duration (hours)**
Team meetings with supervisor	Adminstration	0.5
Team meetings	Adminstration	1.0
Coding	Product development	5.0
Week 10		**6.5**

Timesheet		Ingar
Activity	**Category**	**Duration (hours)**
Team meetings with supervisor	Adminstration	0.5
Coding	Product development	1.5
Team meetings	Product development	1.5
Project reporting	Documentation	2.0
Week 10		**5.5**

Timesheet		**Lars**
Activity	**Category**	**Duration (hours)**
Diagrams and Models	Documentation	2.0
Team meetings with supervisor	Adminstration	0.5
Coding	Product development	2.0
Week 10		**4.5**

Timesheet		**Trygve**
Activity	**Category**	**Duration (hours)**
Team meetings with supervisor	Adminstration	0.5
Coding	Product development	2.0
Team meetings	Product development	1.0
Coding	Product development	2.0
Week 10		**5.5**

Timesheet		0
Activity	**Category**	**Duration (hours)**
Week 10		**0.0**

Timesheet		0
Activity	**Category**	**Duration (hours)**
Week 10		**0.0**

Weekly status report

Started on the GUI of the MVP

Week 12

Timesheet		Ari
Activity	**Category**	**Duration (hours)**
Team meetings with supervisor	Adminstration	0.5
Coding	Product development	3.0
Coding	Product development	7.0
Week 10		**10.5**

Timesheet		Ingar
Activity	**Category**	**Duration (hours)**
Coding	Product development	10.0
Week 10		**10.0**

Timesheet		**Lars**
Activity	**Category**	**Duration (hours)**
Coding	Product development	5.0
Coding	Product development	4.0
Week 10		**9.0**

Timesheet		**Trygve**
Activity	**Category**	**Duration (hours)**
Team meetings	Product development	0.5
Coding	Product development	10
Week 10		**10.5**

Timesheet		0
Activity	**Category**	**Duration (hours)**
Week 10		**0.0**

Timesheet		0
Activity	**Category**	**Duration (hours)**
Week 10		**0.0**

Weekly status report

Finished the MVP

Week 13

Timesheet		Ari
Activity	**Category**	**Duration (hours)**
Team meetings	Adminstration	0.5
Coding	Product development	3.0
Week 10		**3.5**

Timesheet		Ingar
Activity	**Category**	**Duration (hours)**
Team meetings with supervisor	Product development	0.5
Coding	Product development	3.5
Project reporting	Documentation	5.0
Week 10		**9.0**

Timesheet		**Lars**
Activity	**Category**	**Duration (hours)**
Team meetings with supervisor	Adminstration	0.5
Week 10		**0.5**

Timesheet		**Trygve**
Activity	**Category**	**Duration (hours)**
Team meetings with supervisor	Product development	0.5
Week 10		**0.0**

Timesheet		0
Activity	**Category**	**Duration (hours)**
Week 10		**0.0**

Timesheet		0
Activity	**Category**	**Duration (hours)**
Week 10		**0.0**

Weekly status report

Presented MVP to supervisor

Week 14

Timesheet		Ari
Activity	**Category**	**Duration (hours)**
User testing	Product development	2.0
Coding	Product development	5.0
Coding	Product development	7.0
Week 10		**14.0**

Timesheet		Ingar
Activity	**Category**	**Duration (hours)**
Coding	Product development	4.0
Week 10		**4.0**

Timesheet		Lars
Activity	Category	Duration (hours)
Week 10		0.0

Timesheet		Trygve
Activity	Category	Duration (hours)
Week 10		0.0

Timesheet		0
Activity	**Category**	**Duration (hours)**
Week 10		**0.0**

Timesheet		0
Activity	**Category**	**Duration (hours)**
Week 10		**0.0**

Weekly status report

Easter break

Week 15

Timesheet		Ari
Activity	**Category**	**Duration (hours)**
Coding	Product development	3.0
Coding	Product development	5.0
Week 10		**8.0**

Timesheet		Ingar
Activity	**Category**	**Duration (hours)**
Team meetings	Product development	0.5
Project reporting	Documentation	6.0
Coding	Product development	6.0
Week 10		**12.5**

Timesheet		**Lars**
Activity	**Category**	**Duration (hours)**
Project reporting	Adminstration	0.5
Coding	Product development	2.0
Project reporting	Documentation	1.0
Week 10		**3.5**

Timesheet		**Trygve**
Activity	**Category**	**Duration (hours)**
Coding	Product development	4.0
Week 10		**4.0**

Timesheet		0
Activity	**Category**	**Duration (hours)**
Week 10		**0.0**

Timesheet		0
Activity	**Category**	**Duration (hours)**
Week 10		**0.0**

Weekly status report

Started on main rapport, and continuing development to finalize project

Week 16

Timesheet		Ari
Activity	**Category**	**Duration (hours)**
Team meetings	Adminstration	1.0
Project reporting	Product development	4.0
Project reporting	Product development	5.0
Project reporting	Product development	3.0
Week 10		**13.0**

Timesheet		Ingar
Activity	**Category**	**Duration (hours)**
Project reporting	Documentation	20.0
Team meetings	Adminstration	0.5
Week 10		**20.5**

Timesheet		**Lars**
Activity	**Category**	**Duration (hours)**
Project reporting	Documentation	3.0
Coding	Product development	4.0
Coding	Product development	5.0
Team meetings	Adminstration	0.5
Coding	Product development	3.0
Coding	Product development	4.0
Coding	Product development	3.0
Week 10		**22.5**

Timesheet		**Trygve**
Activity	**Category**	**Duration (hours)**
Coding	Product development	6.0
Coding	Product development	5
Team meetings with supervisor	Adminstration	1.0
Coding	Product development	6.0
Week 10		**18.0**

Timesheet		0
Activity	**Category**	**Duration (hours)**
Week 10		**0.0**

Timesheet		0
Activity	**Category**	**Duration (hours)**
Week 10		**0.0**

Weekly status report

Finished the MVP

Week 17

Timesheet		Ari
Activity	**Category**	**Duration (hours)**
Coding	Product development	10.0
Project reporting	Documentation	5.0
Coding	Product development	12.0
Coding	Product development	6.0
Week 10		**33.0**

Timesheet		Ingar
Activity	**Category**	**Duration (hours)**
Project reporting	Documentation	20.0
Week 10		**20.0**

Timesheet		**Lars**
Activity	**Category**	**Duration (hours)**
Coding	Product development	6.0
Coding	Product development	6.0
Coding	Product development	8.0
Diagrams and Models	Documentation	4.0
Project reporting	Documentation	20.0
Week 10		**44.0**

Timesheet		**Trygve**
Activity	**Category**	**Duration (hours)**
Coding	Product development	8.0
Coding	Product development	8
Coding	Product development	6.0
Project reporting	Documentation	24.0
Week 10		**46.0**

Timesheet		0
Activity	**Category**	**Duration (hours)**
Week 10		**0.0**

Timesheet		0
Activity	**Category**	**Duration (hours)**
Week 10		**0.0**

Weekly status report

Finalizing the project and main report

Møteinnkalling nr. 1 for Gruppe 21

Møteinnkallingen går til:

Nils William Lie, Ingar Solveigson Asheim, Ari Maman, Trygve Jørgensen og Lars Mikkel Lødeng Nilsen

Tid og sted: mandag 27.02.2023 klokken 13:40-14:10 NTNU Gløshaugen Realfagsbygget A4-112.

Agenda:

Saksnr.	Saker	Tid	Beslutning	Ansvarlig person
1	Godkjenning av møteinnkalling	1 min		Lars Mikkel
2	Planlagt produkt	10 min		Trygve
3	Domene modell	5 min		Lars Mikkel
4	GUI	5 min		Trygve
5	Visjonsdokument	5 min		Ari og Ingar
6	Orientering om status/fremdrift	5 min		Trygve

Send melding til gruppen om du ikke har mulighet til å delta!

Velkommen!

Mvh Lars Mikkel Lødeng Nilsen

Møteinnkalling Team-møte 1 for gruppe 21

Møteinnkallingen går til:

Surya Kathayat, Ingar Solveigson Asheim, Ari Maman, Trygve Jørgensen og Lars Mikkel Lødeng Nilsen

Tid og sted: mandag 13.03.2023 klokken 11:50-12:10 NTNU Gløshaugen IT-bygget, sydfløy 1. etasje Møterom 119b

Agenda:

Saksnr.	Saker	Tid	Beslutning	Ansvarlig person
1	Generelt om applikasjonen	3 min		Trygve
2	Domene modell	2 min		Lars Mikkel
3	Wireframes	3 min		Trygve
4	Visjonsdokument	8 min		Ari og Ingar
5	Orientering om status/fremdrift	4 min		Trygve

Si ifra om du ikke har mulighet til å delta!

Mvh Lars Mikkel Lødeng Nilsen

Møteinnkalling nr. 3 for Gruppe 21

Møteinnkallingen går til:

Nils William Lie, Ingar Solveigson Asheim, Ari Maman, Trygve Jørgensen og Lars Mikkel Lødeng Nilsen

Tid og sted: mandag 20.03.2023 klokken 13:00-13:30. NTNU Gløshaugen Realfagsbygget A4-112.

Agenda:

Saksnr.	Saker	Tid	Beslutning	Ansvarlig person
1	Diverse kode	30 min		

Send melding om du ikke har mulighet til å delta!

Velkommen!

Mvh Lars Mikkel Lødeng Nilsen

Møteinnkalling Team-møte 2 for gruppe 21

Møteinnkallingen går til:

Surya Kathayat, Ingar Solveigson Asheim, Ari Maman, Trygve Jørgensen og Lars Mikkel Lødeng Nilsen

Tid og sted: mandag 28.03.2023 klokken 09:30-09:50 (20 min). NTNU Gløshaugen IT-bygget, sydfløy 1. etasje Møterom 119c

Agenda:

Saksnr.	Saker	Tid	Beslutning	Ansvarlig person
1	MVP	10 min		
2	Diverse kode	5 min		
3	WIKI	5 min		

Si ifra om du ikke har mulighet til å delta!

Mvh Lars Mikkel Lødeng Nilsen

Møteinnkalling for Gruppe 21 onsdag 19. april

Møteinnkallingen går til:

Nils William Lie, Ingar Solveigson Asheim, Ari Maman, Trygve Jørgensen og Lars Mikkel Lødeng Nilsen

Tid og sted: onsdag 19.04.2023 klokken 16:00-16:30. NTNU Gløshaugen Realfagsbygget A4-112.

Ingen ønsker om å ta opp saker til agendaen har blitt mottatt så langt. Som en konsekvens vil vi kun ha ett punkt på agendaen, som vil være "diverse". Vi vil likevel oppfordre alle deltakere til å bruke anledningen til å ta opp eventuelle saker eller spørsmål de måtte ha under møtet.

Agenda:

Saksnr.	Saker	Tid	Beslutning	Ansvarlig person
1	Diverse	30 min		

Vennligst møt opp i tide og vær forberedt på å delta i diskusjonen. Hvis du har spørsmål eller innspill til agendaen, vennligst gi meg beskjed.

Mvh Lars Mikkel Lødeng Nilsen

Møteinnkalling for Gruppe 21 onsdag 26. april

Møteinnkallingen går til:

Nils William Lie, Ingar Solveigson Asheim, Ari Maman, Trygve Jørgensen og Lars Mikkel Lødeng Nilsen

Tid og sted: onsdag 26.04.2023 klokken 16:00-16:30. NTNU Gløshaugen Realfagsbygget A4-112.

Agenda:

Saksnr.	Saker	Tid	Beslutning	Ansvarlig person
1	Kode	10 min		
2	Rapport	10 min		
3	Diverse	10 min		

Vennligst møt opp i tide og vær forberedt på å delta i diskusjonen. Hvis du har spørsmål eller innspill til agendaen, vennligst gi meg beskjed.

Mvh Lars Mikkel Lødeng Nilsen

Meeting notice 16.02

- Discussed the customer's work tasks, expenses, and income, both private and as a "company."
- Considered if the task description was too narrow and if we should think bigger.
- Explored various areas where the customer could use our program.
- Planning to create a split in the app, with one part for private budgeting and another for businesses. Also, the app should be able to review accounts retrospectively.
- Examined areas where the customer spends money privately and as a business to categorize them.
- Discussed the visual aspect with Tormod, who wanted a feature to track project progress.

Meeting minutes 27.02.2023

- Meeting invitation looks good
- Gave the teaching assistant an overview of the customer, product plan, budgeting (private and corporate), and backlog overview
- Discussion about how to deliver wireframe and vision document
- Went through the wireframe
- Recommendation to narrow down and build upon instead of having a too broad vision
- Recommendation to focus on universal design
- Questions about the vision document
- Impact goals - don't need to be more comprehensive
- Write generally in part 2
- Should everything be in English? Yes, GUI too
- Summary of capabilities, is it enough? It was good enough.
- No need to write redundantly. But the teaching assistant still got 18 pages? Bruh.
- Maybe the submissions will be pushed back?
- Maybe no meeting next week.
- The vision document can be updated after submission.

Meeting with Surya on 13.03

Participants: Surya Kathayat, Ingar Solveigson Asheim, Trygve Jørgensen and Lars Mikkel Lødeng Nilsen

Time and Place: Monday, 13.03.2023 at 11:50-12:10 NTNU Gløshaugen IT-building, south wing 1st floor Meeting room 119b

Agenda

Casenr.	Cases	Time	Reponible person
1	General about the application	3 min	Trygve
2	Domain model	2 min	Lars Mikkel
3	Wireframes	3 min	Trygve
4	Vision document	8 min	Ari og Ingar
5	Status/progress update	4 min	Trygve

Case no. 1:

The domain model and wireframes were presented along with a clear description of the application.

Case no. 2:

The domain model was well-made, but it contained elements that should not be included in a domain model. This was technical information such as which interfaces we should implement in the code. The domain model should be made in such a way that it can be understood by non-technical people.

Case no. 3:

Review of what should be included in the MVP versus what we want to implement in the full application. Surya commented that it is important to continuously conduct user testing at several stages in the development process to be able to correct the design if necessary (wireframes, after MVP, first draft of final product, etc.). It was decided that Ari will conduct a user test of wireframes with Tormod (the customer) in the near future.

Case no. 4:

The vision document was mainly good, but should be complemented with some points: risk analysis, adding the remaining stakeholders and why the product is important to create.

Stakeholders include everyone involved in the project, including teachers, teaching assistants, developers, and the customer.

Case no. 5:

We are on track to complete the MVP by 24.03.23.

Other questions and comments:

We need to start using Scrum, as it is one of the most important learning points in the course. The vision document will be finalized after feedback from this meeting. Conduct user tests and create more iterations of wireframes.

Meeting minutes 20.03.2023

- 5 submissions of MVP
- Discussion of project scope narrowing
- Assigned work tasks
- Code assistance
- TA thinks that the class structure between months and per project looks good, even without an interface.

Meeting with Surya on 28.03

- First 5 minutes were spent on delays and technical issues
- Went through the MVP
- Surya seems positive about the MVP, "it's good"
- Look into universal design
- Confirmation dialog box for actually deleting something
- Clear symbols, must be consistent and try to have them everywhere
- Went through the wiki
- Include text description on some of the diagrams, but not all are necessary, just the most basic principles
- Domain model doesn't need to be thought of as a class diagram, it looks good
- Maybe projectregistry and monthlyregistry aren't necessary?
- Maybe a bit difficult to understand when there are many arrows crossing, can be fixed by changing orientation
- Main is not an object
- Make it a bit more clear
- Use case diagram can also include some text
- Don't use class diagram from IntelliJ since it gets rotated
- The one we have now looked good, not all classes need to be included, the same goes for methods
- Pipeline check, 0 tests, a bit suspicious?
- Gitlab-ci file is not in the correct directory for testing classes
- Brief review of pipelines
- Fix Gitlab Pages
- Make sure to write scrum summaries
- Create a handbook
- Meeting went over by 15 minutes

Meeting minutes 16.04

- Some focus on universal design
- Language options Text is most important
- Contrasts WCAG color calculator
- Write in the report that we have designed with universal design in mind
- ChatGPT can be used as a tool for programming and explaining concepts, but it is not allowed to write the text through ChatGPT
- Same methods in different classes can be a problem
- Methods for an object should only affect the object itself "Follow the list downwards"
- Be mindful of encapsulation
- We may need to upload Javadoc manually The Jar file should be created in the target, but target is in gitignore Jar must be visible and runnable Jar is difficult (according to teaching assistant)
- List of requirements for what needs to be delivered has been posted, we don't need to stress about it Keep things consistent (setters)
- Explain everything, justify justify justify

Meeting minutes 26.04

- Program presentation was made.
- There has been a lot of crunching, so testing is a bit behind.
- JavaFX has been fixed.
- They are trying to follow the Model-View-Controller design.
- The Controller should not store information, but they can push the boundaries a bit as it seems to be working fine.
- They can divide "Update Transaction" and "Add Transaction," and it looks good.
- The text size could be a bit bigger or adjustable.
- The buttons are fine, but the background could be more interesting.
- An "About" section and a quick user manual (from Wiki) are needed.
- Should "Dark Mode" be the default?
- System requirements should be included.
- There was poor source referencing in an example, which they should not do, but they have good source tracking.
- They discussed what is important in a report but often gets forgotten: documentation, diagrams, explanations, and reasons for changes.
- Only one person needs to send the entire assignment, but individuals need to make sure they are in the correct group, and the reflection note needs to be sent individually through Inspera.
- A TA has no idea what it is all about.
- They can contact Surya and Ali regarding the exam, not Grethe.
- They need a link to the GitLab issue board, and they have taken screenshots for the report.
- They are discussing whether to continue as they have been or to have a popup for an error message. The TA suggests making the text more visible.
-

Sprint backlog 0

Doing 5

Write project report
#67

Flesh out System part of wiki
#64

Polish GUI
#94

Fix and write new tests
#93

Fix add image can add other files
#96

Closed 67

Create JAR file
#95

Fix checkstyle warnings
Doing
#92

Convert all controller methods to @FXML methods
Doing
#83

Update javadocs
Doing
#91

Ensure exceptionhandling presents user with understanable error messages(aka no e.printstacktrace)
Doing
#81

GUI update
Doing
#90

Write summation/documents of user test
#79

Ensure deep copying is implemented for best

Sprint backlog 11

- Refactor Bookkeeping to only use transaction instead of income and expense — #88
- Refactor project cohesion and coupling(I.e user addProject vs user getProjectRegistry.addProject) — #87
- Cleanup(javadoc, warnings, checkstyle etc.) — #86
- Convert all controller methods to @FXML methods — #83
- Add MonthlyBookkeeping GUI — #72
- Add MonthlyBookkeeping controller — #73
- WCAG 2.1 Principle 1: Perceivable — #62
- Add MonthlyBookkeepingRegistry controller — #75

Doing 6

- Begin project report — #67
- Flesh out System part of wiki — #64
- Color code projects — #85
- Make dark theme — #84
- Refactor user class to singelton — #63
- Add MonthlyBookkeepingRegistry GUI — #74

Closed 49

- Add image functionality to view project — #82
- Add create new category functionality — #76
- Add images to project functionality — #77
- Add MonthlyBookkeeping class — #69
- Add slideshow functionality to viewProject — #68
- Add MonthlyBookkeepingRegistry class — #70
- Fix warninglabel on new and edit project pages — #71
- Cleanup Project(javadoc, checkstyle, etc.) — #56
- Fix and test equals methods — #57
- Add relevant tests — #51

Sprint backlog 7

- Fix and test equals methods — #57
- Cleanup Project(javadoc, checkstyle, etc.) — #56
- Add slideshow functionality to viewProject — #68
- Fix warninglabel on new and edit project pages — #71
- Add MonthlyBookkeepingRegistry class — #70
- Add MonthlyBookkeeping class — #69
- Add images to project functionality — #77

Doing 6

- Add EditProject to GUI — #53
- Integrate all controller classes — #55
- Cleanup and align GUI — #58
- Fix "new project -> save -> view project" glitch — #61
- Add "Are you sure?" popups — #60
- Add relevant tests — #51

Closed 34

- Refactor and optimize package structure — #42
- Add ViewProject to GUI — #48
- Add functionality for storing a projectRegistry — #54
- Add NewProject to GUI — #47
- Add AllProjects to GUI — #46
- Add testing to classes — #41
- Finish backend according to MVP — #40
- Add teachers and customer as stakeholders in vision document — #37
- Finish MVP domain model — #38
- Create class diagram — #39

Sprint backlog 7

- Add "Are you sure?" popups
  #60
- Cleanup and align GUI
  #58
- Add relevant tests
  #51
- Cleanup Project(javadoc, checkstyle, etc.)
  #56
- Fix "new project -> save -> view project" glitch
  #61
- Fix and test equals methods
  #57
- Integrate all controller classes
  #55

Doing 5

- Add NewProject to GUI
  #47
- Refactor and optimize package structure
  #42
- Add ViewProject to GUI
  #48
- Add EditProject to GUI
  #53
- Add functionality for storing a projectRegistry
  #54

Closed 30

- Add AllProjects to GUI
  #46
- Add testing to classes
  #41
- Finish backend according to MVP
  #40
- Add teachers and customer as stakeholders in vision document
  #37
- Finish MVP domain model
  #38
- Create class diagram
  #39
- User testing
  #35
- Finalize Vision Document
  #36
- Create Bookkeeping Interface
  #33
- Fix Accounting
  #31

Sprint backlog 4

- Finish backend according to MVP
  #40
- Refactor and optimize package structure
  #42
- Add AllProjects to GUI
  #46
- Add testing to classes
  #41

Doing 5

- Add teachers and customer as stakeholders in vision document
  #37
- Finish MVP domain model
  #38
- Add NewProject to GUI
  #47
- Create class diagram
  #39
- User testing
  #35

Closed 23

- Finalize Vision Document
  #36
- Create Bookkeeping Interface
  #33
- Fix Accounting
  #31
- Fix Budgeting
  #32
- Make a transfer catalog class and make accounting and budgeting implement this class
  #25
- Add functionality to Project class
  #30
- Create Expense class
  #26
- Refactor Income, Expense and Transaction
  #27
- Refactor Registry class
  Finished
  #24
- Create Accounting class
  #21

Sprint backlog 4

- Finish backend according to MVP
  #40
- Refactor and optimize package structure
  #42
- Add AllProjects to GUI
  #46
- Add testing to classes
  #41

Doing 5

- Add teachers and customer as stakeholders in vision document
  #37
- Finish MVP domain model
  #38
- Add NewProject to GUI
  #47
- Create class diagram
  #39
- User testing
  #35

Closed 23

- Finalize Vision Document
  #36
- Create Bookkeeping Interface
  #33
- Fix Accounting
  #31
- Fix Budgeting
  #32
- Make a transfer catalog class and make accounting and budgeting implement this class
  #25
- Add functionality to Project class
  #30
- Create Expense class
  #26
- Refactor Income, Expense and Transaction
  #27
- Refactor Registry class
  Finished
  #24
- Create Accounting class
  #21

Sprint backlog 3

Add teachers and customer as stakeholders in vision document
#37

User testing
#35

Finalize Vision Document
#36

Doing 3

Fix Budgeting
#32

Fix Accounting
#31

Create Bookkeeping Interface
#33

Closed 19

Add functionality to Project class
#30

Create Expense class
#26

Make a transfer catalog class and make accounting and budgeting implement this class
#25

Refactor Income, Expense and Transaction
#27

Refactor Registry class
Finished
#24

Create Accounting class
#21

Create ProjectRegistry
#23

Create Project class
#18

Create Income and Expences classes
#19

Create fundamental GUI functionality
#22

Sprint backlog 3

Fix Accounting
#31

Add functionality to Project class
#30

Create Bookkeeping Interface
#33

Doing 4

Make a transfer catalog class and make accounting and budgeting implement this class
#25

Create Expense class
#26

Refactor Income, Expense and Transaction
#27

Refactor Registry class
Finished
#24

Closed 14

Create Accounting class
#21

Create ProjectRegistry
#23

Create Project class
#18

Create Income and Expences classes
#19

Create fundamental GUI functionality
#22

Create Budget class
#20

Implement transfer interface functionality
Finished
#7

Milepæler
Sprint backlog
#8

Innledning
#11

Initalize app package
Finished

Sprint backlog 1

- Make a transfer catalog class and make accounting and budgeting implement this class
  #25
- Refactor Registry class
  #24
- Create Accounting class
  #21

Doing -12

- Create fundamental GUI functionality
  #22
- Create Budget class
  #20
- Create Income and Expences classes
  #19
- Create Project class
  #18
- Create ProjectRegistry
  #23

Closed 8

- Implement transfer interface functionality
  Finished
  #7
- Milepæler
  Sprint backlog
  #8
- Innledning
  #11
- Initalize app package
  Finished
  #10
- Wireframe
  Finished
  #1 Mar 3
- Vision document
  Finished
  #2 Mar 3
- Domain model
  Finished
  #3 Mar 3
- User story example: As a user I would like to view a cake diagram of my spending
  Sprint backlog
  #4

Sprint backlog 8

- Ensure exceptionhandling presents user with understanable error messages(aka no e.printstacktrace) #81
- Ensure deep copying is implemented for best practice #80
- Write summation of user test #79
- WCAG 2.1 Principle 1: Perceivable #62
- Add MonthlyBookkeepingRegistry controller #75
- Add MonthlyBookkeeping controller #73
- Add MonthlyBookkeeping GUI #72
- Refactor user class to singelton #63

Doing 3

- Begin project rapport #67
- Flesh out System part of wiki #64
- Add MonthlyBookkeepingRegistry GUI #74

Closed 50

- Add image functionality to view project Doing #82
- Add create new category functionality Doing #76
- Add MonthlyBookkeepingRegistry class #70
- Add slideshow functionality to viewProject Doing #68
- Add MonthlyBookkeeping class Doing #69
- Add images to project functionality Doing #77
- Fix warninglabel on new and edit project pages Doing #71
- Cleanup Project(javadoc, checkstyle, etc.) Doing #56

Gruppe 21

Personal Banking and Budgeting Program
Vision Document

Version <1.0>

Revision History

Date	Version	Description	Author
13.02.23	0.1	No product criteria or goals set, only document formatting.	Trygve Jørgensen, Ingar Asheim, Ari Maman, Lars Mikkel Nilsen
01.03.23	0.2	Finished all sections except 1.4	Trygve Jørgensen, Ingar Asheim, Ari Maman, Lars Mikkel Nilsen
02.03.23	0.3	Completed first draft of vision document	Ingar Asheim
15.03.23	1.0	Updated based on feedback from Surya	Ingar Asheim
26.04.23	1.1	Fixed grammatical errors and spelling mistakes	Ingar Asheim

Table of Contents

Contents

Vision document

1. Introduction

1.1 Purpose and scope

This vision document is for the project in IDATT1002, and is going to describe all the parties involved, what the project revolves around, what the purpose of this software is going to be and discuss its functionality.

1.2 Definitions, Acronyms, and Abbreviations

GUI	Graphical User Interface
DevOps	A methodology in the software development and IT industry
HCI	Human Computer Interaction
Scrum	A software development methodology
JRE	Java Runtime Environment
JavaFX	A framework for developing GUIs in Java
Java 17	The 2017 version of the Java programming language

1.3 References

Oracle. (n.d.). Oracle JDK 17 Certified System Configurations. Retrieved from https://www.oracle.com/java/technologies/javase/products-doc-jdk17certconfig.html

Java. (n.d.). What are the system requirements for Java? Retrieved from https://www.java.com/en/download/help/sysreq.html

Janssen, J. (2021) It's time to move your applications to Java 17. Here's why—and how. Retrieved from https://blogs.oracle.com/javamagazine/post/its-time-to-move-your-applications-to-java-17-heres-why-and-heres-how

1.4 Overview

The vision document first outlines the goals for the project. It then provides information about the stakeholders and their requirements and obligations. Finally, it shows an overview of the product's features and requirements, as well as information regarding the development of the application.

1.5 Product Position Statement

For	Tormod Malmin
Who	wants an application for the management of personal and business finances
The Personal Banking and Budgeting Program	is a financial management program
That	provides both budgeting and accounting functionality for finances on a month-to-month basis and also for individual projects such as freelance work, vacations, parties, etc., with an easy-to-use user interface.
Unlike	spreadsheet applications
Our product	provides the functionality for financial management without the need to set up formulas and layouts. It also features a slideshow-like user interface that provides a fast and easy way for the user to get an overview of their projects.

2. Project goals

2.1 Impact goals

1. Improve customer's quality of life
2. Allow the customer to be more confident in financial decisions
3. Help the customer grow his business venture

2.2 Result goals

1. A financial management system for personal and business finance
2. Allows the user to manage finances by month
3. Allows the user to manage finances for personal and business projects
4. The system shall be finished by 28.04.2023
5. The system will be free of charge

2.3 Process goals

1. Gain experience in teamwork in software/product development
2. Learn to use version control and DevOps systems for code management
3. Learn how to use wireframes for use in HCI
4. Learn how to use Scrum as a tool to effectively do software development
5. Learn how to write proper documentation

3. Stakeholder and User Descriptions

3.1 Customer description

The customer base of this project is a sole person. This customer is a friend of one of the team members and has a carpentry business where he makes and sells ornaments, decorations and furniture out of wood. With such a narrow user base, the product will be specific to the user's needs. The purpose of this product is to be a tool that can be used to keep track of multiple projects simultaneously while maintaining a budget and possibilities for accounting afterwards.

3.2 Stakeholder Summary

Name	Description	Responsibilities
Ingar Asheim	Developer & Documentation manager	As an experienced programmer, Asheim shall provide guidance during the programming phase of development.
Trygve Jørgensen	Developer, Scrum Master & Team lead	In addition to developing the program, Jørgensen will be responsible for keeping the team's progress on track and leading meetings, both during and outside of Scrum.
Ari Maman	Developer & customer stakeholder representative	Maman is the stakeholder representing the customer and shall communicate the needs of the customer to the development team and vice versa. He will also take part in the development of the application.
Lars Mikkel Nilsen	Developer & Meeting organizer	Nilsen will be responsible for arranging meetings and will be part of the development of the program.
Tormod Malmin	Customer & End user	Malmin shall communicate his needs for the product and be the user during user tests.
Surya Kathayat	Teacher	Kathayat shall provide teaching and assistance when necessary, as well as be part of development meetings.
Grethe Sandstrak	Teacher	Sandstrak shall provide teaching and assistance when necessary.
Muhammad Ali Norozi	Teacher	Norozi shall provide teaching and assistance when necessary.

Nils William Lie	Student assistant	Lie shall provide teaching and assistance when necessary, be a link in communication between the development team and the teachers and participate in development meetings.

3.3 User Summary

Name	Description	Role under development	Stakeholder
Tormod Malmin	Customer and end user	Communicating wanted features and testing the product.	Ari Maman

3.4 User Environment

The user works from home and creates his products in the garage. How much time that is spent on a project varies based on the complexity of the product. One task cycle is going to be either a month or the time it takes for each project. This application will be a desktop application, and the customer has communicated that he possesses a computer capable of running it.

3.5 Key Stakeholder or User Needs

Need	Priority	Concerns	Current Solution	Proposed Solutions
Register a project	High			Creating a project is an isolated process where several fields can be inputted
Get an overview of project backlog	High			Create a slideshow or otherwise visually pleasing interface to get an overview.
Get a monthly update on financials	Medium			Table showing income and expenses

4. Product Overview

4.1 Product Perspective

This product is a new and self-contained system.

Microsoft Excel is a commonly used program for financial management. As a general-purpose spreadsheet software, Excel offers a wide range of functionalities, some of which may be extraneous to financial management. Owing to the intricate nature of the software, users may encounter a steep learning curve. Users also need to create their own formulas, graphs, and tables. Conversely, this program is preconfigured with essential functions for financial management, making it much easier for users to get started.

4.2 Summary of Capabilities

Table 4-1 Personal Banking and Budgeting Program

Customer Benefit	Supporting Features
Customer can quickly get a comprehensive overview of his finances	GUI design will show income and expenses
Customer will have an easier time making financial decisions	The system will show income and expenses, and calculate the difference between them
Customer will be able to manage project-level expenses	The system will have the ability to create budgeting and accounting for specific projects

4.3 Assumptions and Dependencies

The program will be written in Java 17, and the GUI will be made in JavaFX. If these become unavailable, a new programming language and graphics framework will have to be chosen, which could impact system requirements, and certain features might become impossible to implement.

4.4 Risk Analysis

Name	Measures	Likelyhood	Consequences
Developers falling ill	Assure that all members of the team are capable of performing any task that is part of development. Make sure that there is adequate documentation for other developers to step in.	Medium	Medium
Developers not doing their assigned work	Set clear goals that are achievable for the developers. Communicate with the member who is not doing their work to understand why and try to help them if necessary.	Low	High
Time constraints if features or implementation need to change	Assure that the scope of the program is achievable within the development timeframe, with margins, during planning. Use agile development methods to more easily adapt to	Medium	High

	changing requirements.		
Development is more difficult than anticipated	Building in extra time during planning. Easing requirements. Using agile development methods to more easily adapt to changing requirements.	Medium	High

5. Product Features

5.1 Budgeting

The user will have the ability to plan their budget for each month by inserting planned income and expenses into and overview. Each income and expense will have a date, description, category and price.

5.2 Accounting

Accounting is functionally identical to budgeting but allows the user to quickly track what parts of the budget have taken place, as well as tracking unexpected income/expenses that might occur.

5.3 Personal and work categorization

When either budgeting or accounting, it is also defined whether it's personal or work related. In addition, the overview also allows for viewing the total of either budgeting or accounting, where both the personal and work-related expenses and income are displayed simultaneously.

5.4 Project planning

Where several expenses are related to the same project and or the completion date is far out or unknown, planning a project maximizes organization. A project itself has its own name, general description, category, date and notes related to it and can therefore be defined as any abstract concept. Crucially, each project also has its own list of each earnings and expenditure, therefore allowing budgeting of the project itself.

5.5 Project slideshow

The projects slideshow allows the user to neatly display all the relevant information of each project and quickly go change between them. This allows for brainstorming eventual projects and clearly displaying differences between them.

5.6 Automatic saving of product state

As this program runs locally and is considerably lightweight, autosaving the program state will occur every time a change is executed. Since every change is saved, the user should never unexpectedly lose information.

6. Constraints

The GUI will be made with JavaFX, so its design is constrained by that framework’s capabilities.

7. Precedence and Priority

Based on the features discussed above on product features, the priorities will be listed as follows:

- Project planning
- Project slideshow
- Automatic saving of product state
- Accounting
- Budgeting
- Personal and work categorization

Priority is to implement functionality for management of individual products and projects. If this feature is completed, functionality for managing monthly finances will be implemented.

8. Other Product Requirements

8.1 System Requirements

The program will be written in Java 17, so both the developers and users will need operating systems compatible with this version. According to Oracle (n.d.), the certified system configurations for Java 17 include Windows 10 (64-bit), macOS 11.x (Big Sur) and 10.15.x (Catalina), Oracle Linux 8.x (64-bit), Red Hat Enterprise Linux 8.x (64-bit), SUSE Linux Enterprise Server 15 SP2+ (64-bit), Ubuntu Linux 20.04 LTS, 18.04 LTS, and 16.04 LTS (64-bit) (Oracle, n.d.). It might be possible to run Java 17 applications on older operating systems if they meet system requirements (Java, n.d.), however it might pose security risks (Janssen, 2021). The end user will need to have a JRE compatible with Java 17 installed to be able to run the program.

Java (n.d.) lists the following hardware and software requirements:

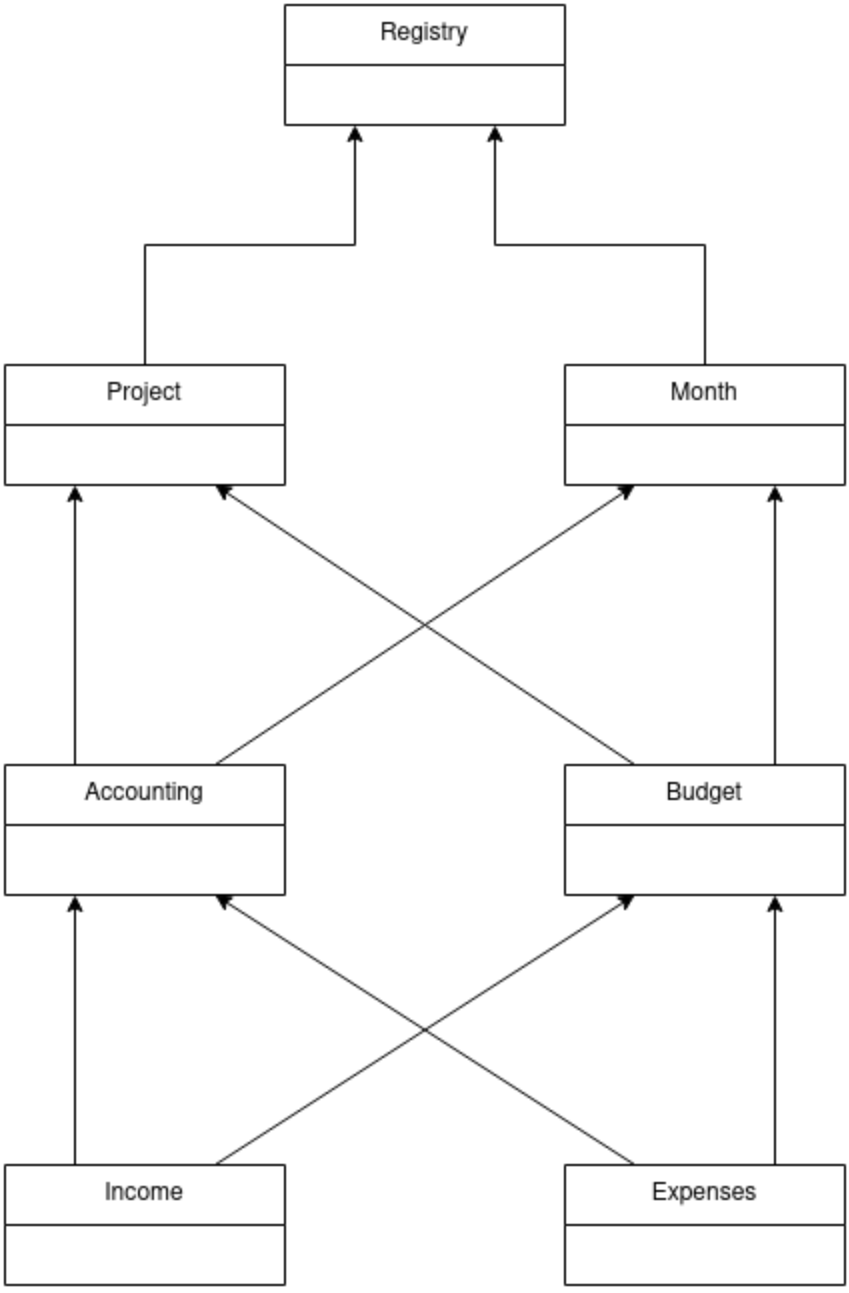
- RAM: 128 MB
- Disk space: 124 MB for JRE; 2 MB for Java Update
- Processor: Minimum Pentium 2 266 MHz processor
- Browsers: Internet Explorer 9 and above, Microsoft Edge, Firefox, Chrome

These requirements should be met by almost any modern computer available to a private consumer. The application requires a monitor, mouse, and keyboard.

8.2 Performance Requirements

The software does not retrieve or send any information to a server, because of this the application should be performant to the point where load times and actions will be experienced instantaneous to the user. File handling needs to be robust enough to avoid corruption of saved data. Tech support will not be available, and there will be no product updates, so it must be fully functional when delivered.


Main
User
- instance: User
- projectRegistry: ProjectRegistry
- monthlyBookkeepingRegistry: MonthlyBookkeepingRegistry
- User()
+ getInstance: User
+ loadUser(user: User): void
+ getProjectRegistry(): ProjectRegistry
+ getMothlyBookkeepingRegistry():
MonthlyBookkeepingRegistry
1..1
1..1
«interface»
Registry
1..1
1..1
ProjectRegistry
- projects: HashMap<LocalDateTime, Project>
- categories: List<String>
- statuses: List<String>
+ ProjectRegistry()
+ addProject(project: Project): void
+ removeProject(project: Project): void
+ updateProject(oldProject: Project, newProject. Project): void
+ getProjects(): List<Project>
+ addCategory(category: String): void
+ getCategories(): List<String>
+ removeCategory(category: String): void
+ getStatuses(): List<String>
MonthlyBookkeepingRegistry
- monthMonthlyBookkeepingMap: Map<YearMonth, MonthlyBookkeeping>
+ MonthlyBookkeepingRegistry()
+ addMonthlyBookkeeping(bookkeeping: MonthlyBookkeeping): void
+ updateMonthlyBookkeeping(bookkeeping: MonthlyBookkeeping): void
+ removeMonthlyBookkeeping(yearMonth: YearMoth): void
+ getMonthlyBookkeeping(yearMonth: YearMonth): MonthlyBookkeeping
+ isYearEmpty(yearMonth: YearMonth): boolean
+ getMonthlyBookkeepingMap(): Map<YearMonth, MonthlyBookkeeping>
1..1
1..1
0..*
0..*
Project
- accounting: Accounting
- budgeting: Budgeting
- images List<File>
- name: String
- description: String
- category: String
- status: String
- dueDate: LocalDate
+ Project(name: String, description: String, category: String, dueDate: LocalDate, status: String)
+ Project(project: Project)
+ getName(): String
+ setName(newName: String): void
+ getDescription(): String
+ setDescription(newDescription: String): void
+ getCategory(): String
+ setCategory(newCategory: String): void
+ getStatus(): String
+ setStatus(status: String): void
+ getDueDate(): LocalDate
+ getAccounting(): Accounting
+ getBudgeting(): Budgeting
MonthlyBookkeeping
- budgetingPersonal: Budgeting
- budgetingWork: Budgeting
- accountingpersonal: Accounting
- accountingWork: Accounting
- yearMonth: YearMonth
+ MonthlyBookkeeping(yearMonth: YearMonth)
+ getBookkeeping(isAccounting: boolean, isPersonal: boolean): Bookkeeping
+ getTotalBookkeeping(isAccounting: boolean): Bookkeeping
+ getYearMonth(): YearMonth
+ isEmpty(): boolean
+ equals(obj: Object): boolean
1..1
1..1
1..1
1..1
«interface»
Bookkeeping
1..1
1..1
1..1
1..1
Accounting
- incomeList: ArrayList<Income>
- expenseList: ArrayList<Expense>
+ Accounting()
+ Accounting(accounting: Accounting)
+ addTransaction(transaction: Transaction): void
+ updateTransaction(oldTrasaction: Transaction, newTransaction: Transaction): void
+ removeTransaction(transaction: Transaction): void
+ getIncomeList(): ArrayList<Income>
+ getExpenseList(): ArrayList<Expense>
+ getTransactions(): List<Transaction>
+ getTotalIncome(): double
+ getTotalExpense(): double
+ equals(obj: Object): boolean
Budgeting
- incomeList: ArrayList<Income>
- expenseList: ArrayList<Expense>
+ Budgeting()
+ Budgeting(budgeting: Budgeting)
+ addTransaction(transaction: Transaction): void
+ updateTransaction(oldTrasaction: Transaction, newTransaction: Transaction): void
+ removeTransaction(transaction: Transaction): void
+ getIncomeList(): ArrayList<Income>
+ getExpenseList(): ArrayList<Expense>
+ getTransactions(): List<Transaction>
+ getTotalIncome(): double
+ getTotalExpense(): double
+ equals(obj: Object): boolean
1..1
1..1
1..1
1..1
«interface»
Transaction
0..*
0..*
0..*
0..*
Income
- date: LocalDate
- description: String
- category: String
- amount: double
+ Income(description: String, category: String, amount: double, date: LocalDate)
+ Income(income: Income)
+ getDate(): LocalDate
+ setDate(date: LocalDate): void
+ getDescription(): String
+ setDescription(description: String): void
+ getCategory(): String
+ setCategory(category: String): void
+ getAmount(): double
+ setAmount(amount: double): void
+ equals(obj: Object): boolean
Expense
- date: LocalDate
- description: String
- category: String
- amount: double
+ Expense(description: String, category: String, amount: double, date: LocalDate)
+ Expense(expense: Expense)
+ getDate(): LocalDate
+ setDate(date: LocalDate): void
+ getDescription(): String
+ setDescription(description: String): void
+ getCategory(): String
+ setCategory(category: String): void
+ getAmount(): double
+ setAmount(amount: double): void
+ equals(obj: Object): boolean


Registry
Project
Month
Accounting
Budget
Income
Expenses