APP 2000

APPLICATION DEVELOPMENT FOR WEB

PROJECT REPORT

Kenneth Andreas Hansen
Jonas El Hbabi Helling
Lucas Leon Svaet Holter
Kristian Martin Tvenning
Mia Trollstøl

Index

Step 1 Team Formation and Project Topic Selection

1.1 Name and email of the group members.

Kristian Martin Tvenning km.tvenning@protonmail.com

Mia Trollstøl miamarietrollstol@gmail.com

Jonas El Hbabi Helling jonas.helling4@gmail.com

Kenneth Andreas Hansen Kenneth.andreas.hansen@gmail.com

Lucas Leon Svaet Holter leonholter23@gmail.com

1.2 Project topic and brief description.

Our chosen project topic is the Task management system.

"This project aims to develop a task management tool where users can create projects and tasks, assign them to team members, set deadlines, and track progress. The tool will support collaboration by allowing users to comment on tasks and mark tasks as completed. The system will also feature a dashboard with a visual representation of task statuses. It will allow for role-based access, with different views for project managers and team members."

In our version of the system, you can assign tasks to others in a group or individual and they will receive annoying notifications until they complete the task. The notices can be set to increase frequency and disturbance until the person completes the tasks. This can be done through different notification types with increasing annoyance.

1.3 Preliminary list of main features.

Create groups

Create tasks

Assign tasks to single person or groups

Set deadlines

Track progress

Annoying notifications

Step 2 Project Scope

Team: Group 8

Mia: Project manager

Kristian: Frontend

Kenneth: Frontend

Lucas: Backend

Jonas: Backend

Equal responsibility and contributions across the application for the finished product.

2.1 Project title and vision statement.

Project title: Nag

Vision: Hold users accountable for completing tasks. Combining task tracking tools with escalating notifications.

2.2 Main problem or need the application will addresses.

Do you live with someone who's procrastinating simple tasks despite wasting lots of time endlessly scrolling apps? Try Nag! This combines the want to use the phone and real-life application of using the phone as a productivity tool rather than for mindless entertainment.

The aim of the project is to address the issue that people don't do the tasks that they have agreed to do. When assigned a task the user will receive notifications escalating in obnoxiousness until the task is completed.

2.3 Benefits of the solution for the end users and key features that will achieve this.

The solution benefits those who don't want to be a nag. Many find it difficult to tell people when they are annoyed with the lack of contribution to performing a task. The system will increase accountability through active reminders that the need to perform a task is persistent. Gentle nudges will escalate to full “howlers” (Harry Potter) if the task is not completed in time. It will also be easier for a group with shared tasks (such as dorms) to monitor if the tasks are done. Despite the nature of the app as a “whip instead of carrot” one might expect people to antagonize it, but we believe people will grow to like it over time after getting into having utilized the *cue-routine-reward* mechanisms (Johannes et al., 2019; Dale et al., 2020) providing a focus shift from instant gratification from the phone – to scur away from privacy disrespect and advertising spam and to utilize it for good in one own’s life.

2.4 Design patterns, technologies, and potential pitfalls.

The usefulness of being designed as a React app is that it is platform independent and can be run through web browsers, circumventing the walled garden of proprietary app stores and it can still be “installed” on the phone and put in the app drawer and give push notifications while not having to have the browser open. The pitfall of the app is that the users need to have a willingness to not just override the whole point of the app by disabling the notifications globally, or that “Do-not-disturb”-mode would kill much of the point. There are also times when people attend lectures, work, and then there’s sleep, so there are also legitimate reasons as to why one would like to set up DND-times.

2.5 Unique requirements specific to the functionality or user experience.

- Need to set up for recurring tasks
- Need for the other participants to validate that the task is done rather than not doing the task and still mark it as checked, but for validation time limit the obnoxiousness of notifications(?)
- UX: system/dark/light mode, font size
- No big data needed, possible to P2P?
- FUDS (Asbjørnsen & Maasø, 2015) (Fritid uten dårlig samvittighet (Spare time without bad consciousness)) For the reward except for joy of having no tasks left, one could earn up “points” to claim a set time where one is “immune” to being nagged

2.6 A list of anticipated features and tools/technologies.

Task assignment tracking

This feature will allow member to assign tasks to other members and track their progress. Functions will be creating tasks, assign tasks, set due date, see status and increase nag mode.

Notifications

Notifications is a feature used to nag members with reminders to complete a task. If the member is slow, you can nag and annoy them until the task is done. The notifications can be in the application and email for this project, since we are making an application for web, not for mobile. *(Kristian's nitpicking notes: If you allow push notifications from your mobile browser you would get mobile global push notifications like normal. React is "mobile first".)*

Group tasks

It will be possible to have a group for collaboration in tasks like a student "kollektiv" with shared and rotating tasks. Here you can assign regular tasks and receive points if you are fast. So, the members of the group can be ranked to. A family can also be a group and assign chores and keep track of allowances. Functions will be to assign group tasks, shared task views and group-based notifications.

Frontend: JavaScript, React

For the user interface we will use JavaScript and React. We will try to create components and manage state efficiently for a seamless user experience.

Backend: Java

The backend programming language is Java and will handle the server-side logic, data processing, and the communication with the database.

Database: PostgreSQL

The database stores task data, room data, notification data and user information. PostgreSQL allows for precise management of the database relations. PostgreSQL guarantees ACID compliance and consistent indexing, which ensures integrity and efficiency in the application.

2.7 Any foreseeable risks and initial plans for addressing them.

The biggest risk for the group is the amount of other school projects and work. The time is limited, and we need to be effective to achieve the result we want. The members need to balance university, exams, work and life to find enough time to dedicate to the project. To address this risk, we have a regular meeting/workday on Tuesdays. So, we meet each Tuesday before class to plan the work week and update on the progress.

Risks: Users might find the notifications annoying, but that is the whole point. Just do what you are already supposed to do, and you won't get nagged.

The strategy for managing them is a clean user guide so the users understand how to set different notifications (persistency, frequency).

Risk: Potential misuse in creating excessive or unreasonable tasks.

Mitigation: Implement task creation guidelines and limits. Allow users to negotiate assigned tasks within the app. Kick votes? Small fines?

Step 3 Design and Architecture planning

Team: Group 8

Mia: Project manager

Kristian: Frontend

Kenneth: Frontend

Lucas: Backend

Jonas: Backend

3.1 Wireframes/mock-ups of the main screens.

Design the main screens of your "Full Stack Flavors" application.

Use tools like Figma, Sketch, or even simple sketches to create wireframes.

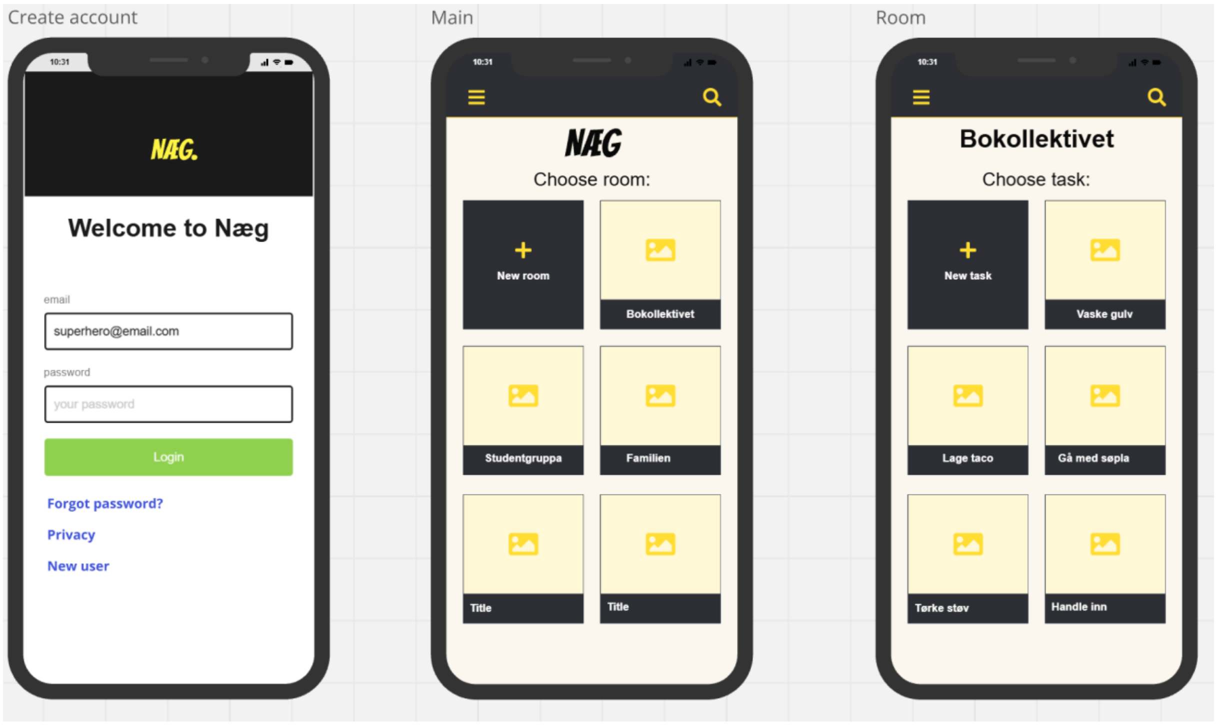
Make sure to get team input on the overall look and feel, as well as the user flow.

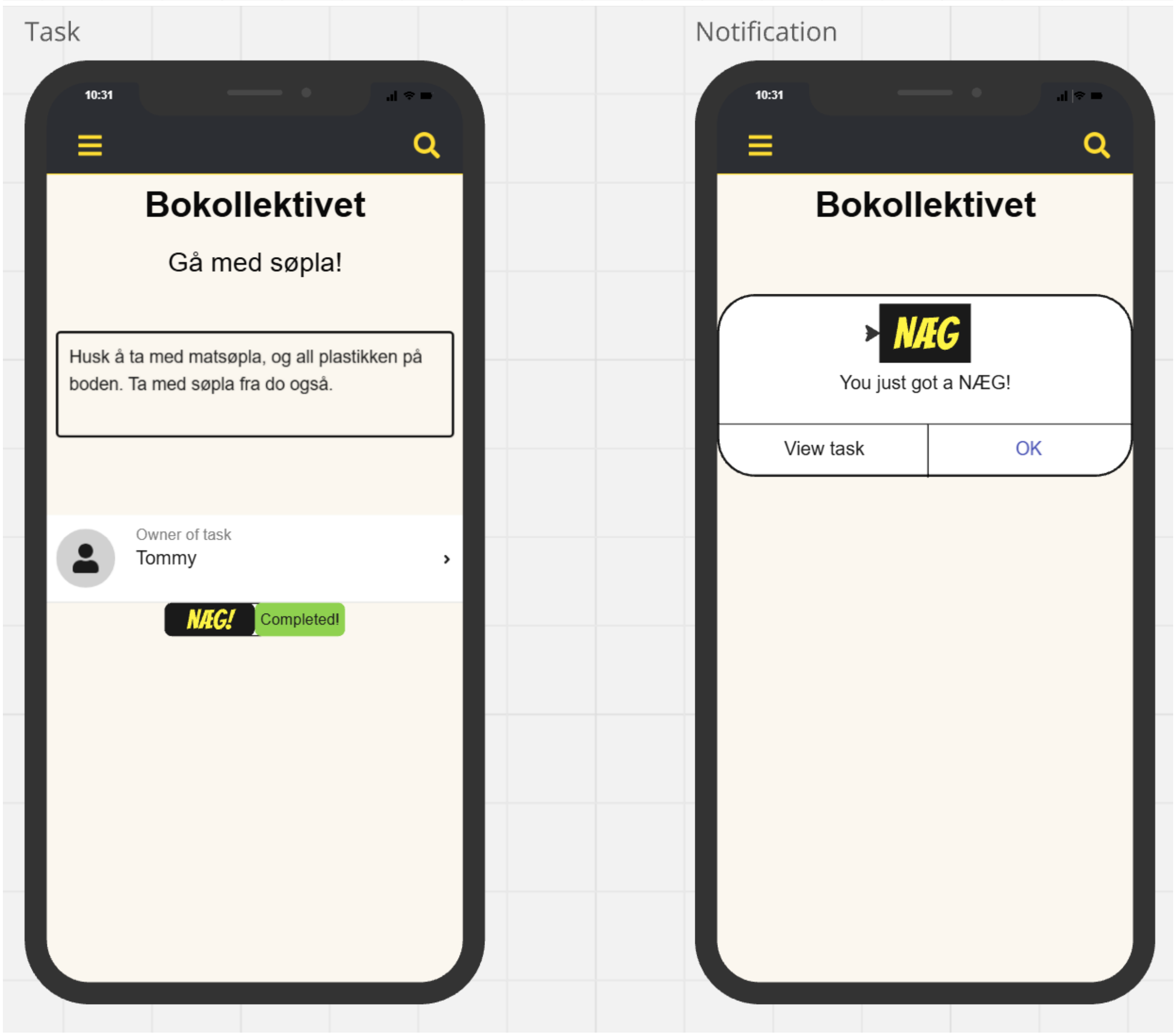
Aim to design a prototype that includes core functionality, showing how users will interact with the application.

The first table is the wireframe-ish prototype.

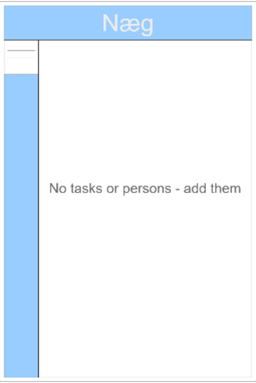
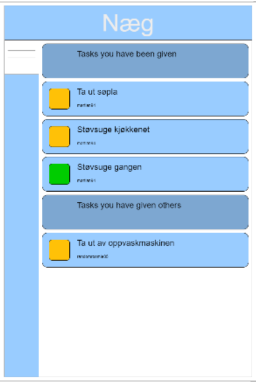
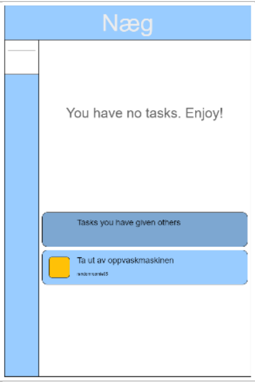
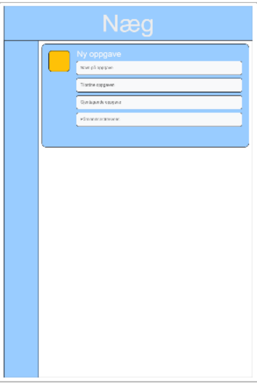
The second table is the high-fidelity mock-ups.

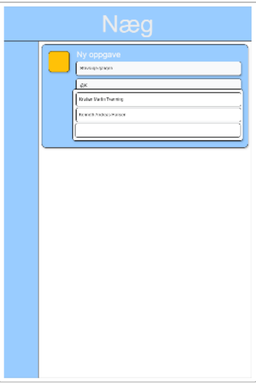
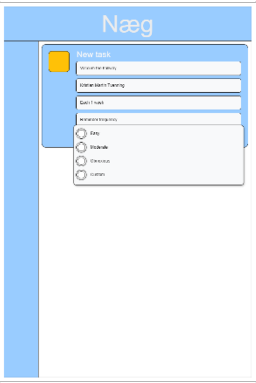
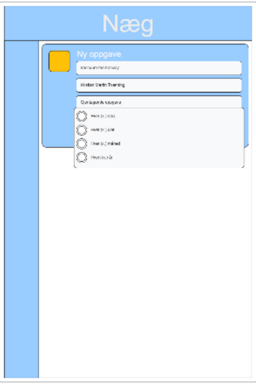
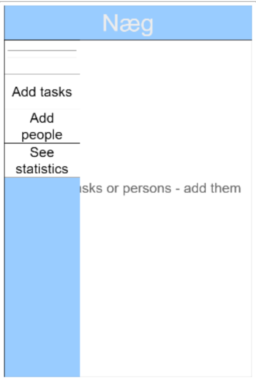
Table 1: Wireframe-ish

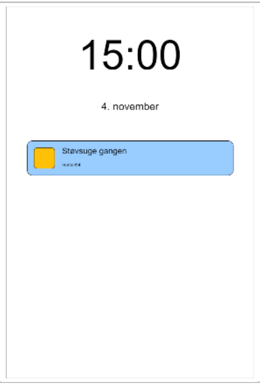
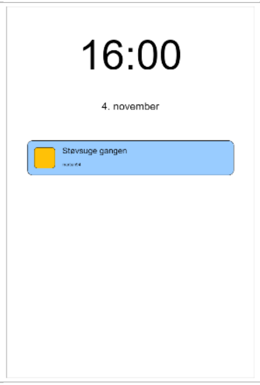
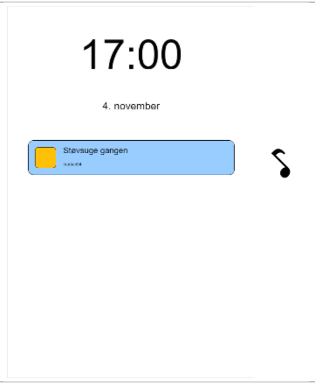
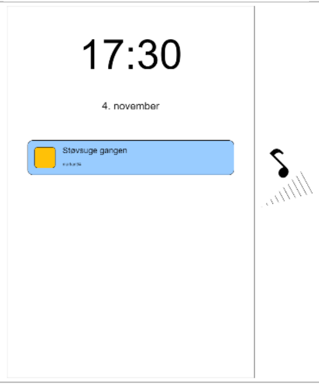




Picture	Description	Team input

Note: As React is “mobile first” the screens are shown in portrait mode		
 Næg No tasks or persons - add them	What you see when the app is freshly installed.	Should have an “Add”-Button or an arrow to the FAB (Frequent Action Button).
 Næg Tasks you have been given 🟨 Ta ut søpla 🟨 Støvsuge kjøkkenet 🟩 Støvsuge gangen Tasks you have given others 🟨 Ta ut av oppvaskmaskinen	App in daily use. You see the tasks that have been delegated to you first, and then the tasks you have delegated to others.	
 Næg You have no tasks. Enjoy! Tasks you have given others 🟨 Ta ut av oppvaskmaskinen	The screen that is shown when you are done with all your tasks.	
 Næg 🟨 Ny oppgave	Screen to Add task. This is the main task for the app and the task the MUI FAB component would be used for	

	Screen to delegate the task to specific people.	
	Screen to let you set notification frequency (and intensity)	
	Option to set up tasks to repeat themselves automatically. Useful for repeating tasks done on	
-	Screen to Add People (the only difference from Add Task is data fields)	
-	Screen to Add Room (the only difference from Add Task is data fields)	
 Næg Add tasks Add people See statistics sks or persons - add them	Burger menu (Side bar) This is where the secondary actions of the app would be, using the functionality you will not use quite as often as Add Task, such as seeing statistics or adding debit/credit cards for paid punishments	

	Notification level 1 – A simple soundless push notification that a task has been assigned to you (slide to mark as read)	
	Notification level 2 – persistent notification that cannot be slid away	
	Notification level 3 – Push notification with standard system notification sound	
	Notification level 4 – Push notification with “painful frequencies” and sound	

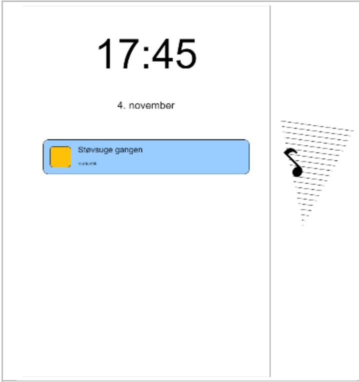
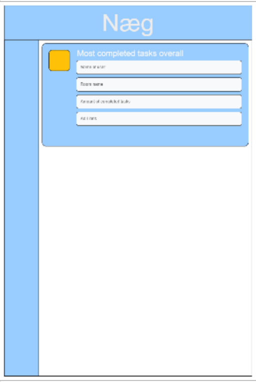
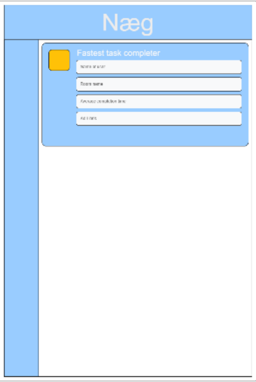
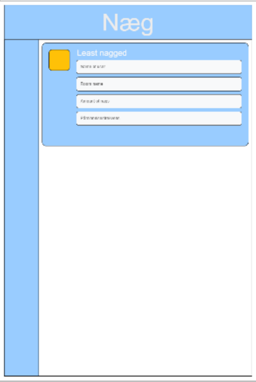
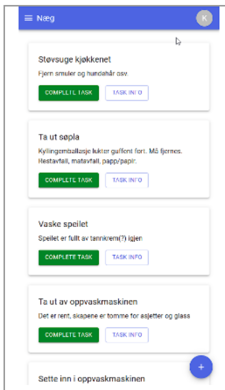
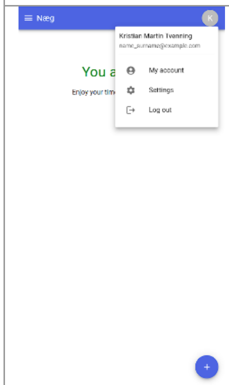
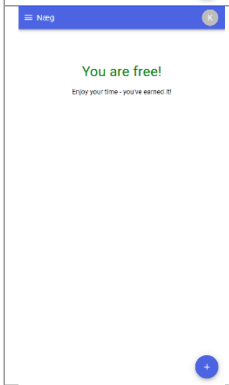
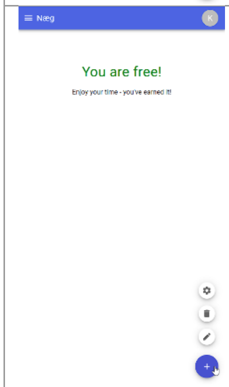
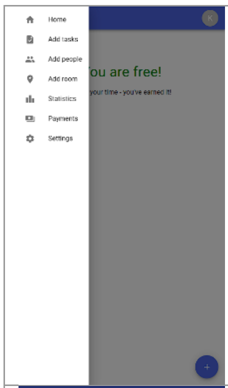
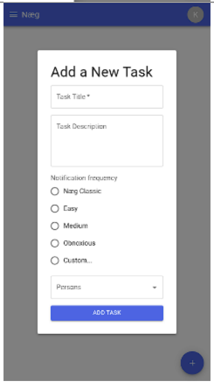
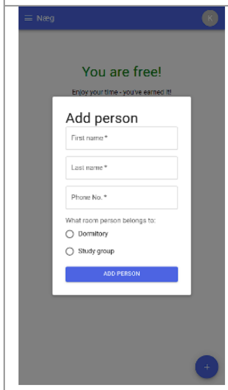
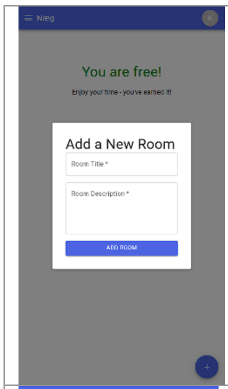
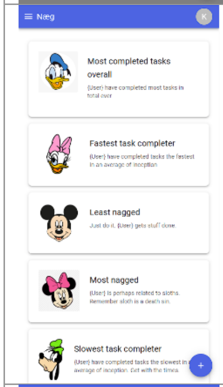
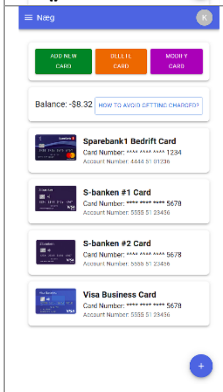
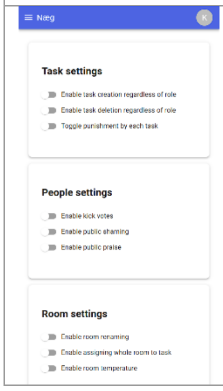
	Notification level 5 – Push notification with painful frequencies on full volume and vibration pattern not at all in accordance with the sound output	
	Statistics – most tasks completed overall	
	Statistics – fastest task completer	
	Statistics: least nagged	

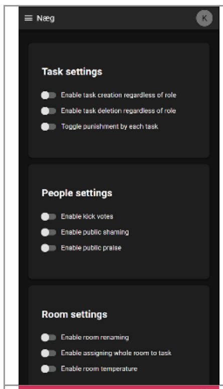
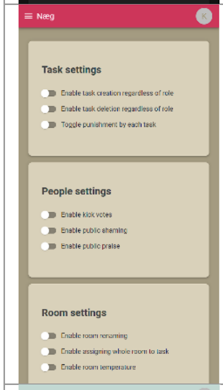
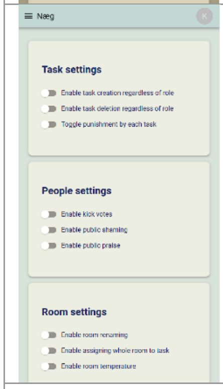
Table 2: High fidelity mockup as of 23rd of Jan, 2025 using the MUI component library

Picture	Description	Team input

 Næg Støvsuge kjøkkenet Fjern smuler og hundehår osv. COMPLETE TASK TASK INFO Ta ut søpla Kyllingemballasje lukter guffent fort. Må fjernes. Restavfall, matavfall, papp/papir. COMPLETE TASK TASK INFO Vaske speilet Speilet er fullt av tannkrem(?) igjen COMPLETE TASK TASK INFO Ta ut av oppvaskmaskinen Det er rent, skapene er tomme for asjetter og glass COMPLETE TASK TASK INFO Sette inn i oppvaskmaskinen	Landing page consisting of the tasks that you have been delegated. When you have done the task, you click the green button. If you want more info, you click the task info button.	
 Næg Kristian Martin Tvenning name_surname@example.com My account Settings Log out	The Popover in the Avatar element lets you log out from the app and have account overview.	Legge til "Add new profile" og/eller "Log in with new account"?
 Næg You are free! Enjoy your time - you've earned it!	When you have done everything, you are greeted with a message.	Forslag: Nye "Freedom-quotes" blir generert hver dag fra f.eks https://www.brainyquote.com/topics/freedom-quotes
 Næg You are free! Enjoy your time - you've earned it!	The frequent action button (FAB) lets phone users add tasks from the thumb (React is mobile first, and Add Task is considered a frequent action).	

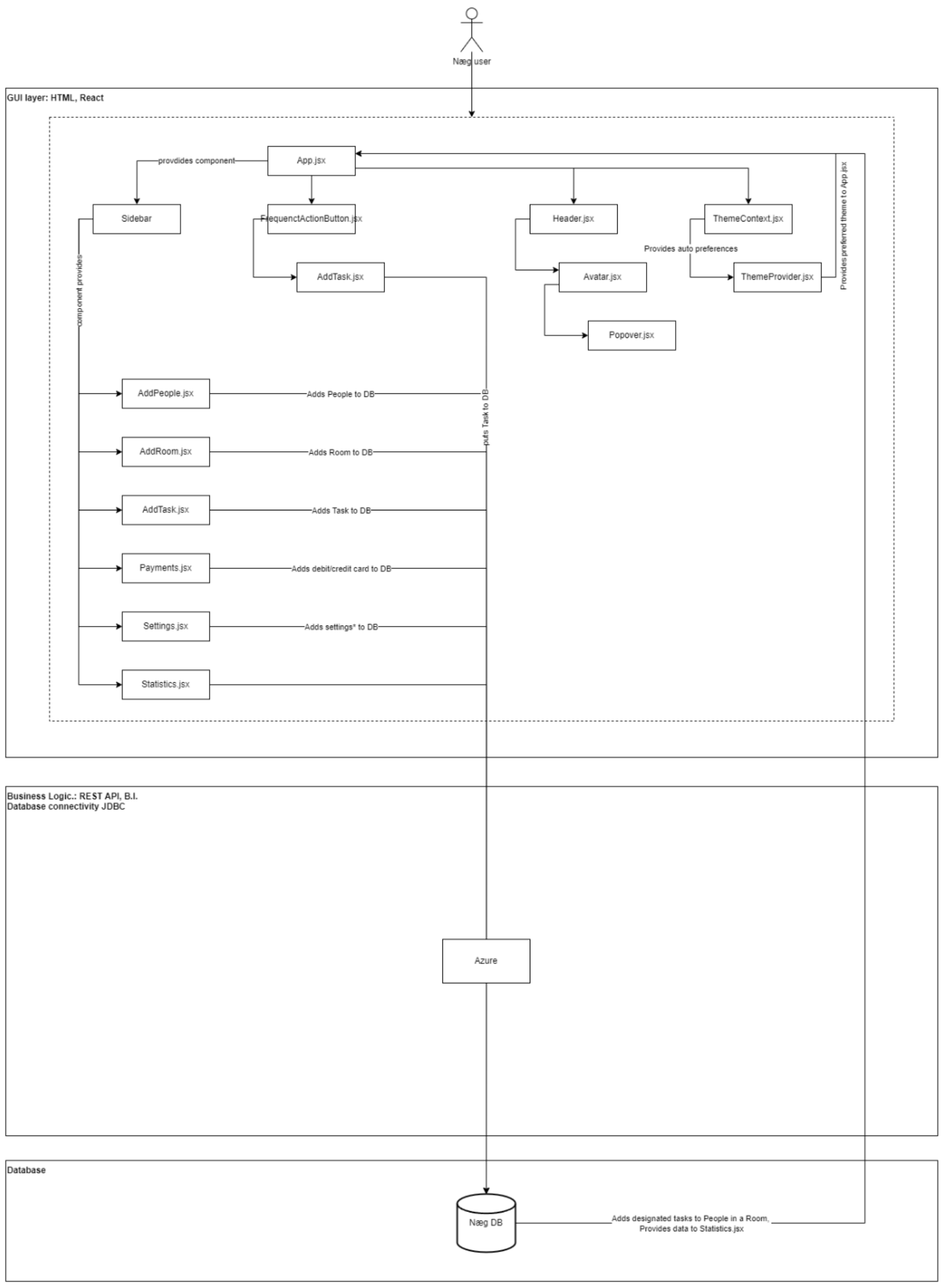
 Home Add tasks Add people Add room Statistics Payments Settings K ou are free! your time - you've earned it! +	The sidebar lets users access options that is considered not to be used that often such as “Add room” or “Add people”.	
 Næg K Add a New Task Task Title * Task Description Notification frequency Næg Classic Easy Medium Obnoxious Custom... Persons ADD TASK +	The Add Task modal. This is the action that is used the most (Available from FAB, accessible with thumb). Note that everything but Task Title is optional.	Add close-cross upper right?
 Næg K You are free! Enjoy your time - you've earned it! Add person First name * Last name * Phone No. * What room person belongs to: Dormitory Study group ADD PERSON +	The Add Person modal. Note that you can add person to formerly created rooms such as if you have a room named Dorm and a cohabitant is moving out whilst you live there, you simply delete the former resident and add the new one instead of creating a new room.	

 Næg You are free! Enjoy your time - you've earned it! Add a New Room Room Title * Room Description * ADD ROOM	The Add Room modal.	
 Næg Most completed tasks overall {User} have completed most tasks in total ever Fastest task completer {User} have completed tasks the fastest in an average of inception Least nagged Just do it. {User} gets stuff done. Most nagged {User} is perhaps related to sloths. Remember sloth is a death sin. Slowest task completer {User} have completed tasks the slowest in a average of inception. Get with the times.	Statistics page.	*Note: this is put on ice.*
 Næg ADD NEW CARD · DELETE CARD · MODIFY CARD Balance: -$8.32 HOW TO AVOID GETTING CHARGED? Sparebank1 Bedrift Card Card Number: **** **** **** 1234 Account Number: 4444 51 01236 S-banken #1 Card Card Number: **** **** **** 5678 Account Number: 5555 51 23456 S-banken #2 Card Card Number: **** **** **** 5678 Account Number: 5555 51 23456 Visa Business Card Card Number: **** **** **** 5678 Account Number: 5555 51 23456	Payment page. This lets users add motivation and punishment for completing and delaying tasks by incorporating money deposits that gets distributed.	Forslag: For å ivareta personvern/kortopplysninger, kunne det vært en metode for å opprette "monopol-penger-pott", så kan heller brukerne innad bli enig om hvordan betaling skal foregå. Evt "klistremerke" belønningssystem. *Note: this is put on ice.*
 Næg Task settings Enable task creation regardless of role Enable task deletion regardless of role Toggle punishment by each task People settings Enable kick votes Enable public shaming Enable public praise Room settings Enable room renaming Enable assigning whole room to task Enable room temperature	Settings page in light mode	Egen "color blind-mode"? Accessibility…

 Næg Task settings Enable task creation regardless of role Enable task deletion regardless of role Toggle punishment by each task People settings Enable kick votes Enable public shaming Enable public praise Room settings Enable room renaming Enable assigning whole room to task Enable room temperature	Settings page in dark mode.	
 Næg Task settings Enable task creation regardless of role Enable task deletion regardless of role Toggle punishment by each task People settings Enable kick votes Enable public shaming Enable public praise Room settings Enable room renaming Enable assigning whole room to task Enable room temperature	Settings page with the Pacific Punch theme.	
 Næg Task settings Enable task creation regardless of role Enable task deletion regardless of role Toggle punishment by each task People settings Enable kick votes Enable public shaming Enable public praise Room settings Enable room renaming Enable assigning whole room to task Enable room temperature	Settings page with the Arctic Atmosphere theme.	

3.2 The system architecture diagram.

Create a high-level diagram that outlines how each component will interact within the app.



3.4 User stories

User stories are simple descriptions of a feature from the user perspective. What the user needs or wants to do and why.

Task management

1. As a user, I want to create tasks with description, due date and level of urgency so I can delegate the task to the user that will perform them
2. As a user, I want to assign the task to another user, so they get the responsibility.
3. As a user, I want to track the progress of the task, so I can nag the user to completion
4. As a user, I want to edit or delete the task so I can make changes and edit the tasks I create.
5. As a user, I wish to nag another user until they complete their task, so the task will be executed in time.

Group tasks

6. As a user, I want to be able to create a group of members to share regular tasks and rotating responsibilities, so we can collaborate in the task management.
7. As a user, I want to view all the tasks assigned to my group, so we have an overview of the group's activities.
8. As a user, I want to be able to comment and rate a task during and after completion, so the group can communicate within the task group.
9. As a user, I want to be able to attach images of the task before and after completion, as a sort of evidence.
10. As a user, I want to be able to receive points for a job well done.
11. As a user, I want the group to be ranged after points of their performance.

Notifications

12. As a user, I want to be notified when a task is assigned to me, so I can get to work a complete the task
13. As a user, I want to configure my notification preferences to what works for me

14. As a user, I want to be able to increase the level of nagging to a member who is slow on completing a task

User features

15. As a user, I want secure log to protect my data
16. As a user, I want a personalized dashboard that I can view my tasks, and the tasks I have given to others.
17. As I user I want the list of tasks I must complete to be in the order I need to complete them, maybe a calendar

3.5 Milestones for the user stories

Milestone 1: Basic task management

Step 3 design and architecture planning

User stories: 1,2,3,4,5

Timeline: last week of January

Goal: Implement task creation, assignment of tasks, filtering and sorting of tasks

Milestone 2: Group tasks

User stories: 6-11

Timeline: end of January (

Goal: Develop group assignment functionality with shared views, comments and file attachments.

Milestone 3: Notifications

User stories: 12-14

Timeline: February

Goal: Set up a system for nagging notifications with customizable preferences and triggers for task updates and reminders.

Milestone 4: Dashboard and calendar view

User stories: 15-17

Timeline: March

Goal: create a dashboard and calendar for the user

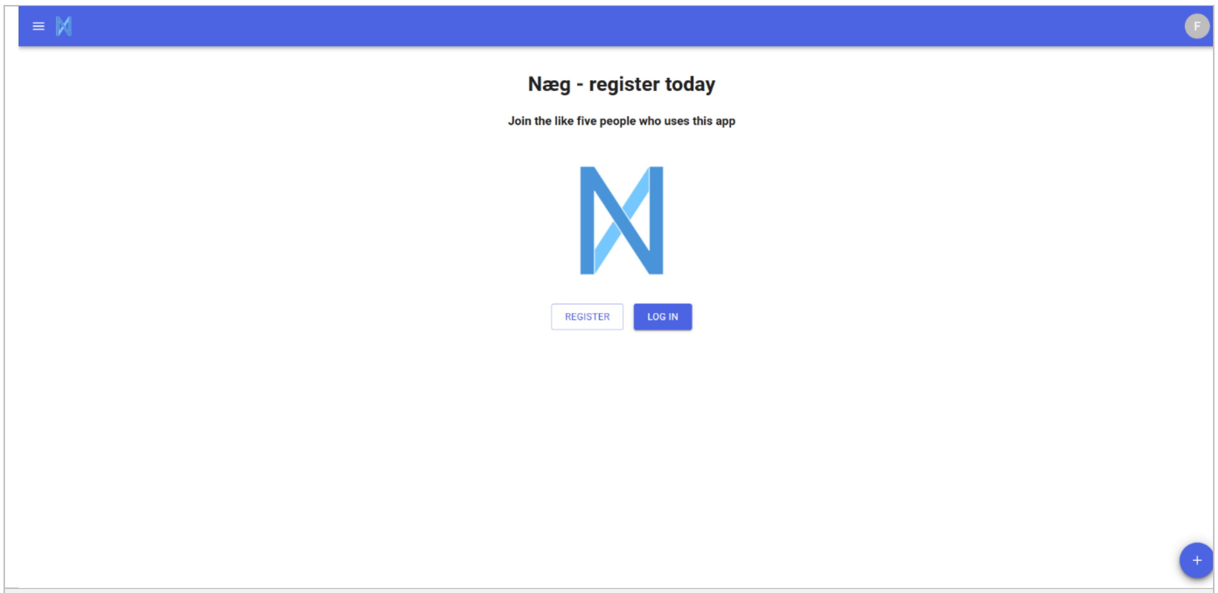
Milestone 5: Testing and deployment

Timeline: April

Goal: deployment on azure and testing to ensure no bugs and secure system.

Step 4 Front-End progress report

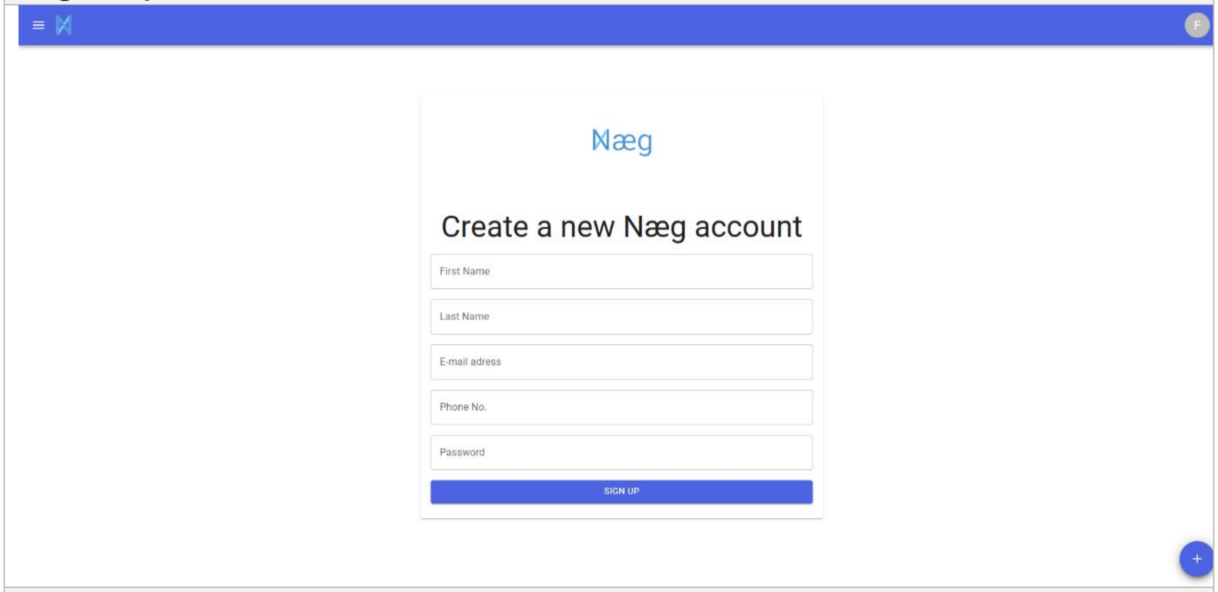
4.1 Screenshots of key pages and components.



Næg - register today

Join the like five people who uses this app

Register.jsx



Næg

Create a new Næg account

First Name

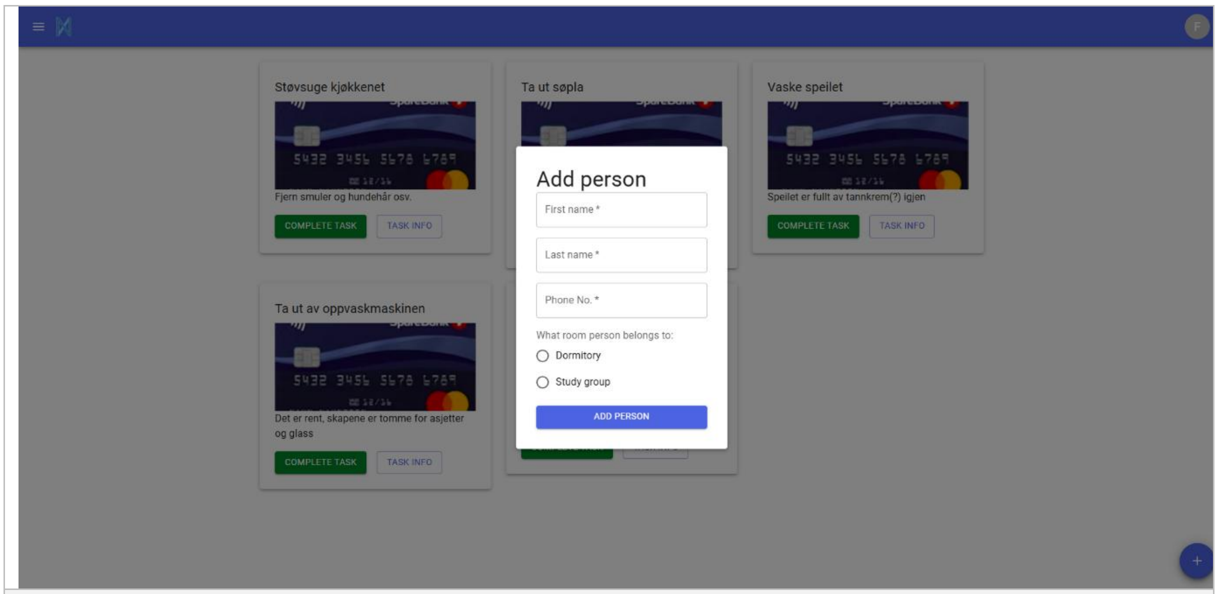
Last Name

E-mail adress

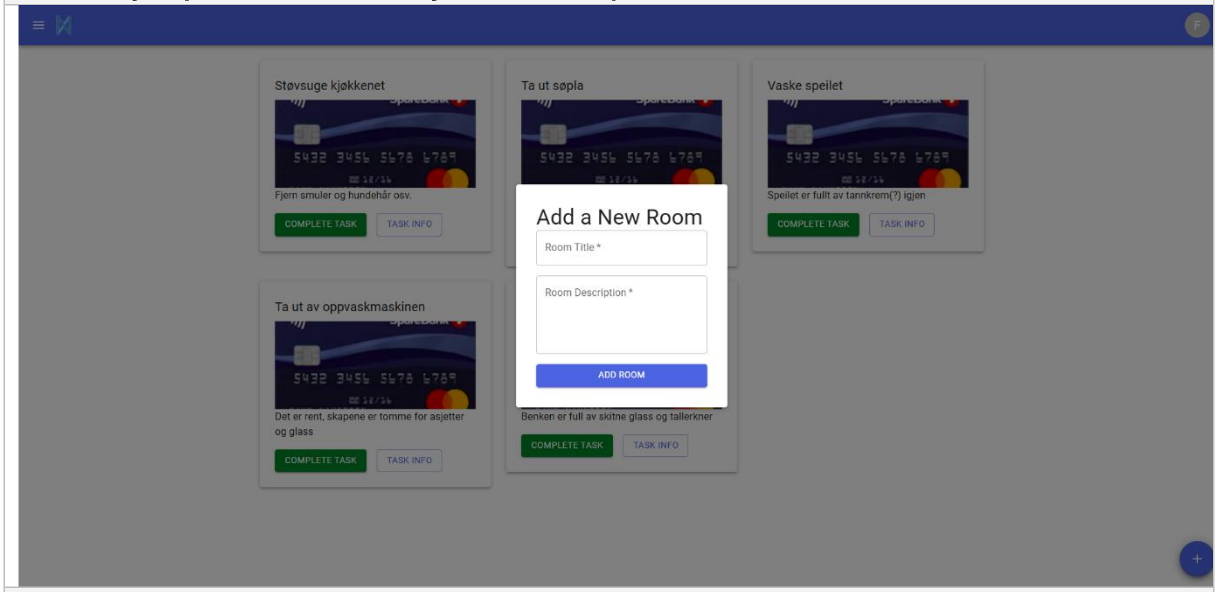
Phone No.

Password

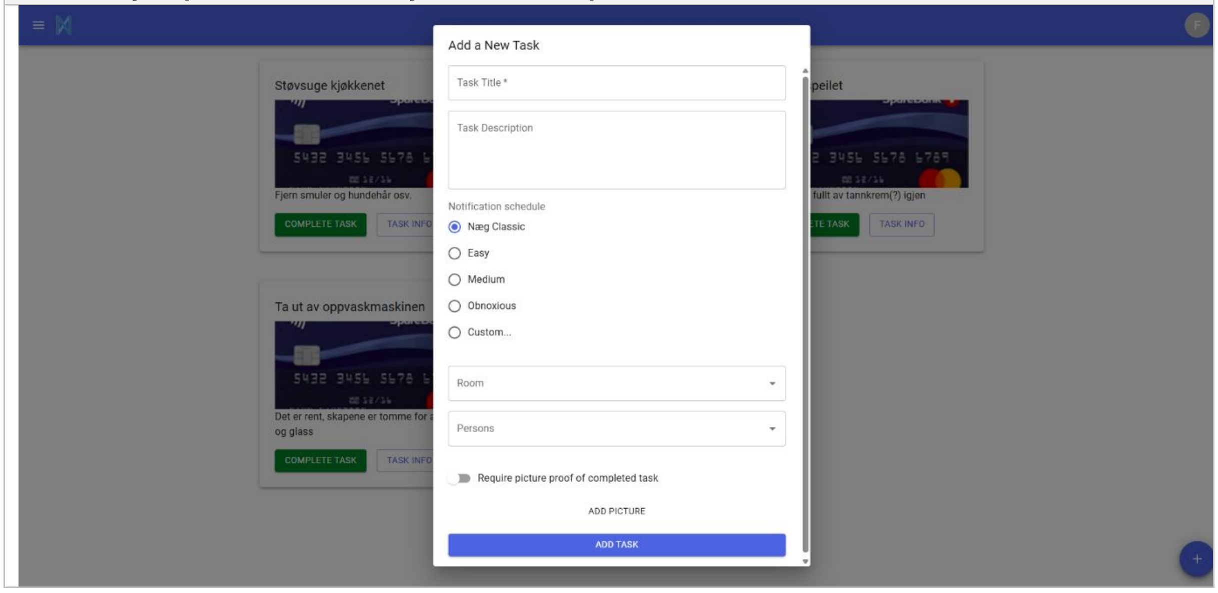
Login.jsx



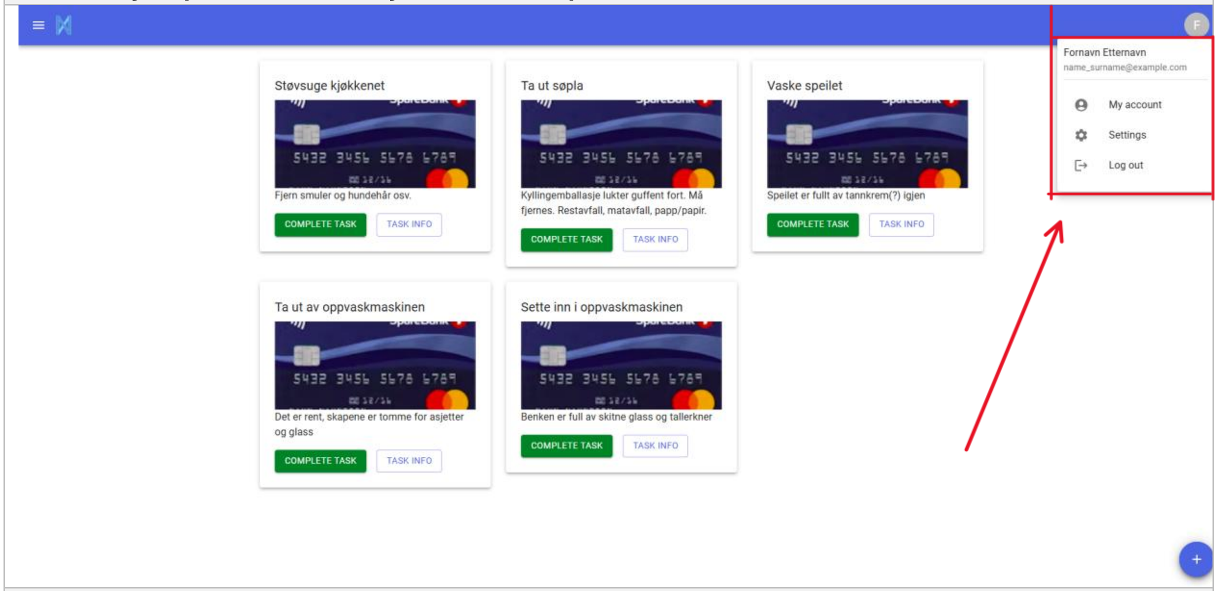
Header.jsx (with AddPerson.jsx as Modal)


Støvsuge kjøkkenet
Fjern smuler og hundehår osv.
Ta ut søpla
Vaske speilet
Speilet er fullt av tannkrem(?) igjen
Ta ut av oppvaskmaskinen
Det er rent, skapene er tomme for asjetter og glass
Benken er full av skitne glass og tallerkner
COMPLETE TASK
TASK INFO
Add a New Room
Room Title *
Room Description *
ADD ROOM

Header.jsx (with AddRoom.jsx as Modal)



Header.jsx (with AddTask.jsx as Modal)



Støvsuge kjøkkenet
Fjern smuler og hundehår osv.
COMPLETE TASK TASK INFO

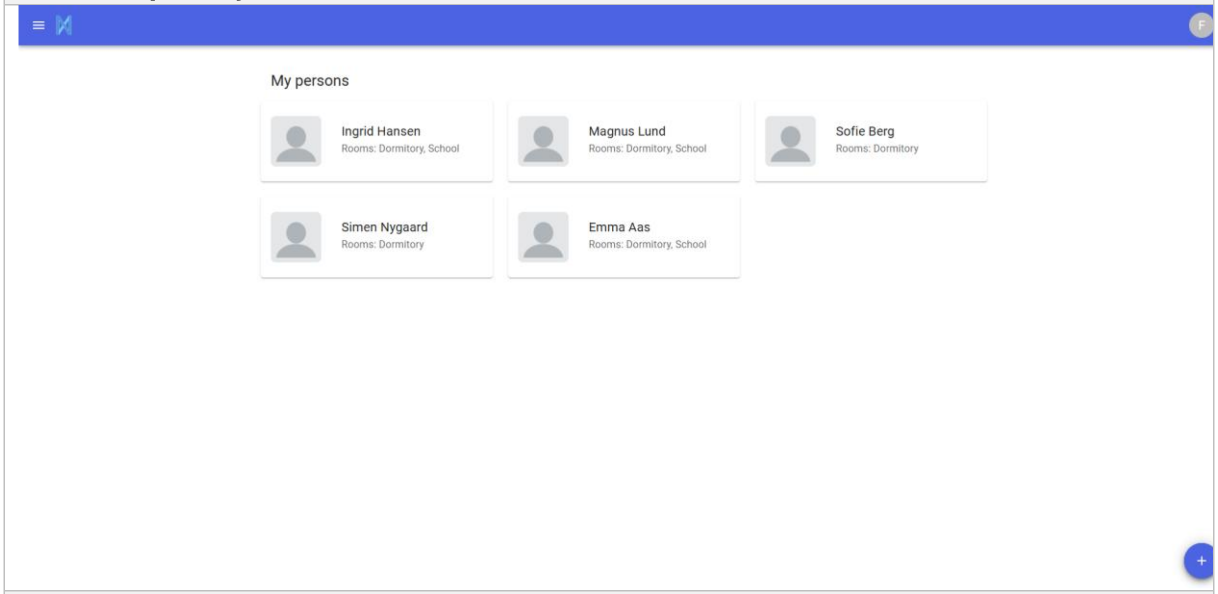
Ta ut søpla
Kyllingemballasje lukter guffent fort. Må fjernes. Restavfall, matavfall, papp/papir.
COMPLETE TASK TASK INFO

Vaske speilet
Speilet er fullt av tannkrem(?) igjen
COMPLETE TASK TASK INFO

Ta ut av oppvaskmaskinen
Det er rent, skapene er tomme for asjetter og glass
COMPLETE TASK TASK INFO

Sette inn i oppvaskmaskinen
Benken er full av skitne glass og tallerkner
COMPLETE TASK TASK INFO

AvatarPopover.jsx



My persons

Ingrid Hansen
Rooms: Dormitory, School

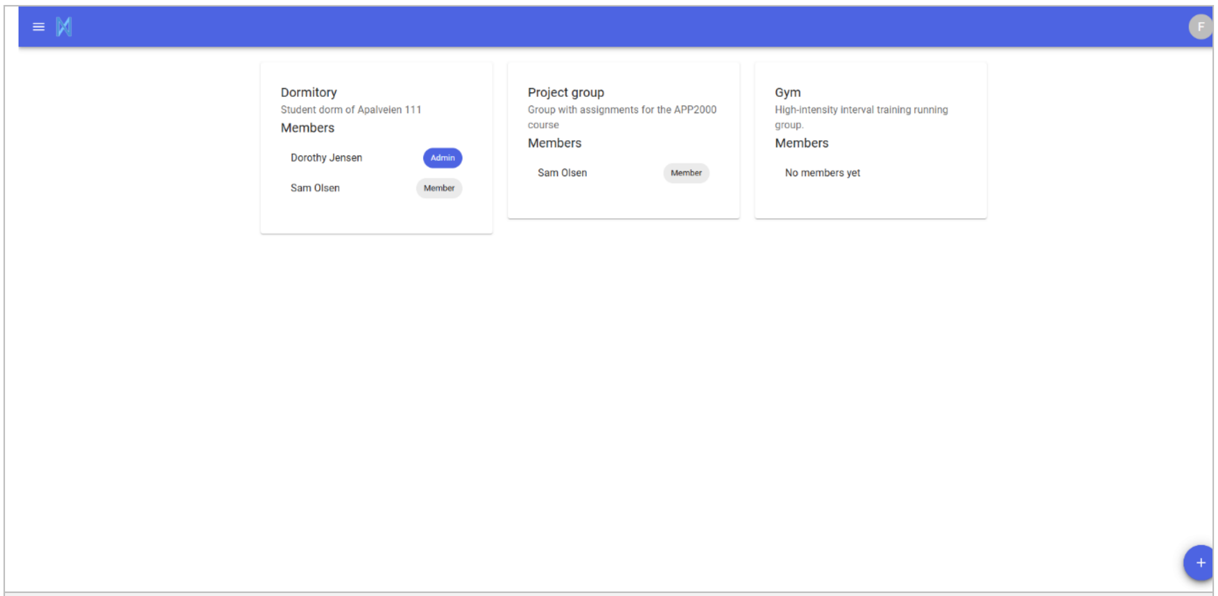
Magnus Lund
Rooms: Dormitory, School

Sofie Berg
Rooms: Dormitory

Simen Nygaard
Rooms: Dormitory

Emma Aas
Rooms: Dormitory, School

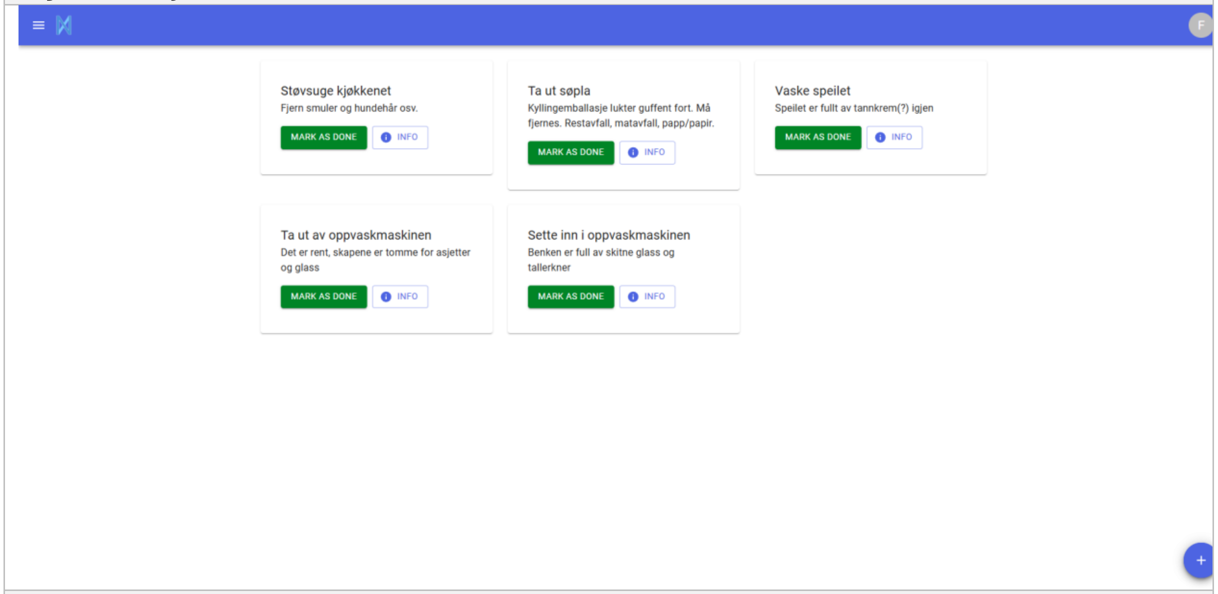
MyPeople.jsx


Dormitory
Student dorm of Apalveien 111
Members
Dorothy Jensen — Admin
Sam Olsen — Member

Project group
Group with assignments for the APP2000 course
Members
Sam Olsen — Member

Gym
High-intensity interval training running group.
Members
No members yet

MyRooms.jsx


Støvsuge kjøkkenet
Fjern smuler og hundehår osv.
MARK AS DONE | INFO

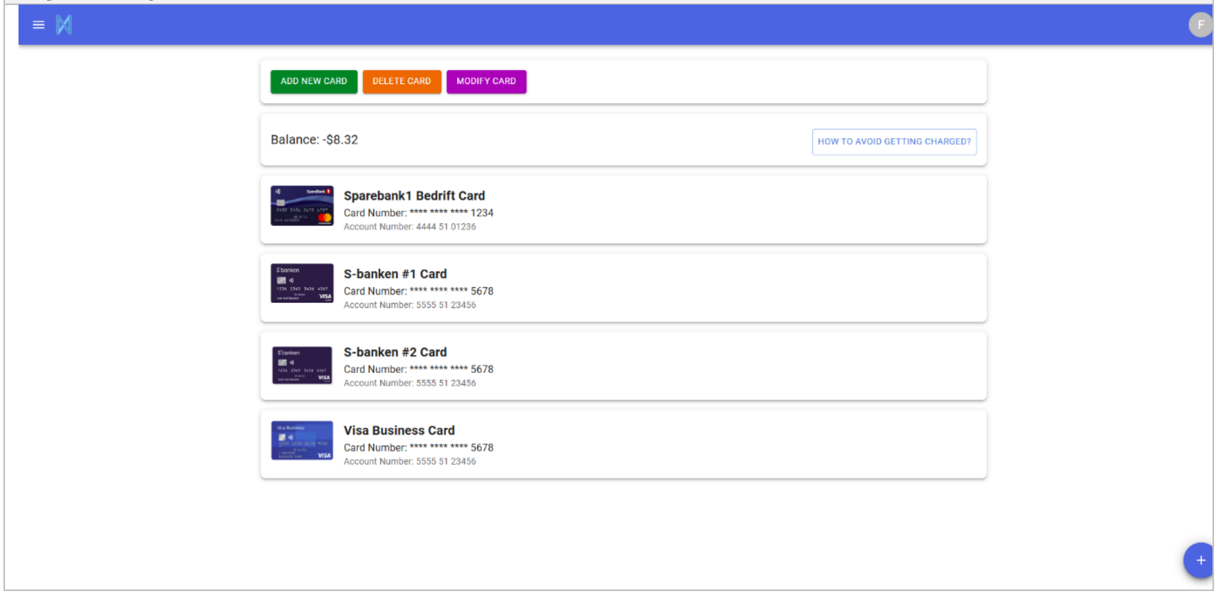
Ta ut søpla
Kyllingemballasje lukter guffent fort. Må fjernes. Restavfall, matavfall, papp/papir.
MARK AS DONE | INFO

Vaske speilet
Speilet er fullt av tannkrem(?) igjen
MARK AS DONE | INFO

Ta ut av oppvaskmaskinen
Det er rent, skapene er tomme for asjetter og glass
MARK AS DONE | INFO

Sette inn i oppvaskmaskinen
Benken er full av skitne glass og tallerkner
MARK AS DONE | INFO

MyTasks.jsx


ADD NEW CARD | DELETE CARD | MODIFY CARD

Balance: -$8.32 — HOW TO AVOID GETTING CHARGED?

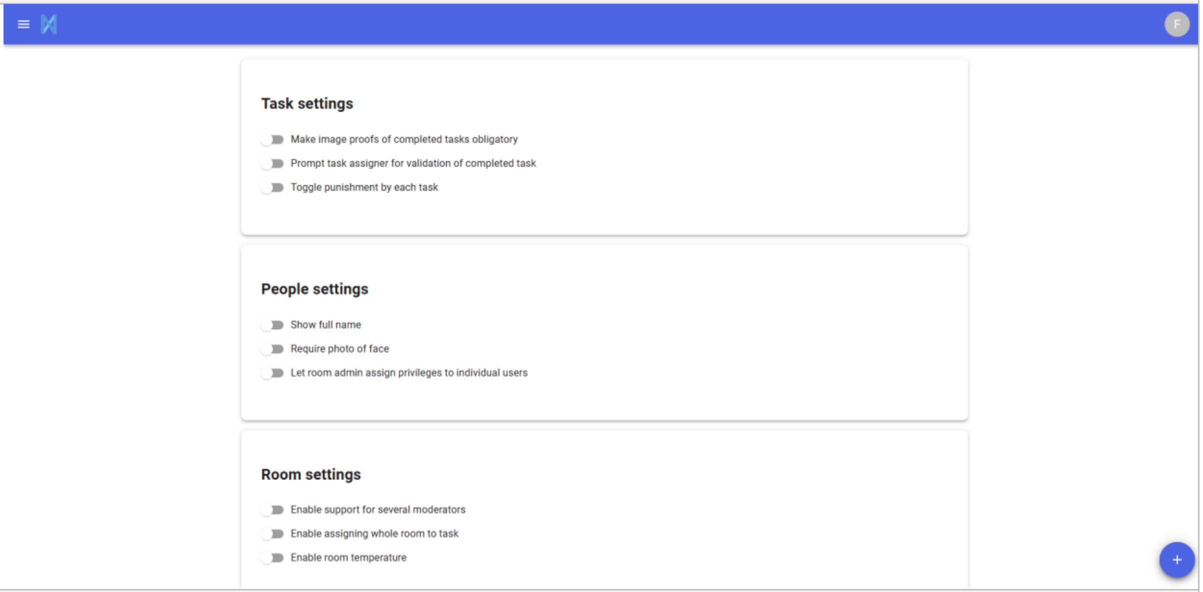
Sparebank1 Bedrift Card
Card Number: **** **** **** 1234
Account Number: 4444 51 01236

S-banken #1 Card
Card Number: **** **** **** 5678
Account Number: 5555 51 23456

S-banken #2 Card
Card Number: **** **** **** 5678
Account Number: 5555 51 23456

Visa Business Card
Card Number: **** **** **** 5678
Account Number: 5555 51 23456

Payments.jsx (put on ice)



Task settings

Make image proofs of completed tasks obligatory

Prompt task assigner for validation of completed task

Toggle punishment by each task

People settings

Show full name

Require photo of face

Let room admin assign privileges to individual users

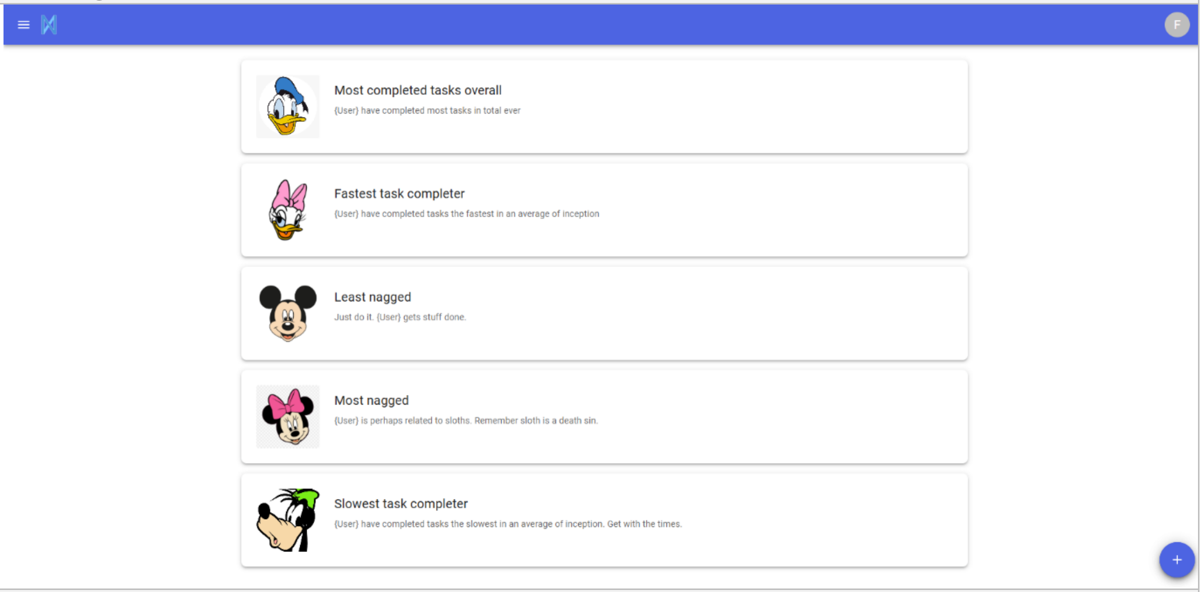
Room settings

Enable support for several moderators

Enable assigning whole room to task

Enable room temperature

Settings.jsx



Most completed tasks overall

{User} have completed most tasks in total ever

Fastest task completer

{User} have completed tasks the fastest in an average of inception

Least nagged

Just do it. {User} gets stuff done.

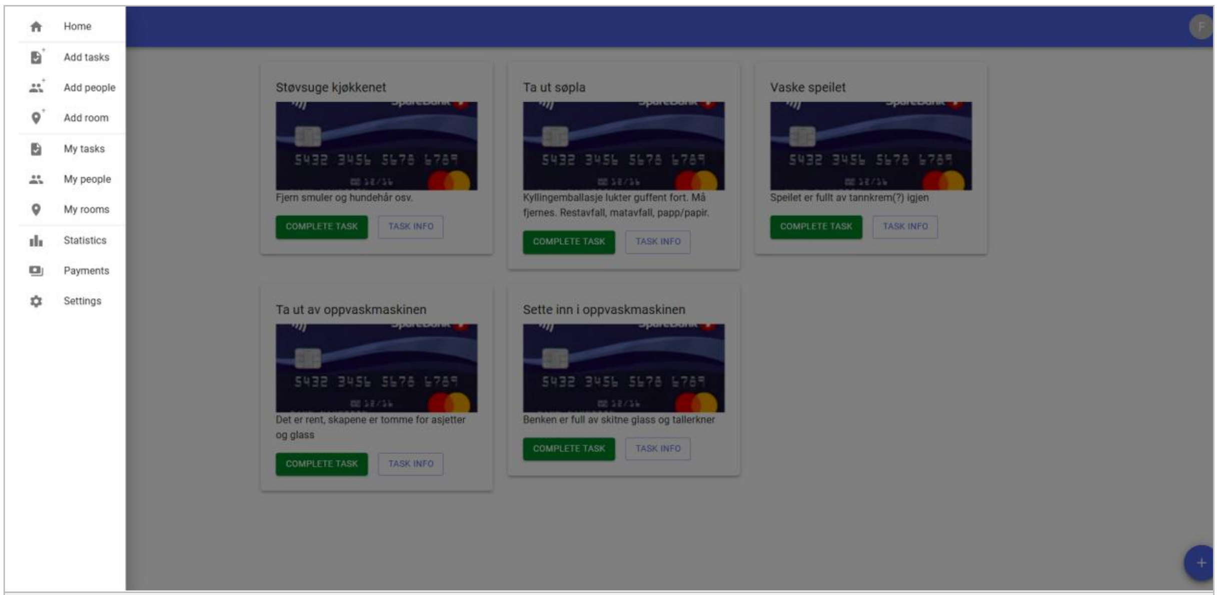
Most nagged

{User} is perhaps related to sloths. Remember sloth is a death sin.

Slowest task completer

{User} have completed tasks the slowest in an average of inception. Get with the times.

Statistics.jsx (put on ice)


Home
Add tasks
Add people
Add room
My tasks
My people
My rooms
Statistics
Payments
Settings

Støvsuge kjøkkenet
Fjern smuler og hundehår osv.
COMPLETE TASK TASK INFO

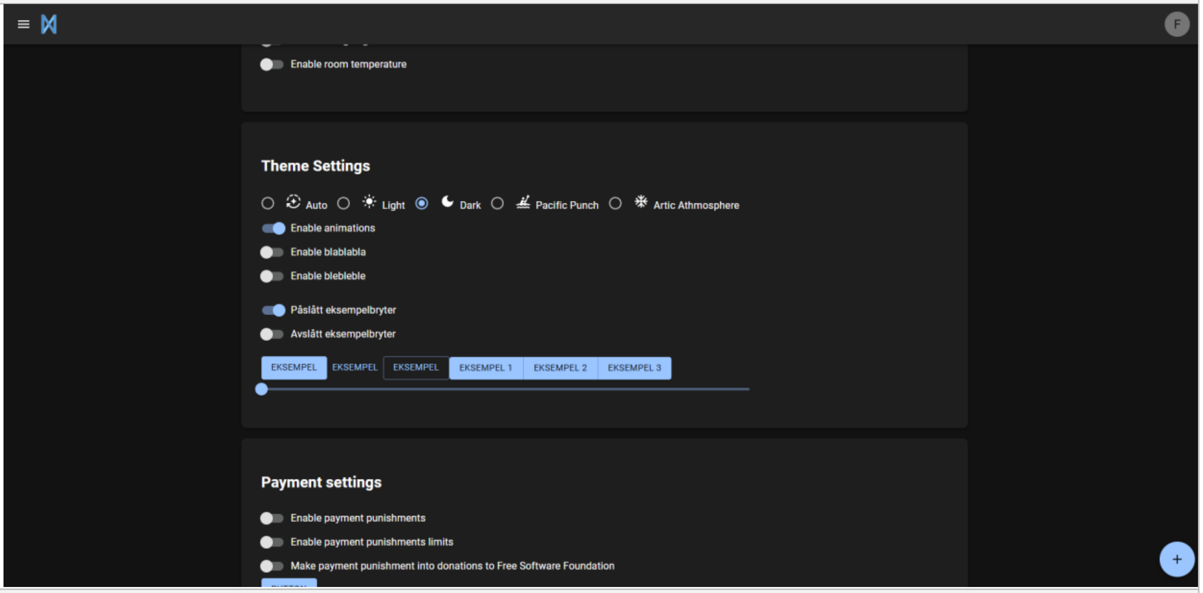
Ta ut søpla
Kyllingemballasje lukter guffent fort. Må fjernes. Restavfall, matavfall, papp/papir.
COMPLETE TASK TASK INFO

Vaske speilet
Speilet er fullt av tannkrem(?) igjen
COMPLETE TASK TASK INFO

Ta ut av oppvaskmaskinen
Det er rent, skapene er tomme for asjetter og glass
COMPLETE TASK TASK INFO

Sette inn i oppvaskmaskinen
Benken er full av skitne glass og tallerkner
COMPLETE TASK TASK INFO

Header.jsx contains <Swipeable drawer> component "Sidebar"


Enable room temperature

Theme Settings

Auto Light [✓ Dark] Pacific Punch Artic Athmosphere
Enable animations
Enable blablabla
Enable blebleble

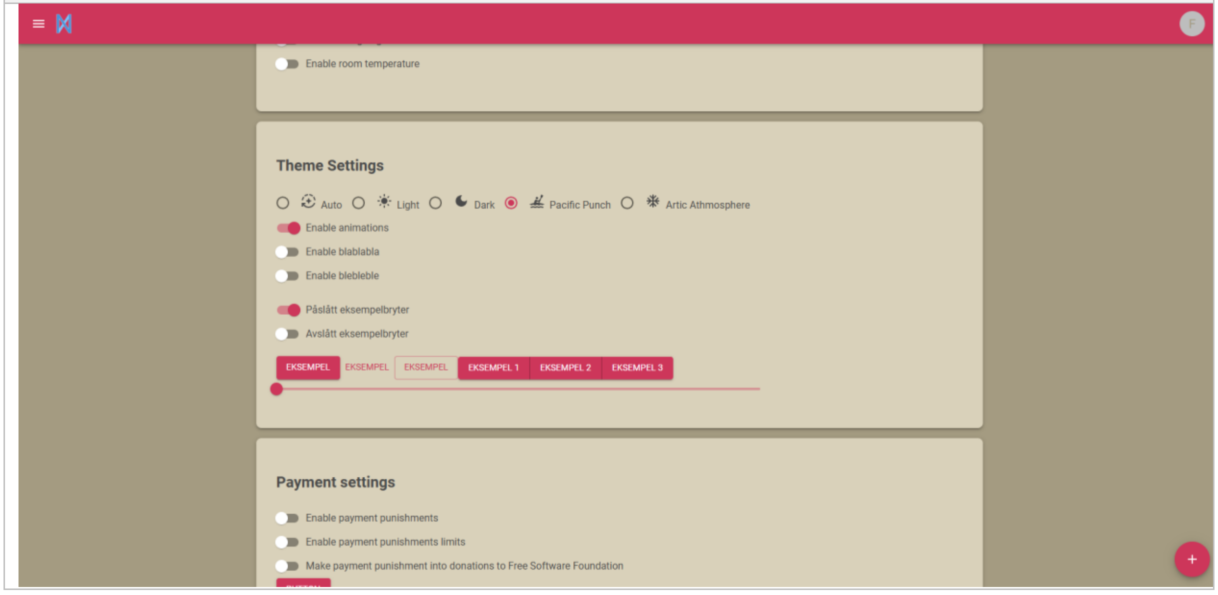
Påslått eksempelbryter
Avslått eksempelbryter

EKSEMPEL EKSEMPEL EKSEMPEL EKSEMPEL 1 EKSEMPEL 2 EKSEMPEL 3

Payment settings

Enable payment punishments
Enable payment punishments limits
Make payment punishment into donations to Free Software Foundation

Dark theme (utilizing ThemeContext.jsx)


Enable room temperature

Theme Settings

Auto Light Dark [✓ Pacific Punch] Artic Athmosphere
Enable animations
Enable blablabla
Enable blebleble

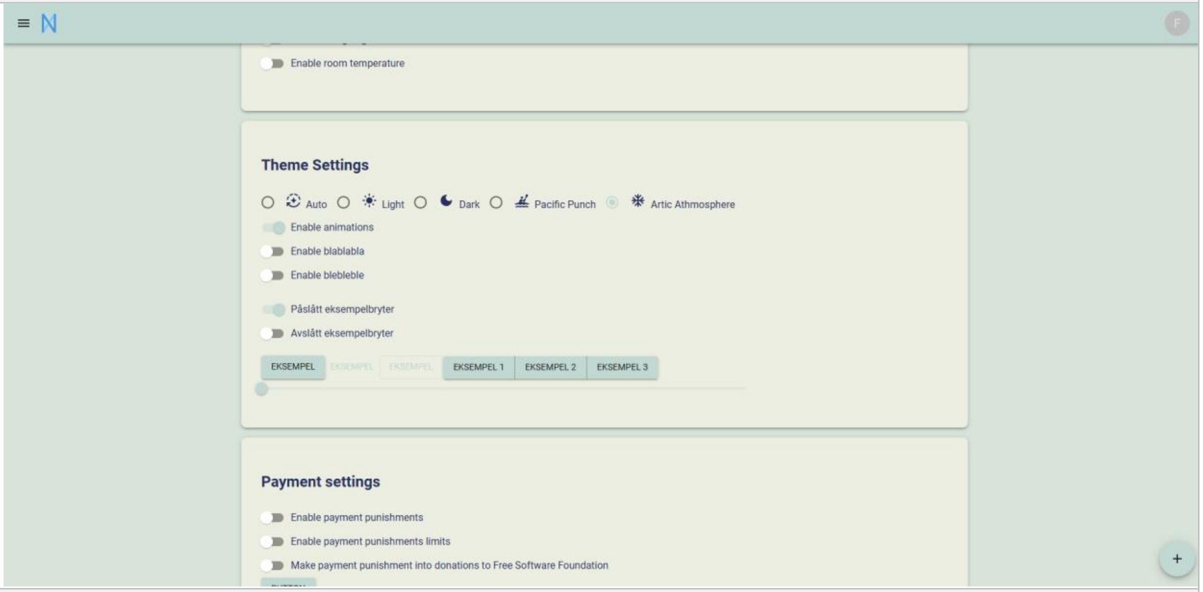
Påslått eksempelbryter
Avslått eksempelbryter

EKSEMPEL EKSEMPEL EKSEMPEL EKSEMPEL 1 EKSEMPEL 2 EKSEMPEL 3

Payment settings

Enable payment punishments
Enable payment punishments limits
Make payment punishment into donations to Free Software Foundation

Pacific Punch theme (utilizing ThemeProvider.jsx)



Enable room temperature

Theme Settings

Auto Light Dark Pacific Punch [✓ Artic Athmosphere]

Enable animations
Enable blablabla
Enable blebleble

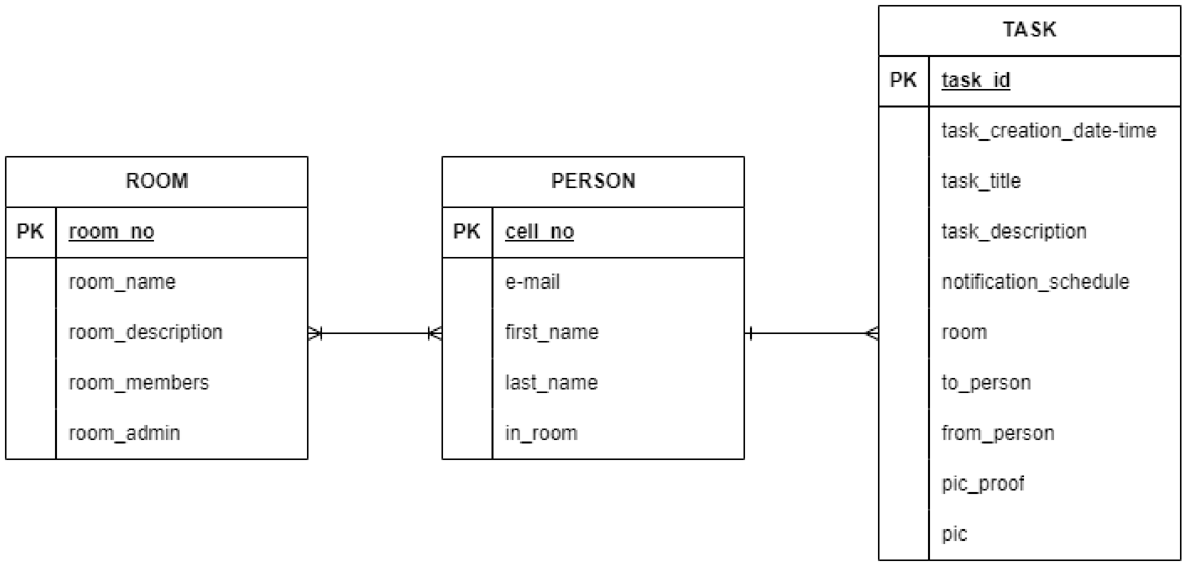
Påslått eksempelbryter
Avslått eksempelbryter

Payment settings

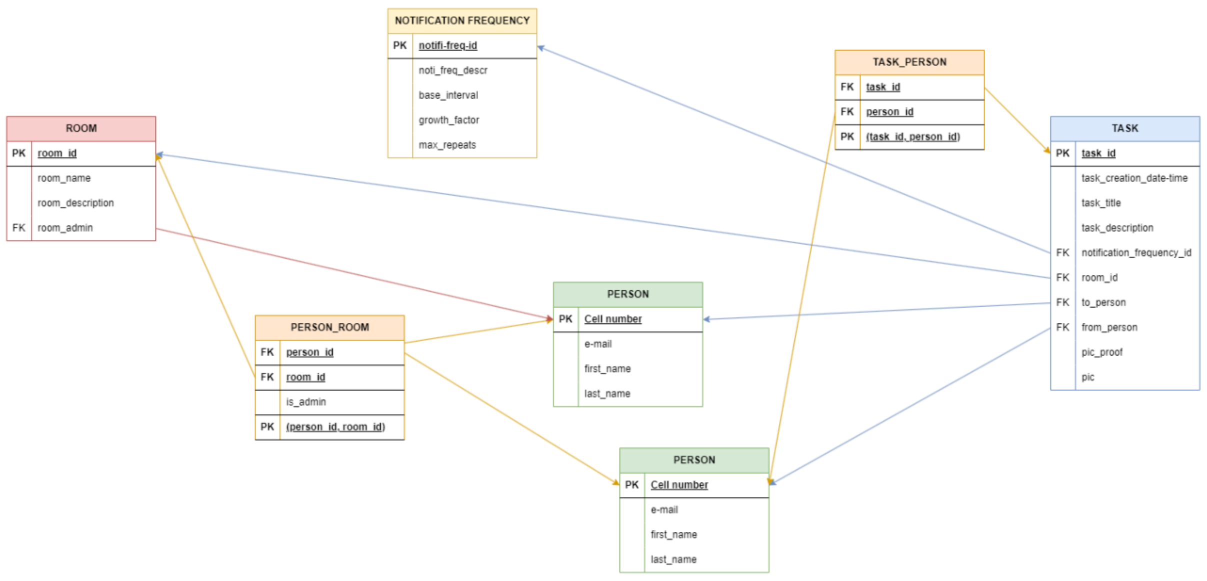
Enable payment punishments
Enable payment punishments limits
Make payment punishment into donations to Free Software Foundation

Arctic Atmosphere theme (utilizing ThemeProvider.jsx)

4. 2 Database design and key entities



A room has persons in it. The persons can create tasks. A room can have at least one or several persons in it. A person belongs to at least one or several rooms. A person can create several tasks. A task is created by one person. Note: this is a fan trap. If a TASK is linked to both ROOM and PERSON but there is no direct ROOM-PERSON-TASK relationship enforced, it can create inconsistencies. For example, a task could be assigned to a person who is not actually a member of the given room, and it does not ensure *to_person* or *from_person* are valid members of the room. This is addressed with associative tables TASK_PERSON and PERSON_ROOM.



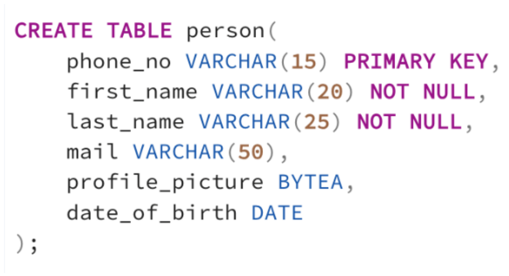
A) The PERSON_ROOM table establishes a many-to-many relationship between PERSON and ROOM. PERSON_ROOM.*is_admin* field ensures a clear distinction between room members and admins.

B) Instead of directly linking TASK to a single *to_person* and *from_person*, TASK_PERSON ensures a task can have multiple people assigned to it. Tasks are associated with people in a structured way rather than allowing direct assignments which could result in orphans.

C) Each TASK still references a *room_id* which ensures that a task belongs to a specific room. Because PERSON_ROOM verifies a person's membership in a room, task assignments will be valid within the room's scope (so that you cannot add tasks to persons who do not exist in the room).

SQL code for the project

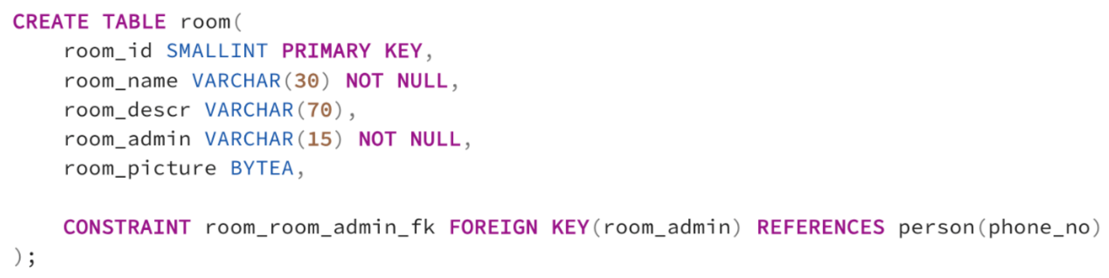
1. Person table



```
CREATE TABLE person(
    phone_no VARCHAR(15) PRIMARY KEY,
    first_name VARCHAR(20) NOT NULL,
    last_name VARCHAR(25) NOT NULL,
    mail VARCHAR(50),
    profile_picture BYTEA,
    date_of_birth DATE
);
```

Profile picture, mail adress and date of birth should be optional for the user as there is no real use for the data. However, we chose to include the attributes for future use cases. VARCHAR for the user's phone number to include signs for country specification.

2. Room table



```
CREATE TABLE room(
    room_id SMALLINT PRIMARY KEY,
    room_name VARCHAR(30) NOT NULL,
    room_descr VARCHAR(70),
    room_admin VARCHAR(15) NOT NULL,
    room_picture BYTEA,

    CONSTRAINT room_room_admin_fk FOREIGN KEY(room_admin) REFERENCES person(phone_no)
);
```

Room description and picture should be optional for the user. Attribute room_admin is a foreign key that references a user's phone number from the person table.

3. Notification frequency table

```sql
CREATE TABLE noti_freq(
    noti_freq_id SMALLINT PRIMARY KEY,
    noti_freq_title TEXT NOT NULL,
    base_interval INTERVAL NOT NULL,
    growth_factor REAL NOT NULL,
    max_repeats SMALLINT NOT NULL
);
```

Table holds data for the different interval presets and should not be interacted with by the user other than read. Interval of notification is dictated by base interval and growth factor to be able to both have a linear notification interval as well as exponential growth.

4. Task table

```sql
CREATE TABLE task(
    task_id INT PRIMARY KEY,
    task_title VARCHAR(50) NOT NULL,
    task_descr TEXT,
    due_date_time TIMESTAMP NOT NULL,
    noti_freq_id SMALLINT NOT NULL,
    creator VARCHAR(15) NOT NULL,
    completed BOOLEAN NOT NULL,

    CONSTRAINT task_noti_freq_id_fk FOREIGN KEY(noti_freq_id) REFERENCES noti_freq(noti_freq_id),
    CONSTRAINT task_creator_fk FOREIGN KEY(creator) REFERENCES person(phone_no)
);
```

Task description should be optional for the user as the title could be descriptive enough. Task creator is a foreign key referencing a user's phone number in the person table. INT data type for task_id since the table holds data for every task, therefore we need to account for a larger number. ”completed” attribute is for separating active tasks from completed tasks.

5. Room for person join table

```sql
CREATE TABLE room_for_person(
    room_id SMALLINT NOT NULL,
    person_id VARCHAR(15) NOT NULL,
    score SMALLINT NOT NULL,

    PRIMARY KEY(room_id, person_id),

    CONSTRAINT room_for_person_room_id_fk FOREIGN KEY(room_id) REFERENCES room(room_id),
    CONSTRAINT room_for_person_person_id_fk FOREIGN KEY(person_id) REFERENCES person(phone_no)
);
```

“score” attribute is included to create a system where the user loses/gains score from completing or failing to complete tasks within the given time period. The composite primary key (room_id and person_id) enables the possibility of one user being in multiple rooms()

6. Task for person join table

```
CREATE TABLE task_for_person(
    task_id INT NOT NULL,
    person_id VARCHAR(15) NOT NULL,

    PRIMARY KEY(task_id, person_id),

    CONSTRAINT task_for_person_task_id FOREIGN KEY(task_id) REFERENCES task(task_id),
    CONSTRAINT task_for_person_person_id FOREIGN KEY(person_id) REFERENCES person(phone_no)
);
```

The composite primary key (task_id and person_id) ensures that one task can be assigned to multiple people.

7. Notification frequency for task join table

```
CREATE TABLE noti_freq_for_task(
    task_id INT NOT NULL,
    noti_freq_id SMALLINT NOT NULL,

    PRIMARY KEY(task_id, noti_freq_id),

    CONSTRAINT noti_freq_for_task_task_id_fk FOREIGN KEY(task_id) REFERENCES task(task_id),
    CONSTRAINT noti_freq_for_task_noti_freq_id_fk FOREIGN KEY(noti_freq_id) REFERENCES noti_freq(noti_freq_id)
);
```

4. 3 Components developed and their functionalities.

AddPeople.jsx

This component lets you add people to a given room.

AddRoom.jsx

This component lets you add a room. You can create rooms so you can divide tasks for e.g. “Campus group” and “Dormitory”.

AddTask.jsx

This component lets you add a task. You add tasks and assign them to people or rooms.

AvatarPopover.jsx

This is a Popover component that lets you log out, go to settings and see profile information.

FloatingActionButton.jsx

This is a standard component using the design principles in MUI, letting the user having thumb access to the most frequently used actions. In Næg, that is “Add new task”.

Header.jsx

This is the component that shows the Næg logo and lets you access the sidebar with the menu options and AvatarPopover.jsx.

Login.jsx

This is the page you will see when you go to Næg or logs out from your session. It lets you log in, and it lets you register a new account by taking you to Register.jsx.

MyAccount.jsx

MyAccount is the oversight of your account that lets you change fields like name, email, passwords, etc.

MyPeople.jsx

MyPeople is a list of all the people in all the rooms you have anything to do with.

MyRooms.jsx

MyRooms is a list of all the rooms you are a member in.

MyTasks.jsx

MyTasks are a list of all the tasks that have been assigned to you.

Payments.jsx

This is a page from when we considered implementing “pay punishment”. In addition to more and more frequent notification, one also can add debit cards that would exponentially charge the users not doing their tasks.

Register.jsx

Let you register a new account.

Settings.jsx

Let you set things the way you like it.

Statistics.jsx

Shows users and their feats.

ThemeContext.jsx

“Talks” with your system and sets the theme to auto, dark or light using your system settings.

ThemeProvider.jsx

Overrides auto/dark/light and sets custom themes like Pacific Punch or Arctic Atmosphere.

4.4 A list of any challenges or issues encountered.

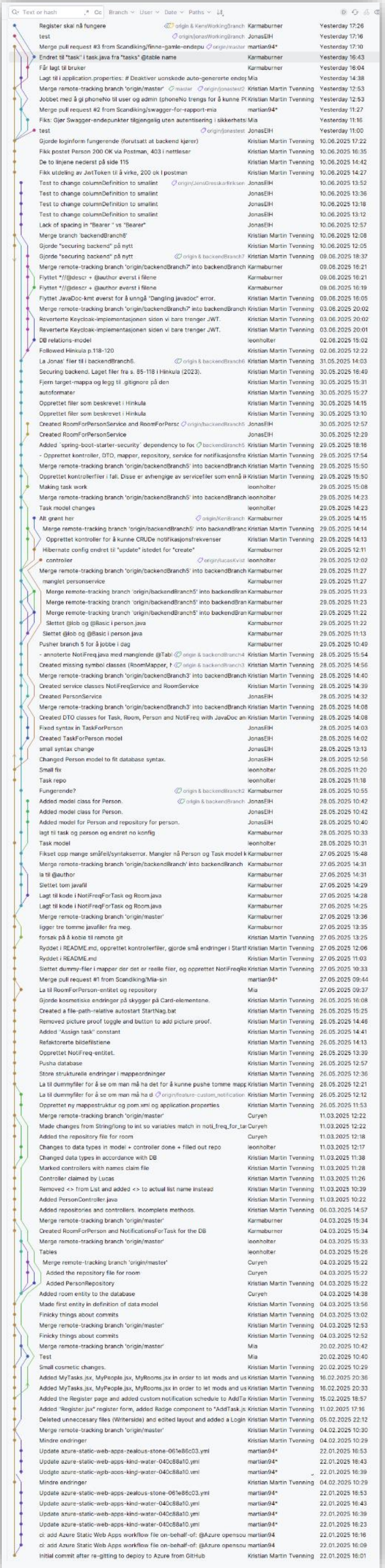
Getting the hang of Git, making sure everyone knows what Git is, what it is used for, and how to use it across team members

Using the terms "Git", "GitHub", "commit" and "push" somewhat interchangeably

Using GitHub in "best practices" like not merging to master branch "all time time", which is unprotected, or uploading "personal trash" files like compiler.xml, class files, target folders...

And it is very frustrating pulling a repo from Github that is working on one computer, and not working on the next computer. When to merge, push, rebase, pull, shelf, stash, force push, and all the WORDS that doesn`t make any sense until we learn it the hard way. It is hard to fully know what to do in the correct order and who can do what, when etc. We have been confused, all of us.

Using GitHub in an organized team manner, because (see picture):



```
Text or hash   .*  Cc   Branch ⌄  User ⌄  Date ⌄  Paths ⌄

Register skal nå fungere                                         origin & KensWorkingBranch    Karmaburner                  Yesterday 17:26
test                                                             origin/jonasWorkingBranch     JonasElH                     Yesterday 17:16
Merge pull request #3 from Scandiking/finne-gamle-endepu         origin/master                 martian94*                   Yesterday 17:10
Endret til "task" i task.java fra "tasks" @table name                                          Karmaburner                  Yesterday 16:43
Får lagt til bruker                                                                            Karmaburner                  Yesterday 16:04
Lagt til i application.properties: # Deaktiver uønskede auto-genererte endep                   Mia                          Yesterday 14:38
Merge remote-tracking branch 'origin/master'      master         origin/jonastest2             Kristian Martin Tvenning     Yesterday 12:53
Jobbet med å gi phoneNo til user og admin (phoneNo trengs for å kunne P(                       Kristian Martin Tvenning     Yesterday 12:53
Merge pull request #2 from Scandiking/swagger-for-rapport-mia                                  martian94*                   Yesterday 11:27
Fiks: Gjør Swagger-endepunkter tilgjengelig uten autentisering i sikkerhets                    Mia                          Yesterday 11:16
test                                                             origin/jonastest              JonasElH                     Yesterday 11:00
Gjorde loginform fungerende (forutsatt at backend kjører)                                      Kristian Martin Tvenning     10.06.2025 17:22
Fikk postet Person 200 OK via Postman, 403 i nettleser                                         Kristian Martin Tvenning     10.06.2025 16:35
De to linjene nederst på side 115                                                              Kristian Martin Tvenning     10.06.2025 14:42
Fikk utdeling av JwtToken til å virke, 200 ok i postman                                        Kristian Martin Tvenning     10.06.2025 14:27
Test to change columnDefinition to smallint                      origin/JensGresskarEriksen    JonasElH                     10.06.2025 13:52
Test to change columnDefinition to smallint                                                    JonasElH                     10.06.2025 13:36
Test to change columnDefinition to smallint                                                    JonasElH                     10.06.2025 13:18
Test to change columnDefinition to smallint                                                    JonasElH                     10.06.2025 13:12
Lack of spacing in "Bearer " vs "Bearer"                                                       JonasElH                     10.06.2025 12:57
Merge branch 'backendBranch6'                                                                  Kristian Martin Tvenning     10.06.2025 12:08
Gjorde "securing backend" på nytt                                                              Kristian Martin Tvenning     10.06.2025 12:05
Gjorde "securing backend" på nytt                                origin & backendBranch7       Kristian Martin Tvenning     09.06.2025 18:37
Merge remote-tracking branch 'origin/backendBranch7' into backendBranch                        Karmaburner                  09.06.2025 16:21
Flyttet *//@descr + @author øverst i filene                                                    Karmaburner                  09.06.2025 16:21
Flyttet *//@descr + @author øverst i filene                                                    Karmaburner                  09.06.2025 16:19
Flyttet JavaDoc-kmt øverst for å unngå "Dangling javadoc" error.                               Kristian Martin Tvenning     09.06.2025 16:05
Merge remote-tracking branch 'origin/backendBranch7' into backendBranch                        Kristian Martin Tvenning     03.06.2025 20:02
Reverterte Keycloak-implementasjonen siden vi bare trenger JWT.                                Kristian Martin Tvenning     03.06.2025 20:02
Reverterte Keycloak-implementasjonen siden vi bare trenger JWT.                                Kristian Martin Tvenning     03.06.2025 20:01
DB relations-model                                                                             leonholter                   02.06.2025 15:02
Followed Hinkula p.118-120                                                                     Kristian Martin Tvenning     02.06.2025 12:22
La Jonas' filer til i backendBranch6.                            origin & backendBranch6       Kristian Martin Tvenning     31.05.2025 14:03
Securing backend. Laget filer fra s. 85-118 i Hinkula (2023).                                  Kristian Martin Tvenning     30.05.2025 16:49
Fjern target-mappa og legg til .gitignore på den                                               Kristian Martin Tvenning     30.05.2025 15:31
autoformater                                                                                   Kristian Martin Tvenning     30.05.2025 15:27
Opprettet filer som beskrevet i Hinkula                                                        Kristian Martin Tvenning     30.05.2025 14:15
Opprettet filer som beskrevet i Hinkula                                                        Kristian Martin Tvenning     30.05.2025 13:10
Created RoomForPersonService and RoomForPersc                    origin/backendBranch5         JonasElH                     30.05.2025 12:57
Created RoomForPersonService                                                                   JonasElH                     30.05.2025 12:29
Added `spring-boot-starter-security` dependency to foc           backendBranch5                Kristian Martin Tvenning     29.05.2025 18:16
- Opprettet kontroller, DTO, mapper, repository, service for notifikasjonsfre                  Kristian Martin Tvenning     29.05.2025 17:54
Merge remote-tracking branch 'origin/backendBranch5' into backendBranch                        Kristian Martin Tvenning     29.05.2025 15:50
Opprettet kontrollerfiler i fall. Disse er avhengige av servicefiler som ennå ik               Kristian Martin Tvenning     29.05.2025 15:50
Making task work                                                                               leonholter                   29.05.2025 15:08
Merge remote-tracking branch 'origin/backendBranch5' into backendBranch                        leonholter                   29.05.2025 14:23
Task model changes                                                                             leonholter                   29.05.2025 14:23
Alt grønt her                                                    origin/KenBranch              Karmaburner                  29.05.2025 14:15
 Merge remote-tracking branch 'origin/backendBranch5' into backendBranc                        Kristian Martin Tvenning     29.05.2025 14:14
  Opprettet kontroller for å kunne CRUDe notifikasjonsfrekvenser                               Kristian Martin Tvenning     29.05.2025 14:13
 Hibernate config endret til "update" istedet for "create"                                     Karmaburner                  29.05.2025 12:11
  controller                                                     origin/lucasKvist             leonholter                   29.05.2025 12:02
Merge remote-tracking branch 'origin/backendBranch5' into backendBranch                        Karmaburner                  29.05.2025 11:27
 manglet personservice                                                                         Karmaburner                  29.05.2025 11:27
  Merge remote-tracking branch 'origin/backendBranch5' into backendBran                        Karmaburner                  29.05.2025 11:23
  Merge remote-tracking branch 'origin/backendBranch5' into backendBran                        Karmaburner                  29.05.2025 11:23
  Merge remote-tracking branch 'origin/backendBranch5' into backendBran                        Karmaburner                  29.05.2025 11:22
  Slettet @lob og @Basic i person.java                                                         Karmaburner                  29.05.2025 11:22
 Slettet @lob og @Basic i person.java                                                          Karmaburner                  29.05.2025 11:13
Pusher branch 5 for å jobbe i dag                                                              Karmaburner                  29.05.2025 10:49
- annoterte NotiFreq.java med manglende @Tabl                    origin & backendBranch4       Kristian Martin Tvenning     28.05.2025 15:54
Created missing symbol classes (RoomMapper, N                    origin & backendBranch3       Kristian Martin Tvenning     28.05.2025 14:56
Merge remote-tracking branch 'origin/backendBranch3' into backendBranch                        Kristian Martin Tvenning     28.05.2025 14:40
Created service classes NotiFreqService and RoomService                                        Kristian Martin Tvenning     28.05.2025 14:39
Created PersonService                                                                          JonasElH                     28.05.2025 14:32
Merge remote-tracking branch 'origin/backendBranch3' into backendBranch                        Kristian Martin Tvenning     28.05.2025 14:08
Created DTO classes for Task, Room, Person and NotiFreq with JavaDoc an                        Kristian Martin Tvenning     28.05.2025 14:08
Fixed syntax in TaskForPerson                                                                  JonasElH                     28.05.2025 14:03
Created TaskForPerson model                                                                    JonasElH                     28.05.2025 14:02
small syntax change                                                                            JonasElH                     28.05.2025 13:13
Changed Person model to fit database syntax.                                                   JonasElH                     28.05.2025 12:56
Small fix                                                                                      leonholter                   28.05.2025 11:20
Task repo                                                                                      leonholter                   28.05.2025 11:18
Fungerende?                                                      origin & backendBranch2       Karmaburner                  28.05.2025 10:55
Added model class for Person.                                    origin & backendBranch        JonasElH                     28.05.2025 10:42
Added model class for Person.                                                                  JonasElH                     28.05.2025 10:42
Added model for Person and repository for person.                                              JonasElH                     28.05.2025 10:40
lagt til task og person og endret no konfig                                                    Karmaburner                  28.05.2025 10:33
Task model                                                                                     leonholter                   28.05.2025 10:31
Fikset opp mange småfeil/syntakserror. Mangler nå Person og Task model k                       Karmaburner                  27.05.2025 15:48
Merge remote-tracking branch 'origin/backendBranch' into backendBranch                         Karmaburner                  27.05.2025 14:31
la til @author                                                                                 Karmaburner                  27.05.2025 14:31
Slettet tom javafil                                                                            Karmaburner                  27.05.2025 14:29
Lagt til kode i NotiFreqForTask og Room.java                                                   Karmaburner                  27.05.2025 14:28
Lagt til kode i NotiFreqForTask og Room.java                                                   Karmaburner                  27.05.2025 14:25
Merge remote-tracking branch 'origin/master'                                                   Karmaburner                  27.05.2025 13:36
ligger tre tomme javafiler fra meg.                                                            Karmaburner                  27.05.2025 13:35
forsøk på å koble til remote git                                                               Kristian Martin Tvenning     27.05.2025 13:25
Ryddet i README.md, opprettet kontrollerfiler, gjorde små endringer i Startf                   Kristian Martin Tvenning     27.05.2025 12:06
Ryddet i README.md                                                                             Kristian Martin Tvenning     27.05.2025 11:03
Slettet dummy-filer i mapper der det er reelle filer, og opprettet NotiFreqRe                  Kristian Martin Tvenning     27.05.2025 10:33
Merge pull request #1 from Scandiking/Mia-sin                                                  martian94*                   27.05.2025 09:44
La til RoomForPerson-entitet og repository                                                     Mia                          27.05.2025 09:37
Gjorde kosmetiske endringer på skygger på Card-elementene.                                     Kristian Martin Tvenning     26.05.2025 16:08
Created a file-path-relative autostart StartNag.bat                                            Kristian Martin Tvenning     26.05.2025 15:25
Removed picture proof toggle and button to add picture proof.                                  Kristian Martin Tvenning     26.05.2025 14:46
Added "Assign task" constant                                                                   Kristian Martin Tvenning     26.05.2025 14:41
Refaktorerte bildefilstiene                                                                    Kristian Martin Tvenning     26.05.2025 14:13
Opprettet NotiFreq-entitet.                                                                    Kristian Martin Tvenning     26.05.2025 13:39
Pusha database                                                                                 Kristian Martin Tvenning     26.05.2025 12:57
Store strukturelle endringer i mappeordninger                                                  Kristian Martin Tvenning     26.05.2025 12:36
La til dummyfiler for å se om man må ha det for å kunne pushe tomme mapp                       Kristian Martin Tvenning     26.05.2025 12:21
La til dummyfiler for å se om man må ha d                        origin/feature-custom_notification  Kristian Martin Tvenning  26.05.2025 12:12
Opprettet ny mappestruktur og pom.xml og application.properties                                Kristian Martin Tvenning     26.05.2025 11:53
Merge remote-tracking branch 'origin/master'                                                   Curyeh                       11.03.2025 12:22
Made changes from String/long to int so variables match in noti_freq_for_ta                    Curyeh                       11.03.2025 12:22
Added the repository file for room                                                             Curyeh                       11.03.2025 12:18
Changes to data types in model + controller done + filled out repo                             leonholter                   11.03.2025 12:17
Changed data types in accordance with DB                                                       Kristian Martin Tvenning     11.03.2025 11:38
Marked controllers with names claim file                                                       Kristian Martin Tvenning     11.03.2025 11:28
Controller claimed by Lucas                                                                    Kristian Martin Tvenning     11.03.2025 11:26
Removed <> from List and added <> to actual list name instead                                  Kristian Martin Tvenning     11.03.2025 10:39
Added PersonController.java                                                                    Kristian Martin Tvenning     11.03.2025 10:22
Added repositories and controllers. Incomplete methods.                                        Kristian Martin Tvenning     06.03.2025 14:57
Merge remote-tracking branch 'origin/master'                                                   Karmaburner                  04.03.2025 15:34
Created RoomForPerson and NotificationsForTask for the DB                                      Karmaburner                  04.03.2025 15:34
Merge remote-tracking branch 'origin/master'                                                   leonholter                   04.03.2025 15:33
Tables                                                                                         leonholter                   04.03.2025 15:26
 Merge remote-tracking branch 'origin/master'                                                  Curyeh                       04.03.2025 15:22
  Added the repository file for room                                                           Curyeh                       04.03.2025 15:22
  Added PersonRepository                                                                       Kristian Martin Tvenning     04.03.2025 15:22
Added room entity to the database                                                              Curyeh                       04.03.2025 14:38
Made first entity in definition of data model                                                  Kristian Martin Tvenning     04.03.2025 13:56
Finicky things about commits                                                                   Kristian Martin Tvenning     04.03.2025 13:02
Merge remote-tracking branch 'origin/master'                                                   Kristian Martin Tvenning     04.03.2025 12:53
Finicky things about commits                                                                   Kristian Martin Tvenning     04.03.2025 12:52
Merge remote-tracking branch 'origin/master'                                                   Mia                          20.02.2025 10:42
Test                                                                                           Mia                          20.02.2025 10:40
Small cosmetic changes.                                                                        Kristian Martin Tvenning     20.02.2025 10:29
Added MyTasks.jsx, MyPeople.jsx, MyRooms.jsx in order to let mods and us                       Kristian Martin Tvenning     16.02.2025 20:36
Added MyTasks.jsx, MyPeople.jsx, MyRooms.jsx in order to let mods and us                       Kristian Martin Tvenning     16.02.2025 20:33
Added the Register page and added custom notification schedule to AddTa                        Kristian Martin Tvenning     15.02.2025 18:57
Added "Register.jsx" register form, added Badge component to "AddTask.js                       Kristian Martin Tvenning     11.02.2025 17:16
Deleted unneccesary files (Writerside) and edited layout and added a Login                     Kristian Martin Tvenning     05.02.2025 22:12
Merge remote-tracking branch 'origin/master'                                                   Kristian Martin Tvenning     04.02.2025 10:30
Mindre endringer                                                                               Kristian Martin Tvenning     04.02.2025 10:29
Update azure-static-web-apps-zealous-stone-061e86c03.yml                                       martian94*                   22.01.2025 16:53
Update azure-static-web-apps-kind-water-040c88a10.yml                                          martian94*                   22.01.2025 16:43
Update azure-static-web-apps-kind-water-040c88a10.yml                                          martian94*                   22.01.2025 16:39
Mindre endringer                                                                               Kristian Martin Tvenning     04.02.2025 10:29
Update azure-static-web-apps-zealous-stone-061e86c03.yml                                       martian94*                   22.01.2025 16:53
Update azure-static-web-apps-kind-water-040c88a10.yml                                          martian94*                   22.01.2025 16:43
Update azure-static-web-apps-kind-water-040c88a10.yml                                          martian94*                   22.01.2025 16:39
Update azure-static-web-apps-kind-water-040c88a10.yml                                          martian94*                   22.01.2025 16:23
ci: add Azure Static Web Apps workflow file on-behalf-of: @Azure opensou                       martian94                    22.01.2025 16:16
ci: add Azure Static Web Apps workflow file on-behalf-of: @Azure opensou                       martian94                    22.01.2025 16:09
Initial commit after re-gitting to deploy to Azure from GitHub                                 Kristian Martin Tvenning     22.01.2025 16:01
```

Time management, delegating tasks, concretize bigger tasks into subtasks, and assigning them to persons rather than “we should have” and let it linger.

Constantly reiterating the DB regarding use case prioritization.

Initial delegation of responsibility (frontend/backend) has proven tough to follow: all group members have mostly been working on solving the same issues as we are learning the framework for the project for the first time.

4.5 Link to the repository

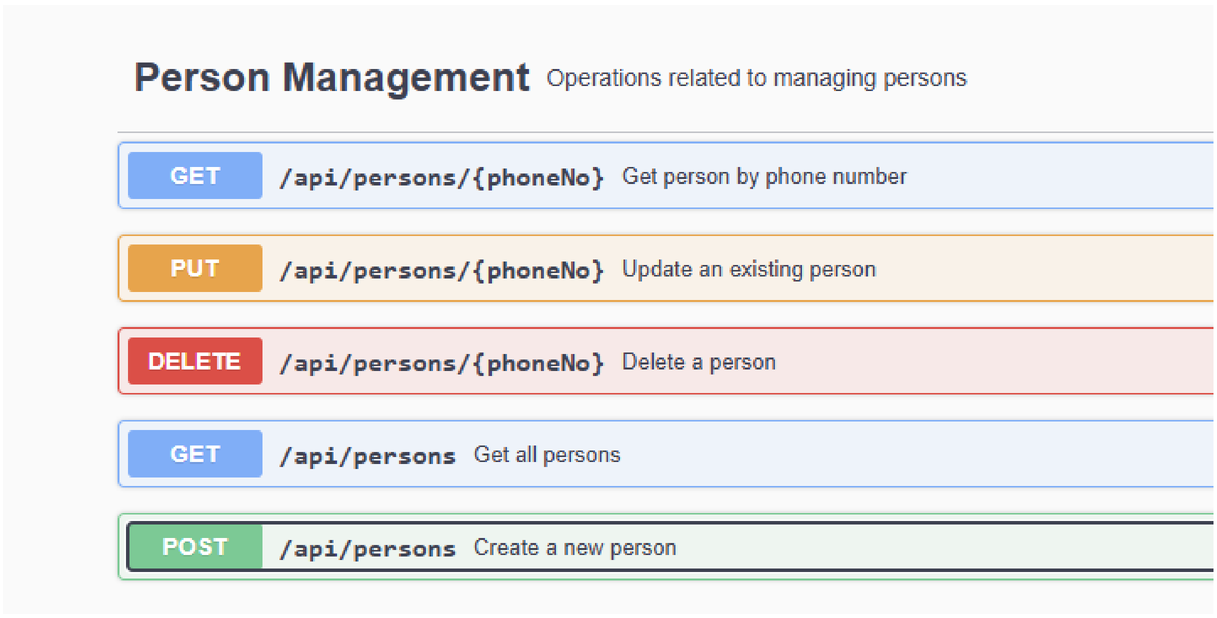
https://github.com/Scandiking/N-g

Step 5 Backend progress report

5.1 API documentation and endpoints

5.1.1 Person Management

Operations related to managing persons



Person Management Operations related to managing persons

GET /api/persons/{phoneNo} Get person by phone number

PUT /api/persons/{phoneNo} Update an existing person

DELETE /api/persons/{phoneNo} Delete a person

GET /api/persons Get all persons

POST /api/persons Create a new person

GET /api/persons/{phoneNo}
Description: Retrieve a person by their phone number.
Path parameter:

phoneNo – The person's phone number.
Response: JSON with person data.
Status codes:

200 OK – Person retrieved.

404 Not Found – Person not found.

500 Internal Server Error – Server error.

PUT /api/persons/{phoneNo}

Description: Update the details of an existing person

Path parameter:

phoneNo – The person's phone number.
Request body: JSON with updated person data.
Response: JSON with updated person data.
Status codes:

200 OK – Update successful.

400 Bad Request – Invalid input data.

404 Not Found – Person not found.

500 Internal Server Error – Server error.

DELETE /api/persons/{phoneNo}

Description: Delete a person by their phone number

Path parameter:

phoneNo – The person's phone number.
Response: None.
Status codes:

204 No Content – Successfully deleted person.

404 Not Found – Person not found.

500 Internal Server Error – Server error.

GET /api/persons

Description: Retrieve a list of all persons

Parameters: None.

Response: JSON array with person data.

Status codes:

200 OK – List retrieved.

500 Internal Server Error – Server error.

POST /api/persons

Description: Create a new person with the provided details

Request- body:

```
{
 "phoneNo": "string",
 "firstName": "string",
 "lastName": "string",
 "mail": "string",
 "profilePicture": [
   "string"
 ],
 "birthDate": "2025-06-11"
}
```

Response: JSON with new person data

Status code:

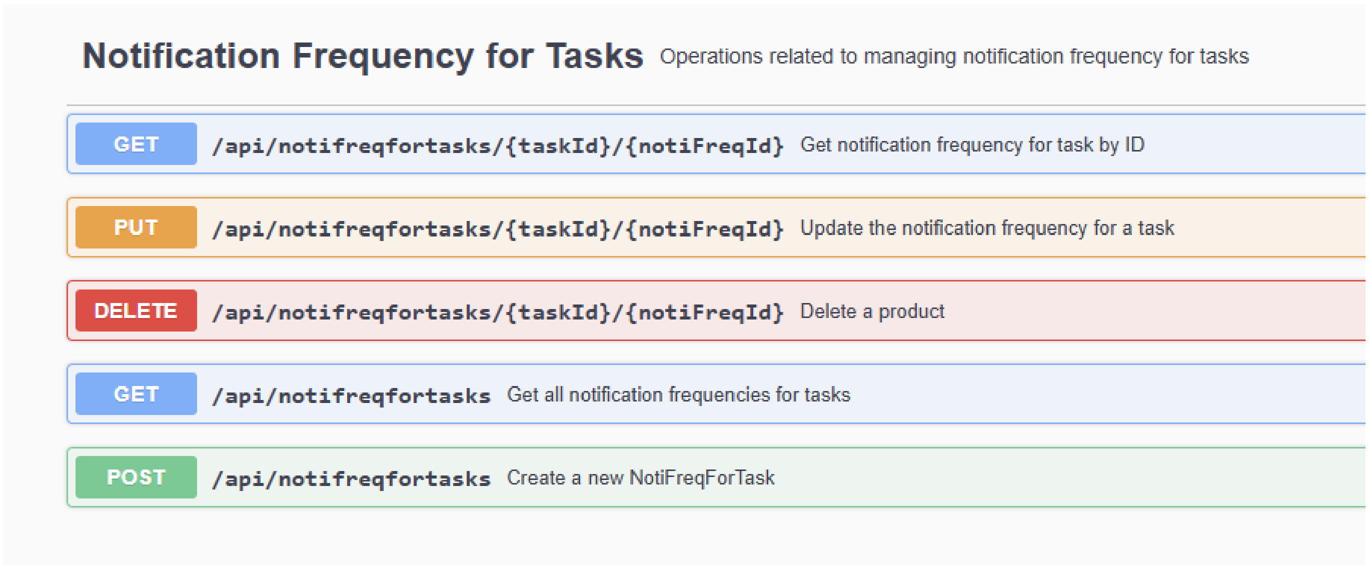
201- Successfully created person

400- Invalid input data

500- Internal server error

5.1.2 Notification Frequency for Tasks

Operations related to managing notification frequency for tasks


Notification Frequency for Tasks Operations related to managing notification frequency for tasks

GET	`/api/notifreqfortasks/{taskId}/{notiFreqId}`	Get notification frequency for task by ID
PUT	`/api/notifreqfortasks/{taskId}/{notiFreqId}`	Update the notification frequency for a task
DELETE	`/api/notifreqfortasks/{taskId}/{notiFreqId}`	Delete a product
GET	`/api/notifreqfortasks`	Get all notification frequencies for tasks
POST	`/api/notifreqfortasks`	Create a new NotiFreqForTask

GET /api/notifreqfortasks/{taskId}/{notiFreqId}

Description: Retrieve a specific notification frequency setting for a task by its task ID and notification frequency ID.

Path parameters:

taskId – The ID of the task.

notiFreqId – The ID of the notification frequency.

Response: JSON object with notification frequency data.

Status codes:

200 OK – Successfully retrieved.

404 Not Found – Not found.

500 Internal Server Error – Server error.

PUT /api/notifreqfortasks/{taskId}/{notiFreqId}

Description: Update the notification frequency for a given task.

Path parameters:

taskId – The ID of the task.

notiFreqId – The ID of the notification frequency.

Request body:

```
{
  "taskId": 0,
  "notiFreqId": 0
}
```

Response: JSON object with updated data.

Status codes:

200 OK – Update successful.

400 Bad Request – Invalid data.

404 Not Found – Entity not found.

500 Internal Server Error – Server error.

DELETE /api/notifreqfortasks/{taskId}/{notiFreqId}

Description: Delete a notification frequency setting for a task.

Path parameters:

taskId – The ID of the task.

notiFreqId – The ID of the notification frequency.

Response: None.

Status codes:

204 No Content – Successfully deleted.

404 Not Found – Entity not found.

500 Internal Server Error – Server error.

GET /api/notifreqfortasks

Description: Retrieve all notification frequency settings for tasks.
Path parameters: None.
Response: JSON array of notification frequency settings.
Status codes:

200 OK – Successfully retrieved list.

500 Internal Server Error – Server error.

POST /api/notifreqfortasks

Description: Create a new notification frequency setting for a task.
Request body:

```
{
  "taskId": 0,
  "notiFreqId": 0
}
```

Response: JSON object of created entity.
Status codes:

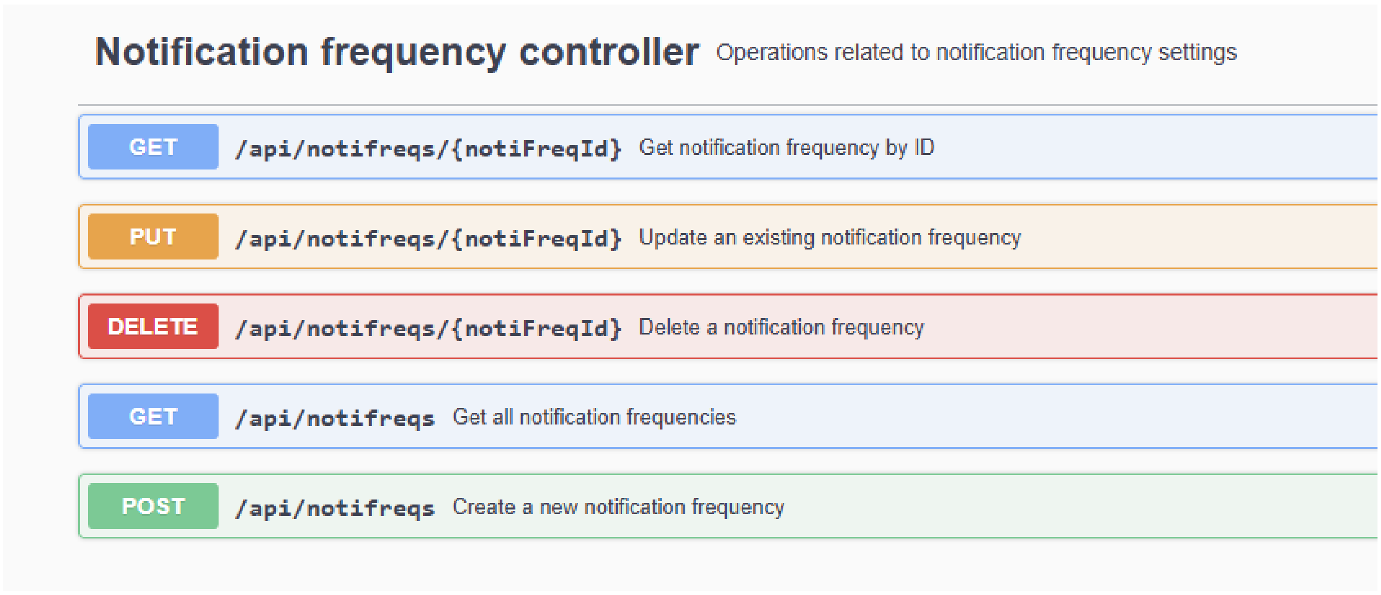
201 Created – Successfully created.

400 Bad Request – Invalid input.

500 Internal Server Error – Server error.

5.1.3 Notification frequency controller

Operations related to notification frequency settings


Notification frequency controller Operations related to notification frequency settings

GET /api/notifreqs/{notiFreqId} Get notification frequency by ID

PUT /api/notifreqs/{notiFreqId} Update an existing notification frequency

DELETE /api/notifreqs/{notiFreqId} Delete a notification frequency

GET /api/notifreqs Get all notification frequencies

POST /api/notifreqs Create a new notification frequency

GET /api/notifreqs/{notiFreqId}

Description: Retrieve a notification frequency setting by its ID.

Path parameters:

notiFreqId – The ID of the notification frequency.

Response: JSON object with notification frequency data.

Status codes:

200 OK – Successfully retrieved.

404 Not Found – Notification frequency not found.

500 Internal Server Error – Server error.

PUT /api/notifreqs/{notiFreqId}

Description: Update a notification frequency setting by its ID.

Path parameters:

notiFreqId – The ID of the notification frequency.

Request body:

```
{
  "notiFreqId": 0,
  "notiFreqTitle": "string",
  "baseInterval": "string",
  "growthFactor": 0,
  "maxRepeats": 0
}
```

Response: JSON object with updated data.

Status codes:

200 OK – Successfully updated.

404 Not Found – Notification frequency not found.

500 Internal Server Error – Server error.

DELETE /api/notifreqs/{notiFreqId}

Description: Delete a notification frequency setting by its ID.
Path parameters:

notiFreqId – The ID of the notification frequency.
Response: None.
Status codes:

204 No Content – Successfully deleted.

404 Not Found – Notification frequency not found.

500 Internal Server Error – Server error.

GET /api/notifreqs
Description: Retrieve a list of all notification frequency settings.
Path parameters: None.
Response: JSON array of notification frequency settings.
Status codes:

200 OK – Successfully retrieved.

500 Internal Server Error – Server error.

POST /api/notifreqs
Description: Create a new notification frequency setting.

Path parameters: None
Request body:

```
{
  "notiFreqId": 0,
  "notiFreqTitle": "string",
  "baseInterval": "string",
  "growthFactor": 0,
  "maxRepeats": 0
}
```

Response: JSON object of the created notification frequency.
Status codes:

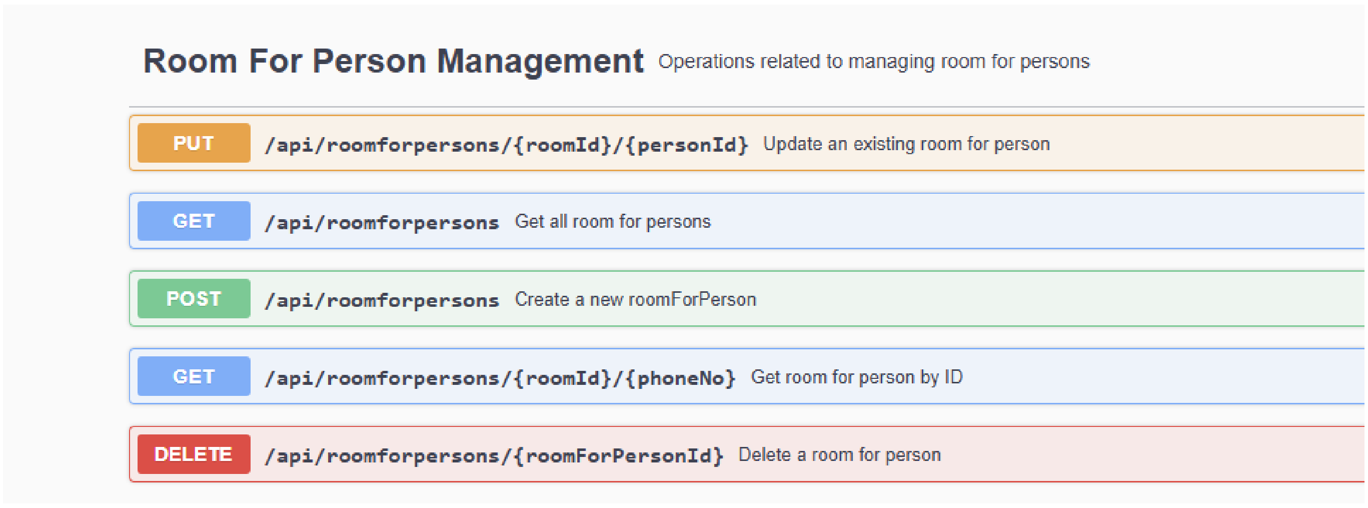
201 Created – Successfully created.

400 Bad Request – Invalid input data.

500 Internal Server Error – Server error.

5.1.4 Room for Person Management

Operations related to managing room for persons



Room For Person Management Operations related to managing room for persons

PUT /api/roomforpersons/{roomId}/{personId} Update an existing room for person

GET /api/roomforpersons Get all room for persons

POST /api/roomforpersons Create a new roomForPerson

GET /api/roomforpersons/{roomId}/{phoneNo} Get room for person by ID

DELETE /api/roomforpersons/{roomForPersonId} Delete a room for person

PUT /api/roomforpersons/{roomId}/{personId}

Description: Update the details of an existing room for person.

Path parameters:

- roomId – The ID of the room.
- personId – The ID (phone number) of the person.

  Request body:

json

```
{
 "roomId": 0,
 "phoneNo": "string",
 "score": 0,
 "room": {
  "roomId": 0,
  "roomName": "string",
  "roomDescr": "string",
  "roomAdmin": "string",
  "roomPicture": ["string"]
 },
 "person": {
  "phoneNo": "string",
  "firstName": "string",
  "lastName": "string",
  "mail": "string",
  "profilePicture": ["string"],
  "birthDate": "2025-06-11"
 }
}
```

Response: JSON object with updated room for person details.
Status codes:

200 OK – Successfully updated room for person.

400 Bad Request – Invalid input data.

404 Not Found – Room for person not found.

500 Internal Server Error – Server error

GET /api/roomforpersons

Description: Retrieve a list of all room for persons.

Path parameters: None.

Response: JSON array of room for person objects.

Status codes:

200 OK – Successfully retrieved list of room for persons.

500 Internal Server Error – Server error.

POST /api/roomforpersons

Description: Create a new room for person with the provided details.

Path parameters: None.

Request body:

json

```
{
 "roomId": 0,
 "phoneNo": "string",
 "score": 0,
 "room": {
  "roomId": 0,
  "roomName": "string",
  "roomDescr": "string",
  "roomAdmin": "string",
  "roomPicture": ["string"]
 },
 "person": {
  "phoneNo": "string",
  "firstName": "string",
  "lastName": "string",
  "mail": "string",
  "profilePicture": ["string"],
  "birthDate": "2025-06-11"
 }
```

}

Response: JSON object of the created room for person.
Status codes:

201 Created – Successfully created room for person.

400 Bad Request – Invalid input data.

500 Internal Server Error – Server error.

GET /api/roomforpersons/{roomId}/{phoneNo}

Description: Retrieve a room for person by room ID and phone number.

Path parameters:

roomId – The ID of the room.

phoneNo – The phone number of the person.
Response: JSON object with room for person details.
Status codes:

200 OK – Successfully retrieved room for person.

404 Not Found – Room for person not found.

500 Internal Server Error – Server error

DELETE/ api/roomforpersons/{roomForPersonId}

Description: Delete a room for person by its ID.

Path parameters:

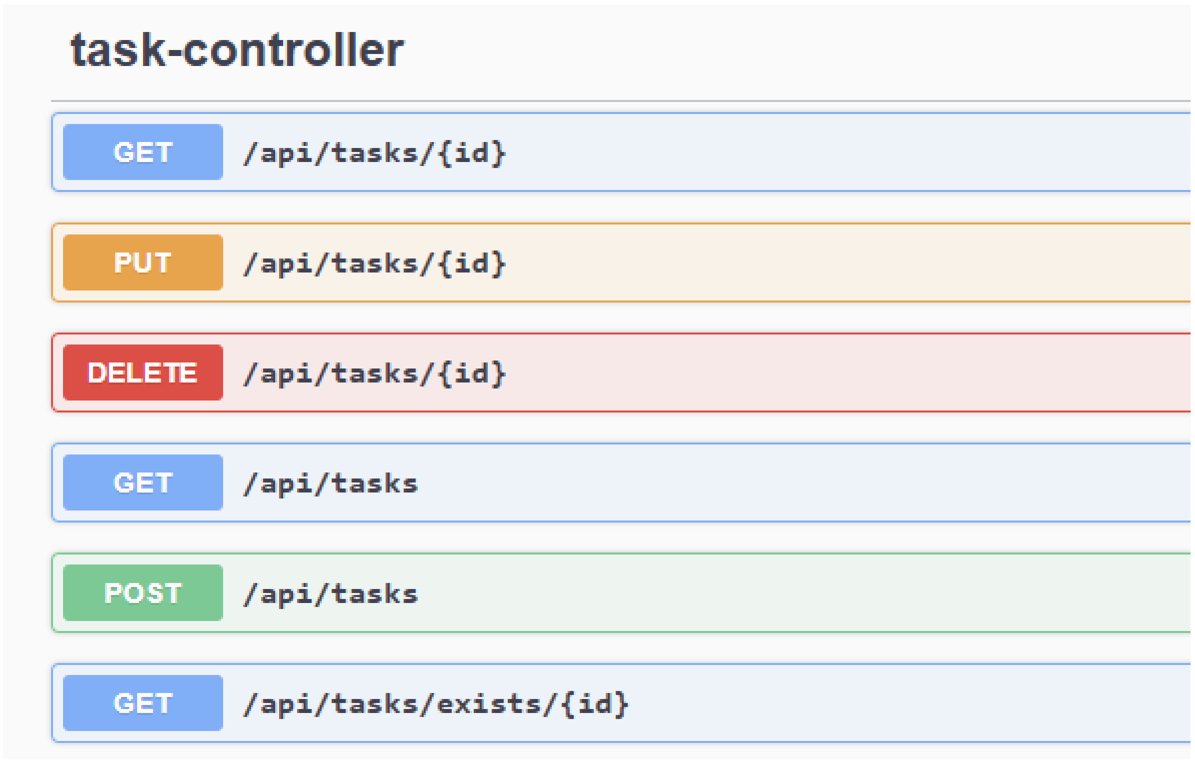
roomForPersonId – The ID of the room for person.
Response: None.
Status codes:

204 No Content – Successfully deleted room for person.

404 Not Found – Room for person not found.

500 Internal Server Error – Server error.

5.1.5 Task controller


task-controller

GET /api/tasks/{id}

PUT /api/tasks/{id}

DELETE /api/tasks/{id}

GET /api/tasks

POST /api/tasks

GET /api/tasks/exists/{id}

GET /api/tasks/{id}

Description: Retrieve a task by its ID.
Path parameters:

id (integer) – The ID of the task.
Response: JSON object with task details.
Status codes:

200 OK – Successfully retrieved task.

Example response body:

```
{
  "taskId": 0,
  "title": "string",
  "description": "string",
  "dueDate": "2025-06-11T14:53:05.458Z",
  "notiFreqId": "string",
  "creator": "string",
  "completed": true,
 "notiFreq": {
  "notiFreqId": 0,
  "notiFreqTitle": "string",
  "baseInterval": "string",
  "growthFactor": 0,
  "maxRepeats": 0
 },
 "person": {
  "phoneNo": "string",
  "firstName": "string",
  "lastName": "string",
  "mail": "string",
  "profilePicture": ["string"],
  "birthDate": "2025-06-11"
 }
}
```

PUT /api/tasks/{id}

Description: Update an existing task by its ID.
Path parameters:

id (integer) – The ID of the task.
Request body: JSON object with updated task details (same structure as GET response).
Response: JSON object with updated task details.
Status codes:

200 OK – Successfully updated task.

DELETE /api/tasks/{id}

Description: Delete a task by its ID.
Path parameters:

id (integer) – The ID of the task.
Response: None.
Status codes: 200 OK – Successfully deleted task

GET /api/tasks

Description: Retrieve a list of all tasks.
Path parameters: None.
Response: JSON array of task objects.
Status codes:

200 OK – Successfully retrieved tasks list.

Example response body:

```
[
 {
   "taskId": 0,
   "title": "string",
   "description": "string",
   "dueDate": "2025-06-11T14:53:05.463Z",
   "notiFreqId": "string",
   "creator": "string",
   "completed": true,
   "notiFreq": {
    "notiFreqId": 0,
    "notiFreqTitle": "string",
    "baseInterval": "string",
    "growthFactor": 0,
    "maxRepeats": 0
   },
   "person": {
    "phoneNo": "string",
    "firstName": "string",
    "lastName": "string",
    "mail": "string",
    "profilePicture": ["string"],
    "birthDate": "2025-06-11"
   }
  }
]
```

POST /api/tasks

Description: Create a new task with the provided details.
Path parameters: None.
Request body: JSON object with new task details (same structure as GET response).
Response: JSON object with created task details.
Status codes:

200 OK – Successfully created task.

GET /api/tasks/exists/{id}

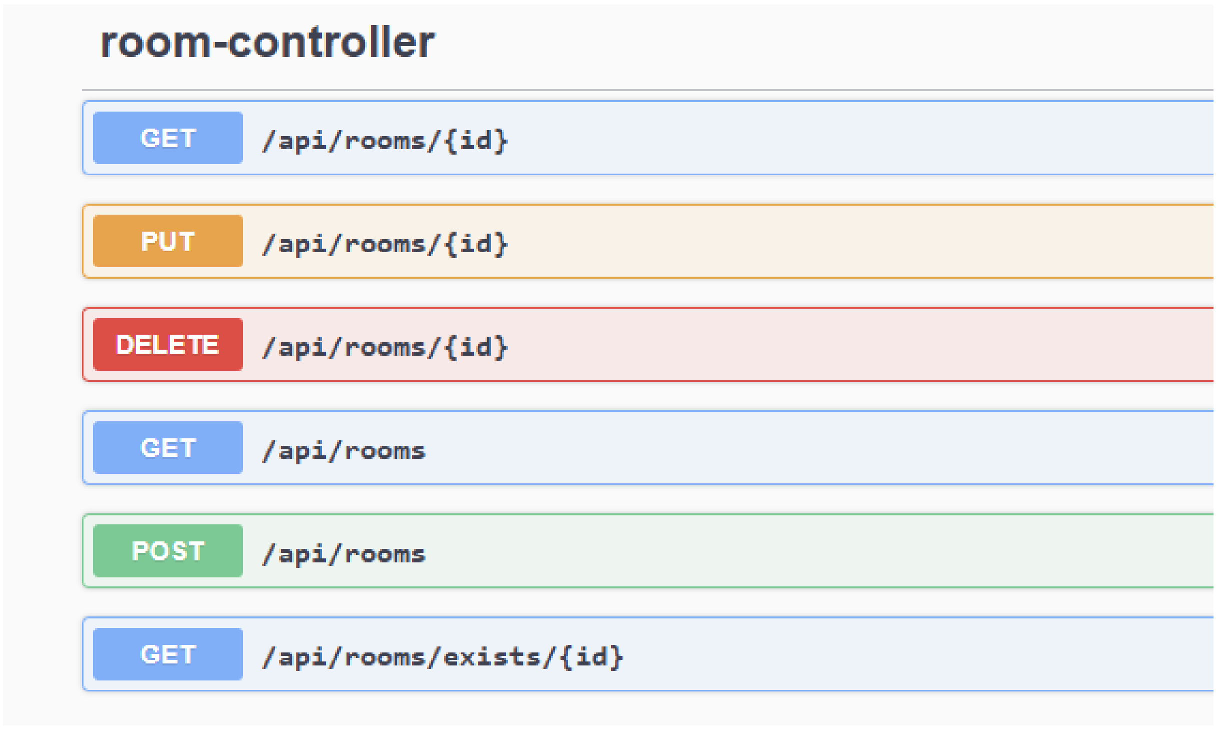
Description: Check if a task exists by its ID.
Path parameters:

id (integer) – The ID of the task.
Response: Boolean (true or false).
Status codes:

200 OK – Successfully checked task existence.

5.1.6 Room controller

Operations for managing rooms



room-controller

GET /api/rooms/{id}

PUT /api/rooms/{id}

DELETE /api/rooms/{id}

GET /api/rooms

POST /api/rooms

GET /api/rooms/exists/{id}

GET /api/rooms/{id}

Description: Retrieve details of a room by its ID.

Path parameters:

id (integer) – The ID of the room.

Response: JSON object with room details.

Status codes:

200 OK – Successfully retrieved room.

Example response body:

```
{
  "roomId": 0,
  "roomName": "string",
  "roomDescr": "string",
  "roomAdmin": "string",
  "roomPicture": ["string"]
}
```

PUT /api/rooms/{id}

Description: Update a room by its ID.
Path parameters:

id (integer) – The ID of the room.
Request body: JSON object with updated room details (same structure as GET response).
Response: JSON object with updated room details.
Status codes:

200 OK – Successfully updated room.

DELETE /api/rooms/{id}

Description: Delete a room by its ID.
Path parameters:

id (integer) – The ID of the room.
Response: None.
Status codes:

200 OK – Successfully deleted room.

GET /api/rooms

Description: Retrieve a list of all rooms.
Path parameters: None.
Response: JSON array of room objects.
Status codes:

200 OK – Successfully retrieved rooms list.

Example response body:

```
[
 {
   "roomId": 0,
   "roomName": "string",
   "roomDescr": "string",
   "roomAdmin": "string",
   "roomPicture": ["string"]
 }
]
```

POST/api/rooms

Description: Create a new room with the provided details.

Path parameters: None.

Request body: JSON object with new room details (same structure as GET response).

Response: JSON object with created room details.

Status codes:

200 OK – Successfully created room.

GET/api/rooms/exists/{id}

Description: Check if a room exists by its ID.

Path parameters:

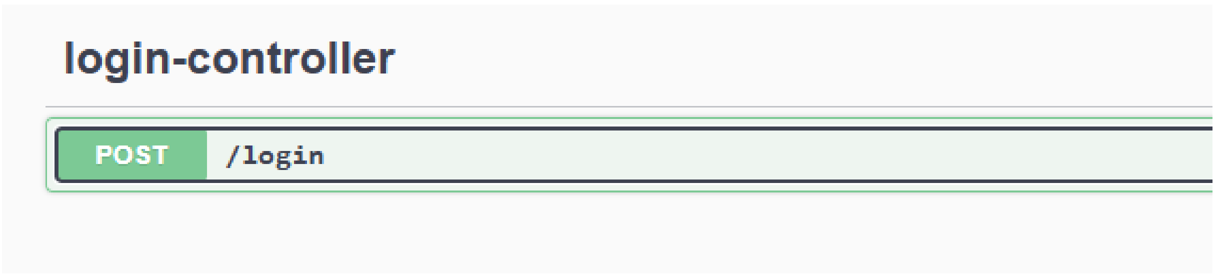
id (integer) – The ID of the room.

Response: Boolean (true or false).

Status codes:

200 OK – Successfully checked room existence.

5.1.7 Login controller



login-controller

POST /login

POST /login

Description: Authenticate a user with username and password.

Parameters: None.

Request body (JSON):

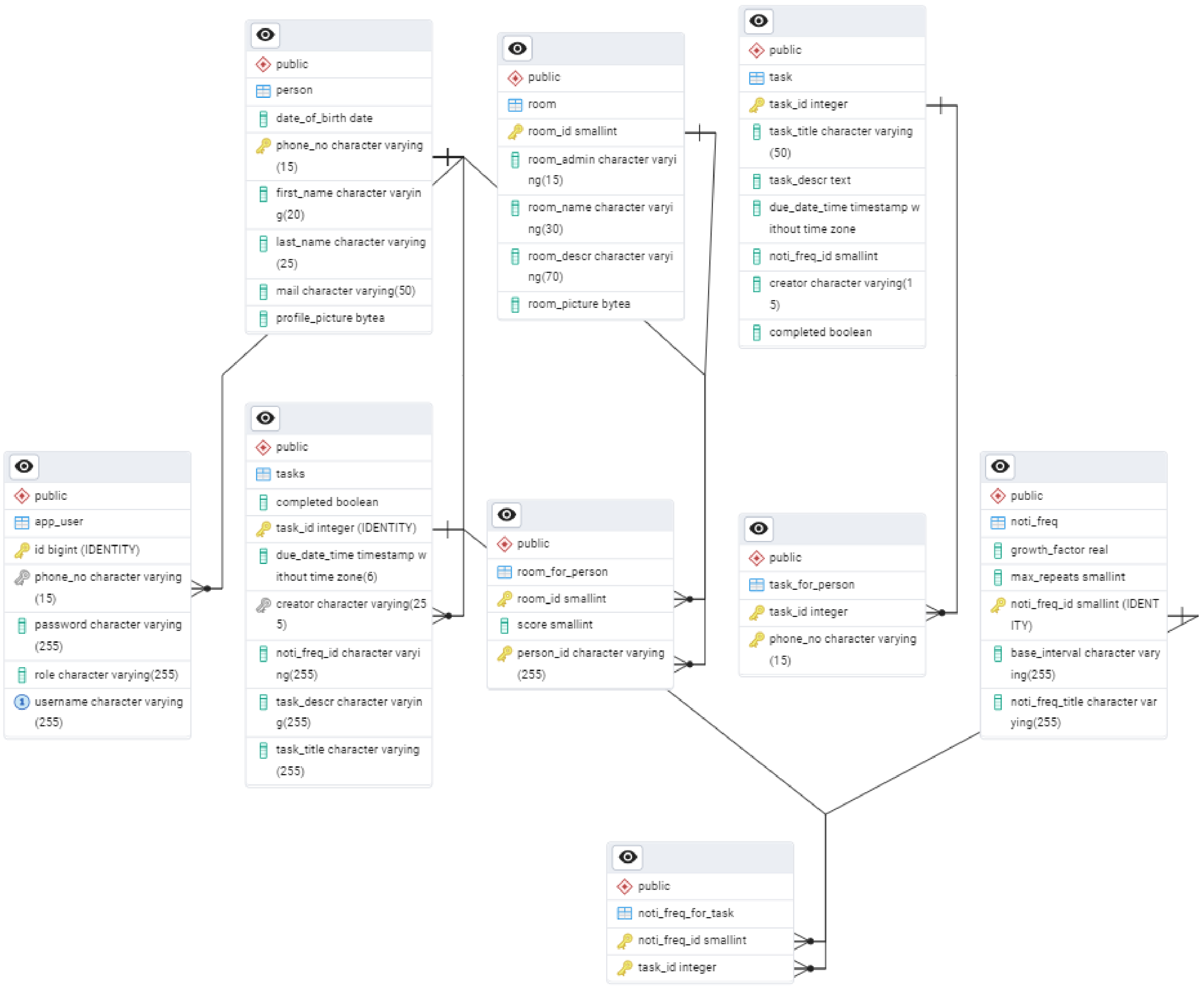
```
{
  "username": "string",
  "password": "string"
}
```

Status codes:

200 OK

Response body: (empty JSON object {} or authentication result, depending on implementation)

5.2 Database schema diagram


public
person
date_of_birth date
phone_no character varying (15)
first_name character varying(20)
last_name character varying (25)
mail character varying(50)
profile_picture bytea
public
room
room_id smallint
room_admin character varying(15)
room_name character varying(30)
room_descr character varying(70)
room_picture bytea
public
task
task_id integer
task_title character varying (50)
task_descr text
due_date_time timestamp without time zone
noti_freq_id smallint
creator character varying(15)
completed boolean
public
app_user
id bigint (IDENTITY)
phone_no character varying (15)
password character varying (255)
role character varying(255)
username character varying (255)
public
tasks
completed boolean
task_id integer (IDENTITY)
due_date_time timestamp without time zone(6)
creator character varying(255)
noti_freq_id character varying(255)
task_descr character varying(255)
task_title character varying (255)
public
room_for_person
room_id smallint
score smallint
person_id character varying (255)
public
task_for_person
task_id integer
phone_no character varying (15)
public
noti_freq
growth_factor real
max_repeats smallint
noti_freq_id smallint (IDENTITY)
base_interval character varying(255)
noti_freq_title character varying(255)
public
noti_freq_for_task
noti_freq_id smallint
task_id integer

5.3 Security implementation details

JWT Token Management

The application implements JSON Web Token based authentication using Spring Security, following a stateless authentication pattern suitable for REST APIs. The token uses local storage which eliminates server-side session storage. The JwtService class handles token generation based on imported libraries and the token gets signed using the Jwts.builder and the key from java.security. Key and returns this as a token. This token is passed to the getAuthUser method and checked if it contains not null && startsWith the Bearer prefix, Jwts parses it and returns a valid user. The tokens are issued upon successful logins to the /login endpoint. AuthenticationFilter validates the JWT token on each request. There is also a OncePerRequestFilter to ensure single execution per request. The token is valid for way too long in a production environment, and there is no token expiration handling.

User Authentication System

Dual Entity Design: Separates Person (business data) and AppUser (authentication data) (separation of concerns).

Password encoding with BCryptPasswordEncoder hashing with proper salt generation, and Key with SignatureAlgorithm.HS256

Security Configuration

Cross-site request forgery disabled (appropriate for JWT-based APIs)

Cross-origin resource sharing enabled with overly permissive configuration considering allowedOrigins: “*”

Stateless session management

The public endpoints are /login and /register, which is kinda necessary... All the other endpoints require authentication.

Registration security

There are several validation measures implemented: required field validation, duplicate email/phone number prevention, automatic password hashing, and proper Person-to-AppUser linkage during user creation. There are no password strength requirements, your password could be “a”. Also, in general there are practically no input sanitization beyond the null checks.

5.5 Known issues and limitations

Database and data management

- Profile picture uploading functionality not implemented, bytea data type unused
- Date of birth field collected but not utilized (violates GDPR’s data minimization principles)
- Notification frequency settings return null values when tasks are created
- For scaling the project, implementation of indexes for some join tables such as task for person for more seamless access
- Trouble using annotations like “@Data” when project is pushed to git and updated on a different computer because of unknown Lombok stubbornness.

User interface (UI) and User Experience (UX)

- Multiple non-functional UI components:
  - Task info button (no action)
  - Complete task button (only removes card from view)
  - Settings page switches (mix of planned and decorative elements)
  - Payments and Statistics page (decorative only)
- Floating Action Button appears on all pages at all times, including login/register
- Avatar popover displays example data instead of actual user information
- Profile page (/profile) displays only the header bar, but the actual component is blank
- Logout functionality incomplete – avatar popover and sidebar remain visible, does not “flush” JWT

- A user can register other users to the database via “add people”. This should either be a function for administrators, or else the user should only be registered to the database in case of filling out a register form. The function could also represent the friends of one user when not logged into administrator, however that could require an additional table that holds the connection between the users
- There are CRUD missing in some places in the UI, like for instance in /myrooms there should be an option for the user to delete rooms that the user administrates, and in /mypeople the user should be able to remove a person they added to their mypeoples list
-

Notification system

- No push notifications
  - Task assignments
  - Room additions
  - General user interactions
- Only local Snackbar responses provide user feedback
- Custom notification frequency option limited to single actual notification
- Custom notification time picker restricted to 00:00 – 01:00 range for second notification

Social features

- Room invitations lack approval/denial mechanism – users added automatically
- No notifications sent when users are added to rooms

Security

- JWT token expiration set to 24 hours
- No password strength requirements, password can literally be “a”

Deployment and code quality

- Application runs locally only, Azure deployment process unclear
- Database deployment strategy undefined

- Unused methods and import statements throughout codebase (deprioritized for GUI development and report requirement fullfilments

Legacy features

- Task cards contain unused Media elements (remnant from “picture proof” feature)

5.6 Setup and deployment instructions

How to install

There are essentially two minuscule different ways; either by using the command line interface, or by double-clicking Autorun.bat, which just does the CLI part automatically + a couple checks.

Instructions for CLI installation.

1. Open a Command window
2. Enter github clone https://github.com/Scandiking/N-g
3. Press enter to continue
4. Unzip the folder
5. Go in the \backend\src\main\java\com\nag and enter javac NagApplication
6. Click enter.
7. Enter java NagApplication to run the backend
8. Click enter
9. Change directory from the already open terminal to /frontend in the newly unzipped folder by entering “cd..” and enter, then “cd /frontend”.
10. Enter ` npm start` to start the frontend (check localhost:3000 or what the console output says if it does not open automatically)

Autorun.bat

Instructions for installing the application using the autorun.bat-file.

1. Download the zip from the sidebar to the right (On GitHub repo)
2. Unzip the folder
3. Click the StartNag.bat file
4. The browser will open Næg in a new tab if everything goes according to the plan, and the console will say what is wrong if it does not go according to plan.

Scandiking/N-g: Task management with nagging

5.7 Submit repository link with a completed backend implementation

Link to the repository

https://github.com/Scandiking/N-g

Please use this branch for evaluation since we had problems with merging to master and we do not dare to mess to much with it so few hours before deadline:

https://github.com/Scandiking/N-g/tree/NavWorkingBranch

5.8 Individual contributions to the project

Jonas: While much of our work overlapped—since we collaborated on integrating the various layers of the app—I took primary ownership of the controller layer. My role was to map incoming HTTP requests to service-layer calls, then serialize the service responses back into HTTP/JSON replies. This is done by using Spring's mapping anootations @GetMapping, @PostMapping, @DeleteMapping, etc to tie HTTP methods and URL patterns to Java methods. Incoming JSON payloads get bound to DTOs by using the @RequestBody (and validated with @Valid). This is a general overview of how the logic behind the controllers work.

Kristian: Kristian started to make a React frontend mock-up based on the MUI component library as per group agreements. My contribution to the MVC framework has been setting up security, following Hinkulas "Securing the backend" chapter, and undoing the Keycloak "tree trunk" as this was not described in the book, and the Baeldung tutorial seemingly skipped a few steps, but as per clarification on Canvas, there still is a JSON Web Token for request authentication. The shortcoming of the security is mentioned in 5.3 and 5.5. Further, I initially wrote the controllers, but as Jonas wrote, we have redone the code a lot of times, including commits to different branches to "make things work", so the toasts in IntelliJ might "lie". And the @author tag is used somewhat "on and off". And a commit does not equal having written the code. There are no freeloaders though, that's for sure.

I also wrote and checked the StartNag.bat because it is practical and aids the user somewhat in compiling the main application, running both the backend and the frontend in one go. And the README.md because I write notes in Obsidian which uses Markdown for formatting, and it is one of the first thing I do when creating a project anyway.

Kenneth: Worked on different CRUD operations, like create user, create room, assign tasks to persons, add people to room. Also worked on polishing UX, making it look smoother but still not happy. As part of my contribution to the project, I was partially responsible for implementing the service layer logic for managing data.

In the service class, I implemented methods for performing all standard CRUD operations. This includes creating a new person, updating existing person data, and deleting a person from the database. To ensure data consistency and error handling, I used @Transactional annotations and threw EntityNotFoundException when attempting to retrieve or delete a person that does not exist.

Additionally, I created two utility methods for mapping between the entity class (Person) and the data transfer object (PersonDTO). This mapping helps to decouple the service layer from the database schema and ensures that only relevant data is exposed to clients, improving both security and maintainability.

I also incorporated validation logic in the creation method to prevent duplicate entries based on the phone number, which serves as the primary key for person records. Throughout the implementation, I tried to follow best practices for structuring service logic in a clean way using Spring's annotation-based configuration.

I also contributed with very dry jokes, occasionally hickups and the smell of boiled eggs.

Lucas: Started the work around the application by discussing the database entities for Næg and going back and forth on their attributes. Wrote script for database and managed the entity relationships.

Even though we technically had different areas of responsibility, much of the work we individually did for the different parts of the application merged into each other because of troubleshooting and other discussions. Furthermore, I contributed by building models in Java around the database entities of Task, Notification_frequency and Task_for_person, as well as the task-, person- and room repositories. Most of the work I did was for the backend of the application. I did very little frontend work, but managed to help fix a routing bug where the navbar profilebutton did not show. I also did some testing on CRUD operations for some of the tables via Postman requests, especially for task.

I came across various challenges when using some of the Lombok annotations that are supposed to reduce boilerplate code, because of beans and constructors not being found. This resulted in taking one step down the advanced-ladder and holding on to the @Getters and @Setters annotations.

Mia: I have been a project manager and worked on keeping track of deadlines and progress (and at times the lack thereof, due to a very busy semester). I have been responsible for the report with the final structure. I specifically contributed to documenting the API endpoints here. This project has had a steep learning curve. I have spent a lot of time getting comfortable with Git. I have created many branches where I worked extensively to fix and get things to work, only to delete them later since they couldn't be merged or issues were resolved in the meantime.

We have had challenges in distributing tasks among us because much of the code depends so heavily on other parts. Therefore, there was a lot of overlapping responsibilities and much collaboration where we worked together to find solutions. The idea of 'næg' itself originates from me and my experience with my partner, who felt I 'nagged' a lot about everything he didn't do/should have done. In our group, we developed the idea further to apply to groups of both students and families living together. The idea is to create a more harmonious daily life where the system handles the nagging, so everyone can be less irritated with each other, and also to help remind about tasks that one might overlook.

I have contributed to coding specifically RoomForPerson, RoomForPersonId, and RoomForPersonRepository. Otherwise, it's been a lot of trial and error, and some fixes to expose endpoints in Swagger without authorization. Had we (or I) had more time this semester, I would have wanted to contribute more, but we have learned a lot together and I appreciate working with everyone in the group.

Sources

Asbjørnsen, D. & Maasø, A. (2015). *Bachelorboka: slik lykkes du som student*. 3. utg. Universitetsforlaget.

Dale, G., Rock, A. J., & Clark, G. I. (2020). Cue-Reactive Imagery Mediates the Relationships of Reward Responsiveness with Both Cue-Reactive Urge to Gamble and Positive Affect in Poker-Machine Gamblers. *Journal of Gambling Studies*, *36*(4), 1045–1063. https://doi.org/10.1007/s10899-019-09864-x

Hinkula, J. (2023). *Full stack development with Spring Boot 3 and React* (4th ed.). Packt Publishing.

Johannes, N., Dora, J., & Rusz, D. (2019). Social Smartphone Apps Do Not Capture Attention Despite Their Perceived High Reward Value. *Collabra: Psychology*, *5*(1), 14. https://doi.org/10.1525/collabra.207