UNIVERSITY OF COPENHAGEN

Bachelor's Thesis

Development of a Task List Application for Nursing Home Caregivers Using DCR Graphs

Aditya Fadhillah (hjg708)
Nikolaj Frølund Schaltz (bxz911)

Supervisor: Thomas Troels Hildebrandt

Datalogisk Institut
Københavns Universitet
Danmark
June 10, 2024

Abstract

The world is experiencing a significant demographic shift, particularly in countries like Denmark, where the elderly population, makes up a large portion of society. This shift places considerable strain on healthcare systems, especially in nursing homes, due to the increasing demand for senior care and the shortage of available workers. To address these challenges, this project aimed to develop a task list application specifically designed for nursing home caregivers, utilizing Dynamic Condition Response (DCR) Graphs as the foundation of the task management system. The project followed a structured development approach based on Process-Aware Information Systems (PAIS) and incorporated methodologies such as Participatory Design and SWOT Analysis. Through Participatory Design, regular interviews were conducted with nursing home caregivers. Through these interviews, it was discovered that their current task management solution lacked detailed task lists. Testing and analysis confirmed the application's reliability and potential for scalability, highlighting areas for future improvement in security and loading times. This project demonstrates the effectiveness of using DCR Graphs to enhance task management in nursing homes, resulting in an efficient and scalable mobile application. This project also highlights key functionalities and provides a foundation for further enhancements.

Contents

1 Introduction

The world is undergoing a significant demographic shift, leading to a rising proportion of the elderly population. This demographic trend is pronounced in countries like Denmark, where the number of individuals over 67 is increasing rapidly in the next few decades. This trend is adversely impacting various sectors of the healthcare system, specifically for senior care. In 2019, older people consumed about one-third of hospital care, and this trend will likely continue to grow as populations age. A study shows that by 2050, the total number of hospital days attributed to individuals over 70 will more than double. This rising demand is placing substantial strain on nursing homes, where the number of available workers does not match the influx of senior residents. In such a scenario, enhancing operational efficiency and improving the quality of care through innovative solutions becomes imperative [1].

An important aspect of modern healthcare innovation involves the integration of mobile devices and applications, collectively known as electronic health (eHealth) and mobile health (mHealth) [2]. These technologies have transformed mobile devices into essential medical tools. By utilizing medical apps, healthcare workers gain instant access to a broad range of medical references and drug information, which greatly enhances point-of-care treatment, the critical moments when healthcare services are delivered to residents. Additionally, these apps support the management of personal files and improve productivity and time management, making them indispensable in the healthcare setting [3].

Building on the foundation of eHealth and mHealth, the project aims to develop a task list application specifically designed for nursing home caregivers. This application will utilize Dynamic Condition Response (DCR) [4] Graphs to model and simulate work processes. It is designed to provide nursing home caregivers with a clear, interactive checklist of tasks to be completed during their shifts. Implementing such a system could not only streamline daily operations but also mitigate the risk of errors by reducing the time staff spend manually logging routine tasks, thereby minimizing the likelihood of forgetting critical information or recording it incorrectly.

In the following section, the background of the project is explained, focusing on how electronic health and mobile health can enhance healthcare facilities. It explores the qualitative requirements for the successful implementation of the mHealth application, existing healthcare technologies, and beneficial cloud architecture types. The methodologies employed include Participatory Design and SWOT Analysis, chosen for their user-centered approach. The tools section introduces the technologies used, such as Process-Aware Information Systems (PAIS) with an emphasis on DCR, the Kivy Python Framework, MySQL, and RESTful API. The case study section focuses on the need for an improved mobile application to enhance the operational capabilities of healthcare professionals in a Danish nursing home. The design section details how the initial problem statement was transformed into a usable mobile application, utilizing DCR Graphs and the PAIS model. The implementation section describes the creation of DCR Graphs, functions for accessing simulations, multi-user support, note creation, and database maintenance. Testing highlights the methodologies used, including unit testing for individual functions and exploratory testing for overall functionality, usability, and visual aspects, ensuring the application's reliability and user satisfaction. The discussion evaluates the use of Participatory Design, analyzes the mobile application through a SWOT analysis, assesses the fulfillment of qualitative requirements, addresses security considerations, and outlines potential areas for future work.

2 Background

This section presents a comprehensive background to provide a deeper understanding of the background information regarding the development of this project. It focuses on how electronic health (eHealth) [2] and mobile health (mHealth) [2] can significantly enhance healthcare facilities, with a focus on a nursing home used as a case study. This section explores the qualitative requirements essential for the successful implementation of the mHealth application. Additionally, it addresses existing healthcare technologies and discusses the different cloud architecture types that could be beneficial for the project.

2.1 eHealth & mHealth

eHealth, encompassing all aspects of electronic health, involves the use of information and communication technologies for health [2]. It includes mHealth which is the integration of mobile technology into the healthcare sector. It involves using mobile devices such as smartphones, tablets, and other personal digital assistants to support medical and public health practices [2]. mHealth applications allow for the collection and sharing of health information more efficiently. Through mHealth, healthcare professionals can access resident records on the go, enhancing decision-making capabilities and enabling immediate action in resident care. Residents themselves benefit from mHealth by being able to monitor their health data, access medical information, and communicate more easily with healthcare providers [3].

Nursing homes can greatly benefit from mHealth, especially when staff are away from their workstation and interacting with residents. This mobility facilitated by mHealth allows staff to respond to residents' needs more swiftly and make informed decisions based on data on their devices. For instance, nurses can update patient records, check medication schedules, and receive alerts about residents' conditions directly on their mobile devices. Additionally, mHealth tools can facilitate better coordination among the caregivers [3, 5, 6, 7].

The purpose of this project is to enhance the existing mHealth architecture by introducing a detailed task list mHealth application specifically designed for caregivers in nursing homes, utilizing the DCR Graph.
The flexibility and immediate access to information that mHealth provides have the potential to make healthcare systems more efficient. However, the successful implementation of mHealth solutions requires careful consideration of user needs to avoid under-utilization and maximize their potential benefits.

2.2 Qualitative requirements

The integration of eHealth and mHealth in healthcare has significantly enhanced the quality of care delivered by healthcare organizations. These technologies enable improved patient management, real-time data accessibility, and better decision-making processes. As the adoption of these technologies expands, focusing on their qualitative requirements is crucial for effective integration into healthcare systems.

Security is a crucial aspect of eHealth systems due to the storage of vast amounts of Electronic Health Records (EHR) [8] and other sensitive data, which pose risks of unauthorized access and data breaches. Implementing rigorous security measures such as data encryption, secure data transmission, and strict access controls is essential. These measures not only keep patient information safe but also uphold the integrity of healthcare data [9]. Additionally, compliance with regulations like the General Data Protection Regulation (GDPR) is necessary to ensure that patient data is handled lawfully and transparently, providing patients with control over their personal information [10].

Alongside security, flexibility in eHealth systems refers to the technology's ability to adapt to the evolving needs of healthcare providers and patients. This includes the capability to modify functionalities as healthcare practices. Systems should be designed to accommodate new modules or updates without requiring significant changes to the existing implementation[9].

A flexible implementation would facilitate scalability, which is a crucial factor to consider. As healthcare facilities grow and the volume of data increases, eHealth systems must be able to expand without losing functionality or compromising security [9].

Moreover, the effectiveness of eHealth applications is critical for their acceptance and use by healthcare workers. These implementations have to be reliable in performing their intended functions, which could be managing patient data, facilitating communication between caretakers, or supporting the diagnosis and treatment of patients. For these applications to be effective, they have to integrate into the existing frameworks[9].

The implementations must be reliable, operating consistently without failures, and delivering accurate information quickly to healthcare workers. This is especially important in critical case situations involving the health status of a patient where accurate data is essential for making informed decisions. For instance, in

emergencies where a patient's condition is changing rapidly, the ability to access current and accurate medical records can mean the difference between an effective treatment and a potentially harmful mistake[9].

While addressing the technical and operational needs, it is equally important to consider the user experience in the design of these applications. The user experience covers how healthcare professionals interact with the technology [9].

The success of digital health technologies, categorized under eHealth and mHealth, requires that they meet high standards of security, flexibility, scalability, effectiveness, and reliability while providing an exceptional user experience. These qualitative requirements help implement health technologies that can lead to more efficient healthcare delivery and improve patient treatment and outcomes.

2.3 Existing healthcare technologies

The integration of mobile devices and applications into healthcare has significantly transformed various aspects of medical practice in Denmark, offering numerous benefits including improved service quality and enhanced healthcare delivery.

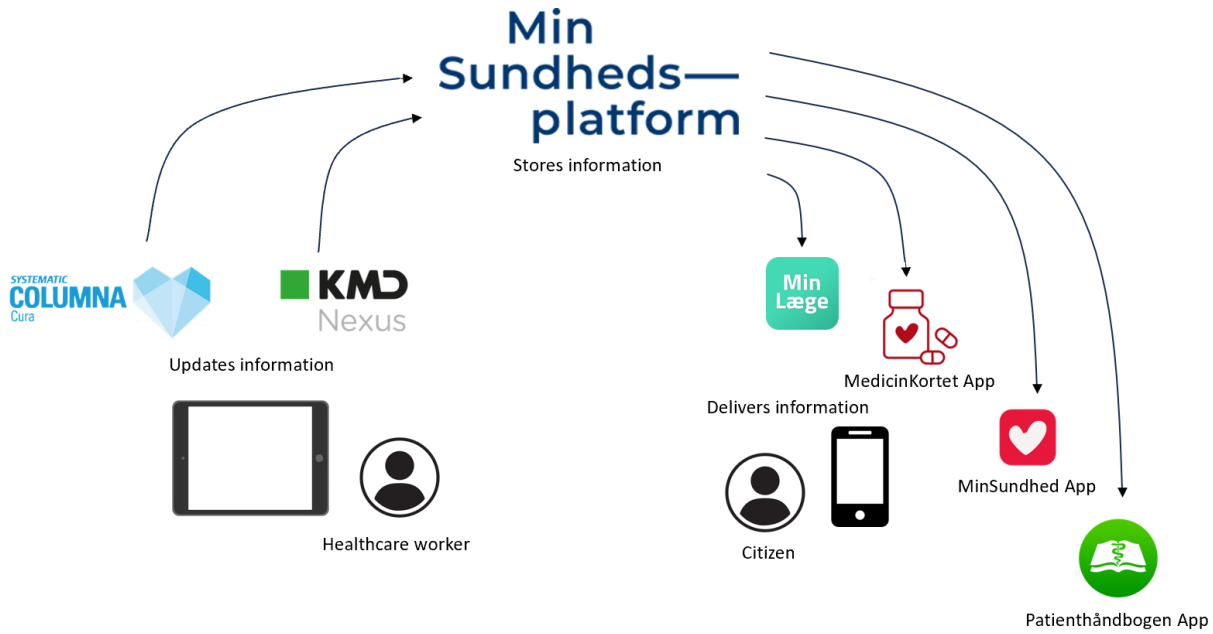


Fig. 1: The flow of health-related information system.

Figure 1 illustrates the flow of health-related information within the healthcare system. The central platform, Min Sundhedsplatform, acts as the core repository where health data is stored. Healthcare workers use platforms like Systematic Columna Cura and KMD Nexus to update or add to existing health records of a patient. These updates are then stored centrally in Min Sundhedsplatform. From Min Sundhedsplatform, the stored health information is delivered to citizens through various applications such as Min Læge, MedicinKortet App, MinSundhed App, and Patienthåndbogen App [11].

For specialized care, particularly for nursing homes, the Columna Cura [12] is widely used. It supports the daily management of nursing tasks by enabling caregivers to plan and document the care of each resident effectively. This professional-oriented tool integrates various aspects of resident care into one platform. Other similar tools such as KMD Nexus are also used in nursing homes B.

This paper aims to create an application that functions as an extension to the existing Columna Cura system, specifically designed to enhance the efficiency and effectiveness of caregivers in nursing homes. While Columna Cura manages broader care aspects, this solution focuses on providing caregivers with an overview of tasks tailored to their daily needs. Additionally, it simplifies the documentation process, allowing caregivers to update task statuses and record residents' details on the go.

2.4 Cloud architecture

Cloud architecture is integral to the project, as DCR, which is the core component of the project, is a cloud service. Cloud architecture includes various components required for cloud computing, including servers, storage, applications, and web services. There are three types of cloud architectures, these are Public, private, and hybrid.

Public cloud architecture delivers services over the internet, shared across multiple organizations. This model is widely adopted due to its scalability and cost-effectiveness. For example, Microsoft Azure provides secure cloud storage while DCR offers processing capabilities. Public cloud services grant access to resources and infrastructure with minimal upfront investment. Private cloud architecture refers to cloud services utilized exclusively by a single organization, offering enhanced security and control over data, as the infrastructure is managed privately. This architecture is well-suited for handling sensitive data due to its secure environment, although it comes with a higher cost. Hybrid cloud architecture combines public and private clouds to create a more versatile environment. In this model, sensitive data might be stored in a private cloud, while less critical data and applications would use the scalability of the public cloud. This approach balances security, cost, and flexibility [13].

By employing cloud architecture, the project integrates DCR for task management. The choice of cloud architecture significantly impacts data handling, processing, and protection, which would directly impact the project's effectiveness and reliability.

3 Methodologies

This project employs a set of well-defined methodologies designed to guide its success and provide a structured framework to follow. In this context, two methodologies were employed: Participatory Design and SWOT Analysis. These methodologies were chosen for their ability to support a user-centered solution.

Participatory Design (PD) aims to develop new technologies with the close involvement of stakeholders and end-users through cycles of requirements gathering, prototype development, implementation, and evaluation. This cycle can be seen in Figure 2. This approach incorporates the experiences, and needs of users into the design process. PD involves collaborative efforts between designers and users throughout the entire development process. PD can be seen as an attempt to understand and involve people throughout the design process to create more appropriate, applicable, and user-friendly products. User involvement within the design process is seen as the key solution to achieving this goal. In this way, all participants within this research can be considered as members of a participatory design group, capable of influencing all aspects of design [14].

This approach often results in a system that is more user-friendly and suited to users' workflows. Additionally, systems designed through PD typically experience higher acceptance rates because users feel a sense of engagement, as they would feel like they are an integral part of the development process. The collaborative nature of PD can lead to innovative solutions that might not emerge in a more traditional, top-down design approach. However, PD is not without its challenges. The collaborative nature of PD can make the design more time-consuming than traditional methods. It often requires more resources. In some cases, it may not be feasible to include all potential users, which can limit the scope of feedback of the design process, which can lead to an incomplete product [15].

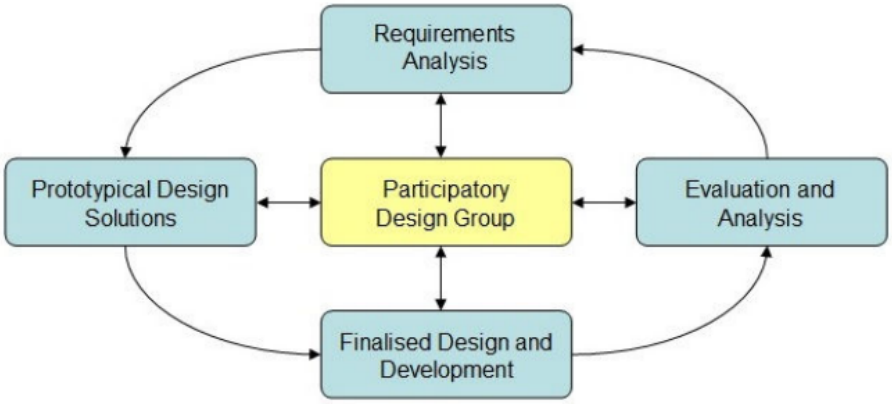


Fig. 2: Participatory Design Process [14].

The SWOT analysis was conducted on the final version of the application to identify and examine internal factors, such as strengths and weaknesses, and external factors, such as opportunities and threats, that can impact the project's success [16].

4 Tools

This section will introduce the various tools used in creating the app. It will explain how a Process-Aware Information System (PAIS) can facilitate the quick creation of easily modifiable task lists for simulating a workday. Additionally, it will describe how the Python framework Kivy enables the rapid and straightforward development of apps. The section will also cover how MySQL can be utilized to interact with and update a database hosted on Microsoft Azure. Furthermore, it will explain the role of the DCR Active Repository, which, through a RESTful API, allows interaction with the simulations via the app. The Design and Implementation sections will detail how these tools work together to create the app.

4.1 Process aware information systems (PAIS)

Process-Aware Information System (PAIS) [17] uses process models to define and administer processes, providing an intuitive and flexible management system that is more adaptable than traditional code-heavy development frameworks. A significant advantage of using PAIS in a mobile health (mHealth) application is its flexibility. The model-based design enables users, including non-technical staff like caregivers and nurses, to modify tasks or adjust process rules directly [17].

Figure 5 illustrates the basic components of a PAIS and their interactions. The Process Model Editor creates executable process models, allowing designers to define and assign tasks. The Organizational Model contains details about roles and organizational structures. The Executable Process Model serves as the blueprint, detailing the sequence of activities and the roles assigned to these tasks.
The Process Engine utilizes the executable process model to manage task execution and handle data through connections to both Process Control Data and Process-relevant Data repositories. The Worklist Manager manages tasks, providing personnel with accurately updated tasks through the End-User Client interfaces. Additionally, Application Services are available to integrate additional data and functionalities in the future.

The use of PAIS in the development of a mHealth application aligns well with the requirements of retirement homes. The system's ability to handle frequent updates to task lists is perfect for facilitating a dynamic adjustment to the care environment for each resident, which makes sure that care delivery can be personal.

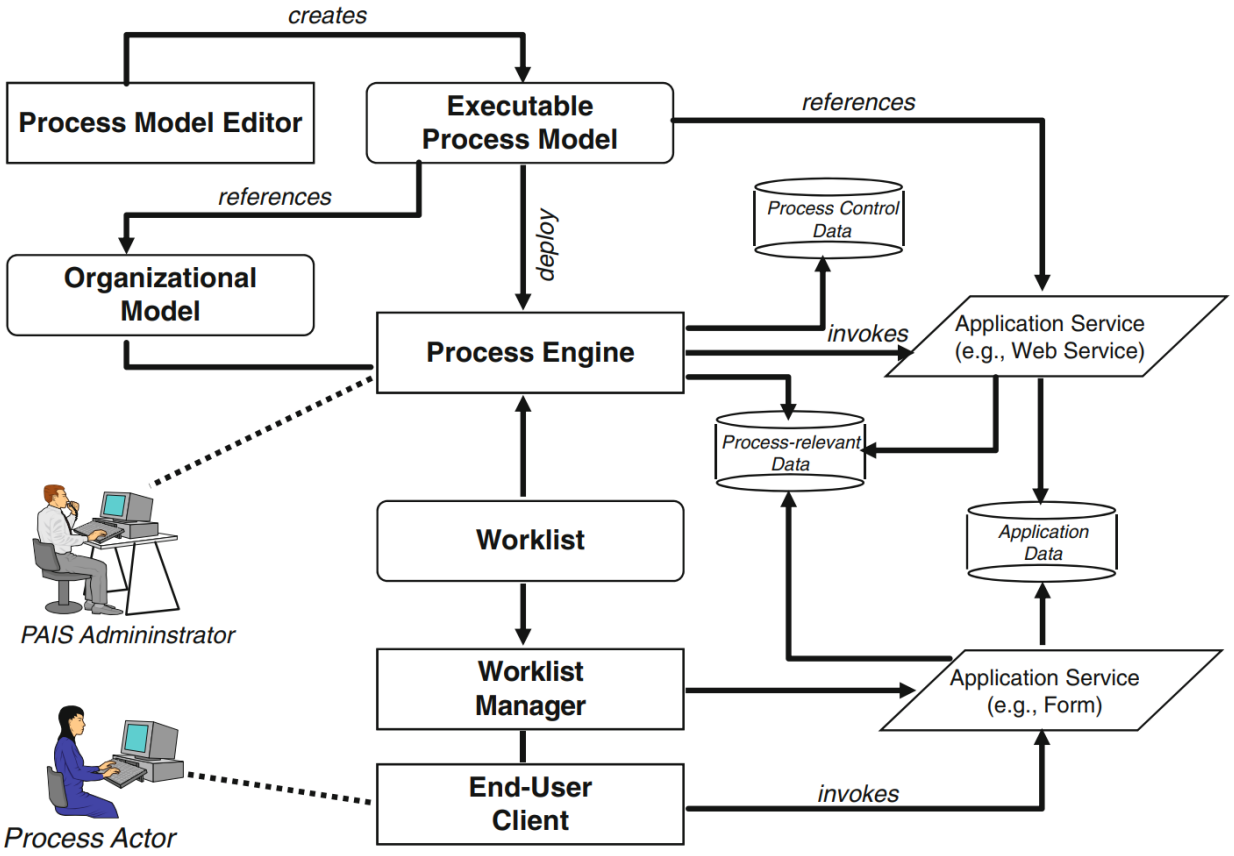


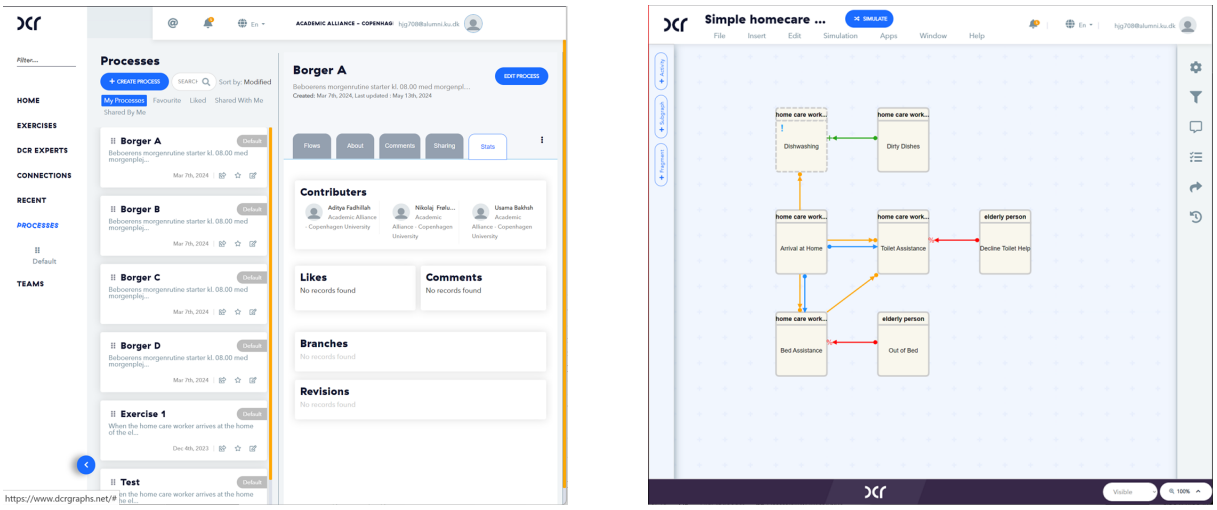
Fig. 3: Basic components of a PAIS [17].

Business Process Model and Notation (BPMN) [18] is recognized as the most widespread PAIS structure worldwide, standardized by the Object Management Group (OMG) [18]. Despite its prevalence, OMG has recognized that BPMN lacks sufficient flexibility for certain applications. As a result, OMG has developed Case Management Model and Notation (CMMN) [18] as an alternative. However, CMMN has not yet achieved widespread adoption and is only used in specific contexts where its structure offers particular benefits. CMMN shares some similarities with the Dynamic Condition Response (DCR) graphs in terms of its structure, which provides the flexibility required for dynamic environments.

4.1.1 DCR

DCR Designer, developed by DCR Solutions, has been chosen as the Process Model Editor for this project. The principle behind DCR Designer is to allow for a constraint-based approach to modeling work that uses graphs. A graph consists of an organized pattern of activities and their assigned rules and actors. A graph can be pictured as a visualization of activities and their relationships [19]. A DCR graph corresponds to a process in PAIS. Additionally, DCR Designer includes a built-in simulation feature to test for errors or application readiness [20]. Figure 4b shows an example of a simple process designed in DCR Designer.

The DCR portal, accessible at dcrgraphs.net, serves as the PAIS Administrator and allows processes to be simulated directly in a browser. As seen in the figure 4a the DCR portal incorporates social media-like features to enhance modeling collaboration, such as users can invite colleagues, send friend requests, comment on models, and browse existing models [19].



(a) The front page of the DCR portal.

(b) An example of a DCR Graph in DCR Designer.

Fig. 4: An example of a DCR Graph in DCR Designer.

The DCR Active Repository, accessible at repository.dcrgraphs.net, serves as a built-in database that is essential in the functionality of DCR Designer. This database allows for the storage and retrieval of process-related data, including comments made by users and records of completed and pending activities. The repository's capability allows all relevant data to be centrally stored, easily accessible, and up-to-date, which is essential for maintaining an accurate overview of activity progress. The DCR Engine, serving as the Process Engine in PAIS, interprets DCR process models and converts them into process data. It manages the execution of these models, making sure that the activities and rules specified in the DCR process models are accurately followed [4].

Activity

A DCR Graph is composed of activities, which represent a specific action or event that can occur within the process. Activities in a DCR Graph can exist in three states, included, pending, and executed. Included means that the Activity is active and will appear in the simulation, and excluded means that it will not. By default, if an activity is not included, it is excluded. Pending means the activity is required to be completed. Pending indicates activities that must be performed during a certain section of the model to move on to the next. Executed means the activity has been completed [20].

These activities can be controlled and modified through rules. Rules are visual connections drawn between activities, shown as arrows. When one activity is complete, these rules can automatically update the states of one or more other activities.

Rule

In the DCR Graph, rules define the relationships between activities. The four main types of connections are Response, Include, Exclude, and Condition.

- The Response rule creates a dependency between activity A and activity B such that B must occur at least once after A has occurred. However, B can still occur even if A never happens. This rule states that if A happens, B must follow.
- The Include relation enables activity B to occur if it was not previously part of the workflow, triggered by the occurrence of activity A.

- The Exclude relation prevents activity B from occurring if activity A has taken place. This exclusion can be overridden if another activity with an Include connection to B occurs.
- The Condition rule specifies that activity B can only occur if activity A has already occurred.

[21]

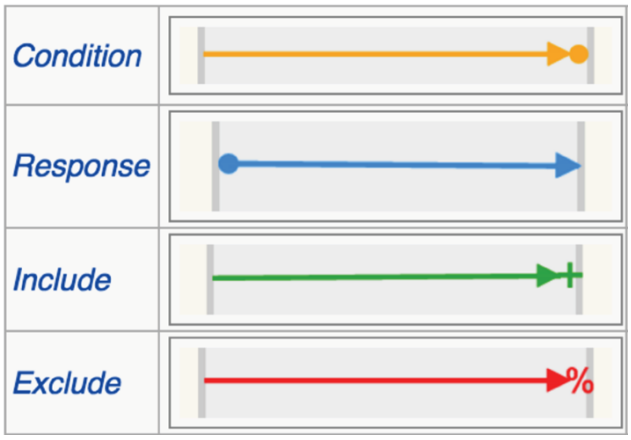


Fig. 5: Four types of rules within DCR Graph.

Role

DCR Designer allows for the creation of roles that are an essential part of managing activities across multiple users within a process. By assigning roles, processes can be streamlined as each user is only permitted to interact with activities that they are responsible for, which in turn increases security. For instance, users can not mistakenly complete activities they are not authorized to perform, and sensitive information is restricted to only those who have access to it [20].

Simulator

The process simulator in DCR Designer allows users to test and run processes, providing a virtual environment to experiment with different scenarios. Users start the simulation from the DCR Designer by clicking "Simulate," then selecting co-workers, and configuring settings. The simulator window displays various elements such as activities grouped by role, current simulation time, a simulation log, process phases, AI recommendations, and the flow of the simulation. Users execute activities by clicking the "Execute" button, which updates the activity list. Once all activities are complete, the log turns green, indicating process completion. Users can then choose to discard or save the simulation [22].
DCR Solutions offers a range of software technologies, these technologies are integrated by Technology Partners into their own system. These technologies enable partners to make use of features such as structured process, end-user process co-development, and post-hoc process intelligence analysis [23]. DCR is designed to integrate seamlessly into existing infrastructure.

The decision to use DCR is supported by its widespread adoption among various companies and industries, which shows its reliability and versatility. This broad usage demonstrates the tool's effectiveness in managing complex workflows, making it a fitting choice for mHealth applications where flexibility is important. During the development of the project, DCR Solutions made a visual update to the rule arrows. As this was a week before the due date, this project will use the old look which can still be enabled on DCR Designer.

4.2 Kivy Python Framework

The user interface of the mobile application was created using the Kivy framework, an open-source Python library designed for developing interactive applications on Windows, macOS, Linux, Android, and iOS. Kivy provides a variety of predefined classes that make application development straightforward and efficient. Essential classes like Button, TextInput, and BoxLayout help build interactive user interfaces. For instance, in the task list application, each task needs to be clickable and disappear once completed, while logging in requires entering text via a keyboard. Kivy's built-in interactive widgets make it particularly suitable for these requirements [24].

A major advantage of Kivy is its cross-platform nature. This feature is vital for projects targeting a wide audience without the need to develop separate applications for different operating systems. Although the initial development focused on Android devices, using a cross-platform framework like Kivy allows the application to be easily adapted for iOS and other platforms.

When run on a computer, the application opens a window that emulates its appearance on a mobile device. To deploy the application on an actual phone or tablet, Buildozer is used. Buildozer automates the packaging and installation process, handling all necessary prerequisites.

Kivy is free to use and comes with extensive documentation, which simplifies the development process. The framework's variety of built-in widgets facilitates the quick creation of interactive applications. Moreover, using Kivy is further justified as this project is heavily inspired by the projects in the University of Copenhagen course "Development of Cloud-Based Health Apps", which also uses Kivy as the application development framework.

4.3 MySQL

The database designed and developed for this project was created with MySQL. MySql is an open-source SQL (Structured query language) database management system that allows for the reading and updating of databases. In this project, the database contains information about the task lists and the caregivers.

MySQL was chosen for this assignment due to its well-documented and user-friendly nature. As a relational database management system, MySQL uses the relational model to store data in different tables, with clear relationships between them. This structure makes the data more organized and easier to read. It allows access to one or more tables at a time, unlike non-relational databases that provide all information at once. This selective access enhances the efficiency of data searches by retrieving only the desired information. MySQL offers several advantages as a database management system. It enables the definition, modification, and creation of databases, as well as the insertion, updating, deletion, and retrieval of data. Additionally, MySQL provides an integrity system that maintains data consistency. This system enforces data integrity rules using constraints such as primary keys, unique indexes, and foreign keys. Moreover, MySQL includes a concurrency control system that supports shared data access, managing simultaneous data operations by multiple users [25].

In this project, the data managed by MySQL is stored in the cloud using Microsoft Azure. This cloud-based storage allows caregivers and administrators to interact with task lists and caregiver information from anywhere with an internet connection.

4.4 RESTful API

RESTful APIs use the Representational State Transfer (REST) model, a design approach for building large-scale distributed systems on the web. These web services are identified by unique addresses, which help in locating resources and services on the web. RESTful services interact through a consistent set of operations, including GET, PUT, DELETE, and POST, using the HTTP protocol. Each request and response contains all the necessary information to process the message [26].

The communication and interaction between the mobile application and the DCR process model is facilitated through RESTful API, which accesses the capabilities of the DCR Active Repository. The DCR Active Repository uses the RESTful model to access the DCR Process Models, enabling various GET and POST calls to obtain information about the state of the simulation or to post updates that modify it. This repository supports essential functions such as creating, updating, retrieving, and deleting simulations and their activities. By leveraging HTTP authentication and requiring a username and password to access the DCR Active Repository, the RESTful API facilitates efficient and secure data transfer and interaction between the mobile application and the DCR Process Models [26, 4].

In this project, the RESTful API was used to enable the mobile application to send requests to the DCR Active Repository and receive responses in a structured format, such as XML. This setup allows the mobile application to manage tasks such as querying the status of tasks, updating task progress, and handling user interactions with the task list in real-time.

An external engine such as the DCR Active Repository, was chosen over developing a custom engine because it offers a mature and scalable solution that integrates seamlessly with the DCR Designer. This choice offers comprehensive functionalities required for managing complex workflows and simulations, significantly reducing development time and effort.

5 Case Study

This case study explores the need for an improved mobile application designed to enhance the operational capabilities of healthcare professionals in a Danish nursing home facility. Reflecting on existing technologies in healthcare. The focus of this project is the implementation of a task-list application tailored specifically for caregivers. The application is designed to streamline their daily tasks by providing tools that assist them at point-of-care.

5.1 Hotherhaven Plejehjem

Hotherhaven Plejehjem is a Danish nursing home comprised of two main sections, a somatic department for general residents and a specialized unit for residents with dementia. The facility is designed to take care of a diverse group of elderly residents, providing them with tailored care for their specific needs.

The somatic department of Hotherhaven consists of 31 housing units spread across two divisions. These divisions are structured to provide comprehensive care and support to the elderly without dementia. The dementia department is slightly larger as it consists of 35 housing units distributed among three divisions. Each section features common living areas, which include corridors, communal living rooms with kitchens, dining areas, and relaxation zones designed to promote social interaction. The use of life history tools helps in creating personalized care plans that resonate with the personal experiences and backgrounds of the residents, enhancing their comfort and engagement. The organization's approach to caring for residents with dementia emphasizes the need for empathy and creativity from the staff. These traits are highly valued by both residents and their families [27].

For this project, the focus will be on one specific division within the dementia department of Hotherhaven Plejehjem. By concentrating on this specific division, the project can more effectively enhance care delivery and operational efficiency for a targeted group.

5.2 Columna Cura

The nursing home currently utilizes the Columna Cura system, a care and case management solution that is employed across a third of all municipalities in Denmark [12]. This system facilitates the daily management of general nursing tasks and provides caregivers with robust tools for managing resident information and medication schedules. It serves as a dependable platform, enabling caregivers to easily access vital data and efficiently update resident records.

Despite its robust features, Columna Cura faces challenges in adapting to the specific needs of each resident. Its standardized approach is beneficial for routine care but may lack the necessary flexibility required to handle unexpected situations or the personalized care demands that often arise. Furthermore, the interface and user experience of Columna Cura, though comprehensive, may not always be intuitive during point-of-care situations. Caregivers often find themselves navigating through multiple screens to add new information about a resident, which can disrupt the flow of their work. This typically forces caregivers to document their routines only after completing all their tasks, potentially leading to missing information or inaccuracies due to forgotten details.

While Columna Cura provides structured and reliable support for routine tasks, there is potential for enhancements to improve its adaptability and ease of use in real-time situations. Addressing these limitations could significantly improve the effectiveness of care delivery.

5.3 Target users

The targeted group for the tasklist application within Hotherhaven Plejehjem is the caregivers working in the dementia department. The caregivers in this division are tasked with responsibilities that range from basic assistance to medical management specific to each resident's needs [27].

The staff is categorized into two main groups: Social- og Sundhedsassistent (healthcare assistants) and Social- og Sundhedshjælper (healthcare helpers) C. These roles are designed to complement each other, where healthcare helpers assist the healthcare assistants in delivering care. Healthcare assistants are qualified to manage more complex care, including the adjustment and oversight of medication prescriptions for residents. Healthcare helpers support the assistants by performing daily care activities and administering medication under the supervision and direction of the healthcare assistants A. The task list application is designed to enhance the efficiency and effectiveness of both groups. By providing a user-friendly interface with real-time updates of tasks, the application helps all caregivers manage their workflow.

6 Design

In this section the design of the project is detailed, showcasing how the initial problem statement was transformed into a usable mobile application. The primary objective was to develop an application for caregivers in a nursing home that utilizes DCR Graphs to manage their tasks. Using the PAIS model, the design process followed a structured approach. This section will discuss the design considerations, methodologies, and tools used to create an efficient mHealth solution.

6.1 Structure of the application

The problem statement for this project was to develop a mobile application for caregivers in a nursing home, utilizing DCR graphs to manage their tasks. To achieve this, the project employed the PAIS model as its backbone. Figure 6 illustrates how the different elements of the project align with various components of PAIS. The Process Model Editor, DCR Designer, is used to create and design executable process models. The Organizational Model incorporates roles, determining who can perform each activity within these process models. Once designed, the process models are deployed using the DCR Active Repository. In the DCR

Active Repository, the DCR Engine handles data interactions with the Process Control Data and Process-relevant Data repositories.

For PAIS Administrators, the project utilizes the DCR Portal to access and manage different process models and it also supports collaborative process design. The Worklist, which connects the DCR Active Repository with the application, is managed through a RESTful API. This API facilitates dynamic updates and interactions between the process engine and the worklist manager. The mHealth application structure includes the End-User Client, which interfaces with the Worklist Manager. This setup enables users to execute tasks through a simple, user-friendly interface. Application Data is stored using MySQL, providing a robust and scalable data storage solution. Potential future integration with CURA could enhance the system's capabilities.

This design takes advantage of the strengths of PAIS to create a flexible and scalable solution for managing healthcare workflows.

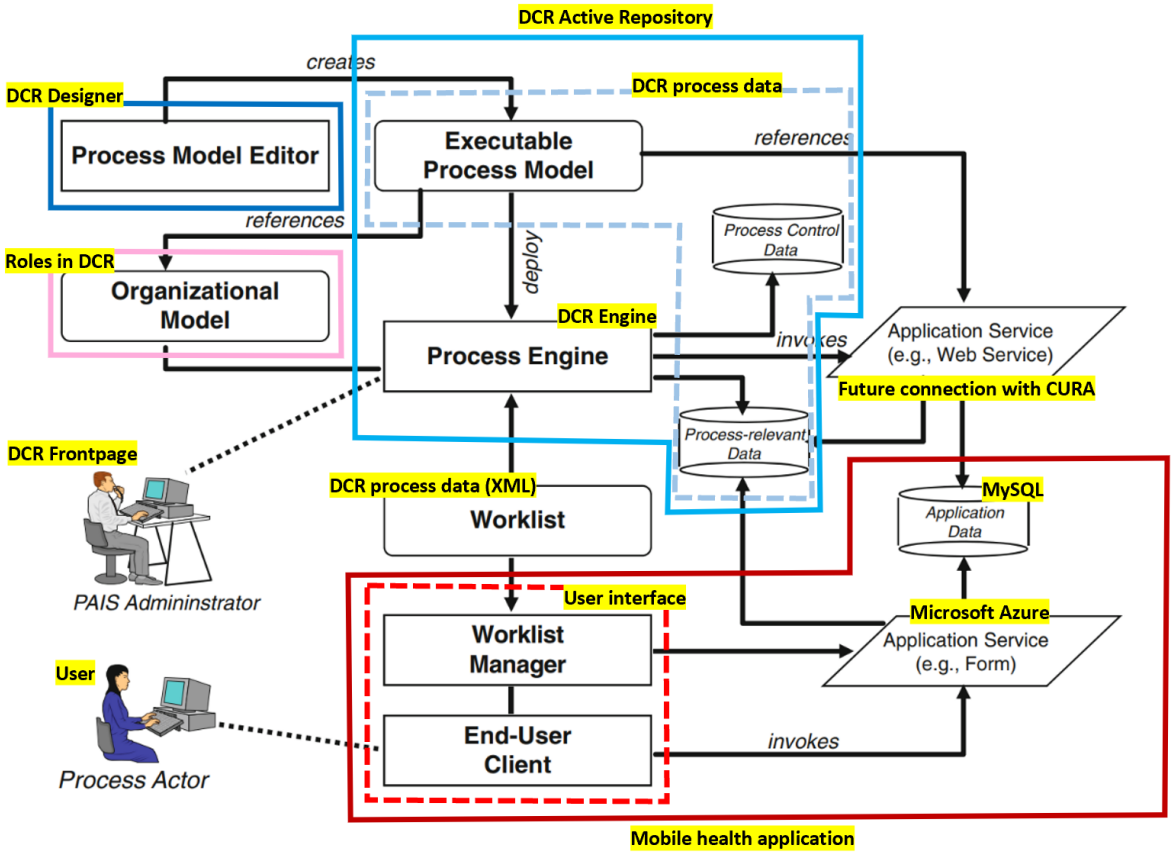


Fig. 6: Components of the project align with the PAIS framework.

The mobile health application architecture integrates technologies such as Python, the Kivy framework, MySQL, DCR Active Repository, and DCR Designer. Python handles backend development, while Kivy creates a versatile and user-friendly interface. MySQL manages data storage, including IDs for graphs, simulated processes, and hashed user credentials. Communication with the DCR Active Repository is facilitated via a RESTful API, enabling the execution and updating of DCR processes created and edited in the DCR Designer. This setup supports simultaneous multi-user access, with the application regularly sending API calls to the DCR Active Repository to keep updated, eliminating the need for manual task simulation within the app.

Figure 7 shows a generalized version of the PAIS model from figure 6. This figure highlights the important connections between the different tools used in the application development, helping to visualize the relationships between the components.

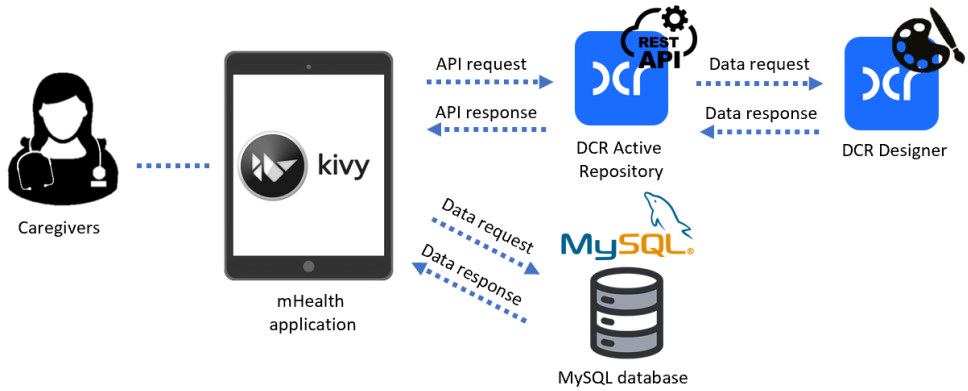


Fig. 7: Illustration of the architecture of the mobile application.

An example of these connections is when a user attempts to log in, as seen in figure 8. The application checks the MySQL database to verify the user's identity. The application also makes an API call to the DCR Active Repository to verify if the user has a DCR account. Upon successful login, the user can select a resident to manage. This action triggers the application to make an API call to start a new process in the DCR Active Repository. The DCR Active Repository, in turn, utilizes the DCR Graphs created by the DCR Designer. These models define various activities, roles, and rules that structure the workflow for the selected process. Once the process is initiated, the DCR Active Repository generates the appropriate tasks. At the same time, the graph ID and simulation ID of the process are saved in MySQL. The application then makes another API call to get these tasks displayed in the application as clickable buttons, provided by the Kivy framework. When these buttons are clicked, an API response is sent to the DCR Active Repository, updating the simulation.

Fig. 8: Step-by-step connection of log in and resident selection.

Another example of these connections is a user attempting to create a note, these connections are shown in figure 9. The user clicks the "Ny note" button to access the note screen, where they can write all the necessary documentation regarding the resident. The user can choose to make it a generalized note without assigning it to a specific activity, or they can choose from all tasks that are assigned to them. The application retrieves these tasks through an API call to the DCR Active Repository, filtered only to include tasks from

the current simulation, and for the tasks assigned to the user role. When the note is uploaded, it is stored in the DCR Active Repository, and the notes for this specific simulation are shown in the "Noter" screen. Notes created during different simulations are also stored in the DCR Active Repository and can be accessed anytime as long as the process is not deleted.

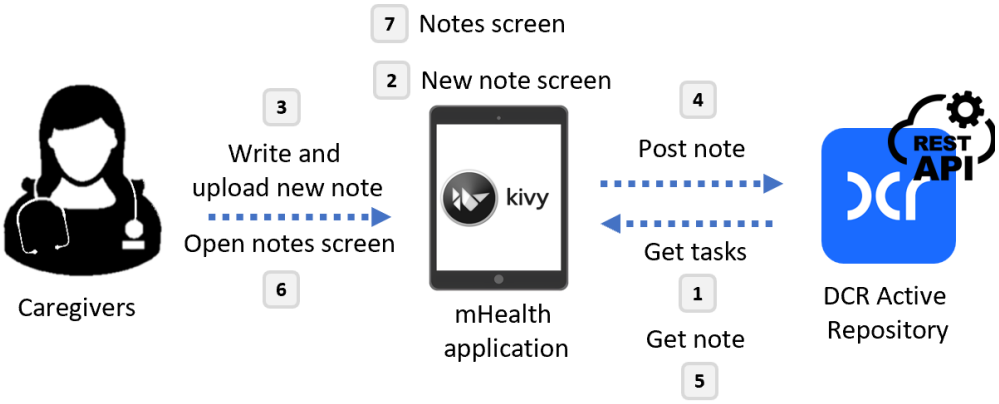


Fig. 9: Step-by-step connection of note function.

To show the interaction between users and the task list application, a UML use case diagram was utilized. In this UML use case diagram 10, caregivers and administrators are depicted as the primary users. Both primary users can log in and out. Caregivers have access to functionalities such as choosing residents, executing tasks, reading, and writing notes. Administrators perform tasks such as adding new users, terminating simulations, and removing users.

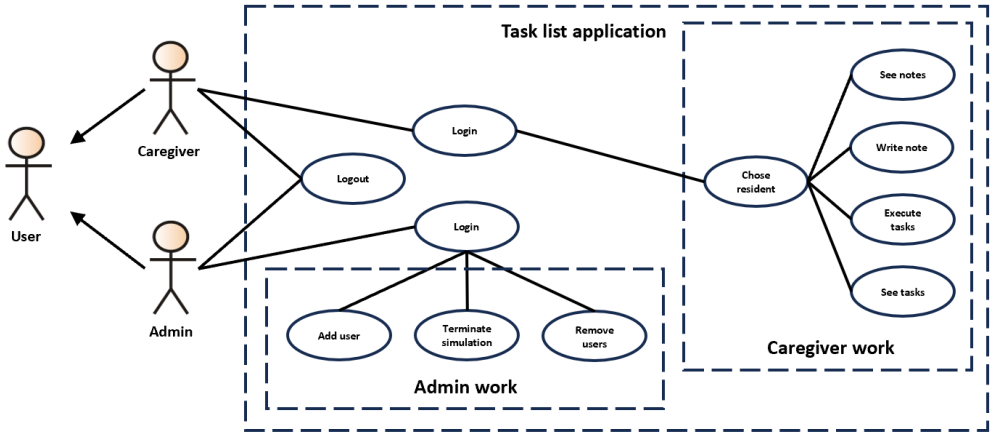


Fig. 10: UML use case diagram for the task list application.

The project effectively utilizes the PAIS model to develop a reliable application for caregivers in nursing homes. The detailed architecture facilitates secure data handling and real-time updates, with figure 8 and 9 illustrating the step-by-step interactions for login, resident selection, and note creation. The UML diagram 10 illustrates how users interact with the system.

6.2 Interviews and feedback through Participatory Design

The principles of Participatory Design guided the initial steps of the project. This approach is intended to involve end-users actively in the design and development process to create a final product that accurately reflects their needs and work practices. The early stages of implementation focused on setting clear objectives, identifying stakeholders, planning engagement strategies, and collecting data to inform the design.

To begin, the project's goals and scope were clearly defined. The application needed to:

- Integrate with the DCR Graph for its task list functionality.
- Support user login for multiple users.
- Allow caregivers to select from a list of residents.
- Comply with GDPR requirements by enabling user deletion.

These objectives provided a clear framework for the project's development.

Next, stakeholders were identified to give input into the design process. The primary stakeholders were caregivers, as the project aimed to evaluate the feasibility of using the DCR Graph in nursing homes. Although cooperation with the nursing home's administration or leadership was not actively pursued, it was acknowledged that their involvement would be necessary for a full implementation. For this proof-of-concept stage, the focus remained on the caregivers.

An engagement plan was then devised to involve caregivers in the design process. The ideal plan included weekly meetings with caregivers to gain insights into their work schedules and gather feedback on the application's development. Regular interaction with caregivers was intended to make the application user-friendly and aligned with their needs.

Insights were gathered through interviews and surveys. The plan was to conduct weekly interviews with caregivers. However, maintaining this schedule proved challenging due to holidays, shift changes, and other scheduling conflicts. Despite these challenges, regular contact was maintained with one caregiver, providing valuable insights into the caregivers' workflows.

For each of these interviews, the questions and interactions remained the same. The interviewer showed the caregiver the current development of the application and asked them to evaluate the current iteration of the application as well as give examples of features that would benefit the app. As well as any limitations they might experience with the Cura System. It was also during these interviews that the different residents' schedules were noted down to be made into the DCR Graphs.

Using the data collected from interviews, an affinity diagram was created to organize and prioritize the feedback. This diagram helped identify key features that would be realistic and impactful enough for the application.

From the affinity diagram (Figure 11), several features were selected to be part of the application:

- Note management: Caregivers needed the ability to write and view notes easily.
- Shift-specific Task Management: The current application used by caregivers, Cura, only displays the overall routines for a resident without distinguishing between different work shifts. This lack of shift-specific task management clashes with DCR's role-specific nature.
- More detailed routines: The application will show the individual tasks that is done in a routine, contrary to Cura which only lists the routine names.

- Admin system: The new application introduced an admin system that allows tasks to be assigned to specific shifts. Admins can manage caregiver shifts, delete caregivers from the database, and start a new task list for a new day.
- Improved organization and error reduction: Assigning shift-specific tasks means that caregivers only see the tasks relevant to their current shift.

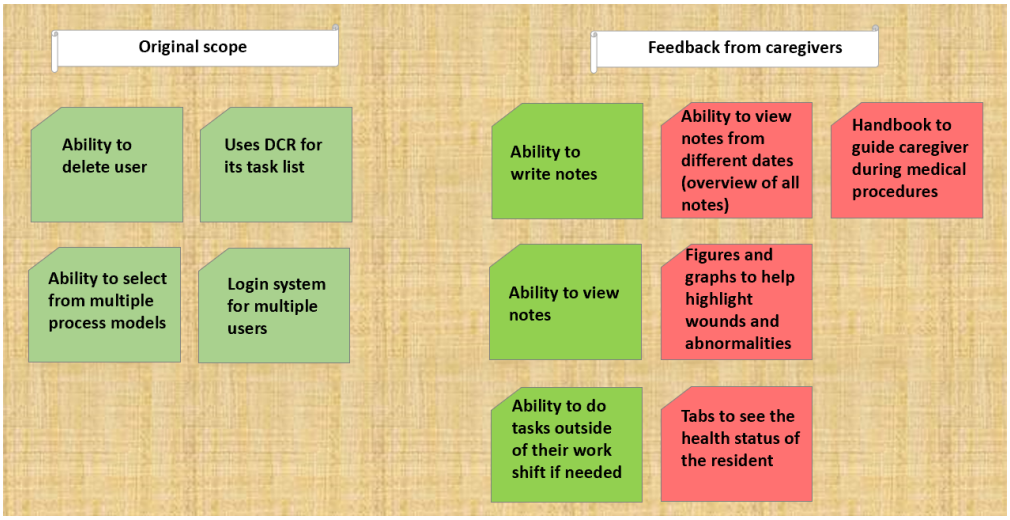


Fig. 11: Affinity diagram of project objectives and feedback from caregivers.

Some desired features could not be implemented due to time constraints and technical considerations:

- Graphs and interactive drawings: The feature to mark areas on a patient's body, such as wound locations or other abnormalities, was acknowledged as important. This feature was not prioritized to focus on the task list functionality.
- Viewing notes from different dates: While the ability to write notes was implemented, the function to view notes from different dates was deferred. Secure storage of all notes in the DCR Active Repository allows data to be accessed in the future.
- Tabs to view health status: The functionality to have tabs for viewing the health status of the resident was not implemented.
- Handbook for caregivers: A handbook to guide caregivers during medical procedures was not included.

These features, while valuable, were beyond the immediate scope and resources available for the project.

A final interview was conducted following Participatory Design principles, where users tested the prototype and provided feedback. This included assessing the DCR Graph to evaluate the user's ability to add new activities. Based on this final interview, the simplicity and ease of use of the application are significant strengths, as the application allows for minimal training. However, the visual design is too basic, and it needs better graphics and color schemes. Additionally, the app's speed, particularly during login, requires improvement to enhance the overall user experience. The note-taking function is useful, but there is a need for better categorization and a search function to make finding specific notes easier. The inclusion of shift-specific task management is highly appreciated, as it reduces confusion and aligns tasks with appropriate work periods. The admin system for managing caregiver shifts and tasks is also seen as a valuable addition D.

Regarding the DCR Graph, users found it provided a clear and organized schedule for tasks, which they appreciated. However, they struggled to understand the significance of the arrows and found adding new activities initially challenging. With guidance, they could comprehend the steps, but they mentioned that

remembering all the steps could be difficult without regular training. Incorporating instructions or a guide within the application could help users remember how to perform various tasks D.

Applying Participatory Design principles effectively guided the development of the project. The project achieved its primary objectives, including integration with the DCR Graph and supporting user logins while identifying and addressing feedback from stakeholders.

7 Implementation

This section will describe the implementation of the tools introduced in the Tools and Design section of the app. It will explain the creation of DCR Graphs and the functions required to access simulations using the DCR Active Repository. Additionally, it will cover how the application supports simultaneous use by multiple users and the process of creating and storing notes. The section will also detail the maintenance of the database using MySQL, which is used to store login data and simulation information.

7.1 DCR Graph

DCR Graph was instrumental in developing a task list manager for caregivers. Figure 12 showcases the comprehensive process model for "Borger A," covering the entire workflow from "Morgenpleje" to "Natpleje." This model illustrates the tasks assigned to different shifts and the logical rules governing these activities. DCR Graphs activities describe the tasks in the application.

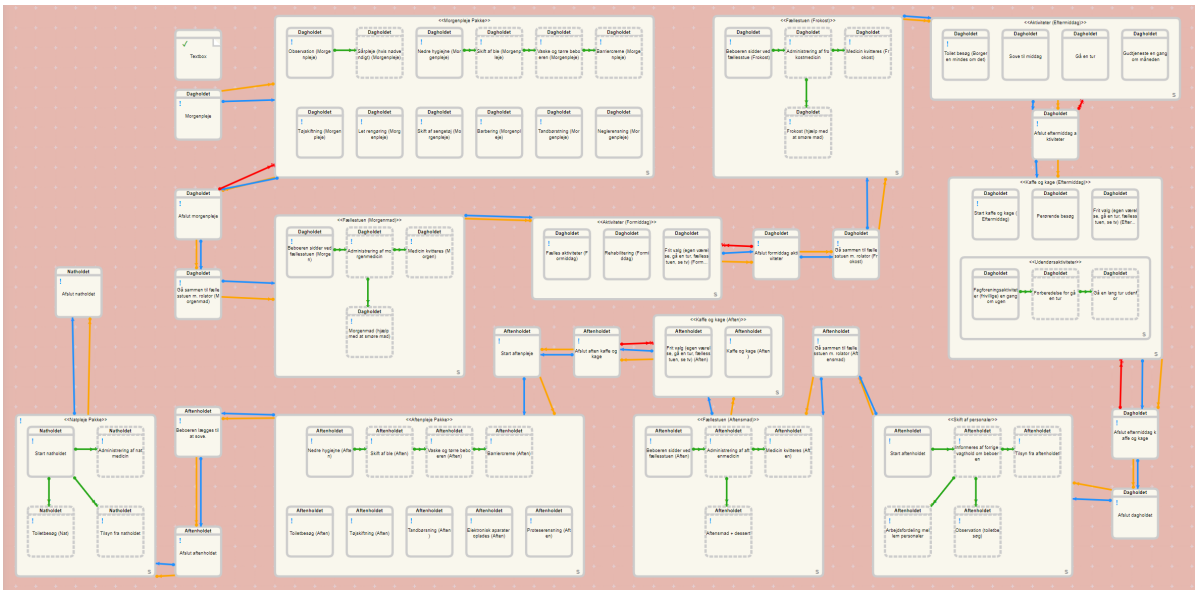


Fig. 12: DCR Graph for Borger A.

In the DCR Graph, some activities are grouped within subprocesses. This design states that all pending activities within a subprocess must be completed before progressing to the next set of activities. This structure helps maintain order and provides a clearer overview of the entire workflow. This subprocess condition applies only to pending activities, allowing non-pending activities to be skipped.

Dotted lines in the model indicate dormant activities. These activities remain inactive until triggered by a connected activity via an Include rule. This mechanism prevents unnecessary activities from appearing in the task list manager.

The Response rule maintains a logical sequence in the workflow. The Condition rule enforces that activities appear on the task list manager only after the completion of the previous activity or group of activities. The Exclude rule removes non-pending activities that are skipped in a subprocess, keeping the task list manager updated by displaying only relevant activities.

It was not possible to implement a process that could reset from the beginning, replicating a daily cycle. The idea is that at the beginning of a new day, an admin would need to end the previous simulations and start a new simulation, representing a new day.

In each of the process models, there is a single activity that is converted to a textbox datatype. This feature allows for the storage of notes, providing caregivers the ability to record information within the task list manager. This textbox datatype is not assigned to a specific role, meaning that all roles can create notes. An explanation of how the notes are made is given in the API subsection 7.2.

Figure 12 is just one example of four DCR Graphs that were implemented, each designed for the specific activities caregivers must perform for a particular resident, these can be seen in Appendix F. The structured and straightforward workflow of the DCR Graphs allows caregivers to modify and add new activities to the existing workflow. The logical sequence and clear rules aim to make it easy for caregivers to follow and adapt the system to their needs.

7.2 API

The API is used to interact with the DCR graphs. Once a simulation of the graph has been started, the simulation data is saved on the website repository.dcrgraphs.net. It contains all the data about all users and all processes made on dcrgraphs.net. The API calls are then used to get or update the information. The data is accessed through different subdirectories, an example is the link: "https://repository.dcrgraphs.net/api/graphs/1706874/", which will give all information about the process with the graph id "1706874". For this project, only get and post calls are used.

The Get call is used for several purposes within the application. It retrieves all enabled tasks at the current point in the simulation. It also obtains all tasks available for a user concerning a resident, which is used to add tasks to notes. Additionally, the Get function checks that the inputted username and password correspond to a valid user, enabling the login process. It retrieves the name of the graph, allowing the user to choose between residents, and shows all the residents the user has access. Finally, it retrieves all the notes written about a resident during the simulation.

The Post call is used for updating the process simulation after a task has been marked as done. It is used for creating new simulations and for uploading notes.

The API calls are done by using the `Session()` class from the requests python library, which has built-in get and post functions. The list of all possible DCR Active Repository API calls can be seen on repository.dcrgraphs.net/index.html. The calls to the DCR Active Repository require logging in to a DCR Solutions account. This is done by defining the `requests.Session().auth` variable as a tuple containing the username and the password.

Listing 1 shows how the API is used. `getEnabledEvents()` is the most important function as it is used to get the list of all tasks in the graph that are currently able to be done. It does this by first setting the `requests.Session().auth` to the authentication given to the function. Then it makes the get call, "https://repository.dcrgraphs.net/api/graphs/" + str(graph_id) + "/sims/" + sim_id + "/events?filter=only-enabled", which inputs the graphs id and the id of the simulation and returns only the enabled events, which are the tasks. The function then saves the response, which is then formatted and made into a dictionary as that is what the `createButtonsOfEnabledEvents` uses to create the buttons.

```python
def getEnabledEvents(graph_id: str, sim_id: str, auth: (str, str)):
    # Connects to the API
    req = requests.Session()
    req.auth = auth
    response = req.get("https://repository.dcrgraphs.net/api/graphs/" +
                       str(graph_id) + "/sims/" + sim_id + "/events?filter=only-enabled")

    # Reformats the xml to json
    events_xml = next_activities_response.text
    events_xml_no_quotes = events_xml[1:len(events_xml)-1]
    events_xml_clean = events_xml_no_quotes.replace('\\\"', "\"")
    events_json = xmltodict.parse(events_xml_clean)
    return events_json
```

Listing 1: Python function to get activities from the DCR Active Repository.

7.2.1 Simultaneous use of the application

The API enables the simultaneous use of the application. When a user marks a task as completed, the API updates the simulation. And when a nursing home worker accesses a resident the API is used to retrieve the simulation of the process. This works when only one nursing home worker accesses a resident at a time. But in cases where multiple workers complete tasks simultaneously, the apps of all caregivers need to be updated in real-time, eliminating the need for manual updates. This is achieved using the `kivy.Clock` class to schedule events, as demonstrated in listing 2.

```python
    reload_event = lambda dt: createButtonsOfEnabledEvents(graph_id, sim_id, auth, button_layout)
    Clock.schedule_once(reload_event, 5)
    global scheduledReloads
    scheduledReloads.append(reload_event)
```

Listing 2: The code snippet that schedules the task list screen to be reloaded.

The code structure for reloading the task list and the notes is the same. `Clock.schedule_once` is used to schedule the function to be called only once in 5 seconds. The scheduled reload is then added to a global list, which is used when stopping the scheduled event. All scheduled reloads are stopped and removed from the list of reloads by the `stopReloads` function shown in Listing 3, which is called whenever the screen changes. This is done as otherwise, the scheduled event would still happen which would make it impossible to leave the notes screen or the task list screen. This is done to ensure that scheduled events will not occur and will not cause the task list screen to be permanently shown.

```python
def stopReloads():
    global scheduledReloads
    for reload in scheduledReloads:
        Clock.unschedule(reload)
    scheduledReloads = []
```

Listing 3: Python function to stop reload.

The chosen solution involves updating the application every 5 seconds for the task list screen. This approach prevents the application from updating too often which could cause problems with marking tasks as completed. Given that only around 50 events are completed during a typical workday, the application does not require constant updates. Notes update every 60 seconds, as reloading the notes screen causes it to reset and display the first notes again, resulting in a poor user experience, as it requires the user to scroll down to read newer notes.

7.2.2 Writing and reading notes

As explained in the previous DCR section 7.1, notes are saved in the DCR graph simulation by writing the notes in the `textbox` activity. This is done by the `uploadNote()` function, shown in listing 4.

```
def uploadNote(self, instance):
    if self.notes_box.text != "":
        url = (f"https://repository.dcrgraphs.net/api/graphs/{self.graph_id}
                    /sims/{self.simulation_id}/events/textbox")
        auth=(self.username.text, self.password.text)
        req = requests.Session()
        req.auth = auth
        json = {"dataXML": f"Date/time: {datetime.now().strftime('%Y-%m-%d %H:%M')}
                \nNoteret af: {self.username.text} \nAktivitet:
                {self.selected_activity_notes.text} \n{self.notes_box.text}"}

        req.post(url, json = json)
```

Listing 4: Python function to upload notes to the DCR Active Repository.

This function makes the same API call as `executeEvent`, but since the `textbox` has the datatype `textbox`, it can be executed and sent in JSON text format. This JSON text contains the note that was written by the caregivers. But some extra data is added, this is the date and time the note was written, the username of the caregiver who wrote the note, and which task the note was written for. Giving the nursing home valuable info regarding their caregivers and residents. The `textbox` is an activity and has therefore been set as executed from the beginning of the graph as it should not be shown as a task.

Reading the notes is done by calling the `showNotesScreen()` function, which calls `getNotes()` to get the list of notes and use the `showNotesLayout()` function to display the notes.

`getNotes()` uses an API call to retrieve all the information about the current simulation (see listing 5). `getNote()` then splits the data into parts, with each part starting with the string `"$<$event "`, ensuring that each part contains exactly one note. It uses the regular expression `data="([^"]+)"`, which matches everything between `data="` and `"`, extracting each note. The DCR Active Repository cannot handle the symbols `"`, `&`, and newline (`\n`) as they are used in other contexts. That is why these symbols are converted to `"`, `&`, and `
` respectively when uploaded. `getNote()` replaces these placeholder symbols using `.replace`.

```python
def getNotes(self, instance):
    url = (f"https://repository.dcrgraphs.net/api/graphs/{self.graph_id}/sims/{self.simulation_id}/")
    auth=(self.username.text, self.password.text)
    req = requests.Session()
    req.auth = auth
    response = req.get(url)

    splitEvents = response.text.split("<event ")
    allNotes = []
    for event in splitEvents:
            match = re.search(r'data="([^"]+)"', event)  # Match the pattern 'data="<looked for text>"'
        if match:
            allNotes.append(match.group(1).encode('latin1').decode('utf-8'))
    allNotes = [note.replace("&#xA;", "\n").replace("&amp;", "&").replace
                ("&quot;", '"') for note in allNotes]
    return allNotes
```

Listing 5: Python function to get notes from the DCR Active Repository.

7.3 Utilizing MySQL in the Implementation

The implementation utilizes MySQL to manage and store essential data for the mobile application. The data are saved on a database hosted in the cloud facilitated by Microsoft Azure. The database consists of multiple tables, each with rows and columns. Each row represents a different entry, and each column represents a different value. These tables are `dcrusers`, `dcrprocesses`, and `dcrgraphs`.

Email	Role
1eb4f018f04e6531be595fe765203383a3e4a14...	a25c56aa35c63e9b8bfa8972c8f33097957a898...
97927c4ecddc4d517037b5d83f2b8e46e68d1da...	1df52b58de10dba53600c9230d50f6727661d2c...
cd3c68d11179c4c2986aee27e8c7d9bd8391daf...	c1c224b03cd9bc7b6a86d77f5dace40191766c4...
f2fbc59ba4c7010a79b83a964daa28015ef21ecf...	8a1162ca6bdf68dafd236a5f2acf8a33b144cc9c...
NULL	NULL

(a) `dcrusers` table showing users with their hashed emails and roles.

GraphID	SimulationID	CreatedDate	IsTerminated
1706803	2013849	2024-06-04	0
1706870	2013850	2024-06-04	0
1706872	2013851	2024-06-04	0
1706874	2013852	2024-06-04	0
NULL	NULL	NULL	NULL

(b) `dcrprocesses` table showing four active simulated processes of four different residents.

GraphID
1706803
1706870
1706872
1706874
NULL

(c) `dcrgraphs` table showing four GraphIDs of four residents.

Fig. 13: The three tables used in the application.

The `dcrusers` table stores the email addresses and the role of the users as hashed data, as shown in figure 13a. Hashing is done to ensure the security of the nursing home workers in case of a data breach. The hashing is done by the `hashData()` function which uses the `sha256()` function from the Python library `haslib`. Once some data has been hashed it is impossible to unhash, therefore queries to the `dcrusers` table, will always return the hashed data making the data secure.

```
CREATE TABLE dcrusers (
    Email varchar(255),
    Role varchar(255),
    CONSTRAINT PK_Email PRIMARY KEY (Email)
);
```

Listing 6: The MySQL query used to create the `dcrusers` table.

The primary key is the Email field, meaning that each user has to have a unique Email. This table is essential for user authentication and role management within the application. When a user attempts to log in, the application queries this table to verify the user's identity and role.

The `dcrprocesses` table manages the various processes initiated within the application. Each entry includes a `GraphID`, `SimulationID`, `CreatedDate`, and `IsTerminated`. The `GraphID` corresponds to the ID of the DCR process model created for a specific resident, while the `SimulationID` is the ID of a simulated process. The `IsTerminated` boolean indicates whether a simulation has been terminated, which occurs when an administrator clicks the "Terminate all processes" button. This boolean ensures that an active process is not disrupted by a new process, each time the resident is selected. This allows a fresh process to be started when needed. In figure 13b all the simulated processes are active, which is shown as "0" in the `IsTerminated` column. The `CreatedDate` provides the date that the simulation was created. The `SimulationID` is the primary key, ensuring each process has a unique identifier.

The `dcrgraphs` table stores the `GraphID` of all available DCR process models for residents, as shown in figure 13c. This table only includes `GraphID`, with `GraphID` also functioning as the primary key. The `GraphID` is to be inserted separately from the application as it would not be managed by caregivers. The intended purpose is that a new `GraphID` will be inserted each time a new resident is registered.

To connect to the database using `MySQL` in the code, the `mysql.connector` module is used. The function takes the login information of the database and establishes a connection, enabling MySQL calls. Once a connection is established, a database cursor is created to execute SQL commands and queries. This is done by the `dbQuery` function 7.

It tries to connect to the database and then depending on the first word in the query it can either return data or change the database tables. The different types of MySQL call this project will use are `SELECT`, `INSERT`, `UPDATE`, and `DELETE`. The tests will also use `TRUNCATE` to reset the tables to ensure the tests will always work and not interfere with the rest of the app. If the query is a `SELECT` query it will return the result of the query. To avoid having to format the result, a statement can be added to `SELECT` queries which will either fetch one or all of the matches from the query. The `SELECT one` statement allows using the query directly in an if statement making the code easier to understand and read. An example of this is in `connectToSim` where it is used to check if the simulation has already been created or if a new one should be made.

An example of using the else part of `dbQuery` is `forceTerminateAdmin` which uses `UPDATE` to change all `IsTerminated` values to `TRUE` in `dcrprocesses`. Its sole purpose is to update or change the data in the table and therefore requires no return data.

Establishing a connection once, rather than every time dbQuery is called would be ideal, but it is impossible because of how the `mysql.connector` class works. This is because every time a new function is called the connection is disabled, creating the need to connect to the database when making a MySQL call. Since connecting to the database is rather slow it causes the application to load for multiple seconds. Which is when logging in and every time a resident is chosen. This is because both functions require multiple queries to ensure safety and to make sure that an existing simulation is connected to, rather than creating a new one every time.

```python
def dbQuery(query, statement=None):
    try:
        db = mysql.connector.connect(**login)
        ...
    else:
        if query.startswith("SELECT"):
            cursor = db.cursor()
            cursor.execute(query)
            if statement == "one":
                result = cursor.fetchone()[0]
            if statement == "all":
                result = cursor.fetchall()
            cursor.close()
            db.close()
            return result
        else:
            cursor = db.cursor()
            cursor.execute(query)
            db.commit()
            cursor.close()
            db.close()
```

Listing 7: The function that facilitates database connection.

```python
def forceTerminateAdmin(self, instance):
        dbQuery("UPDATE DCRprocesses SET IsTerminated = true WHERE IsTerminated = false;")
```

Listing 8: The function to terminate all active processes.

The integration of MySQL in the project ensures a structured and secure way of managing data essential to the application's functionality.

8 Testing

Testing is a crucial part of the application development process, ensuring that the application is reliable and can complete the desired tasks. It helps identify and fix bugs early and ensures that the application meets user expectations and requirements. This section will detail the testing methodologies employed to test the application and to ensure its quality and reliability. Two test approaches were used: unit testing and exploratory testing. Unit testing focuses on verifying the correctness of individual functions through automated tests, while exploratory testing examines the overall functionality, usability, and visual aspects of the app. It is not possible to unit test every function as most of them change the visual user interface and cannot be tested automatically. This makes Exploratory testing the perfect addition to Unit tests, providing a comprehensive test sheet that can identify potential issues, ensuring that the application will work as expected.

8.1 Testing the functions

Unit testing is a test method used to test each function individually and in isolation. The purpose is to ensure that each function works as intended and follows all requirements. Unit tests are written at the beginning

of the development process and serve as tests that ensure the functions work properly when changes are made to the code. This allows for automatically testing the functions whenever changes are made to the code. As the functions are tested in isolation it allows for quick and precise tests, that allow the developer to easily find and fix errors, opposite integration tests that test bigger parts of the code at the same time. [28]

Unit testing was chosen as the function testing method for this application as it is important that each function is tested individually, and as the application as a whole is already being tested using Exploratory Tests. This makes using tests where multiple functions are tested as a single test unnecessary.

The objective of unit testing is to isolate a small section of code, verify the correctness, and test every function and procedure. All the unit tests written for this project follow the same procedure to ensure that the tests will work every time, and will not interfere with the normal use of the app. The procedure is as follows: First, everything the test will use or have an effect on will be reset. Then the test cases will be made and tested. After the test is done, everything the test affected will be reset. An example of this is the test for `forceTerminateAdmin()`, which is used to terminate all processes allowing new ones to be made, as shown in listing 9.

```python
def test_forceTerminateAdmin():
    mainApp = MainApp()
    #Resets the dcrprocesses table by removing all data, and checks that it is empty
    dbQuery("Truncate dcrprocesses;")
    assert dbQuery("SELECT * FROM dcrprocesses", "all") == [], "The dcrprocesses table should be empty."

    #Create two processes that will be terminated
    dbQuery("INSERT INTO dcrprocesses (GraphID, SimulationID, CreatedDate, IsTerminated)
        VALUES (123, 456, '2024-05-23', 0)")
    dbQuery("INSERT INTO dcrprocesses (GraphID, SimulationID, CreatedDate, IsTerminated)
        VALUES (111, 222, '2024-05-23', 0)")

    #Checks that the two processes has been added and arent terminated
    result = dbQuery("SELECT GraphID FROM dcrprocesses WHERE IsTerminated = 0", "all")
    expected_result = [(111,), (123,)]
    assert result == expected_result, "The 2 inserted processes should not be terminated."

    #Force terminates all processes by calling the function that will be tested
    mainApp.forceTerminateAdmin(mainApp)

    #Checks that the two processes has been terminated
    result = dbQuery("SELECT GraphID FROM dcrprocesses WHERE IsTerminated = 0", "all")
    assert result == [], "All processes should be terminated."

    #Resets the dcrprocesses table by removing all data
    dbQuery("Truncate dcrprocesses;")
```

Listing 9: Python function to test terminate the functionality of admin.

The test follows the procedure by emptying the `dcrprocces` table. Then it adds two new processes and checks that they have been added and are not terminated. It then calls the `forceTerminateAdmin` function which should set the `isTerminated` values to 1 and checks that it has been done. At last, it removes all processes from the `dcrprocesses` table, ensuring that the test will not have an affect on the normal application usage.

Some tests needed to be rewritten during the production of the code as the functions were updated and refactored which in some of the functions added or removed functionality. An example of this is also

`forceTerminateAdmin()`, as originally the name of the graph was added as a column in `dcrprocesses`. The test needed to be updated such that the two processes, that were added as test processes, were added correctly.

All tests were successful and show that, by themselves, each of the functions, that were tested, works as expected. The list of functions that were tested using unit testing can be seen in Appendix H, and the list of functions that were unable to be tested can be seen in Appendix I.

8.2 Testing the application

Exploratory testing is the process of testing software by exploring it. Contrary to unit tests the test cases aren't meticulously planned out with detailed documentation created beforehand. As Exploratory testing is done on software, it allows testing all visual elements of the app, making it the perfect way to test the app. Exploratory testing allows for rapid feedback during testing and enables quick fixes to the code when an error is found. This makes it easier to adapt and design the app, as the visual aspect of the application can easily be inspected and evaluated.

Three of the most common ways to do exploratory testing are Scenario-based, strategy-based, and Freestyle [29].

Scenario based involves creating realistic scenarios of how the application will be used by the nursing home caregivers. It is used to test how the application will perform during everyday uses. It ensures that the application can handle various interactions and scenarios which gives a comprehensive assessment of its functionality. Scenario-based testing requires knowledge about the day-to-day use cases of the apps and are created by the intended users of the app. In this case, the scenarios were based on answers given during interviews.

Strategy based involves creating specific tests that will test different parts of the code. It is designed to stress test the application and helps identify specific problems identified by the developer. Strategy-based requires extensive knowledge about the application and is based on the developers testing vulnerable parts of the app. This type of test is therefore created by the developers of the app.

Freestyle involves using the software in an ad-hoc manner. There is no maximum coverage and it is used to get to know some software and to test others work. It is used when the testers are not the developers.

As the task list application is small and does a very specific job, there is no reason for the ad-hoc way of testing that the Freestyle approach provides. As the application is used for healthcare purposes, GDPR is a maximum priority, therefore Scenario-based and Strategy-based are better choices, making Freestyle redundant.

8.2.1 Scenario based

The scenarios can be created by the test team or by the end user. The scenarios were created based on interviews with the caregivers, as it allows for as close to real-life scenarios as possible.

Scenario based test 1
The first test is a normal work day for a caregiver on the "Dagholdet" shift while doing the tasks required for all four residents. It is designed to show that multiple notes can be written at different times and that tasks can be done for multiple residents by the same Nursing home caregiver. It will also test that doing a task for one of the residents will not affect the other residents. This scenario test was designed based on an interview E with a healthcare caregiver at the nursing home.

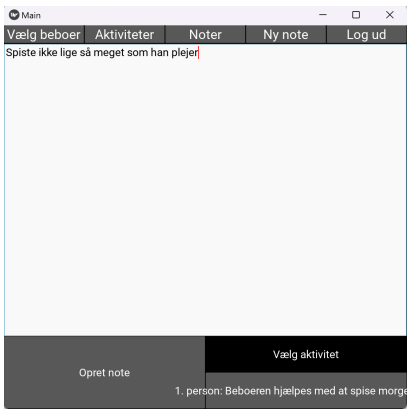
The scope of the test was successful. The user was able to swap between the four residents and do the tasks at different points. The remaining tasks were the same when they accessed the resident's process again. Doing a task for one of the residents did not change the tasks available in the other residents' task list. The

user was able to see their notes when accessing a resident again. The application did not crash at any point.

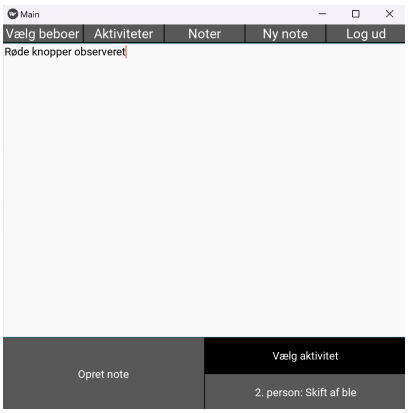
An unforeseen error was found during the test, "Borger C"'s DCR graph was not correct, and two of the tasks that should be accessed by "Aftenholdet" were able to be seen and completed by the "Dagholdet" user. Upon further inspection of the graph, it was found that the two tasks did not have a role added to them. While this error did not remove any usability of the app, it did reveal a breach in the safety of the app. While it may not be a significant issue in this instance, it has the potential to reveal sensitive information about the resident since anyone can view and complete the tasks, regardless of their role.

Scenario based test 2
The second test is a scenario where multiple nursing home caregivers complete tasks on the same resident, "Borger B". This test should show that the application can update itself and show the tasks that are available after another caregiver has completed a task. It should also show that the notes will be updated. The application should also not crash or cause any app-breaking situation. This test ensures that multiple caregivers do not accidentally perform the same task for a resident. This should help to prevent unnecessary work and avoid potential critical health issues such as administrating a double dose of medication to a resident.
This test would also confirm that multiple nursing home caregivers can be assigned the same role.
The scenario is as follows: Caregiver 1 completes a few tasks followed by Caregiver 2 completing some, and this continues until all tasks are done. Both caregivers write a note each to ensure that both notes can be seen by both caregivers. Pictures showing the two nursing home caregivers writing their notes and being able to see them can be shown in figure 14.

The original scope of the test was successful. Both nursing home caregivers were able to see the list of tasks being updated within 5 seconds of the other nursing home caregiver completing a task. Nursing home caregiver 2 was also able to see their note and nursing home caregiver 1's note. But an application breaking situation did happen when both nursing home caregivers accessed "Borger B" at the same time and a simulation hadn't been created yet. This happened as both nursing home caregivers' apps accessed the `dcrprocesses` table at the same time and saw that a new simulation had to be made. Both apps then created a new simulation each and added them to the `dcrprocesses` table. This resulted in the application looking normal but when nursing home caregiver 1 completed a task it would not update on nursing home caregiver 2's app, as they were on two different simulations. When they tried to access the simulation again the application would crash, as the application is designed such that there should only be one of each `graphid` with an `isTerminated` value of 0. As there were two, it crashed.



(a) Caregiver 1s note to "Beboeren hjælpes med at spise Morgenmad".



(b) Caregiver 2 writes a note to "Skift af ble".

(c) Caregiver 2 checks that they can see both of the written notes.

Fig. 14: Picture a and b show the 2 notes that were written and picture c shows that it is possible to read both notes.

Scenario based test 3
The third test is to complete all tasks available to the "Dagholdet", "Aftenholdet" and "Natholdet" roles for a single resident, "Borger D". One note will be written by each of the three roles to test that the notes persist for all three of the roles. The goal of the test is to ensure that the application will work as expected, as a whole during a 24-hour cycle at the nursing home without crashing or errors. It will also test that there can be nursing home caregivers assigned to each of the three different roles.
The scope of the test was successful. It was possible to add the three users each with a different role as seen in figure 15. The emails and roles are hashed to protect the data, row one has the "Natholdet" role, row two has the "Dagholdet" role, row 4 has the "Aftenholdet" role, and row 3 is the "Admin".

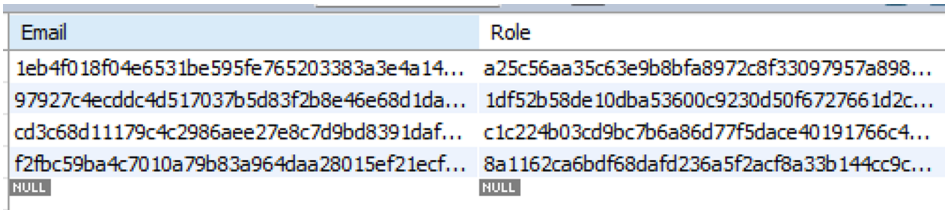


Fig. 15: The users and their roles added to the `dcrusers` table.

All three notes were able to be seen by all three nursing home caregivers. It was possible to complete all available tasks in all three roles without crashes or errors. The notes can be seen in Figure 16.

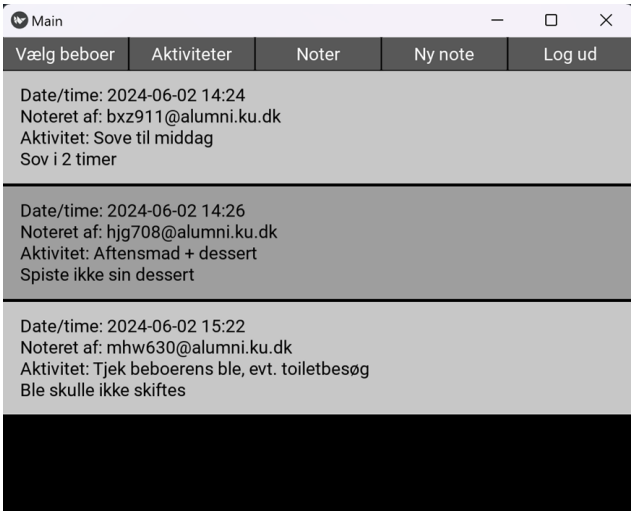


Fig. 16: The 3 notes written during the workday by different roles and users.

8.2.2 Strategy based

Different strategies can be used to test the code, this project will use Error guessing and boundary value analysis. Strategy-based testing requires that the tester has extensive knowledge of the code. Therefore the test cases were created and performed by the developers. [29]

Error guessing
Error guessing is when the tester uses past experience and intuition to guess potential areas of the software that might fail. Error guessing was used to test that logging in with a wrong username or password should result in nothing happening. The application should not crash or give any information to the user. This is important as the application should never crash nor should the application give any information to someone that should not have access to the information.

Multiple tests were created for this purpose.

1. Login with the right username but the wrong password to a user that has access to the graphs.
2. Login with the wrong username.
3. Login with the correct username and password to a user who does not have access to the graphs.
4. Login with the correct username and password to a user that has access to some graphs but not all.

These tests were chosen based on how the code works. For all of the tests, a resident was chosen and an activity was done at random. This was done to ensure that activities would show and be completed when pressed. Originally the login process worked by checking if the inputted username was in the `dcrusers` database table. If that was the case then the buttons to choose a resident were shown.

The tests showed that in tests 1, 3, and 4 the application showed all 4 graphs. Nothing happens in test 2. Tests 1 and 3 were able to see the titles of the residents' graphs but when they chose a resident no tasks were shown. Test 4 worked like the other two, but they were able to see the tasks in the graphs that they had access to. These tests showed major flaws with the implementation as the application user was shown sensitive information in the form of the names of the processes.

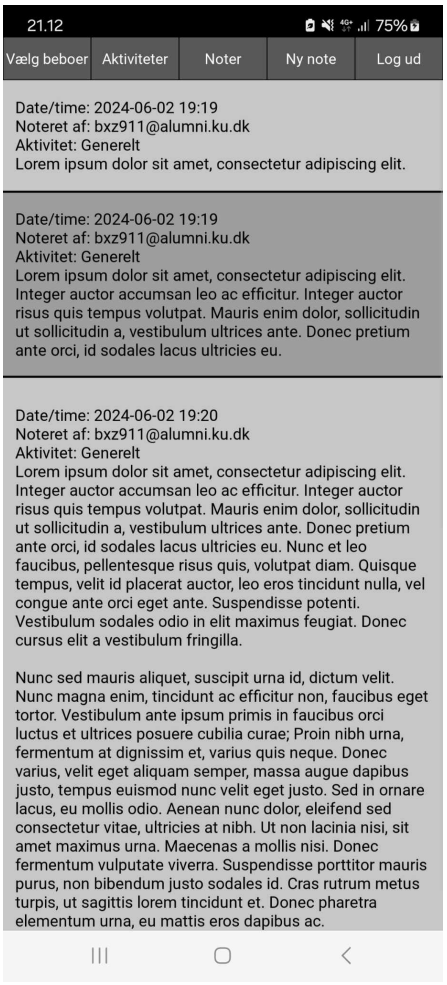
These flaws were fixed and the new login process works by first doing a RESTful API call to the `dcrgraphs` to check that the user is valid and then a MySQL query is done to check that the username is in `dcrusers`, lastly, a RESTful API call and a MySQL query work together to create the resident buttons that the user has access to. This new version passes all the tests resulting in no crashes for all of them. The tests show that for tests 1, 2, and 3 no residents are shown, and in test 4 only the residents they have access to are shown.

Boundary value analysis
Boundary value analysis is when testing that the minimum and maximum bounds of values work as intended. This will be used to test the size of written notes. The notes do not have a minimum or maximum size that can be tested. However, the size of each text block depends on the amount of newlines that have been written in the text and on how long is between each newline. This could result in the notes looking visually unappealing if they're too long. During an interview C, it was stated that the longest notes are usually 3-5 sentences long.

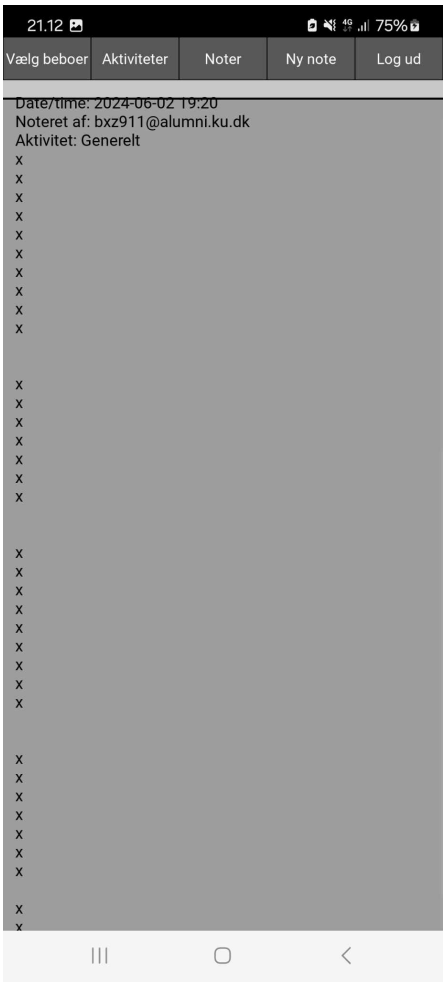
Multiple inputs were tested to ensure that the notes look good and are easy to read. Each note should be contained in its own square and the entire note should be readable. The five different input lengths and their reasoning are below.

1. Write a note without text - This is to ensure that no note will be created if the text box is left empty.
2. A one-sentence long note. - This is to ensure that a short note looks appealing.
3. A five sentences long note. - This is to ensure that the longest notes the nursing home caregivers write looks appealing.
4. A very long note with over 15 sentences. - This is to test what the notes look like when they're way longer than needed.
5. A note that consists of many single chars and newlines. - This is to ensure that the notes look good if they divide the notes into paragraphs.

All notes were written for the same simulation and can be seen in figure 17. The note for test one was not written as expected and the notes for tests two through four looked as expected with no overlap between them. While the note in test five is still readable, the text is on the top edge of the note square making the date harder to read. This is not perfect, but as it is still readable, and since the application is a proof of concept, this implementation is deemed acceptable.



(a) Test notes one to four.



(b) Test note five.

Fig. 17: The 5 notes written for the boundary value analysis test.

All errors found during testing were fixed unless otherwise stated. The tests that found errors were redone to ensure that all errors were indeed fixed. Furthermore, the data obtained from these tests highlight the importance of including the intended user of applications when developing tests. As they can be a big factor in ensuring that the tests are as accurate as possible.

The tests that were conducted provided insights into whether it is possible to create a task list application used by nursing home caregivers. Through testing, the application has become robust and reliable and can handle the tasks it was designed for.

9 Discussion

This discussion will evaluate the use of Participatory Design for this project. The section then analyses the mobile application through a SWOT analysis, examining its strengths, weaknesses, opportunities, and threats. It will assess how effectively the application meets the qualitative requirements defined at the beginning of the project, considering user experience, functionality, and performance. Additionally, it will delve into the security considerations that were prioritized and implemented throughout the development process to maintain data integrity and user privacy. Finally, it will outline potential areas for future work, highlighting enhancements and features that could further improve the app.

9.1 Result from Participatory Design

Despite challenges in maintaining regular feedback sessions during Participatory Design, valuable insights were gathered and used to prioritize essential features. Some desired features, such as graphical tools and health status tabs, were shelved due to constraints but remain future considerations. Final user testing highlighted the application's strengths in functionality and ease of use while pointing out areas for improvement, particularly visual design and note management. It also highlights the need for training or a guide to better utilize the DCR Graph.

The project has now reached the stage where the design needs to be finalized, and the solution implemented based on the latest test runs and feedback from the caregivers. Despite users pointing out that the application lacks visual appeal, the decision has been made to keep the application as a proof of concept. This choice highlights the main objective of this project, which is to show that the DCR Graph can effectively serve as the foundation for creating a task list manager.

9.2 SWOT

In the final version of the application, a SWOT analysis was conducted to assess its strengths, weaknesses, opportunities, and threats related to the solution. This strategic planning tool helps to identify internal and external factors.

Strengths

The application has several strengths that make it a robust solution for task management in nursing homes, First, it will decrease the occurrence of errors since tasks will be clearly defined for caregivers. This clarity enhances the overall quality of nursing care, as tasks are more tailored to the individual needs of each resident. Additionally, the application provides easier access to reading and writing notes, making documentation more efficient. The application is designed to meet the specific needs of caregivers, resulting in high usability. Its simple, user-friendly, and intuitive design allows caregivers to use it effectively with minimal training. The high adaptability of DCR, which serves as the backbone of the application, enables seamless integration with existing systems like Cura through the DCR Solutions Technology Partner program. This program enables software provider partners to integrate DCR technology into their applications, providing case management solutions and incorporating DCR as a core feature of their original software. Additionally, the application is highly scalable, allowing for the easy addition of new features.

Weaknesses

Despite its strengths, the application also has some weaknesses. One significant limitation is the need for a wireless internet connection as the application uses cloud-based computing. Additionally, the proposed application requires mobile devices, such as smartphones and tablets. While this is not an issue for the chosen nursing home, which already provides tablets to its workers, it could be a barrier for other facilities. There is also a need to train caregivers before they can effectively use the application. Caregivers may need incentives to adopt the new application instead of sticking to the original system they are accustomed to.

Opportunities

The application presents several opportunities for expansion and improvement. One significant opportunity is the potential introduction and implementation of the solution in other nursing homes. By demonstrating the benefits of the solution, it can be adopted more widely, improving task management and care quality across multiple facilities. Additionally, the DCR system used in this application could enhance other processes in various healthcare sectors not just nursing homes. Its flexibility and scalability make it a valuable tool for managing complex workflows in different medical and caregiver environments.

Threats

Several threats could impact the success of the application. One potential issue is the possibility of DCR deciding to change its API system. If such changes render the current API incompatible, the application might face significant functionality issues, requiring extensive updates. Furthermore, updates to improve the existing Cura system could trivialize the current method of using the DCR Process Model to create a task list manager. If Cura's updates address the same issues more effectively or offer better integration, the application might be seen as too niche and lose its competitive edge.

The SWOT analysis highlights the application's potential as a valuable tool for improving task management and care quality in nursing homes. The strengths demonstrate its capability to improve task management and care quality, while the identified weaknesses highlight areas for further enhancement. The opportunities point to future scalability and adaptability across various settings. Awareness of potential is essential to maintain the application's relevance.

9.3 Fulfillment of Qualitative Requirements

Throughout the development process, the project has considered qualitative requirements to achieve high standards of usability, reliability, and adaptability. These considerations were integrated into every stage of development, from initial planning and design to implementation. By taking these qualitative requirements into consideration, the project aimed to create a solution that not only met functional needs but also provided a seamless, efficient, and user-friendly experience.

Flexibility was a key qualitative requirement identified early in the project. The application needed to be adaptable to various scenarios and capable of evolving with changing user needs.

- The DCR Graph framework was chosen for its inherent flexibility. DCR Graphs allow for easy modifications and additions to the workflow, enabling caregivers to update and adapt task lists without extensive technical knowledge. This capability allows the application to respond to changes in the care environment.
- Continuous user input was taken into consideration to keep the application aligned with user needs. The application could be adjusted based on their feedback, maintaining its usefulness.
- The application was designed to be scalable, facilitating the addition of new features and functionalities as needed. This scalability allows the application to grow and adapt, accommodating more users, more residents, additional tasks, and new functionalities.

Effectiveness was another crucial qualitative requirement for the application. The goal was to create an application that improved caregivers' ability to manage tasks efficiently and accurately.

- From the beginning, the application's design was based on the needs and workflows of the caregivers. The application was customized to integrate smoothly into their daily routines by conducting interviews and gathering feedback. This user-centered approach allowed the application to address real-world challenges faced by caregivers.
- The use of DCR Graphs allowed for a detailed and structured representation of the different tasks. This task management system listed all necessary tasks and assigned them to the appropriate shifts.

Reliability is an important aspect of any application, especially in a healthcare setting where consistent performance can directly impact the quality of care.

- The backend infrastructure of the application was designed with reliability in mind. Microsoft Azure provides reliable cloud storage, while MySQL, a widely used relational database management system, offers a consistent environment.

- By using the simulation function, new process models can be tested before connecting them to the application itself. The DCR simulation tool allows developers to simulate different scenarios and workflows in a controlled environment. Helping with identifying potential issues in the process models without affecting the live application.
- Extensive testing was conducted on the application, involving several stages to verify its reliability. These stages include unit testing and exploratory testing. Unit testing involves evaluating individual functions of the application in isolation to confirm they work as expected. Exploratory testing is a more informal and flexible approach where testers actively explore the application to uncover potential issues. This type of testing is useful for identifying usability problems, edge cases, and unexpected behaviors.

Securing the application is crucial, especially in a healthcare setting where sensitive patient information is handled.

- All communication between the application and its connection to cloud services is secured using HTTPS. By employing HTTPS, the integrity of the data is maintained, making it difficult for unauthorized parties to access the information.
- The login system verifies that every personnel capable of using the application must be authenticated by the DCR system and granted access to the relevant processes by an administrator. Users are assigned specific roles by an administrator, restricting their access to only the data and functions necessary for their duties. This role-based access control (RBAC) limits access to sensitive information to those who need it, reducing the potential for data breaches.
- Compliance with GDPR and other relevant data protection regulations is strictly maintained. This allows users to manage their data privacy settings, access their data, and request the deletion of their personal information. The DCR system supports GDPR compliance by securely storing notes in the DCR Active Repository. To further enhance security, notes can be encrypted, as they may contain sensitive data. The encryption key can be stored locally to prevent data breaches in the cloud storage of the DCR Active Repository. By utilizing the DCR Active Repository, the application guarantees that entire processes, including all associated sensitive data, can be deleted when necessary.

Creating a **positive user experience** was an important aspect of the application's development process.

- By conducting interviews and gathering feedback from actual users, the application can fully represent their daily routine and therefore can help them with completing them optimally.
- The application was designed with simplicity and ease of use in mind. Clear labeling of tasks, simple navigation menus, and a natural flow of activities helped create an intuitive user experience.

By integrating these qualitative requirements into every state of development, the project successfully created a task list application that is flexible, effective, reliable, secure, and user-friendly. These efforts resulted in a comprehensive solution that meets the standards required in a healthcare setting, therefore improving the quality of care provided.

9.4 Security Considerations

While the current implementation of the application addresses several aspects of data security and user privacy, there are opportunities for further enhancement. Several strategies could have been employed to improve the security of the application.

Implementing data encryption would enhance the security of the stored data. While some security measures are already in place for caregivers' information, further encryption of data, such as notes taken during nursing care, would better align with privacy requirements and GDPR. Encrypting these notes is crucial, as they contain sensitive personal information about the residents that must be protected against unauthorized access.

Currently, the solution uses a public cloud architecture, storing data in Microsoft Azure and using DCR Active Repository for processing simulations, both of which are cloud services. To increase security, transitioning to a hybrid cloud architecture could be considered. In a hybrid model, sensitive data can be stored encrypted in a public cloud, with the encryption key kept locally. This way, unauthorized access to the cloud data is useless without the locally stored keys.

Additionally, the nursing home or Cura could also be part of the DCR Solutions Technology Partner program. This program would allow them to have access to the DCR Active Repository within their internal system, making it safer since data and processes would not be stored in the cloud. Sensitive healthcare data could be at risk of being stored unsafely in the cloud during the simulation of a DCR Graph in DCR Designer. To address this risk, integrating DCR Designer into the internal system might be a solution, allowing for the simulation and testing of new processes without exposing sensitive information. However, the current challenge is that no companies or industries have yet adopted DCR Designer for their internal systems.

By incorporating these additional security measures, the application would comply with GDPR, and therefore provide a more secure environment for managing sensitive healthcare data.

9.5 Future work

9.5.1 See and utilize the data from previous process simulations

Currently, there is no way to access old terminated simulations in the application. This function was deemed unnecessary as the application is a proof of concept of being able to use a DCR Graph as the basis for a task list application. Further work with the application should consider enabling this, as being able to see old notes could help with caring for the residents. Ways to implement this could be to simply add an extra screen when selecting a resident that shows all the simulations that have been added to the `dcrproccess` table for that resident. Another way could be to add a new screen that allows searching for all notes written for a task or all notes written by a specific caregiver.

A challenging but potentially very helpful solution could be to save all data about all simulations. This includes which caregiver completed what tasks and when they were done on different days. This would enable the creation of diagrams that could help show the average time a specific task is completed or how often an optional task is done. This could help find irregularities in the day-to-day tasks or problems with the residents, that a caregiver normally would not catch. Or help find the correlation between when a task is done and how the rest of the resident's day turns out.

9.5.2 Start all simulations

During scenario-based test 2, it was found that if two caregivers started a new simulation at the same time it would break the application. Two solutions to this error were thought up. The first was to allow multiple non-terminated processes with the same `graphid` to be added to the `dcrproccesses` table. This fix presented new issues, as a way to choose between which simulation to access would have to be created. But it would also cause problems as there should only be one simulation that isn't terminated at a time. The second solution was to create a new function that starts the simulation of all residents and adds it as a button on the admin screen. This fits the application's current functionality as admins are already able to remove all users and terminate all graphs.

Due to time constraints and since the chance of two caregivers trying to access the same resident at the same time before a simulation started is very low, this error was deemed as not a top priority to fix. Therefore, this was not implemented as the application is intended to be considered a proof of concept. However, it should be a top priority to fix it for further work with the application.

9.5.3 Decreasing the runtime of the application

In the "Utilizing MySQL in the implementation" section 7.3 it was described how database queries are slow. This creates a problem as it disrupts the flow of the application, especially since the application does not indicate it is loading. Fixing the runtime while logging in and adding a loading indicator should be a top priority if the application were to be further developed and used by nursing home caregivers. Reducing the runtime can be achieved by only establishing a connection to the database once or by reducing the time it takes to establish the connection. Removing the requirement to connect every time a query needs to be done is impossible when using MySQL.connector, therefore a new library or another database manager is required. Reducing the time it takes to connect could be reduced by locally hosting the database at the nursing home facility and requiring connecting to the same network.

10 Conclusion

This project aimed to introduce innovative solutions to enhance task management in nursing homes, particularly addressing the challenges posed by an aging population. Faced with a growing elderly demographic, especially in Denmark, healthcare systems experience increased pressure, requiring improvements in efficiency and care quality. To address these issues, a task list application utilizing Dynamic Condition Response (DCR) Graphs was developed. The project has demonstrated the effectiveness of creating a task list application using DCR Graphs to enhance task management in nursing homes. By leveraging DCR Graphs and a structured development approach through PAIS. the project has produced a mobile application that provides an interactive checklist for caregivers. Which could potentially streamline daily operations and reduce errors associated with routine tasks. The integration of DCR Graphs, MySQL, and the DCR Active Repository has resulted in an effective and scalable solution tailored to the needs of nursing home caregivers.

The development process, guided by Participatory Design, actively involved caregivers, resulting in valuable insights into their routines and identifying useful functionalities. One of these functionalities was the note system, which the project succeeded in implementing allowing caregivers to write notes directly on their tablets anywhere in the nursing home, eliminating the need to return to their workstations. Although some functionalities and features considered during development were not implemented due to time constraints, the project still serves as a solid proof of concept for a mobile application in a healthcare setting using DCR Graphs. Testing and a SWOT analysis revealed the application's reliability and potential for scalability to other facilities, it also highlighted areas for future improvement, particularly in security and loading times. Despite these areas needing enhancement, the project successfully developed a task list application that effectively addresses operational challenges in nursing homes.

References

[1] A. Oksuzyan, A. Hohn, J. K. Pedersen, R. Rau, R. Lindahl-Jacobsen, and K. Christensen, "Preparing for the future: The changing demographic composition of hospital patients in Denmark between 2013 and 2050." *PLOS ONE*, 2020.

[2] European Parliament Members' Research Service, "ehealth – technology for health," Briefing, March 2015, accessed: 02.05.2024.

[3] M. Esteves, M. Esteves, A. Abelha, and J. Machado, "A proof of concept of a mobile health application to support professionals in a portuguese nursing home," *Sensors*, 2019. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6767027/

[4] DCR-Solutions-A/S. Dcr active repository. [Online]. Available: https://documentation.dcr.design/documentation/dcr-active-repository/

[5] R. Breitschwerdt, S. Robert, and O. Thomas, "Mobile application systems for home care: Requirements analysis & usage potentials," 2011. [Online]. Available: http://aisel.aisnet.org/amcis2011_submissions/152

[6] M.-H. Kuo, S.-L. Wang, and W.-T. Chen, "Using information and mobile technology improved elderly home care services," *Health Policy and Technology*, vol. 5, pp. 131–142, 2016. [Online]. Available: https://doi.org/10.1016/j.hlpt.2016.02.004

[7] G. Manzano-Monfort, G. Paluzie, M. Díaz-Gegúndez *et al.*, "Usability of a mobile application for health professionals in home care services: A user-centered approach," *Sci Rep*, 2023. [Online]. Available: https://doi.org/10.1038/s41598-023-29640-7

[8] R. S. Evans, "Electronic health records: Then, now, and in the future," *IMIA Yearbook of Medical Informatics*, pp. S48–S61, 2016.

[9] K. Wiegers and J. Beatty, *Software Requirements, Third Edition.* Microsoft Press, 2013.

[10] Information Commissioner's Office, *Guide to the General Data Protection Regulation (GDPR)*, August 2018, accessed: 09.06.2024. [Online]. Available: https://ico.org.uk/media/for-organisations/guide-to-the-general-data-protection-regulation-gdpr-1-0.pdf

[11] M. Kierkegaard. Oversigt over sundhedsapps. Accessed: 01.05.2024. [Online]. Available: https://www.regionh.dk/Sundhed/Patientguiden/undersoegelse-for-en-sygdom/Vaerktoejer-og-huskelister-der-giver-overblik/Sider/Oversigt-over-sundhedsapps.aspx

[12] S. I. Buhl, "Hjemmeplejens digitale virkelighedskløft," *DJES*, May 2022. [Online]. Available: https://creativecommons.org/licenses/by/4.0/

[13] G. Cloud. (2024) What is cloud architecture. Accessed: 06.06.2024. [Online]. Available: https://cloud.google.com/learn/what-is-cloud-architecture

[14] C. Wilkinson, "Evaluating the role of prior experience in inclusive design," Ph.D. dissertation, 12 2011.

[15] F. Kensing and J. Blomberg, "Participatory design: Issues and concerns," *Computer Supported Cooperative Work (CSCW)*, vol. 7, pp. 167–185, 1998. [Online]. Available: https://doi.org/10.1023/A:1008689307411

[16] J. S. Keld Bødker, Finn Kensing, *Professionel it-forundersøgelse - grundlag for brugerdrevet innovation.*

[17] M. Reichert and B. Weber, *Enabling Flexibility in Process-Aware Information Systems: Challenges, Methods, Technologies.* Springer Berlin Heidelberg, 2012. [Online]. Available: https://doi.org/10.1007/978-3-642-30409-5

[18] V. Paradigm. (2023) What is case management model and notation (cmmn). Accessed: 15.05.2024. [Online]. Available: https://www.visual-paradigm.com/guide/cmmn/what-is-cmmn/

[19] L. R. Christensen and T. Hildebrandt, "Modelling cooperative work at a medical department," *C and T*, 2017.

[20] DCR. (2021, May) Dcr whiteboard. Accessed: 06.06.2024. [Online]. Available: https://documentation.dcr.design/documentation/introduction-to-business-rulesvalue/

[21] ——. (2024, June) Introduction to business rules. Accessed: 06.06.2024. [Online]. Available: https://documentation.dcr.design/documentation/introduction-to-business-rulesvalue/

[22] ——. (2024, April) Dcr simulator. Accessed: 15.05.2024. [Online]. Available: https://documentation.dcr.design/documentation/dcr-simulator/#Render_the_process_model

[23] ——. (2024) Become our partner. Accessed: 06.06.2024. [Online]. Available: https://dcrsolutions.net/partners/

[24] M. Virbel, T. Hansen, and O. Lobunets, "Kivy–a framework for rapid creation of innovative user interfaces," 2011.

[25] dev.mysql.com. What is mysql? [Online]. Available: https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html

[26] Q. Z. Sheng, X. Qiao, A. V. Vasilakos, C. Szabo, S. Bourne, and X. Xu, "Web services composition: A decade's overview," *Information Sciences*, vol. 280, pp. 218–238, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025514005428

[27] A. Møller, "Tilsynsrapport: Hotherhaven, stevns kommune," December 2017.

[28] Geeksforgeeks. (2024) Unit testing – software testing. Accessed: 26.05.2024. [Online]. Available: https://www.geeksforgeeks.org/unit-testing-software-testing/

[29] ——. (2024) Exploratory testing. Accessed: 30.05.2024. [Online]. Available: https://www.geeksforgeeks.org/exploratory-testing/

11 Appendix

A Survey Interviews

Spørgsmål til medarbejdere på Plejecenter Hotherhaven

Navn (Forkortelse er også fint)**:** Personale A

Stilling: Social- og Sundhedsassistent (SSA)

Hvordan bliver dagens opgaver planlagt? Og hvordan bliver det besluttet hvem der skal gøre hvad?

Hverdag der en Sundhedsassitent (SSA). Peronal som uddelegerer dagens opgaver. Først SSA uddeler SSA opgaver til andre SSA personale og når det er gjort uddeler resten af SSH (Sunhedshjælper) opgaver til andreSSh personale. Den der uddeler altid spørge andre kellega hvem eller hvilke beboere de vil gerne have på den dag. Udddelegering forgår i vores system "Cura" der giver overblik over relevante informationer eller dagens opagver og borgere. Personalet bruger det til at dokumentere relevante oplysninger i borgerjournalen.

Kan du give mig en tidlinje på hvad I laver på en typisk dag?

Vi starter med at mødes på vores grupperum mindst 30 minutter. Bagefter snakker vi lidt om vores beboere (per afdeling). Når det er kl 8 starte vi at gør klar mad til vores beboere. Vi er også igang med at tjekke om der er nogen der er klar til til morgenpleje. Når vi er færdig med at pleje, giver vi morgenmad og medicin i fællesrummet. Bagefter går vi og dokumentere i følge "Cura". Kl 12 frokost tid, derefter toiletbesøg eller gå en lille tur. Dokumentering kl 14 kaffe og kage tid. Inden vagtskift giver vi rapport til næste vagt.

Hvilke opgaver skal laves omhandlende én beboer på en typisk dag?

- nedre/ovre hygiejne
- rengøring
- medicingivning
- give mad / støtte / spise
- toiletbesøg / sove til middag
- dokumentering

Kan du går i dybde i hvordan I tilsyner en beboer (giv et eksempel for et bestemt beboer uden at nævne deres navn)?

Evt. er der nogen opgaver som skal gøres før en anden opgave? (mad før medicin)

En beboer der har en Parkinsons sygdom – jeg giver først medicin inden jeg starte morgenpleje og give mad

Tager I noter under tilsynet? Hvis ja, hvordan?

Ja.
Medarbejde har sin egen Ipad og skriver i ”Cura” hvis der er ændringer i beboers tilstand. Vi finder det rigtig fokus område også skriver vi fx. forhøjet blodtryk eller ødemer under cirkulation.

Er oplysingerne givet af den medicinske historie tilstrækkelige?

Ja

Hvilke yderligere tekniske værktøjer anvendes under plejen?

Vi bruger Varportalen, som er en vidensdatabase med procedurer og instrukser til brug for samfundsfagligt personale og det er presentereret med illustrationer og videoer.

Hvordan registrer behovet for medicin og hvordan registreres det?

Igennem ”Cura” under P.N.
Du skal registere det for eks. Paracetamol mod smerte eller hovedpine. Skrive hvor meget beboer har fået og vurdere bagefter om hvis der har været noget effekt.

Ind i kontoret, hvad bliver dokumenteret/noteret omkring beboeren?

Det bliver dokumenteret beboere helbredstilstand og deres funktion tilstand

Hvad er formålet med dokumentationen?

For at give uddybe beskrivelse af beboeres helbredstilstand.

Ekstra noter

Spørgsmål til medarbejdere på Plejecenter Hotherhaven

Navn (Forkortelse er også fint)**:** Personale B

Stilling: Social- og Sundhedshjælper (SSH-elev), Klinten

Hvordan bliver dagens opgaver planlagt? Og hvordan bliver det besluttet hvem der skal gøre hvad?

SSA fordeles i fællesskab alle SOL-ydelser til dagen i Cura.
I de enkelte team planlægges de øvrige opgaver (SEL-ydelser)

Kan du give mig en tidlinje på hvad I laver på en typisk dag?

7 – 7:30:
Planlægning og koordinering
7:30 – 10:30/11:
Medicingivning, morgenpleje, morgenmad til beboere og løbende dokumentering af tilsyn.
11:45 – 12:30:
Frokost til beboere
12:30 – 30:
Oprydning
13 – 13:30:
Dokumentation, spise pause til personale
13:30 – 15:
Kaffe til beboerne, oprydning og hygge

Hvilke opgaver skal laves omhandlende én beboer på en typisk dag?

Medicingivning, mad, personlig pleje, bad, at påklæde evt. Madning af beboer, dokumentation

Koordinering med kollegaer og SSA vedrørende beboere

Kan du går i dybde i hvordan I tilsyner en beboer (giv et eksempel for et bestemt beboer uden at nævne deres navn)?

Evt. er der nogen opgaver som skal gøres før en anden opgave? (mad før medicin)

Tilsyn ca. en gang i timen eller hver 1.5 time
Medicin før mad til en af mine beboer

Tager I noter under tilsynet? Hvis ja, hvordan?

Ja – dokumenter medicingivning

Er oplysingerne givet af den medicinske historie tilstrækkelige?

Ja for mig som SSH; jeg henter oplysninger i Cura

Hvilke yderligere tekniske værktøjer anvendes under plejen?

Jeg anvender kun Cura

Hvordan registrer behovet for medicin og hvordan registreres det?

…

Ind i kontoret, hvad bliver dokumenteret/noteret omkring beboeren?

- SSH ændringer I habituel tilstand -> tidlig opsporing samt opfølgning på udvikling af borger -> vurdering fx. i forhold til medicin behov, ændre tiltag, osv.
- Aftaler
- Alle opgaver i dagsorden inklusivt medicin
- POV-medicin

Hvad er formålet med dokumentationen?

- Lovpligtige!
- Koordinering mellem vagtplan og tværfaglig kommunikation fx med Ergo ansvarlig, forflytning vejledning af demens koordinering, osv
- Tidlig opsporing
- Dokumentation af udført ydelse/opgaver
- komuunikation mellem ledelse + komunal bestyrelse + pårørende

Ekstra noter

Spørgsmål til medarbejdere på Plejecenter Hotherhaven

Navn (Forkortelse er også fint)**:** Personale C

Stilling: Sygeplejerske

Hvordan bliver dagens opgaver planlagt? Og hvordan bliver det besluttet hvem der skal gøre hvad?

Dagens opgaver bliver fordelt om morgenen.
Det er blandt andet mig som sygeplejerske der sørger for Personalet har opagver at fordele, da det er mig der opretter dem i journalsystemet.

Kan du give mig en tidlinje på hvad I laver på en typisk dag?

7 – 12:
Borger overblik (opgave overblik)
12 – 15:
Administrative opgave + Adhoc opgaver

Hvilke opgaver skal laves omhandlende én beboer på en typisk dag?

Som sygeplejerske laver jeg kun sygeplejefaglige opgaver sårpleje etc.

Kan du går i dybde i hvordan I tilsyner en beboer (giv et eksempel for et bestemt beboer uden at nævne deres navn)?

Evt. er der nogen opgaver som skal gøres før en anden opgave? (mad før medicin)

Det fremgår af journalen, hvilke opagver der gøres først

Tager I noter under tilsynet? Hvis ja, hvordan?

Det optimale svar er her, ja

Er oplysingerne givet af den medicinske historie tilstrækkelige?

Ja

Hvilke yderligere tekniske værktøjer anvendes under plejen?

Der bruges Ipad for at tilse en borgers døgnrytme

Hvordan registrer behovet for medicin og hvordan registreres det?

Der er afkrydsningssystem

Ind i kontoret, hvad bliver dokumenteret/noteret omkring beboeren?

Alle afvigelser af borgers dagligdag

Hvad er formålet med dokumentationen?

At sikre den bedste pleje eller behandling hos borger

Ekstra noter

Vi bruger Cura journalsystem

Spørgsmål til medarbejdere på Plejecenter Hotherhaven

Navn (Forkortelse er også fint)**:** Personale D

Stilling: Social- og Sundhedshjælper (SSH)

Hvordan bliver dagens opgaver planlagt? Og hvordan bliver det besluttet hvem der skal gøre hvad?

Ved starten af vagten, fælles planlagt opgaver, hvor opgaverne blive deles ude mellem 3 personale.Om spisning, personligepleje ogklar til sengetid

Kan du give mig en tidlinje på hvad I laver på en typisk dag?

Kl. 15.00 – Ved starten af vagten, får report fra dagvagten
Kl. 15.30 – Oprydning og gøre ting klar til middagsmad
Kl. 15.30-16.00 – Hjælpes med borger på toiletsbesøg og andre aktivitet
Kl. 17.30 – Laves klar til middagsmad og start med spisning
Kl. 19.30 – Gives kaffe/kage
Kl. 20.30 – Hjælpes nogle borger til sengetid

Hvilke opgaver skal laves omhandlende én beboer på en typisk dag?

Til borger det kunne ikke klar selv med spisning og andre gør det selv. De beboere der skal hjælpes hjælpesmed spisning og personligpleje og andre opgaver.

Kan du går i dybde i hvordan I tilsyner en beboer (giv et eksempel for et bestemt beboer uden at nævne deres navn)?

Evt. er der nogen opgaver som skal gøres før en anden opgave? (mad før medicin)

Tilsyn til borger, hjælpes med toiletbesøg, medicin givning som bliver ordinere og hjælpes med spisning

Tager I noter under tilsynet? Hvis ja, hvordan?

Hvis nødvendig, skrives i systemet (CURA)

Er oplysingerne givet af den medicinske historie tilstrækkelige?

Ja

Hvilke yderligere tekniske værktøjer anvendes under plejen?

Hjælpemidler

Hvordan registrer behovet for medicin og hvordan registreres det?

Det er skrevet i systemet (CURA)

Ind i kontoret, hvad bliver dokumenteret/noteret omkring beboeren?

Alt dokumentation omkring beboeren bliver skrevet/noteret i systemet (CURA)

Hvad er formålet med dokumentationen?

Som bevis på at borger får ydelse sombliver visiteret til

Ekstra noter

For de demente, deres dagligdag er ikke ens, deres mønstre af ”blomstring er bliver fremme til måne”.
Nogen af dem er helttil måne. Mak kan ikke orientere dem individuelt som ens, dordi der er psykisk og fysisk forskellige.

Spørgsmål til medarbejdere på Plejecenter Hotherhaven

Navn (Forkortelse er også fint)**:** Personale E

Stilling: Social- og Sundhedsassistent (SSA)

Hvordan bliver dagens opgaver planlagt? Og hvordan bliver det besluttet hvem der skal gøre hvad?

Dagens opgaver er planlagt ud fra de opagver der er hos de enkelte borger.
Kompentencer er medvirkende til de opgaver der bliver fordelt.

Kan du give mig en tidlinje på hvad I laver på en typisk dag?

1. Tager borgere op
2. Udfører SSA opgaver hos andre borgere
3. Kontakt med de andre (læge, pårørende, osv.)
4. Medicin dispensering

Hvilke opgaver skal laves omhandlende én beboer på en typisk dag?

1. Vask i sengen
2. Støtte strømper
3. Tøj på i sengen
4. Hjælpe ud af sengen
5. Hjælpe påbadeværelse
6. Redde seng
7. Give morgenmad + medicin
8. Gå en tur med borger
9. Tænde for TV
10. Give formiddags måltid
11. Hjælpe ind på toilet
12. Give frokost + medicin
13. Hjælpe ind på toilet
14. Hjælpe i seng til middagslur

Kan du går i dybde i hvordan I tilsyner en beboer (giv et eksempel for et bestemt beboer uden at nævne deres navn)?

Evt. er der nogen opgaver som skal gøres før en anden opgave? (mad før medicin)

Der er en del medicin som skal gives på specifikke tider (Parkinson) der er også medicin som skal gives før eller efter er måltid (penicillin), der er en del medicinering som er vigtig at give før man udfører andre opgaver.
Det kan også være en borger med KOL da skal de gerne have medcin (spacer) før de laver noget fysisk.

Tager I noter under tilsynet? Hvis ja, hvordan?

Jeg tager noter på min lille blok, men det er kun forsi jeg ikke kan have min tablet med rundt

Er oplysingerne givet af den medicinske historie tilstrækkelige?

Ikke altid det sker at der er en borger som har fået sin medicin senere end der står på FMK og da kan den næste medicingivning ske for tidligt.
Fx. paracetamol gives giver 4 time borgere har fået kl. 09.00 fordi da fik borgere morgenmad og så får borgeren igen kl. 12.00 => 3 timer hvilken er en fejl

Hvilke yderligere tekniske værktøjer anvendes under plejen?

Der bruges VAR fx. ved kateterskift eller ved stomipleje. Der kan også bruges promedicin, når der skal undersøges om der er overenstemmelse imellem de forskellige mediciner.

Hvordan registrer behovet for medicin og hvordan registreres det?

Hvis man har mistanke om at der er en ”forkert” medicingivning. Hvis fx. borgeren har fået PN medicin flere gange (dagligt) og over en længere periode, da skal der tages kontakt til sygepljerske eller læge for at få ændret deres medicin i FMK.

Men der registreres og generelle ændringer i forhold til de 12 sygeplejefaglige problemområder og da vil det være muligt at kunne trække oplysninger over en længere periode.

Ind i kontoret, hvad bliver dokumenteret/noteret omkring beboeren?

Gerne alt i forhold til de 12 sygeplejefaglige "problemer" men specielt når der er afvigelser. Desuden så bliver der også noteret omkring "vare indkøb" til borgeren, medicin, stomiremedier, bleer, ernæringsdrikke, kateterremedier, osv. Samt kontakt til pårørende og andre kontakter.

Hvad er formålet med dokumentationen?

Formålet er at vi sørger for den bedste mulige pleje for den enekelte borger og at borgeren ikke er afhængig af at det er den samme person som kommer hver dag samt 24/7

Ekstra noter

Det var mening at vi skulle bruge mindre tid på overlevering fra et vagtlag til et andet og at vi ikke skulle bruge så meget tid på at sidde og skrive i kardex. Men som plejepersonale bruger jeg mere tid på at sidde og dokumentere end egentlig at være sammen med borgerene.

Spørgsmål til medarbejdere på Plejecenter Hotherhaven

Erna Maruf – SSA, Demens afdeling, Fyrtårnet

Hvordan bliver dagens opgaver planlagt? Og hvordan bliver det besluttet hvem der skal gøre hvad?

Cura applikationen bruges til planlægning mellem medarbejderne hvor de uddelegerer forskellige opgaver. Derudover har de en fysisk tavle som viser hvilke personaler kommer og hvornår de har fri, og hvilken afdeling de skal være.

Der er en bliver valgt til at fordele opgaver til personaler, hvor man så spørge personaler hvilke borger de vil tage. Alle personaler har en Ipad hvor de kan markere de borgere de vil have. Arbejdsfordeler markere hvilke personaler skal være anden person til visse borgere (6 borgere). Opgaverne er allerede lavet hvor de bare skal markere hvis de er gjort. Det er arbejdsfordeler der tilføjer ting på opgaveplanen, men der er allerede en automatisk arbejdsplan for en normal arbejdsdag.

Kan du give mig en tidslinje på hvad I laver på en typisk dag?

07.00 – 07.30: morgenmøde
07.30 – 08.00: morgenmad klar
08.00 – 10.30: borgerpleje
10.30 – 12.00:
- Dokumentering
- Borger aktiviteter

12.00: frokost
13.00: toilet besøg
13.30: Dokumentering
14.00:
- Kaffe og aktiviteter

14.45: rapport til aftenvagt

Hvilke opgaver skal laves omhandlende én beboer på en typisk dag?

08.00:
- sårpleje
- nedre hygiejne
- øvre hygiejne
- lette rengøring

08.30:
- medicin givning
- morgenmad (støtte med at spise)

12.00:
- medicin givning
- frokost (støtte med at spise)

13.00
- toilet besøg
- sove til middag

14.00:
- kaffe og kage

Kan du går i dybde i hvordan I tilsyner en beboer (giv et eksempel for en bestemt beboer uden at nævne deres navn)?

Evt. er der nogen opgaver som skal gøres før en anden opgave? (mad før medicin)

En beboer som har en behov for insulin skal have deres blodsukker måles før de skal spise. For at deres blodsukker niveau, og finde ud af om de har brug PN medicin (efter behov).

Tager I noter under tilsynet? Hvis ja, hvordan?

Ja direkte til Ipad, nogle gange bruger vi blok noter. Men som regel dokumenterer vi på fællesstuen efter vi er færdig med vores opgaver.

Er oplysningerne givet af den medicinske historie tilstrækkelige?

Ja, gennem systemet kan man læse om borgerens medicinske historie. Det kan man se under fanen 'Medicin' for de specifikke borger. Desuden kan personalet (assistent eller sygeplejerske) give efter behov medicin gennem systemet.

Hvilke yderligere tekniske værktøjer anvendes under plejen?

Ja, vi bruger også 'varportal.dk' til for eksempel kigger på vejledende guide eller videoer om hvordan man administrerer pleje eller medicin.

Hvordan registrer behovet for medicin og hvordan registreres det?

Fast administration medicin registres ved at markere under opgaver at personalet har givet medicinen. Efter behov medicin dokumenteres når de skal administreres.

Det er lægen der ordinerer medicin, men det er personaler der registrerer og bestille medicin.

Ind i kontoret, hvad bliver dokumenteret/noteret omkring beboeren?

Beboers nye observation som for eksempel afføringsskema, blodsukkermålinger, egenomsorg.
Tidlig opsporing altså om beboers ændret adfærd, hvis ændringer i tilstanden er alvorlige og ikke kan vente til næste triageringsmøde. Fx nye sår, kvalme, et fald, rødt øje, blod i afføring.

Kommunikerer med lægen hvis borgeren har behov for nogle nye PN-medicin, fx Paracetamol.

Hvad er formålet med dokumentationen?

Dokumentation er vigtigt fordi det giver information videre til de andre personaler.
Og fordi det giver et sammenhæng blik på beboerens medicinske historie, og deres helbred. Give information til eksterne personaler som læger og sygeplejerske i en tilfælde hvor beboeren bliver indlagt.

Ekstra noter

Under ikke ordinær situation som når der sker et fald vil tidsplanen på applikationen ikke passer, så det er muligt at for eksempel morgenplejen først er færdig kl. 12 i stedet for 11.

B Preliminary Interview (Danish)

På plejehjemmet, der kombinerer somatisk og demensafdelinger, er arbejdsdagen struktureret omkring de opgaver, som tilsynsmedarbejdere får tildelt om morgenen. Disse opgaver udføres for beboerne afhængigt af deres individuelle behov.

Somatisk afdeling: Denne afdeling tager sig af beboere med begrænset bevægelsesevne. Personalet kan frit bevæge sig ind på denne afdeling efter behov.

Hjemmeplejeopgaver: I modsætning til plejehjemmet kan beboere inden for hjemmeplejen generelt klare sig selv med personlige opgaver som at gå på toilettet og bade. Medarbejderne i hjemmeplejen har primært til opgave at købe ind, dosere medicin og udføre rengøring.

Dokumentationspraksis: Medarbejderne på plejehjemmet noterer normalt ikke under udførelsen af deres opgaver, men skriver rapporter, hvis der observeres ændringer i beboernes sædvanlige adfærd. På kontoret dokumenteres observationer af beboernes afføring samt eventuelle ændringer i deres ernæringsmæssige indtag. Hvis der opdages nye sår, bliver det også noteret som en del af en tidlig opdagelse.

Personalestruktur:

- Kontordamer (Fagkoordinatorer): Ansvarlige for at lave vagtplaner, placere bestillinger og håndtere personaleledelse, herunder udvikling af medarbejdernes kompetencer.
- Assistenter: Har et højere ansvarsniveau med autorisation til at dispensere medicin, håndtere insulin, forberede søndagsmad og udføre behandlinger. De bruger CURA-systemet til at vælge og tilsyne beboere, inklusive en tjekliste for medicinadministration.

Arbejdsfordeling:

- På hverdage bliver medarbejdere på plejehjemmet tildelt 3-4 beboere, og i weekender 4-5 beboere.
- Medarbejdere i hjemmeplejen tager sig af 11-12 beboere.
- Specifikke plejehjem og hjemmepleje systemer inkluderer Nexus i Hørsholm og CURA i Stevns Kommune.
- Normalt betjenes 15 beboere af 5 medarbejdere, men i weekenderne kan dette antal være 15 beboere til 3 medarbejdere.
- Nogle gange er det nødvendigt, at to medarbejdere arbejder sammen om en enkelt beboer.

Plejehjemmet, omtalt i interviewet, er Hotherhaven, hvor personalet udfører disse tilsynsopgaver som en væsentlig del af deres daglige arbejde.

C 6 short interviews throughout the develop process

C.1 Interview 1:

- Kunne godt lide enkeltheden ved login-skærmen.
- Fandt det nemt at vælge borgere fra listen.
- Mente, at opgavelisten var ligetil, men ville foretrække, hvis opgaverne var grupperet efter hold.
- Nævnte, at muligheden for at skrive noter er meget nyttig, siger at de skriver op til 5 sætninger per note.

C.2 Interview 2:

- Task list funktionen fungerer godt, men mangler stadig hold-specifikke opgaver.
- Notefunktionen fungere fint. Man kan måske gøre sådan så man kan se noterne efter man har skrevet dem.
- Spurgte, om det er muligt at markere områder på en patient's krop (fx sårsteder) - dette ville være meget hjælpsomt.

C.3 Interview 3:

- Forslog faner til hurtige sundhedsstatus-tjek på borgere.
- Nævnte behovet for en håndbog i appen til at guide procedurer. En link til en ekstern hjemmeside hvor der allerede er guides til det.

C.4 Interview 4:

- Opgaver nu tilpasset til forskllige vagthold, reducerer forvirring.
- Stadig behov for at kunne se noter fra forskellige datoer.

C.5 Interview 5:

- hold-specifik opgaver, gør arbejdet lettere.
- Admin-system tilføjet, meget positiv feedback på styring af skift og opgaver.

C.6 Interview 6:

- Snakke om at gøre det muligt at slette en borger
- Bemærker, at det visuelle design stadig er meget grundlæggende, foreslår forbedringer.

D Final Interview (Danish)

Interview med en Social- og Sundhedsassistent (SSA om appen)

Interviewer: Kan du fortælle mig lidt om din oplevelse med appen og hvad du synes?

SSA: Lad os starte med loginskærmen. Login-processen var ret enkel. Jeg indtastede mit brugernavn og adgangskode".

Interviewer: Hvad med at vælge borgere?

SSA: Skærmen til at vælge borgere er meget klar. Det gør det nemt at vælge den borger, jeg arbejder med.

Interviewer: Hvad med opgavelisten for borgerne?

SSA: Når jeg vælger en borger, vises tasklist. Opgaverne er nemme at forstå og klikke på. Jeg kan godt lide, at opgaverne er specifikke for skift.

Interviewer: Hvad synes du om note-funktionen?

SSA: Note-funktionen er ret funktionel. Jeg kan nemt skrive noter og uploade dem. Men jeg tænker, at når man skal vælge aktiviteterne så kan det godt gøre bedre ved at for eksempel kategoriser dem eller en søge felt.

Interviewer: Og hvordan synes du det er at se noter?

SSA: At se noter er ligetil. Noterne er listet efter dato og aktivitet, hvilket er godt for at holde styr på observationer og opdateringer. Man kan måske gøre så man kan læse hvem der har lavet noterne.

Interviewer: Alt i alt, hvad synes du så om appen?

SSA: Jeg synes, at appen er meget praktisk og nem at bruge. Enkelheden er en stærk side, fordi det sikrer, at alle hurtigt kan lære at bruge appen uden meget træning. Men det visuelle design er ret fladt. Som i Cura så savner jeg en søgefunktion som skal gøre det nemt for arbejdere at søge efter bebeoere og noter.

Interview med en Social- og Sundhedsassistent (SSA om DCR)

Interviewer: Kan du fortælle mig lidt om din oplevelse med DCR Process Model?

SSA: Jeg kan godt lide, at DCR Process Model viser en klar oversigt over opgaverne. Dog, forstår jeg ikke hvad betydningerne er med pilene.

Interviewer: Kan du prøve at tilføje en ekstra aktivitet i Morgenpleje?

SSA: lad mig se... Jeg tror, jeg skal klikke her... Nej, det var ikke rigtigt. Måske her? Jeg er ikke helt sikker på, hvordan jeg gør det.

Interviewer: Lad mig vise dig det. Først skal du vælge en activity, du kan se at der bliver lavet en ny box. Derfter klikker du på boxen og hold shift inde hvor du så klikker på 'Morgenpleje Pakke'. Nu er den nye aktivitet forbundet med pakken. Du kan nu navngive den her aktivitet, så det svare til hvad aktiviteten er.

SSA: Ah, jeg forstår. Det giver mening nu. Men jeg må indrømme, det kan være lidt svært at huske alle disse trin. Måske med lidt træning, kunne det være at jeg lære det.

Interviewer: Det er helt forståeligt. Med træning bliver det helt sikkert lettere. Er der noget andet, du synes kunne forbedres?

SSA: Generelt synes jeg, det er et godt system. Det er bare et spørgsmål om at vænne sig til det. Måske kunne nogle instruktioner eller en guide i appen hjælpe os med at huske, hvordan man gør forskellige ting.

Interviewer: Det er en god idé. Tak for din feedback.

E A normal workday for a caregiver

En normal arbejdsdag:

- 07.00 - 07.30
  - Rapport fra nattevagten
  - Fordeling af opgaver i personalelokalet
  - En gang om ugen laves der triagering for hele beboere (status for beboere: grøn (normal), gul (ny rutine), rød (særlig opmærksom))
  - Intern kommunikation om beboere, deling af information
- 07.30 - 08.00
  - (Assistent) tjekker bestilte medicin
  - (Hjælper) borddækning til morgenmad i fællesområdet
- 08.00 - 11.00
  - Beboere morgenrutine
- 10.30 (Personalelokalet)
  - Dokumentering af de beboere man havde om morgenen
  - Dispensering af medicin (30 min)
  - (De andre) rydder op efter morgenmad
  - (Hjælper) aktiviteter til beboere
- 11.45
  - Borddækning til frokost
  - Middagsmedicin
- 12.00 - 13.00
  - Frokost (køkkendamer laver frokost)
  - 3 personaler i fællesstuen sammen med 11 beboere
  - 2 personaler hjælper 4 beboere med at spise ind i deres værelse
- 13.00
  - Hjælper beboere med at gå på toilettet
  - Personaler hjælper deres valgte beboere
  - Hvis de ikke skal hjælpe beboere, rydder de op efter frokost
- 13.30
  - Beboere går til deres værelse
  - Dokumentering
  - Dispensering af medicin
- 14.00
  - Kaffe og kage (ikke samme antal af personaler da der nogle få af dem tager hjem kl 13)
- 14.45 (Personalelokalet)

  - Rapport til aftenholdet
- 15.00
  - Fri

Beboere morgenrutine:

1. 08.00 (Morgenpleje) (0-4: 2) (vejledning eller fuld hjælp)
   - Sårpleje eller observation
   - Nedre hygiejne (vaske underlivet, skifter ble)
   - Skifter tøj
   - Børste tænder
   - Barbering
   - Rede/skifte sengetøj
   - Lette rengøring (toilet, fejning)
   - En gang om måneden vægtes beboeren
   - Går sammen til fællesstuen (ved hjælp af en rolator)
2. 08.30 - 08.45 (Fællesstue)
   - Sidder på de siddepladser de pleje at sidde på.
   - Morgenmedicin (Med specifikke behov: mix med yoghurt) (kvitteres)
   - Morgenmad (0-4: 2) hjælpe med at smørre mad
3. 08.45 (Morgenpleje) (0-4: 1) (kun vejledning)
   - Morgenplejepakke
   - Går sammen til fællesstuen (går alene)
4. 09.15 (Fællesstue)
   - Morgenmedicin (kvitteres)
   - Morgenmad (0-4: 1) vejledes med at spise
5. 09.30 (Assistere som anden person - sengeliggende beboere) (0-4: 4)
   - (Beboeren har fået medicin og morgenmad af første personen)
   - Nedre hygiejne (vaske og tørre) (Assistent giver sårpleje)
   - Hjælp med at flytte beboeren med at sidde på kørestolen (med loft lift)
   - Skifter tøj
   - (Første person giver øvre pleje)
   - Lette rengøring
   - Rede sengen
6. 10.00
   - (Første personen bliver på værelset med beboeren (TV, musik, radio))
7. 10.00 (Assistere som anden person - aggressivt demens) (0-4: ?)
   - (Fysisk mæssig kan nogen beboere stadig meget, men deres kognitivt tilstand er meget nedsat)
   - (Beboeren kan godt finde ud af at vandre rundt på området)
   - (Hvis beboeren er ud for sit værelse, bringes de tilbage på værelset)
   - Morgenplejepakke

F Demonstration of the app

F.1 Demonstration of a morning shift

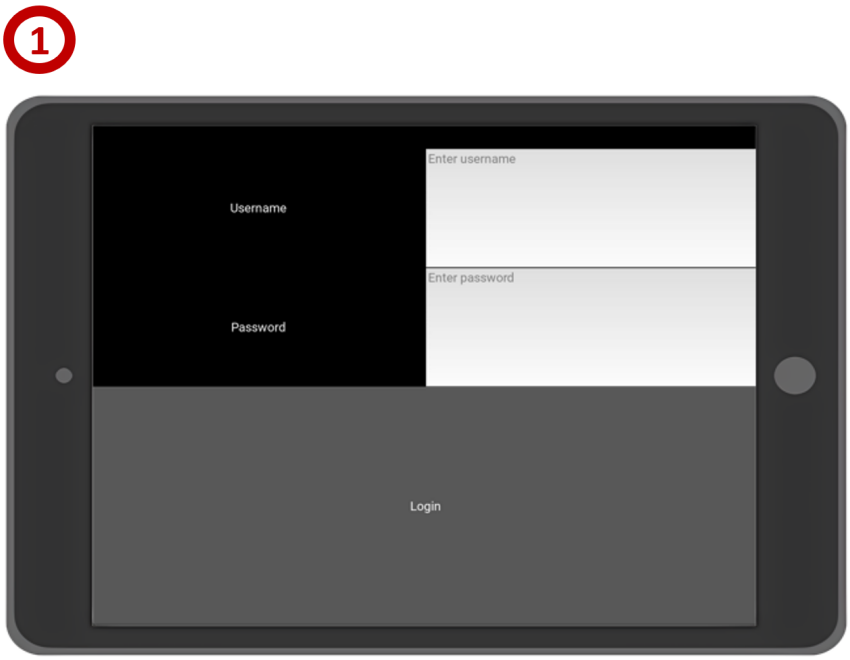


Fig. 18: Log in with the correct email and password.

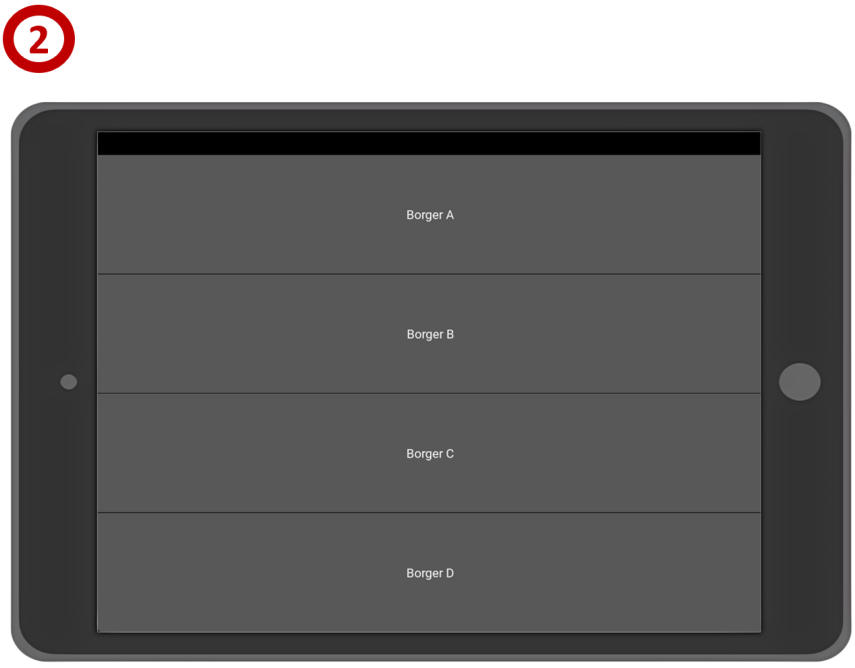


Fig. 19: Select a resident.

Fig. 20: App switch to task screen. Select the first task.

Fig. 21: Name of a task group in blue. Required tasks are in yellow. Optional tasks in white.

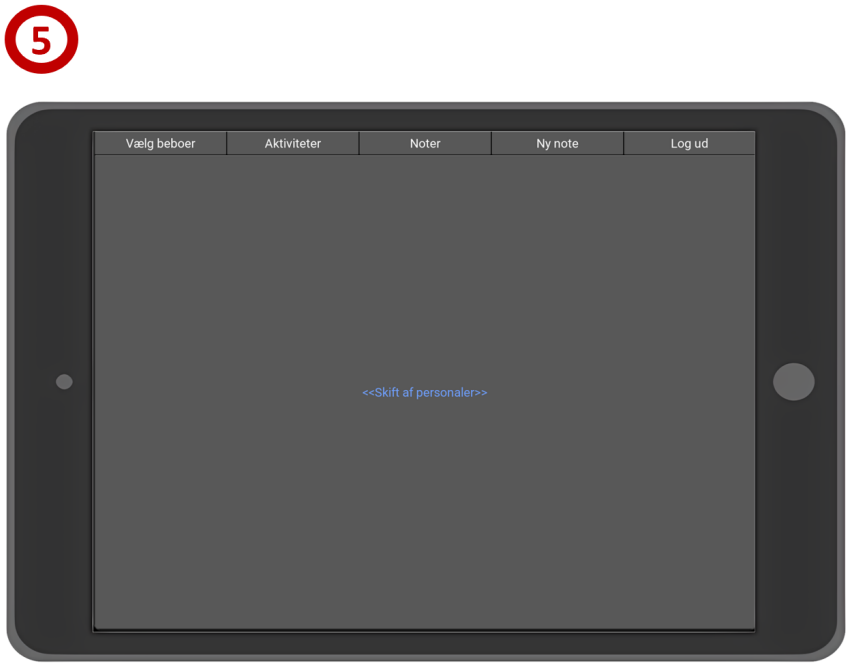


Fig. 22: Exhaust all tasks. Log out.

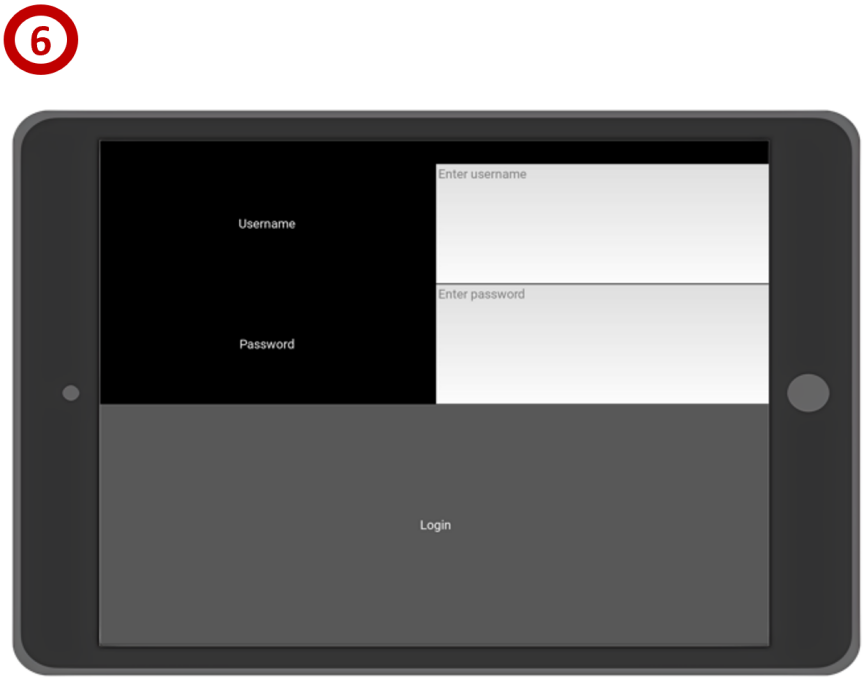


Fig. 23: Back to login screen.

F.2 Showcase of the note functionality

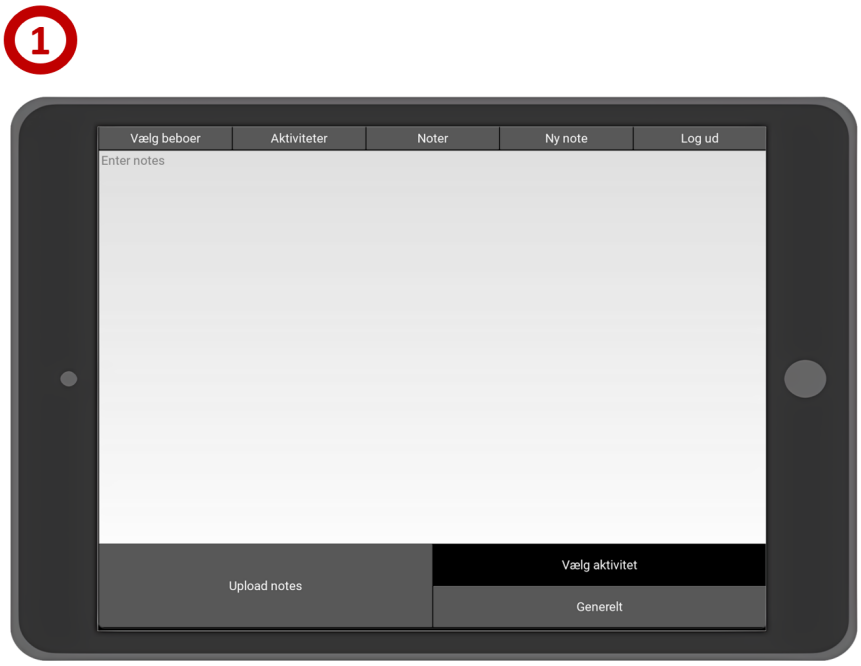


Fig. 24: In the new note screen the user can write a note for assigned activities.

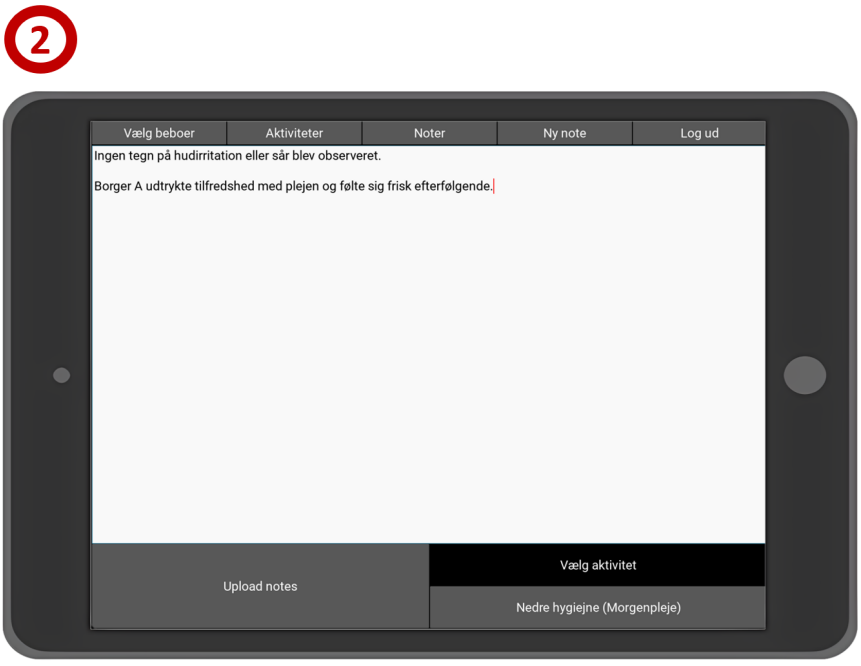


Fig. 25: By clicking on the activity button in the bottom right corner a drop-down menu will show all the available activities. By choosing one of the activities the uploaded note will have a tag for that activity.

Fig. 26: In the notes screen the uploaded note can be seen, other than the actual text note itself, it also contains information about when it was created, who made the note, and which activity is assigned to it.

F.3 Showcase of the admin system

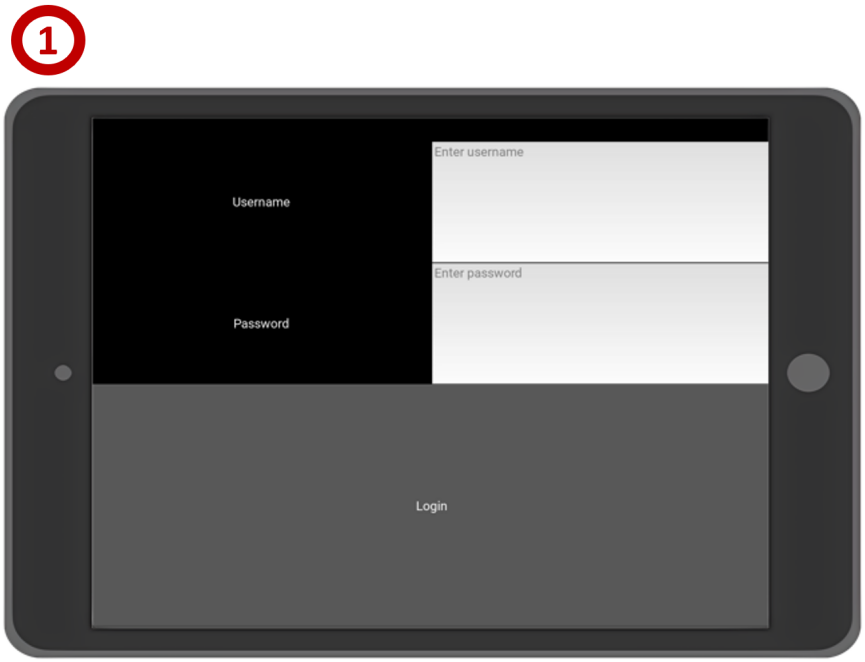


Fig. 27: In the login screen, if the user is an admin the user can access the admin system by inputting the appropriate email and password.

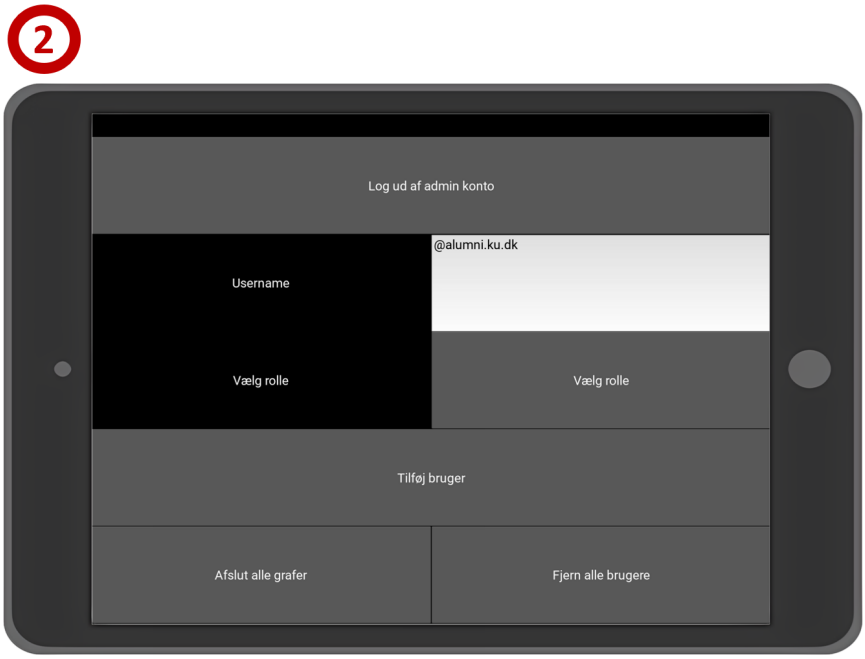


Fig. 28: In the admin screen the user can add new users to the database, allowing them to use the task list application. It can terminate all current active processes, and delete all users from the database.

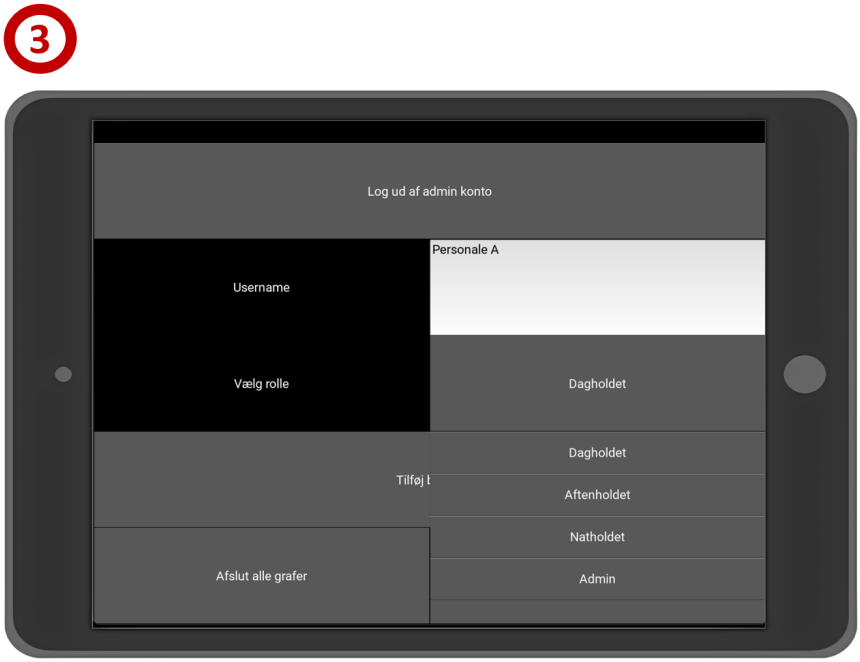


Fig. 29: A drop-down menu will show all the roles that can be assigned to the user. The admin system allows the same user to have multiple roles.

G DCR Graphs used for the demonstration of the application

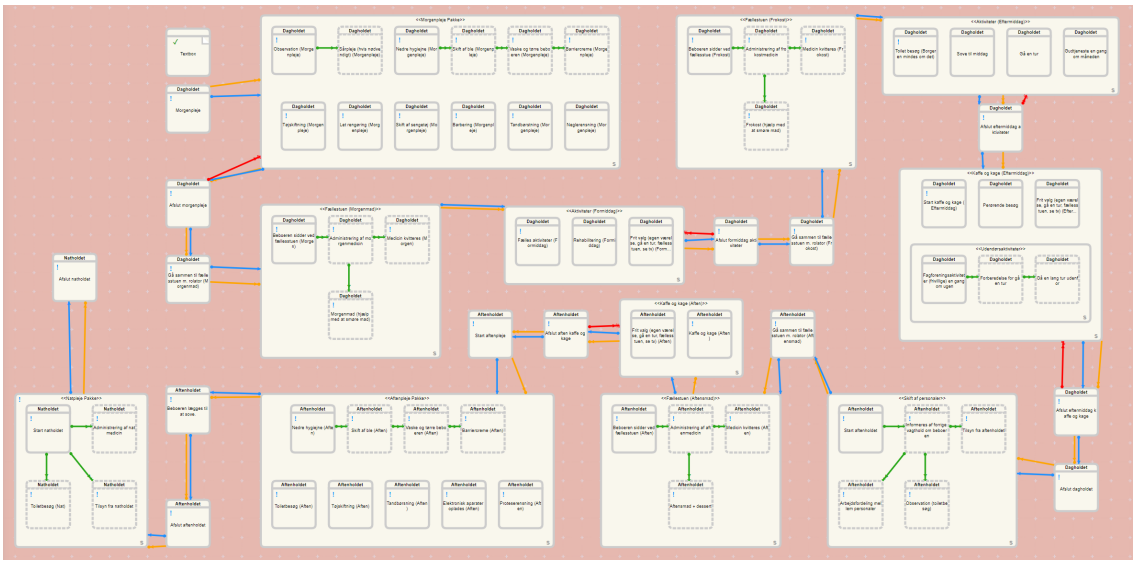


Fig. 30: DCR Graph for "Borger A".

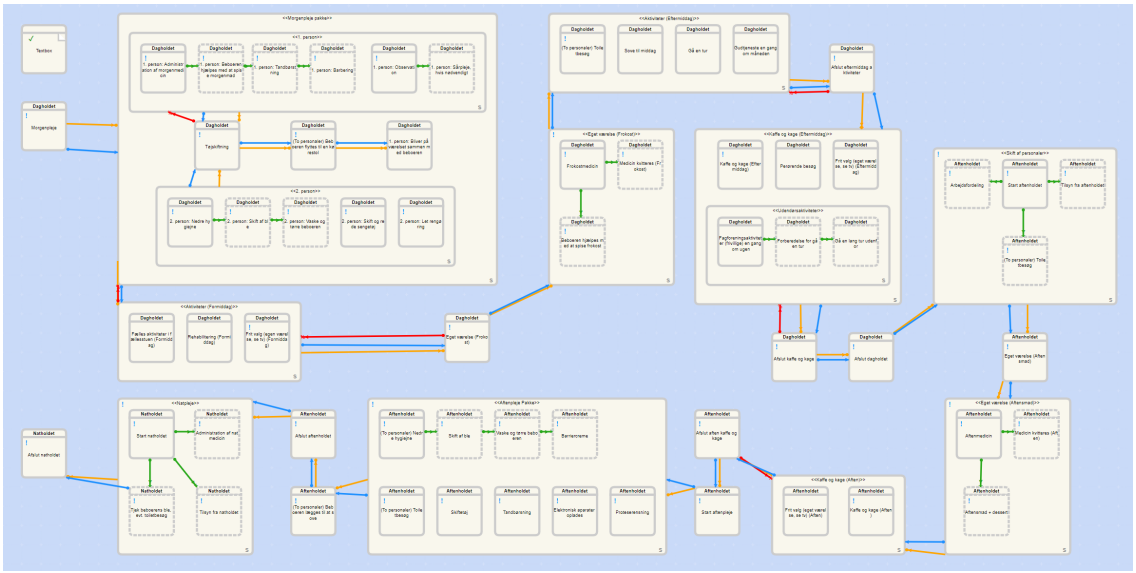


Fig. 31: DCR Graph for "Borger B".

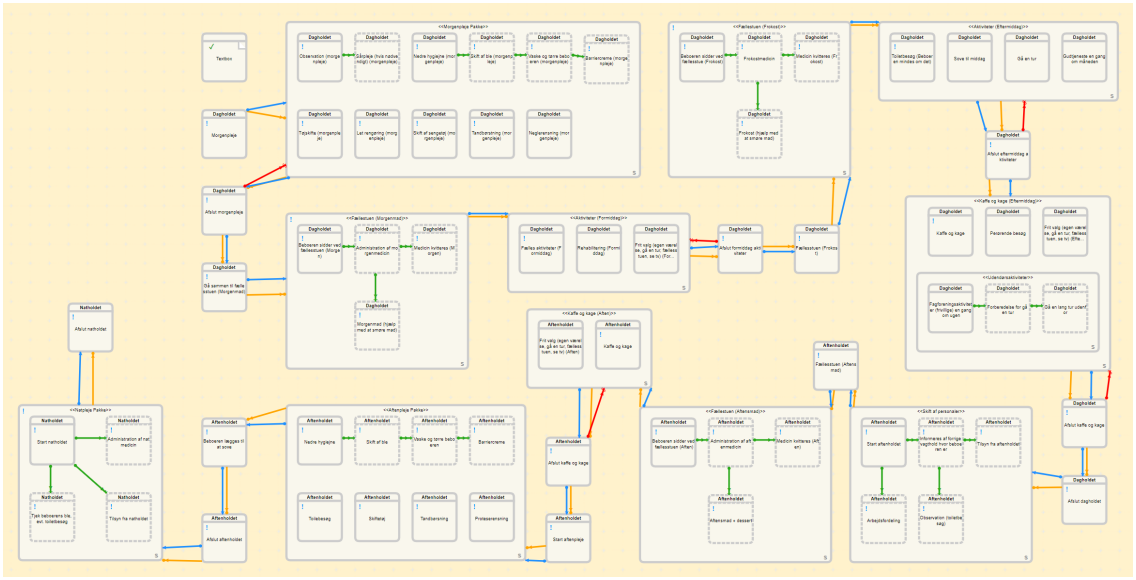


Fig. 32: DCR Graph for "Borger C".

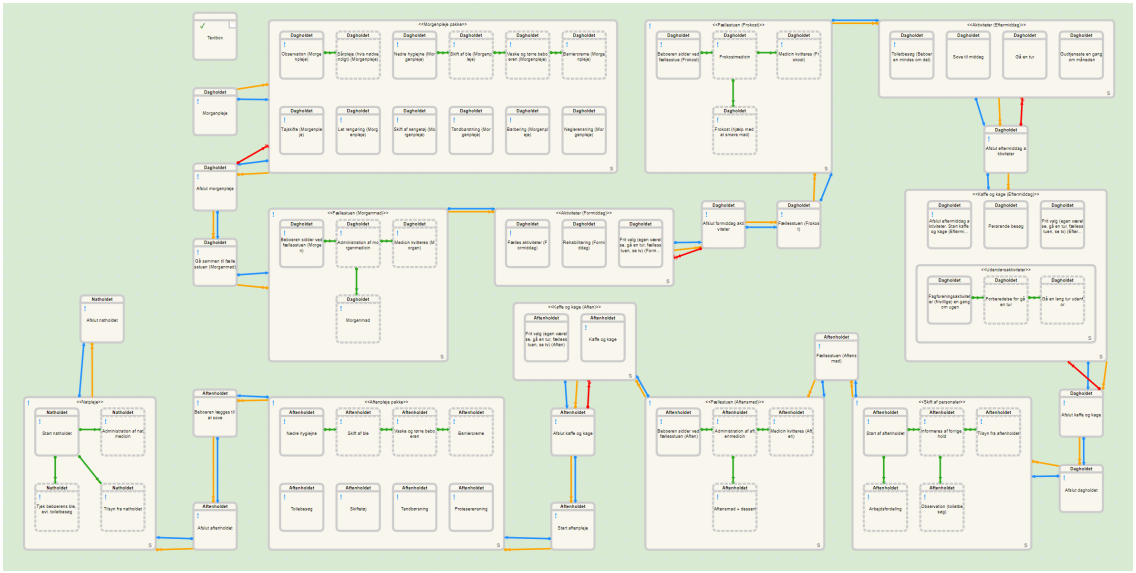


Fig. 33: DCR Graph for "Borger D".

H The functions that was tested with unit tests

1. hashData
2. getEnabledEvents
3. stopReloads
4. dbQuery
5. SimulationButton.__init__
6. PatientButton.__init__
7. PatientButton.choosePatient
8. MainApp.addUser
9. MainApp.deleteAllUsers
10. MainApp.createNotesFields
11. MainApp.calculateNumLines
12. MainApp.getGraphTitle
13. MainApp.forceTerminateAdmin
14. MainApp.getRole

I The functions that couldn't be tested by unit tests

The following functions could not be tested because they are pertained to visual elements.

1. createButtonsOfEnabledEvents
2. MainApp.build
3. MainApp.topBar
4. MainApp.loginScreen
5. MainApp.adminScreen
6. MainApp.login
7. MainApp.hideTopBar
8. MainApp.showTopBar
9. MainApp.eventsScreen
10. MainApp.showNotesScreen
11. MainApp.showNotesLayout
12. MainApp.choosePatientScreen
13. MainApp.writeNotesScreen
14. MainApp.addActivityToNotes
15. MainApp.cleanScreen

The following could not be tested as they require graphid and password/username which could not be given to the test because of the way the code has been written. Or another reason.

1. SimulationButton.executeEvent
2. MainApp.startNewSim
3. MainApp.uploadNote
4. MainApp.getNotes
5. MainApp.clearNotes
6. MainApp.connectToSim - it does not return anything which makes it impossible to write unit tests for.