Title of the Project: Microsoft Power Apps for NHS

Year of Submission: 2023

Exam Candidate Code: YMRY2

Name of Supervisor: Kouadri Mostefaoui Ghita

Degree Programme: MSc Computer Science

Abstract

Objective and Challenges: This project centers on the development of three essential applications tailored to address prevailing challenges within the NHS system. The primary objectives include streamlining hospital bed management, refining task management for social care, and optimizing shift management for healthcare organizations. Amongst the major challenges, the UK hospitals grapple with an intricate bed management system, social care workers face the onus of repetitive data entry tasks, and healthcare institutions wrestle with inefficient staff redeployment and shift management.

Methodology and Execution: Our approach for each application commenced with a thorough analysis of the requirements. Post-analysis, wireframes were developed which subsequently transitioned into Figma prototypes. These prototypes were presented to the client to ensure alignment with their needs. After incorporating client feedback, ER diagrams were created, and a Dataverse database equipped with dummy data was set up. The final stage involved building the applications in Microsoft Power Apps and harnessing the capabilities of Power Automate to streamline and automate task flows. After rigorous testing through both Test Studio and manual methods, we showcased the final Apps to the client for their closing feedback.

Results and Current Progress:

- Bed Management App: Aims to overhaul the bed management system by offering real-time updates on bed availability and providing an intuitive UI for bed and patient status. This application is finalized.
- Social Care Task Management App: Designed to curtail the time social care workers spend on redundant data entry tasks. It facilitates task management within units, records essential details, and automates task setups. This application is finalized.
- E-Roastering App: Seeks to reform shift management by ensuring an optimal staff presence on-site, computing maximum work hours, and incorporating a clock-in/out feature within the application. This application is finalized.

Table of Contents

1. Introduction *5*

1.1. Overview **5**

1.2. Aims and Goals: **6**

1.3. Project Execution: **6**

1.4. Report Structure **7**

2. Context *8*

2.1. Similar Solutions **8**

2.2. Tools **9**

2.3. Additional Research **9**

3. Requirements and Analysis *11*

3.1. Bed Management App **11**

3.2. Social Care Task Management App **13**

3.3. E-Rostering App **16**

4. Design and Implementation *21*

1. Social Care Task Management App **21**

2. E-Rostering App **21**

2.1. Database 21

2.2. E-Rostering – Employees 21

5. Testing *30*

5.1. Unit and Integration Testing **30**

5.2. Responsive Design Testing **31**

5.3. Browser Compatibility Testing **31**

5.4. Stress Testing **32**

5.5. User Acceptance Testing **32**

5.6. Client **33**

6. Conclusions and Project Evaluation *34*

6.1. Summary of Achievements **34**

6.2. Critical Evaluation **35**

6.2.1. Functionality 35

6.3.1. User Interface and User Experience Design 35

6.3. Future Work **37**

List of References *38*

Appendices *39*

1. System Manual **39**

2. User Manual **39**

3. Supporting Documentation and Diagrams **40**

3.1.Bed Management App Wireframes 40

3.2.Bed Management Figma Prototype 43

3.3.Social Care Task Management App Wireframe....45
3.4.Social Care Task Management App Figma....47
3.5.E-Rostering App Employee Use Case Specifications....51
3.6.E-Rostering App Employees Wireframe....55
3.7.E-Rostering App Employees Figma....57
3. Test Results and Test Reports....61
4. Code Listing – E-Rostering Employees....70

1. Introduction

1.1. Overview

In recent years, the healthcare sector, particularly the National Health Service (1) of England, has confronted unprecedented challenges. With hospitals grappling with their longest ever elective waiting lists and a rapid rise in inpatients due to the pandemic, efficient management systems have become more essential than ever before. These systems need to be agile and robust, ensuring they can adapt to changing patient needs swiftly. However, the pre-configured legacy systems currently in place are proving inadequate, struggling to keep up with the fluid requirements of modern healthcare (2).

Our collaboration with Stephanie Stasey, Principal CSAM at Microsoft who boasts nearly 20 years in Digital Transformation and significant achievements such as the NHS App and the #NHSEmpower initiative, has led to the development of three applications tailored to address the most pressing concerns:

1. Power Platform Solution for Bed Management: With bed management being an essential aspect of hospital operations, there's a pressing need for real-time data about patient locations, transfer information, and hospital capacity. The pandemic has intensified this need, revealing the inadequacy of traditional systems to cope with evolving patient demands (3)

2. Power Platform Solution for Social Care Task Management: Handwritten logs and data re-entry are consuming up to 25% of frontline care workers' weekly hours, diverting valuable time from actual care provision (4) Coupled with the fact that many are juggling multiple roles within the sector (5), it becomes evident that an efficient digital solution is not just preferable but imperative.

3. Power Platform Solution for eRostering: Efficient scheduling of healthcare personnel, matching skills with patient needs, and managing logistics can lead to significant savings in time and resources. Utilizing scheduling solutions could reduce travel time, potentially allowing for a saving of 34 minutes per clinician each week and enabling them to attend to more patients (6)

This project aims not just to replace outdated systems but to revolutionize the way the NHS operates, ensuring they're better equipped to serve patients in today's challenging environment.

1.2. Aims and Goals:

Aims:

- Enhance proficiency in full-stack development, thereby attaining a holistic understanding of both front-end and back-end aspects.
- Dive deep into the intricacies of database development and explore ways to seamlessly connect it with user interfaces.
- Foster an environment of rigorous testing, ensuring the robustness and efficiency of the applications.

Goals:

- Deliver the *Bed Management App* to overhaul the current bed allocation procedures and present real-time updates on availability.
- Finalize the *Social Care Task Management App*, aimed at simplifying task management and minimizing repetitive data entry.
- Implement the *E-Roastering App* to transform shift management, ensuring optimal staff allocation and maximizing productivity.

1.3. Project Execution:

The methodology adopted for this project was iterative. It began with an in-depth analysis of requirements, followed by the creation of wireframes and prototypes in Figma. Client feedback played a pivotal role in refining these prototypes. After integrating this feedback, we embarked on the journey of developing the ER diagrams and setting up a database in Dataverse. However, this journey was not devoid of challenges.

The project's execution was the result of the collaborative effort of a four-student team, of which I was a part. We maintained a rigorous schedule, holding weekly sprint meetings amongst ourselves, weekly interactions with our supervisor, and consistent sessions with our client. To streamline our processes, two central shared files were maintained: one detailing client meeting questions and responses, and the other outlining our weekly tasks.

Our work spanned three months, with each month dedicated to different stages of the app development.

- In June, our focus was on app 1, working on its requirements, prototypes, and the initial building phase on Power Apps.
- July saw the culmination of app 1 and the commencement of app 2, with emphasis on its prototypes and the Dataverse database creation.
- By August, we had successfully navigated through app 2's completion and initiated app 3.

I undertook specific roles during this period:

- I collaborated with Judy on creating app 1's Figma prototypes.
- I independently managed the wireframe creation for app 2.

- For app 2, I single-handedly designed the ER diagram and set up the database within Dataverse.
- In app 3's development phase, I provided feedback and modifications on the ER diagram initially crafted by Leeyuu and later implemented those changes in the Dataverse database.
- I was solely responsible for app 3's mobile version on Power Apps and its subsequent testing.

1.4. Report Structure

This report, while providing a comprehensive view of the project's scope across all three applications, will focus on areas where my contribution was key.

Chapter 2: Context Within this chapter, the reader is introduced to the necessary background that sets the stage for our project. We'll delve into the significance of our chosen tools and how our initiative aligns with existing solutions.

Chapter 3: Requirements and Analysis This chapter offers a deep dive into each application, breaking down its problem statement, Moscow requirements, use cases, and visual blueprints through wireframes, Figma prototypes, and ER diagrams.

Chapter 4: Design and Implementation In this chapter, we unravel the design intricacies of the project. From discussing the database and its integration in Dataverse to the broader design aspects of the app, we detail the application's architecture and the components it comprises. While we discuss certain vital implementation points in the main content, a full exposition of the code is provided in the appendix for those seeking a deeper technical understanding.

Chapter 5: Testing Our testing approach unfolds in this section. We elaborate on our multi-faceted strategy, from unit and integration tests in Test Studio to manual, stress, user acceptance, and responsive tests. The chapter also highlights unique test cases, offering a snapshot of test results and coverage with detailed reports in the appendix.

Chapter 6: Conclusions and Project Evaluation Concluding the report, we review the project's achievements, ensuring they resonate with the goals set out initially. A critical analysis of the project ensues, reflecting on its objectives, applicability, and overall design. We also muse over potential future expansions, setting a clear distinction between current accomplishments and potential next steps.

2. Context

2.1. Similar Solutions

The modern healthcare environment is rapidly evolving, particularly in the realm of task management and collaboration. Over the years, multiple software solutions, as well as manual methods, have come into play, each addressing unique challenges that professionals in the healthcare domain face.

One such solution is **Medics BedManager**, which I came across in a 2020 YouTube video titled "Intelligent Bed Management Software". This solution informed the design of our Figma prototype. It served as a benchmark, showcasing how intelligent software could revolutionize bed management, ultimately ensuring that patients receive timely care (9).

CareTasker is another noteworthy solution, tailored for social care professionals. It stands as a task management tool, but with a specific caveat: it lacks automation capabilities and the feature to generate shift reports. This pointed out a niche where our solution could provide added value (10).

Microsoft, a juggernaut in the tech industry, has also ventured into tools that cater to healthcare professionals. As per Emma Williams, Corporate Vice President of Viva Studios, healthcare staff necessitate tools that bolster collaborative workflows, especially in challenging times like the COVID-19 pandemic (11). Microsoft's solution to this is a suite of tools within Microsoft 365 tailored for care teams. This includes **Scheduled Virtual Visits** where providers and patients can engage without the need for in-person visits, and **Targeted Communication** in Teams, enhancing connectivity and reducing wasted time searching for contact details of healthcare staff (11).

Another Microsoft solution is **Shifts** in Microsoft Teams. This tool, as explained in a 2020 YouTube video, allows managers and employees to effectively manage schedules. Managers can relay messages swiftly, update shift schedules, and share important documents, while employees can view their upcoming shifts, seek shift changes, and request off days (12). The emphasis here is on keeping Firstline Workers connected and synchronized, highlighting the importance of streamlined communication in healthcare.

Our solution draws inspiration from these existing tools, aiming to integrate the best features and overcome their limitations, providing a comprehensive, efficient, and user-friendly platform for the healthcare sector.

2.2. Tools

Our choices in tools were rooted in both their capabilities and the evident benefits reflected in industry research. Below is an in-depth analysis of each:

• PowerApps: PowerApps was pivotal for our project because of its low-code application development capabilities. This allowed us to swiftly prototype and develop our solutions. As highlighted by The Total Economic Impact of Power Apps by Forrester Consulting, organizations can experience up to a 74% reduction in app development cost using PowerApps (7). This reduction not only streamlines processes but also ensures that projects remain cost-efficient.

• Power Automate: To further enhance our solutions' efficiency, we integrated Power Automate. This tool was crucial for automation, especially in the generation of task flows, which significantly reduced the need for manual intervention and subsequent human errors.

• Dataverse: Our choice for the primary data source was Dataverse. The reason for this choice extended beyond its user-friendly interface. It provided us with the capability to manage and organize data with superior ease compared to traditional databases. In a landscape where data-driven decisions are paramount, Dataverse's robustness and flexibility were unparalleled.

Our decision to utilize Microsoft's Power Platform was further reinforced by McKinsey's Developer Velocity Research. According to their findings, organizations that adopted such platforms noticed 4-5 times higher revenue growth and a 55% surge in innovation (8). The Power Platform allows organizations to modernize existing applications, develop new intelligent cloud-native applications, scale using Fusion Teams, and standardize processes with DevOps. Our aim was to harness these benefits to provide NHS with state-of-the-art solutions.

2.3. Additional Research

Engaging directly with our clients provided invaluable insights that greatly enriched our project's direction. Through these interactions, the requirements for the apps were elucidated with greater clarity. More importantly, the client delineated distinct user personas, enabling us to tailor the app functionalities to suit specific user needs. Furthermore, to facilitate practical design and testing, the client shared dummy data for all the apps, offering a tangible perspective on the necessary records. These collaborative sessions were instrumental in bridging any understanding gaps and ensuring our solutions were both relevant and efficient.

For the first app, the client provided us dummy data for 2 of the tables in the database, admissions and bed inventory. The data had the structure below.

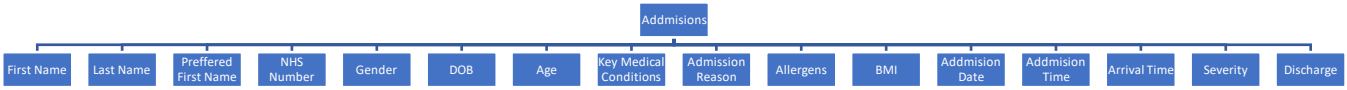


Figure 1 – Admissions Dummy Data

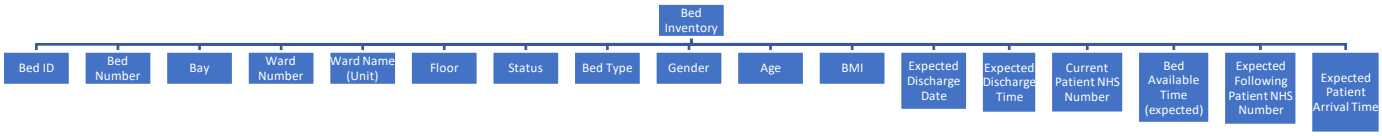


Figure 2 – Bed Dummy Data

For the second app, the client provided us with a table for the tasks table in the database, to help us better understand its format. It has the variables listed below.

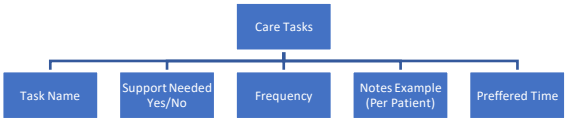


Figure 3 – Care Tasks Dummy Data

For the third app, the client offered us some dummy data on the care roles, with the following structure:

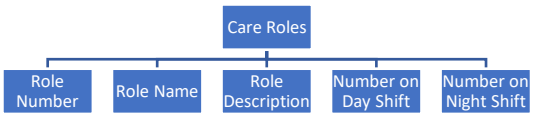


Figure 4 – Care Roles Dummy Data

3. Requirements and Analysis

3.1. Bed Management App

Problem Statement

In the vast and intricate healthcare system of the NHS, the challenge of bed management has become increasingly pronounced. Efficient utilization of beds within a hospital isn't simply about ensuring a physical space for every patient. It's centered on the principle that beds should be occupied for the shortest necessary duration, be swiftly ready for the next patient, and appropriately matched to the patient's specific care requirements.
The dynamic nature of a hospital means that patient statuses, ranging from admissions to transfers and discharges, can change rapidly. It becomes imperative to have access to real-time data on patient locations, ensuring that errors like double bookings are avoided, rooms are promptly cleaned and prepared, and patient wait times are minimized.

Adding another layer of complexity, knowledge about patient transfers in real-time can substantially streamline logistics, ensuring transfers don't inadvertently result in bed shortages. Further, having immediate insights into current bed capacities, complemented by predictive data on expected discharges, becomes crucial. Such data can significantly aid in capacity planning, reduce wait times for incoming patients, and assist medical teams in optimally scheduling procedures.

Recent data highlighted by the NHS in December 2022 showcased the escalating problem. Hospitals now face the longest elective waiting lists on record. This surge in demand is, in large part, the aftermath of the pandemic, which saw an unprecedented rise in inpatient numbers. The rapidity of this influx is putting a strain on hospitals, making it immensely challenging to quickly reconfigure bays and beds.

Compounding this challenge is the reliance of many hospitals on pre-configured legacy systems for managing beds. While these systems might have been adequate in more predictable times, they have shown to be ill-equipped to handle the rapidly changing needs of today's patients. These systems' limitations in flexibility, adaptability, and real-time features aren't just technical issues. They have a direct and tangible impact on patient care, overall hospital efficiency, and the satisfaction of both patients and healthcare workers.

Addressing this need effectively and providing a robust solution to these challenges is the driving force behind the development of the Bed Management App.

Moscow Requirements

The requirements were build starting from the clients' initial requirements and adjusted based on feedback on prototypes. This Figure was created by Hannah and Leeyu.

ID	Functional requirements	Priority
R1	Bed management teams must be able to make bed closures and openings	M
R2	Nurses and bed managers must be able to mark patient severity in real-time	M
R3	Allocation of beds and bays must be simple	M
R4	Bed status must be real-time tracking (updated every 5 minutes or less)	M
R5	RFID and air tag must be applied to track live bed states (updated every 5 minutes or less)	M
R6	When patiens discharging, tasks of cleaning must be trigger	M
R7	When patiens transfering, tasks of portering must be trigger	M
R8	Bed and capacity management must be contained in the app	M
R9	App could contain Care Control Centre view	C
R10	App could contain Integration into Cerner EPR (using FHIR)	C

Figure 5 -Bed App Moscow Requirements

Wireframes

From the requirements, four versions of wireframes were created, updated based on client's feedback. See the final version in *Appendices 3.1.* The wireframes were created by Hannah and Leeyu.

Figma Prototype

From the wireframes, a Figma Prototype was created. See *Appendices 3.2.* I have worked on the prototype in collaboration with Judy. (about 60% of the work).

3.2. Social Care Task Management App

Problem Statement

The urgency and criticality of the problem confronting the social care sector in the UK is underscored by the extensive time drain experienced by frontline care workers and line managers. Currently, up to a quarter of their workweek is monopolized by data entry, a situation only aggravated by the persistent reliance on handwritten logs and the consequent re-entry of this data into outdated computer systems (4),The primary concern here is the palpable diversion of time and attention from actual caregiving tasks, an alarming scenario given the evolving demands of modern care.

The contemporary social care landscape demands a holistic approach. Care recipients, in their interaction with the system, anticipate a structure that prioritizes their well-being, focusing on personalized, proactive, and preventative care. The expectation leans heavily towards an efficient, fair, community-based, and integrated system, as outlined in the "Time to care" report (13). This mandates a significant overhaul in the way tasks are recorded and processed.

Furthermore, the multi-faceted roles many individuals juggle within the sector [Skills for Care 2014] further emphasize the need for an efficient digital solution. This not only optimizes their workload but ensures they can deliver the personalized, co-created, and outcome-driven care the modern recipient expects.

Considering the common care tasks such as recording food consumption, medicines administered, and well-being of the person supported, it's clear that the current manual recording system is inadequate. For instance, recording minute details like fluid consumption, the willingness of a care recipient to engage in activities, or even their demeanor requires precision and time. The current handwritten logs, compounded by the need for re-entry into digital systems, makes this process cumbersome, error-prone, and time-intensive.

In light of these identified problems, the proposed solution aims to revolutionize the process. The solution will leverage the capabilities of the Power Platform, targeting the efficient digitization of these tasks within Microsoft Teams. By integrating automation for tasks, enabling user-friendly recording methods, and offering features like incident reporting and shift handovers, it seeks to streamline the task management process. Additional functionalities, if time permits, will further enhance the system, from monitoring dashboards to in-depth analytics on care recording.

Moscow Requirements

The requirements were build starting from the clients' initial requirements.

Must have:

1. Utilise Tasks/To do application within Microsoft Teams.
2. Or build or repurpose a Power App within Microsoft Teams, as suggested by the example from GitHub - Microsoft/NurseEmpowerment.
3. Integration and automation for daily, weekly, and monthly tasks from a list.
4. Enable recording for employees with limited command of written English – utilising standard icon libraries where possible.
5. Enable recording of employee name, date, and timestamp on tasks.
6. Enable recording of tasks via the app on mobile phones (Teams app or Power App).
7. Enable recording of willingness/refusal of care.
8. Enable refusal reason from a list with mandatory selection if an option is selected and free text with a 20 character minimum.
9. Enable picker of weekly, monthly tasks as well as mandatory daily tasks.
10. Create an incident reporting option within the application.
11. Create a summary of care at shift handover for all people within the ward/round highlighting areas of concern, worrying behaviors, and any incidents.

Should have (time permitting):

1. Monitoring dashboard for each location, and alerts for supported individuals who refuse to medicate, eat, take fluids, or show other triggers for managers to address.
2. Automate storing the daily handover for the unit.
3. Automate storing the daily record for each supported individual and monthly summary.
4. Enable recording via speech where feasible.
5. Produce well-being graphs, trends, and analyze patterns.
6. Analyze timeliness of care recording.
7. Analyze trends and patterns in incidents, such as day of the week, time of day, or other factors that can aid in care improvement.

Could have:

1. Utilise language translation, allowing employees to record notes in their native language and translate these to English, while also retaining a copy in the original language for reference.
2. Write back data over 6 weeks old to Azure storage to optimize data management.

User Personas

The target user are the hospital employees.

Wireframes

I have created the app wireframe based on the client's description of requirements. See *Appendices 3.3.*

Figma Prototype

From the wireframe and requirements, a Figma Prototype was created by Judy. See *Appendices 3.4.*

ER Diagram

Based on the final version of the Figma Prototype, I have created the ER Diagram that would support the app.

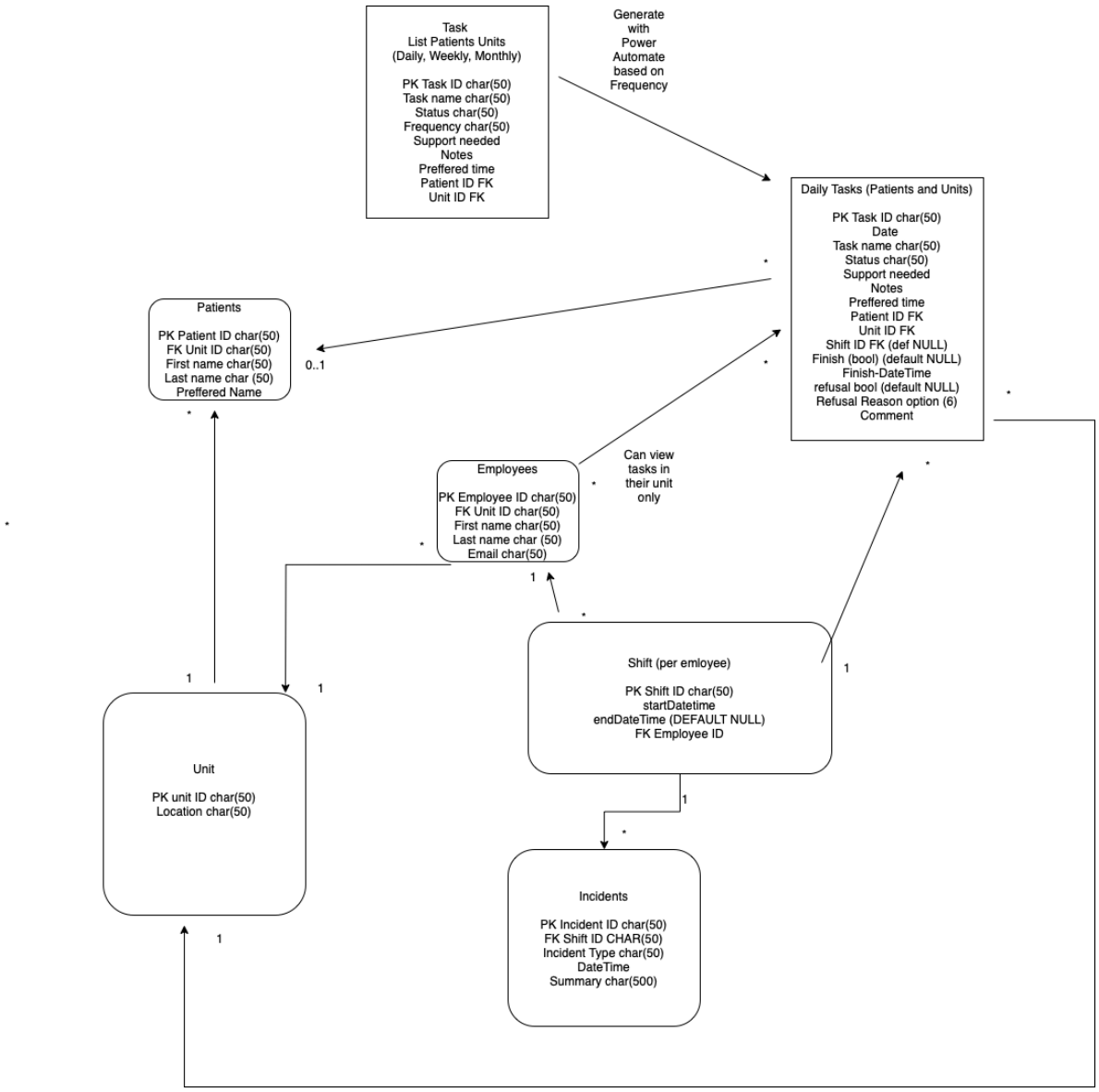


Figure 6 – Social Care ER Diagram

3.3. E-Rostering App

Problem Statement

The intricacies of healthcare operations, particularly in the scheduling and deployment of clinical staff, have long presented a myriad of challenges. The complexities arise from the necessity to marry the diverse skill sets of healthcare personnel with specific patient needs, all while ensuring optimal logistics. It is imperative to understand the profound ramifications of inefficient rostering: not only does it stifle optimal patient care, but it also levies an undue burden on healthcare professionals, affecting their productivity and well-being.

The premise established in the introduction – that harnessing the capabilities of eRostering can engender significant savings in both time and resources – underscores the urgency to pivot towards a digital solution. Data from client research suggests that eRostering can play a pivotal role in efficiently redeploying staff, all the while ensuring that patient care requirements are precisely matched with the right clinical skill sets. Additionally, the strategic geographic matching facilitated by advanced rostering systems could yield a reduction in travel time by between 20% to 30%. This, in essence, could translate to a saving of approximately 34 minutes per clinician every week, thereby allowing them to cater to more patients and optimizing patient turnaround (6)

However, the benefits of eRostering are not confined to operational efficiencies alone. A more profound, often overlooked, advantage is its potential to bolster the qualitative aspects of healthcare. eRostering can significantly improve patient safety and care quality, serving as a catalyst for better staff recruitment and retention strategies. By empowering clinicians to influence services directly, it can usher in a marked improvement in morale, alleviate stress, and can substantially reduce dependencies on external staffing agencies (1)

Considering the gravity of these challenges and the palpable advantages of an eRostering solution, the project delineates a set of critical requirements. At its core, the solution necessitates the creation of a shift booking application embedded within the Microsoft Power Platform, ensuring seamless integration with Microsoft Shifts. This would empower employees with a comprehensive view of their roster, while also offering functionalities like clocking on/off shifts. To enhance flexibility and adaptability, the solution mandates features like a rolling roster pattern, roster draft and publish options, quick shift swaps, and an ability to log approved holidays, non-working hours, and contracted hours for every employee.

Moscow Requirements

The requirements were build starting from the clients' initial requirements and edited in the following client sessions.

Must have:

1. Shift booking application built on Microsoft Power Platform.
2. Employees can view roster (shifts).
3. Employees can clock on/clock off a shift using NFC tags.
4. Enable rolling roster pattern of every 2, 4, or 6-week repeat.
5. Enable roster minimum service levels by role for five different roles per unit.
6. Enable roster red, amber, and green flag to show if minimum staffing level has been met, ensuring safe working hours and an appropriate skill mix.
7. Enable roster draft and publish options.
8. Enable quick shift swap changes.
9. Ability to hold table of approved holidays.
10. Ability to hold table of non-working hours. (availability times per employee)
11. Ability to hold table of contracted hours per employee.
12. Ability to roster training and meeting time. (Activities)
13. Ability to request time off.

Should have (time permitting):

1. Live monitoring dashboard for hospital/ICB of employees clocked in/out in each location, and alert for locations under minimum staffing level.
2. Ability to roster community services by location rooting (home visits) with shortest travel distances.
3. Ability to roster 24/7 services.
4. Ability to re-assign employees from other departments, but hold the preferred department on the employee table.
5. Suitability criteria for employees for each unit held in a table.
6. Enable rolling roster pattern to suggest alternate employees to cover a shift based on the table of suitability criteria for the shift.
7. High, medium, and low-priority staffing units flag + emergency roster option.
8. Integration with HR system for holidays, sick leave, authorized absences, contracted hours per employee, and non-working hours.
9. Creation of weekly and monthly timesheet for payroll both as a data table and an easy-to-read weekly timesheet.

Could have:

1. Dashboard for all employees' attendance by team/department, attendance reporting by week.
2. Auto-approval of shifts clocked in up to 1 minute late and clocked out up to 1 minute early, 15 minutes.
3. Manual weekly approval of all shifts outside of target with pay rules recommendations.
4. ML to recommend (based on previously worked shifts, availability, and skills) the most appropriate employees to cover the shift.

5. Offer overtime to employees via a secondary Teams application.

User Personas

Based on the requirements, two user personas were identified: the manager and the employees. The manager will benefit from using a web application with the following capabilities: creating, publishing, and editing shifts, viewing shifts per employee, viewing employees' available times and clock on records, approve employees' holiday requests.

The employees will benefit from using a portable mobile application to easily clock-in/out, submit holiday requests, view their records (including holidays, clock records, schedule, available times).

Use Cases

The use cases and diagram were created by me.

ID	Use Case Name	Actor	Access
UC1	Login	All users	All users
UC2	Display if minimum service levels are satisfied	System	Manager
UC3	DisplayShifts/Activities	System	Manager – all employees Employees – his data
UC4	Draft new shifts with roster pattern	Manager	
UC5	Publish drafts	Manager	
UC6	Swap shifts	Manager	
UC7	Publish Activity	Manager	
UC8	DisplayAvailableTimes	System	Manager – all employees Employees – his data
UC9	DisplayClockRecords	System	Manager – all employees Employees – his data
UC10	DisplayHolidays	System	Manager – all employees Employees – his data
UC11	Approve/Decline Holiday Requests	Manager	
UC12	SendHolidayRequest	Employee	
UC13	ClockIn	Employee	
UC14	ClockOut	Employee	

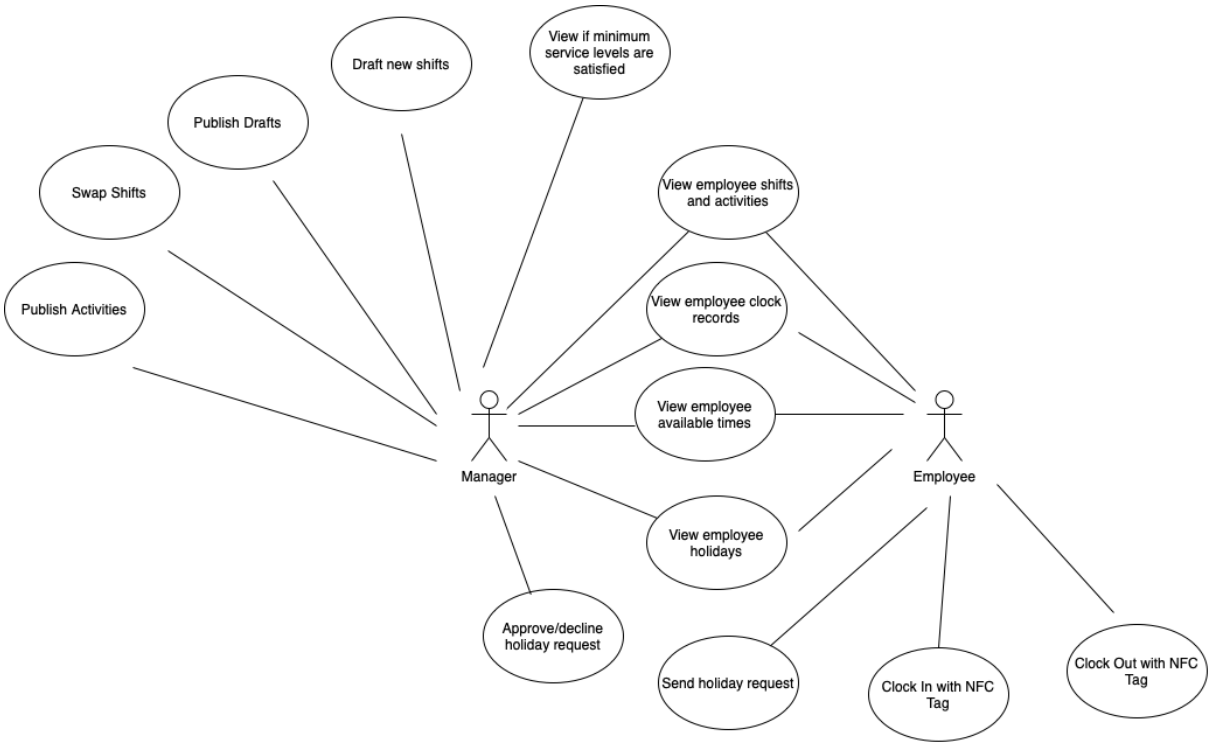


Figure 7 – Use Case Diagram

A more detail view of the use cases can be found in *Appendices 3.5.* Since I have designed the employee version of the app, I have focused on creating use case specifications for the use cases involving the employee.

Below is the matching between the requirements and the use cases for the employee app.

Requirement	Use Case
2.Employees can view roster (shifts).	UC3 DisplayShifts/Activities
3.Employees can clock on/clock off a shift using NFC tags.	UC13 ClockIn UC14 ClockOut
9. Ability to hold table of approved holidays.	UC10 DisplayHolidays
10. Ability to hold table of non-working hours. (Availability times per employee)	UC8 DisplayAvailableTimes
11.Ability to hold table of contracted hours per employee.	UC9 DisplayClockRecords
12.Ability to roster training and meeting time. (Activities)	UC3 DisplayShifts/Activities
13.Ability to request time off.	UC12 SendHolidayRequest

Wireframes

Leeyu created the app wireframe from the requirements. The wireframe for the employee app can be found in *Appendices 3.6.*

Figma Prototype

From the wireframe, Leeyu created a Figma Prototype. The Prototype for the employee app can be found in *Appendices 3.7.*

ER Diagram

The ER diagram was created by Leeyu in collaboration with me. Leeyu created the initial draft of the diagram, and I have given her feedback and made some adjustments.

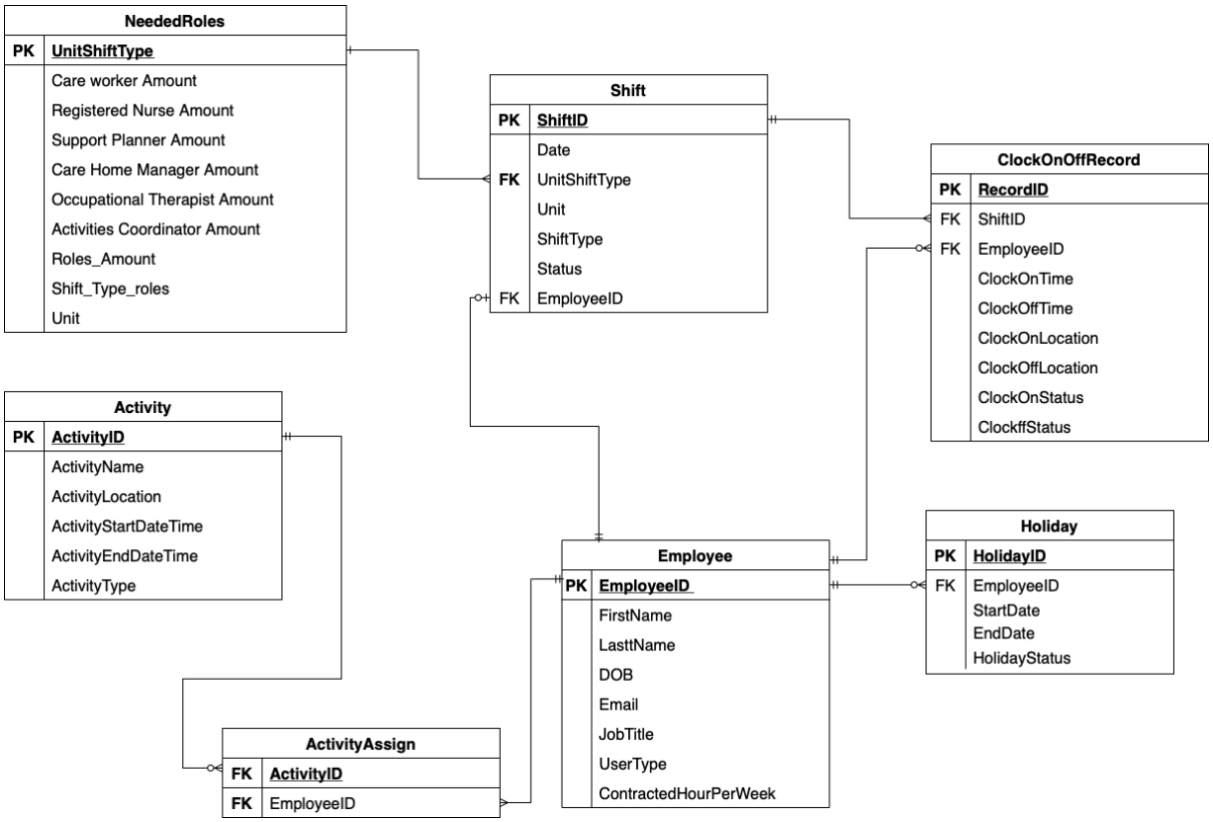


Figure 8 - ER Diagram ERoster App

4. Design and Implementation

1. Social Care Task Management App

Based on the ER Diagram (*Figure 6*), I have created the database in Dataverse. The dummy data from the client in *Figure 3* was useful in generating the *Task* and *Daily Tasks* tables. All the data generated is dummy data, as the client is not allowed to provide real hospital data. The one-to-many relationships between the tables were created in Dataverse by using look-up columns for the foreign keys.

2. E-Rostering App

2.1. Database

Based on the ER diagram *(Figure 8)* I have created some of the tables in Dataverse in collaboration with Leeyu.

The app is divided into 2 apps since there are 2 users: the admin and the employees.

2.2. E-Rostering – Employees

I have worked individually on building this app.

The app is a mobile compatible application that has 5 main functionalities:

1. To allow employees to view their daily schedule.
2. To allow employees to view their clock in and clock out records for a specified period along with the total worked hours.
3. To allow employees to view their holiday records and send new requests.
4. To allow employees to view a calendar with their available times.
5. To allow employees to clock in and clock out.

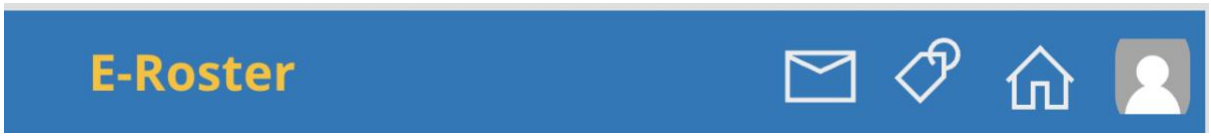


Figure 9 - Header

The app has one component – the header. The header is present on all the main pages and allow users to navigate back to the home page and to click the “Tag” icon to clock in or clock out.

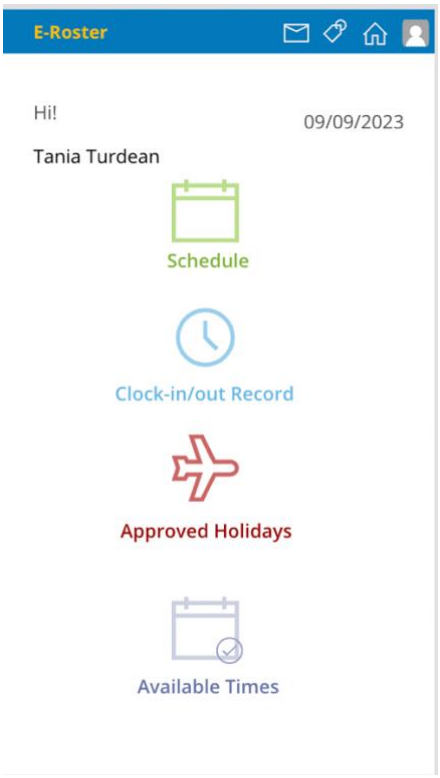


Figure 10 – Home page

The home page allows users to navigate to the main pages. It is important to note that this app is only displaying the logged in user's records, by retrieving the user's email and looking up the Employee ID corresponding to that email. On the home page, the logged in user's full name is displayed.

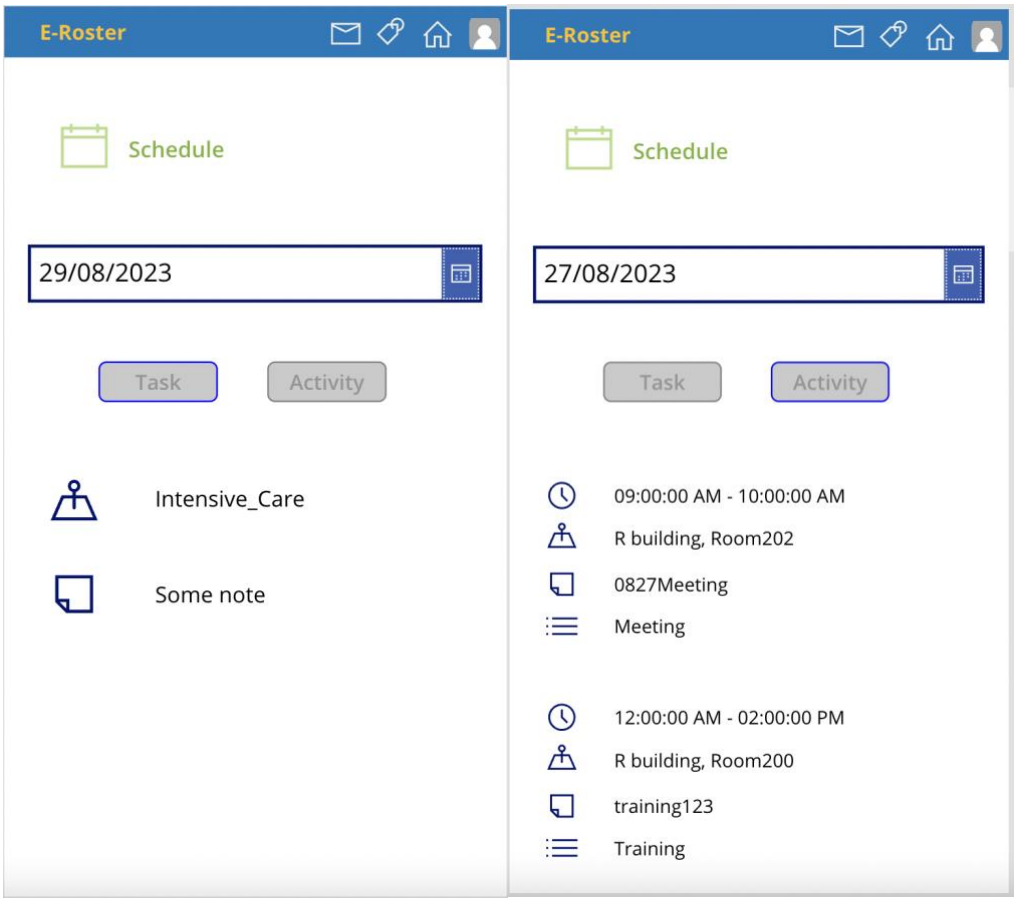


Figure 11 - Schedule

The schedule page has a date picker and 2 buttons: task and activity. The user selects a date and one of the buttons. The buttons have a corresponding variable that is set to True when selected and sets the other button's variable to False. If the variable is true, the border is highlighted, and the corresponding components are set to visible.

For the task listing, the *Shift_Tables* is queried. The rows selected are the ones with the corresponding date, status as published, and the employee ID of the logged in user. The unit and notes variables are displayed as text since I am assuming there can only be one shift in day.

For the activity listing, a gallery is displayed to allow displaying more activities per day. For each activity (meeting or training) we display the start and end time, location, name and type. Two tables are queried. First, we select the rows from *Assign_Activit_Tables* that have the user's employee id. Then, we select the *activity_meeting_id* column from those rows and look up the rows in *Activity_Tables* that have matching IDs. The last step is to select the rows with the matching date.

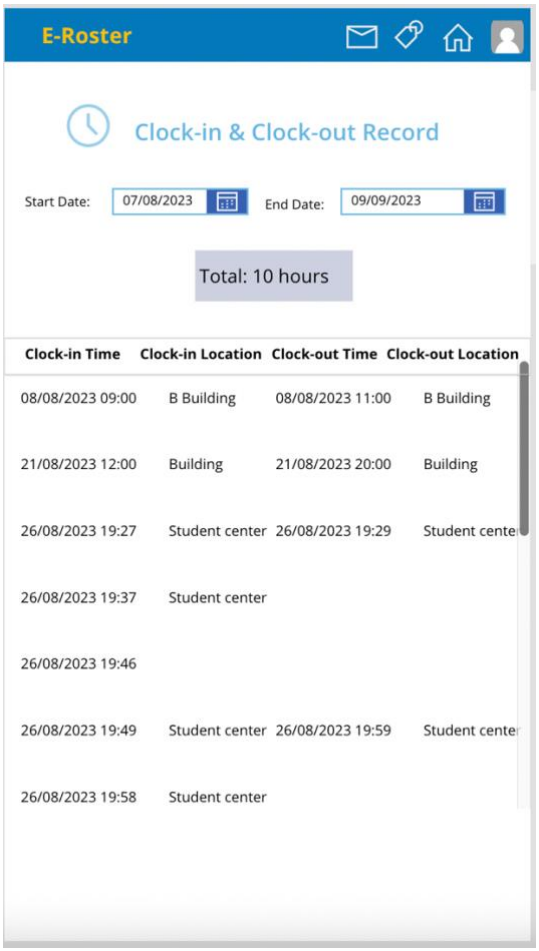


Figure 12 – Clock Records

The clock in records page is displaying the user's clock in/out time and location for the selected dates by querying the corresponding table and selecting the employee's ID and the start and end dates. It is sorted ascending by date.

To calculate the total worked hours for the selected period, a collection is created on app start. It contains 3 columns: employee id (only logged in user's), clock in time, and hour difference (the difference between the clock on and off in hours). The field uses this collection to sum up the hour rows that are in the time period selected by the user.

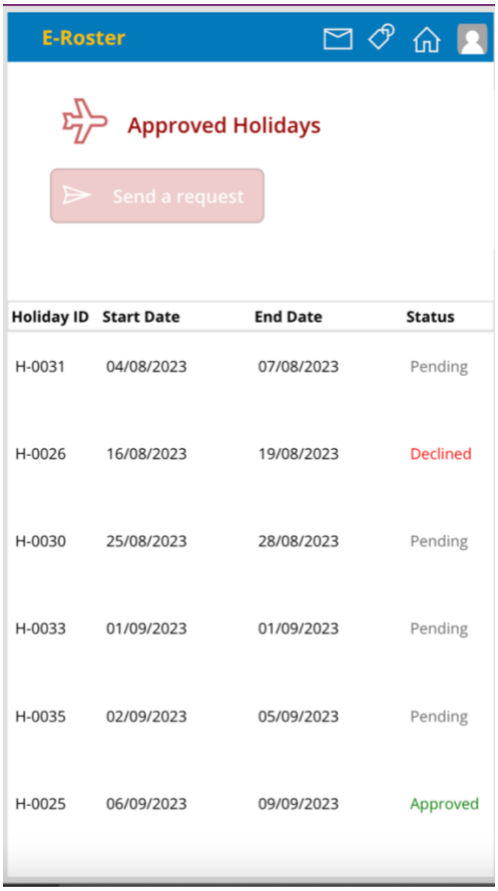


Figure 13 – Approved Holidays

The Holiday Page displays the logged in user's holiday records sorted by date. The status column is colored according to the value.

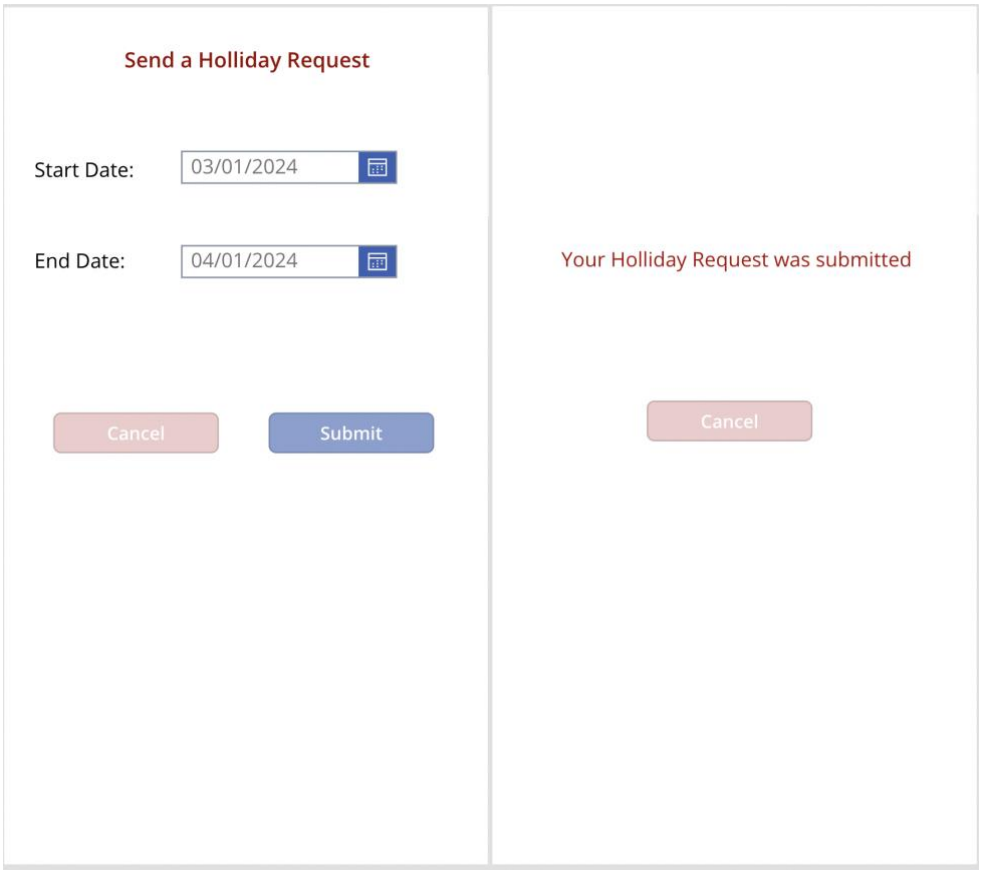


Figure 14 – Send Holiday Request

Users can send holiday requests by selecting a start and end date. When clicking submit, a put request is sent to the database which creates a new record with the logged in employee's ID, the selected dates, and the status set to "Pending".

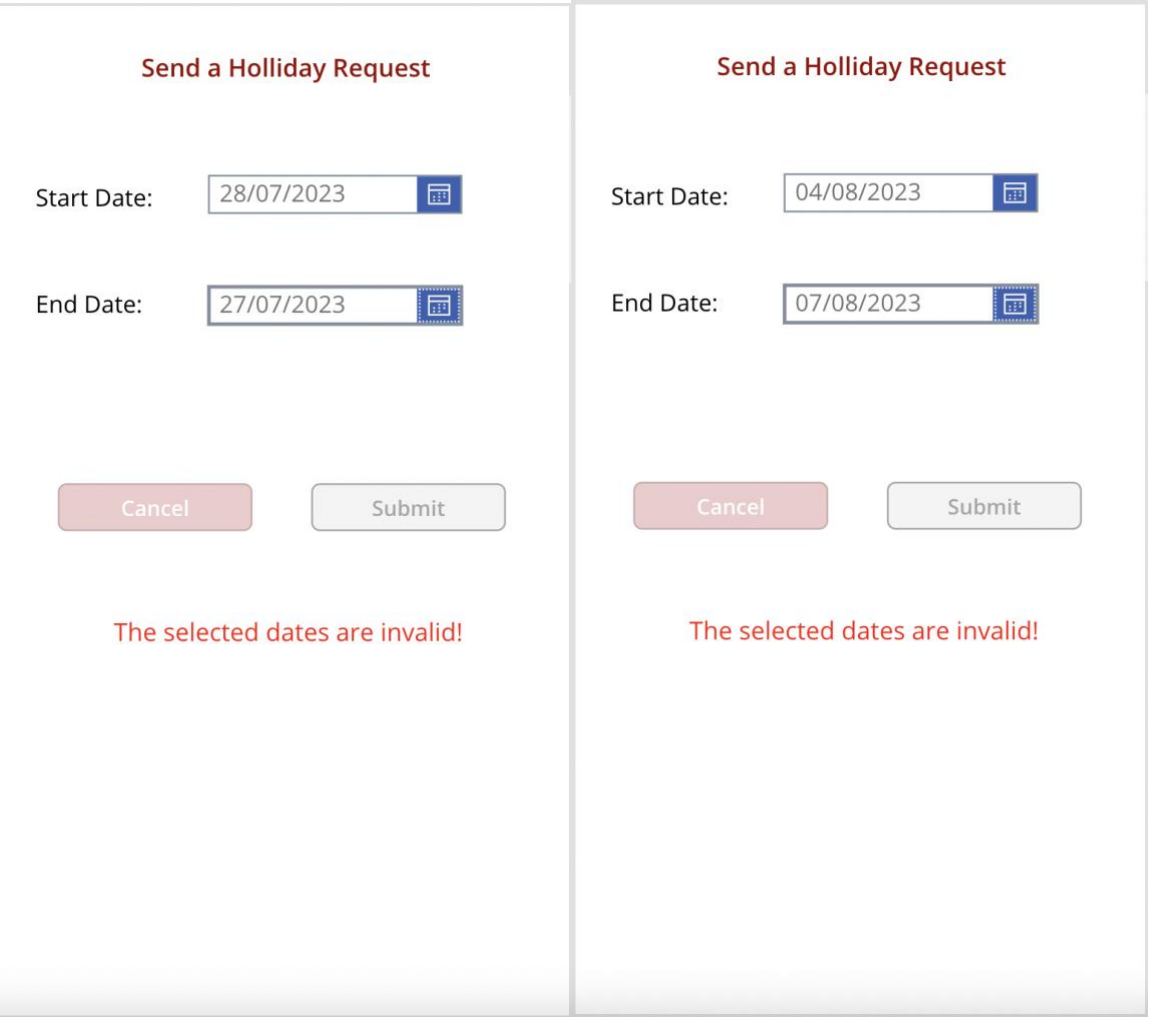


Figure 15 – Invalid Holiday Request

The “Submit” button is disabled and an error message appears whenever a user selects 2 invalid dates. A date is “invalid” in two cases. The first one is that the “End Date” is before the “Start Date”. The second one is that the period selected is already in the employee’s holiday records either as “pending” or “approved” request.

Figure 16 – Available Times

The “Available Times” page displays a user’s available times in white. Available times means the employee is not on shift and not on holiday. The red entries are the pending and approved holidays, the orange is a published day shift and the blue is a published night shift.

For the blue month header, a variable was created at the start of the app which stores the first date of the month. The text is the year and month value for that variable. The “forward” arrow increments the month value of the variable with one unit. The “previous” arrow decrements by one unit.

The calendar is a gallery generated using a formula. It creates a sequence of 42 items to represent a month's days, accommodating for months spanning between 4 to 6 weeks. The formula starts from the first date of the month, adjusts to ensure the correct weekday start, and then extracts day numbers for display, including any preceding or following days from adjacent months to complete the grid.

The shifts and holidays are nested galleries inside the calendar gallery.

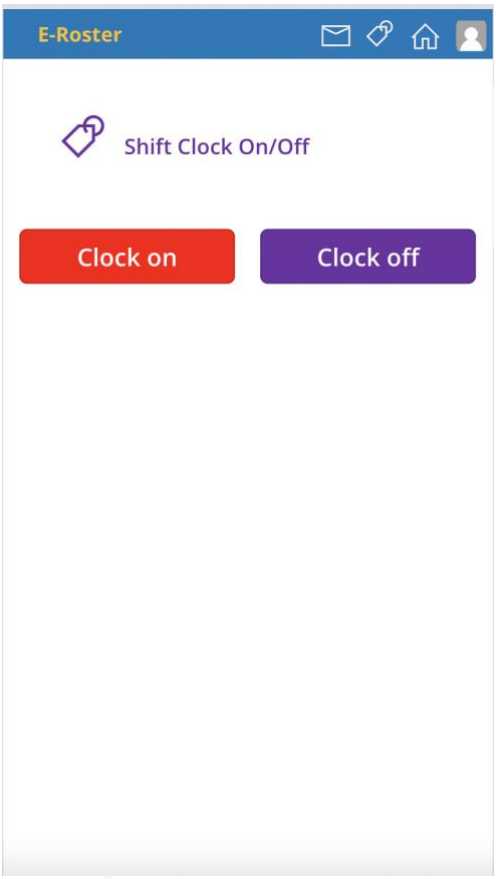


Figure 17 – Clock On/Off

The employee uses this page to clock on and off a shift. After pressing one of the buttons, the employee is prompted to scan an NFC tag, which will be placed in each clinic and will store the location. After scanning the tag, the data is added to the clock records.

For the clock on, new row is created with the employee id, the location stored in the tag, the time of the clock on, and the status of clock on is set to “Yes” while the status of clock off is set to “No”.

For the clock off, the corresponding clock on row is found using the LookUp function. It checks that the employee ID is matching, and that the clock on was made less than 16 hours ago and that the clock off status is “No”. The it adds the location and time of the clock off and sets the status to “Yes”.

5. Testing

5.1. Unit and Integration Testing

This testing stage was performed using Test Studio in Power Apps. Four test suites were created, one for each page: Holiday Request, Schedule, Holiday, Clock records. There is no test suite for the Available Times page since one known limitation of Test Studio is Nested Galleries. (14) Also, there is no test suite for the Clock In page, since a physical NFC tag is needed to use the functions of the page.

Holiday Request

This suite is made up of five test cases. All test cases passed.

Test Case Name	Description
Send Valid Holiday Request	To test sending a valid request, the rows from the holiday table are counted before and after submitting the request. It is asserted weather the number of rows is incremented by one. Two random dates are generated, and there is an assert statement after which check if the dates are valid. If so, the test continues.
Send invalid request - Date 1 Larger - button disabled check	Two variables which store dates are created. The end date is set to be before the start date. The test checks weather the “Submit” button is disabled.
Send invalid request - date 1 larger - check error message appears	Two variables which store dates are created. The end date is set to be before the start date. The test checks weather the error message is visible.
Send invalid request - Dates already taken - check button disabled	The start and end date are set to be 2 existing dates from holiday record with status “pending” or “approved”. The test checks weather the “Submit” button is disabled.
Send invalid request - Dates already taken - check error message visible	The start and end date are set to be 2 existing dates from holiday record with status “pending” or “approved”. The test checks weather the error message is visible.

Schedule

This suite is made up of three test cases. All test cases passed.

Test Case Name	**Description**
Check Task is highlighted on click	Checks if the Task button is highlighted on click and the Activity button is not.
Stress test - check Activity is highlighted after multiple clicks	Checks if the Activity button is highlighted on click and the Task button is not. Check is made after multiple clicks as a stress test.

Check the right elements are visbile for Task	Checks the corresponding Task elements are visible and the Activity elements are not visible.
Check the right elements are visbile for Activity	Checks the corresponding Activity elements are visible and the Task elements are not visible.
Check correct data identified for a date and task	Checks the task location and notes match the table for a given example.

Holiday

This suite is made up of one test case. All test cases passed.

Test Case Name	Description
Check if holidays are for logged in employee	Checks if the number of rows in the table for the employee are the same as the number of rows in the displayed gallery.

Clock Records

This suite is made up of two test cases. All test cases passed.

Test Case Name	Description
Logged in employees values in gallery with applied date filters	Creates 3 variables that store the employee ID and the selected dates. Loops through all the rows of the gallery and checks if the records match the 3 variables.
Total hours	Selects 2 dates and computes the expected total hours. Formats it as the required text “Total x hours” and asserts weather the value in the box is matching.

The full tests (step, screen, action) can be found in *Appendices 4*.

5.2. Responsive Design Testing

Power Apps Play tool allows the developer to test for compatibility with 17 different types of mobile phones. The app is responsive to all of the phone screen sizes.

5.3. Browser Compatibility Testing

Compatibility testing was undertaken using PowerMapper SortSite. The application has no compatibility issues with any browser.

Browser	Edge	Firefox	Safari	Opera	Chrome	iOS		Android
Version	114	114	16	99	114	≤ 15	16	114
Critical Issues	✓	✓	✓	✓	✓	✓	✓	✓
Major Issues	✓	✓	✓	✓	✓	✓	✓	✓
Minor Issues	✓	✓	✓	✓	✓	✓	✓	✓

Figure 18 – Browser Compatibility Test Results

5.4. Stress Testing

As mentioned in the Unit and Integration section, the Activity and Task buttons on Schedule page were stress tested and passed the test. The two buttons (Task and Activity) were pressed multiple times and the correct data was still displayed.

For the clock on and off functions, manual tests were performed. After multiple trials, it was found that sometimes the NFC tag location cannot be read and only the time is added to the database.

One issue that was fixed during the tag stress tests was the clock off function. It was found that if a user clocked in multiple times during a 16 hour window, the second clock off was replacing the values of the first one. This was because there was no condition that checked that the clocked off status is set to “No”. The condition was added.

5.5. User Acceptance Testing

User acceptance tests were essential for identifying additional bugs and app functionalities that might be challenging for new users, given that the test participants were not familiar with the website.

The first step was defining a list of tasks, as displayed in the first table. These tasks represent the essential functional requirements that were previously determined. Then, a table in Excel was created to be filled out by each tester. They were asked to indicate ”Pass” or ”Fail” for each task and provide relevant comments. Participants were given the tester’s phone with the published app and were asked to interact with the app while completing the Excel file. The audience is made of 3 people: a medical UCL student, an engineering UCL student, and a User Experience Designer.

The first table presents the completion rate for each task. The second table compiles the comments provided by the testers, which provided insights into why some tests failed, and consequently, offered valuable information for future improvements. The main findings were that users cannot find the clock in/out page, the available times calendar would benefit from a legend, and selecting start and end dates should be made at the same time.

Task	Pass/Fail
Look at your Schedule for tasks for 29 August	3/3
Look at your schedule for Activities for 27 August	3/3

Check you clock in/out records for a period of time	3/3
Check your total hours for 10th - 27th august in clock in records	3/3
Check your holiday records	3/3
Submit Holiday request	2/3
View your available times for September and August	3/3
Clock In your shift	3/3
Clock Out your shift	3/3

Comments
Send Holiday Request - Specify invalid dates error - holiday request already submitted ; would be helpful to see when you select what is already selected
Available Times - Add 'Holiday' and 'Shift' text to holiday pending/approved - more clear ; night and day shift; different colors for pending/approved
Clock in/out - easier if there is a shortcut in the home page to go there (another button like the others)
Clock in/out - when I clocked on it would be nice to show a timer with how many hours I already worked
Schedule - user didn't know he needs to click on task button
When you select 2 dates, be able to select at the same time in the calendar similar to airplane tickets
Available Times - legend: white available, user doesn't know what is blue and orange
Clock in/out - user found it difficult to find the button in the header
The app will benefit from a back button because the home page button not obvious
Choose start and end at the same time in calendar
Send Holiday Request - the button for send request looks like unable to click on - in UI we add more contrast for buttons
Send Holiday Request - the reason I couldn't find send request is because I thought approved holidays was in the past and was not looking there
Available Times - Pending should be grey
Clock in/out - not easy to find the button in header and it is a daily thing to do so should be bigger or in the main menu maybe even the first menu item cause its daily

5.6. Client

Feedback from client was collected not only at the end, but also in the development stage of the app during the weekly meetings. The client requested an accumulated hours box in the Clock Records page. This request was immediately implemented.

During the final demo meeting, the client provided useful feedback for future work. Firstly, the client spotted a spelling mistake on Holiday Page for Holiday. For the Send Holiday Request function, the client suggested to add one more restriction: restrict submission of holidays if the date is in the past. At the end of the app demo, the client provided an overall

feedback: the solution is a really good baseline to start building a solution upon and can be already used in hospitals in its current state. There are more features that can be implemented, such as recording breaks, or medical appointments.

6. Conclusions and Project Evaluation

6.1. Summary of Achievements

The Roster Employee App addressed half of the third goal in *Section 1.2.:* Implement the *E-Rostering App* to transform shift management, ensuring optimal staff allocation and maximizing productivity. The other half (admin view) is addressed by another student.

From the Moscow requirements in 3.3., I have selected the ones relevant to the employee version of the app and displayed them below, with their priority and state.

Requirement	Priority	State
Shift booking application built on Microsoft Power Platform.	Must	Done
Employees can view roster (shifts).	Must	Done
Employees can clock on/clock off a shift using NFC tags.	Must	Done
Ability to hold table of approved holidays.	Must	Done
Ability to hold table of non-working hours. (availability times per employee)	Must	Done
Ability to hold table of contracted hours per employee.	Must	Done
Ability to roster training and meeting time. (Activities)	Must	Done
Ability to request time off.	Must	Done
Ability to roster 24/7 services.	Should	Done
Integration with HR system for holidays, sick leave, authorized absences, contracted hours per employee, and non-working hours.	Should	TBD
Creation of weekly and monthly timesheet for payroll both as a data table	Should	TBD

and an easy-to-read weekly timesheet.		
Dashboard for all employees' attendance by team/department, attendance reporting by week.	Could	TBD
Auto-approval of shifts clocked in up to 1 minute late and clocked out up to 1 minute early, 15 minutes.	Could	TBD
Manual weekly approval of all shifts outside of target with pay rules recommendations.	Could	TBD
Offer overtime to employees via a secondary Teams application.	Could	TBD
Recording breaks or medical appointments (identified during client demo)	Could	TBD
Key Functionalities (Must Have)		100% completed
Optional Functionalities (Should Have and Could Have)		11% completed

6.2. Critical Evaluation

6.2.1. Functionality

The application covers the key functionality of the requirements. Given the short timeframe, the focus was put on the “Must Have” requirements. The application successfully incorporates all of them and is ready to be used in hospitals, as the client stated. Future work for this application will include the optional functionalities.

6.3.1. User Interface and User Experience Design

Visibility of system status

The system should always keep the user informed about what is going on through appropriate feedback within reasonable time. This requirement is satisfied. For the Send Holiday Request function, the user is informed weather the dates are invalid or if the request was successfully submitted.

Match between system and real world

All the icons in the app match their use cases to keep the user informed of what is happening.

User control and freedom

Not developed.

Consistency and standards

The system has continuity across the platform: all the headers, tables and icons are on the same level for pages, tables have the same formatting, date pickers are consistent.

Error prevention

Users cannot submit invalid holiday requests as the “Submit” button is disabled. What can be added is error prevention for the clock in/out page. A “Are you sure you want to clock in?” message can be shown to prevent users from clocking in/out by mistake.

Recognition rather than recall

Ass seen from testing, users cannot find the clock in/out page because it does not appear in the main menu, only in the header. Moreover, users cannot find the “Send holiday request” button, because of its fading color.

Flexibility and efficiency of use

There are no shortcuts implemented.

Aesthetic and minimalist design

This principle has been respected across the app: there is no irrelevant information.

Help users recognise, diagnose and recover from errors

Error message should be expressed in plein language, and precisely indicate the problem. Although an error message is displayed for sending invalid holiday request, the message does not specify the error reason. Could specify why the dates are invalid and could show the already taken requests in the selection calendar.

Help and documentation

There is no documentation in the app.

6.3. Future Work

Feature	Description
Holiday Request Restriction	Restrict to only submit future dates.
Breaks	Record employee breaks and add that to clock records to get a more accurate calculation for worked hours.
Send Holiday Request Button	Make the button more visible – can add it to the main menu
Clock In Menu	Add the tag icon and clock in to the main menu, as the first entry.
Error message holiday request	See error type for holiday request, i.e. end date before start date, dates already exist in records.
Calendar picker	Pick start and end date at the same time.
Calendar Holiday Request	When picking the dates, see in the calendar what you cannot pick (already selected dates).
Available Times legend	Add a legend for the available times and color each value differently.
Back Button	To improve visibility, add a back button to all of the pages.
Sick leaves and authorized absences	Add these to an absences table, along with holidays.
Payroll	Calculated from worked hours (in clock records)
Attendance	Table storing employee attendance from shift and clock in.
Shift punctuality	Late clock in checker (15 minutes)

List of References

1. NHS. Internet]. Place of publication: NHS; 2023 [cited 2023 08 20]

2. NHS England. [Internet]. England: NHS England; December 2022 [cited 2023 08 22].
3. NHS England. [Internet]. England: NHS England; 2023 [cited 2023 08 22].
4. Department of Health and Social Care. [Internet]. 2023 [cited 2023 08 22].

5. Skills for Care [Internet]. Place of publication: Skills for Care; 2014 [cited 2023 08 23].

6. Newton Europe Consultancy. [Internet]. Place of publication: Newton Europe Consultancy; May 2018 [cited 2023 08 23].

7. Forrester Consulting. The Total Economic Impact of Power Apps [Internet]. March 2020 [cited 2023 08 30].

8. McKinsey. Developer Velocity Research [Internet]. Place of publication: McKinsey; 2023 [cited 2023 08 30].

9. Intelligent Bed Management Software [Youtube]. 2020 [cited 2023 09 01]. Available from: https://www.youtube.com/watch?v=BTmeEo3Pfwc

10. CareTasker [Internet]. [cited 2023 09 01]. Available from: https://caretaskr.com

11. Williams E. Empowering care teams with new tools in Microsoft 365 [Internet]. Microsoft; 2020 Mar 6 [cited 2023 09 01]. Available from: https://www.microsoft.com/en-us/microsoft-365/blog/2020/03/06/empowering-care-teams-with-new-tools-in-microsoft-365/

12. How to manage a Shifts schedule in Microsoft Teams [Youtube]. 2020 Jul 31 [cited 2023 09 05]. Available from: https://www.youtube.com/watch?v=Ka_oWhyD3V8

13. Times to Care [Internet]. 2023 April [cited 2023 09 07]. Available from: https://www.adass.org.uk/media/9685/adass-time-to-act-april-2023.pdf

14. Microsoft. Test and monitor your app [Internet]. Microsoft; 2023 [cited 2023 09 09]. Available from: https://learn.microsoft.com/en-us/power-apps/maker/canvas-apps/test-studio

Appendices

1. System Manual

- Go to https://make.powerapps.com
- Go to the “Solutions” tab on the vertical side bar
- Click “Import solution” on the top menu
- Download the zip file from Github: https://github.com/taniaturdean/Power-apps-roster-employees
- Import the zip file
- After the solution is imported, it can be edited since it is an unmanaged solution

2. User Manual

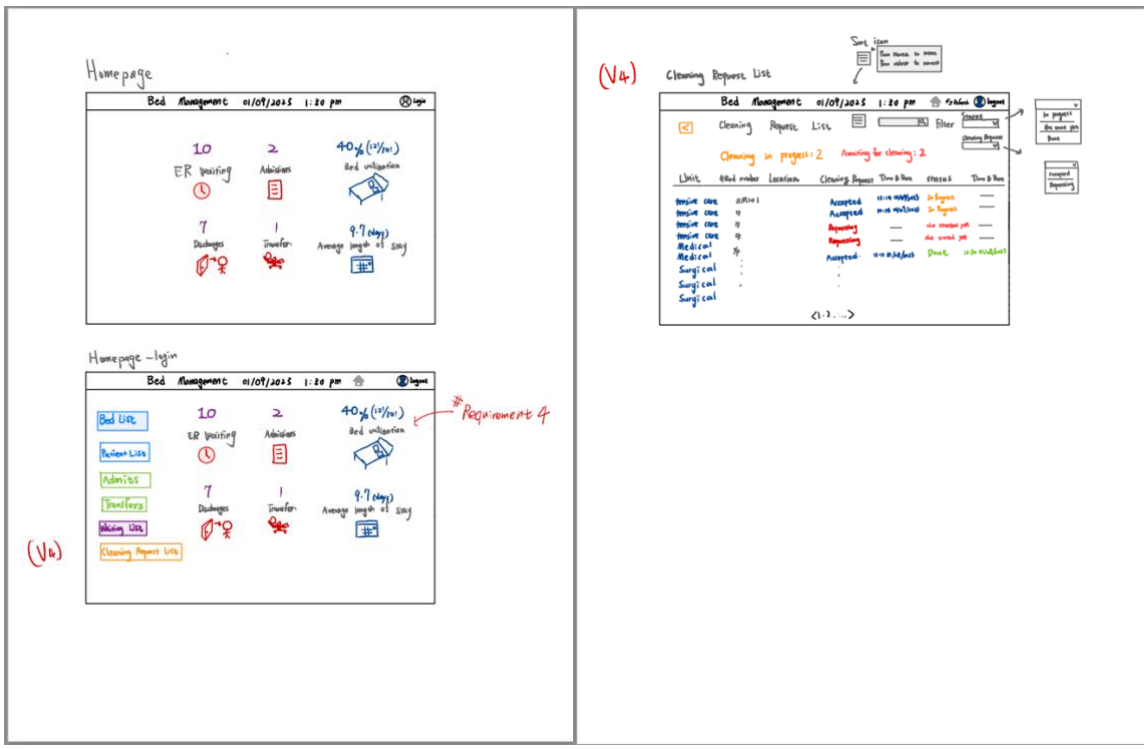
Go to this link:
https://apps.powerapps.com/play/e/default-aea48856-3e55-45be-ba30-15334fd23af9/a/e96acb10-7eae-4f2f-a26b-e0aff6bd8900?tenantId=aea48856-3e55-45be-ba30-15334fd23af9&hint=8e200c9b-0eac-4f22-9454-1e6cdb6bcca6&source=sharebutton&sourcetime=1694374697503

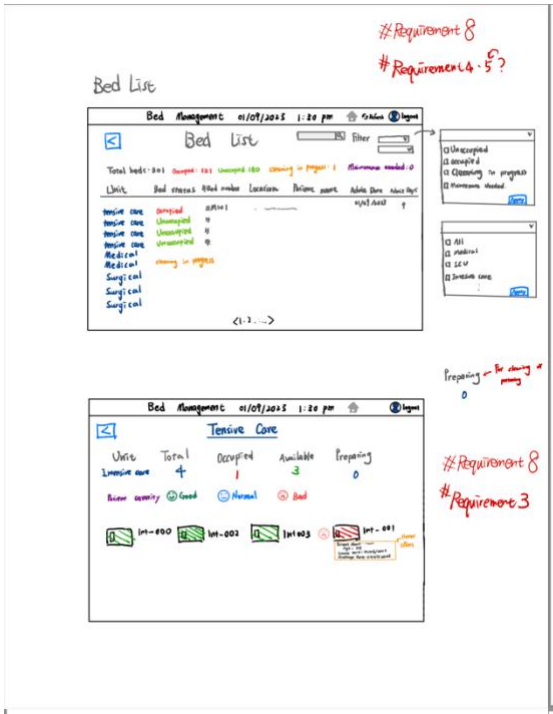
Login to this account to explore the app:
taniaturdean@uclmicrosoftnhs.onmicrosoft.com
Intropic12!@

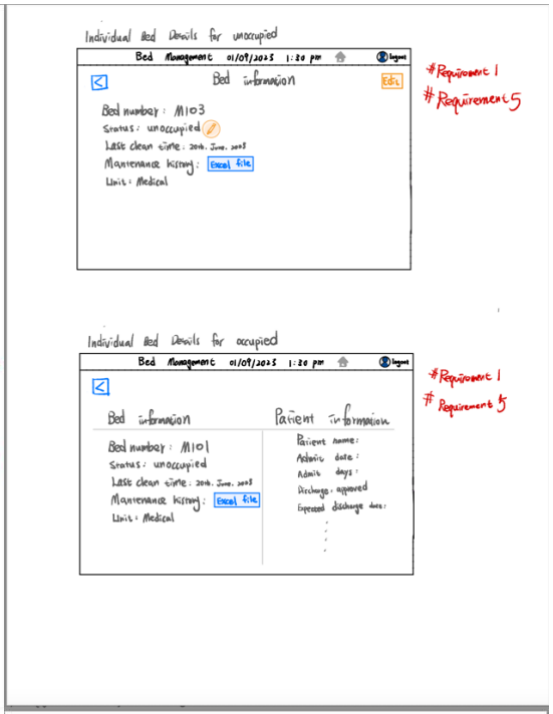
Demo Video: https://www.youtube.com/watch?v=J2DMgQ-Xi9E

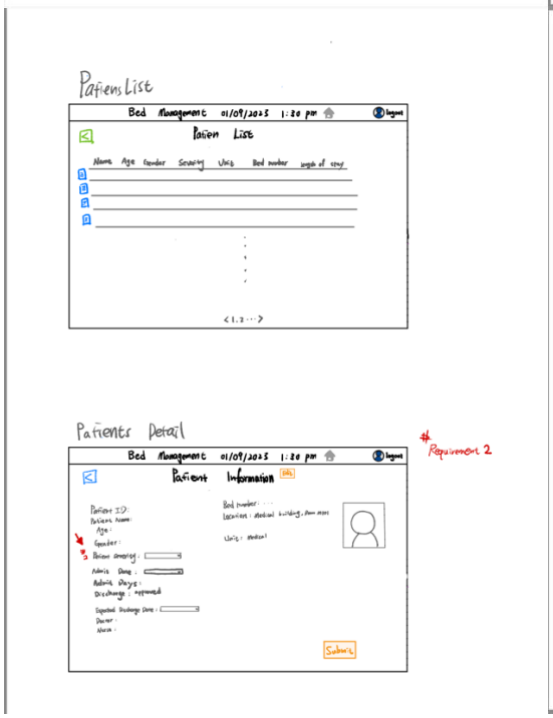
3. Supporting Documentation and Diagrams

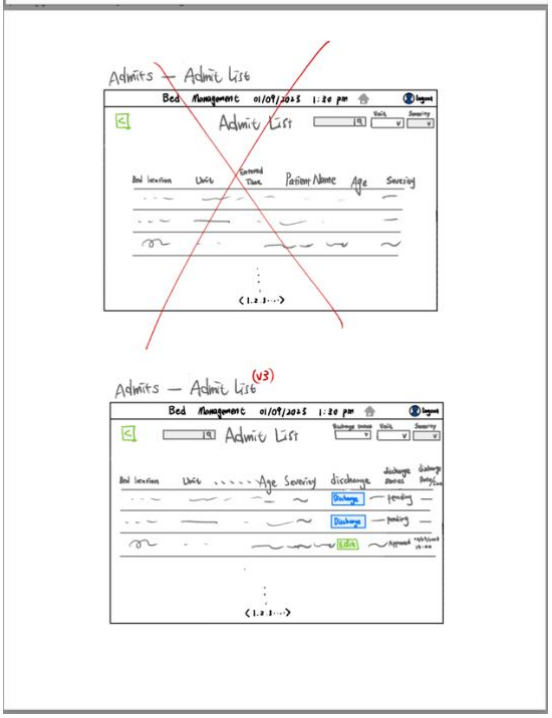
3.1.Bed Management App Wireframes



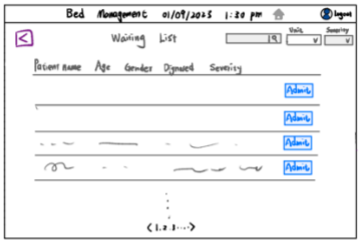

#Requirement 8
#Requirement 4, 5?
Bed List
Bed Management 01/09/2025 1:20 pm
Bed List
Filter
Unit
Bed status
Bed number
Location
Patient count
Admin
Tensive Care
Medical
Surgical
Unoccupied
occupied
Cleaning in progress
Maintenance needed
All
Medical
ICU
Invasive care
Preparing = for cleaning & painting
0
Tensive Care
Unit
Total
Occupied
Available
Preparing
Invasive care
4
1
3
0
Room cleanliness
Good
Normal
Bad
bed-001
bed-002
bed-003
bed-004
#Requirement 8
#Requirement 3


Individual Bed Details for unoccupied
Bed Management 01/09/2025 1:20 pm
Bed Information
Edit
Bed number: M103
Status: unoccupied
Last clean time: 20th June, 2025
Maintenance history:
Excel file
Unit: Medical
#Requirement 1
#Requirement 5
Individual Bed Details for occupied
Bed Management 01/09/2025 1:20 pm
Bed Information
Patient Information
Bed number: M101
Status: unoccupied
Last clean time: 20th June, 2025
Maintenance history:
Excel file
Unit: Medical
Patient name:
Admit date:
Admit days:
Discharge approved
Expected discharge date:
#Requirement 1
#Requirement 5

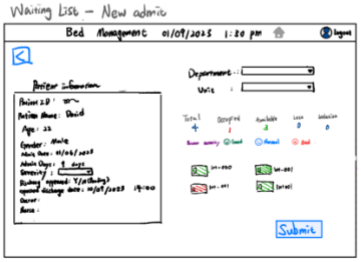

Patients List
Bed Management 01/09/2025 1:20 pm
Patient List
Name
Age
Gender
Severity
Unit
Bed number
Length of stay
Patient Detail
Bed Management 01/09/2025 1:20 pm
Patient Information
Edit
Patient ID:
Patient Name:
Age:
Gender:
Patient severity:
Admit Date:
Admit days:
Discharge approved
Expected Discharge Date:
Doctor
Notes:
Bed number:
Location: medical building, 2nd floor
Unit: Medical
Submit
#Requirement 2


Admits — Admit List
Bed Management 01/09/2025 1:20 pm
Admit List
Bed location
Unit
Expected Time
Patient Name
Age
Severity
Admits — Admit List (v3)
Bed Management 01/09/2025 1:20 pm
Admit List
Discharge status
Unit
Severity
Bed location
Unit
Age
Severity
discharge
discharge status
Discharge
pending
Discharge
pending
Edit
Approved

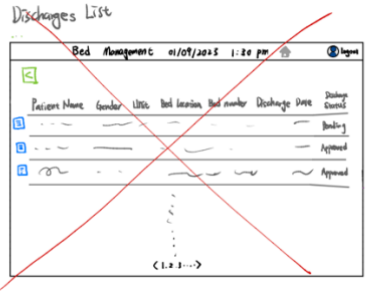
Waiting List



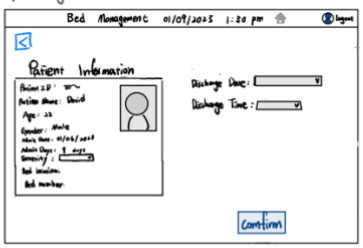
Waiting List – New admit



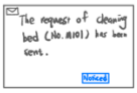
(V3) Discharges List



(V3) Discharge

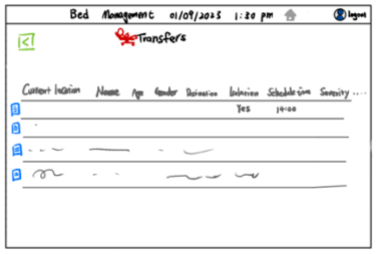


After clicking on the "confirm" button pop-up screen

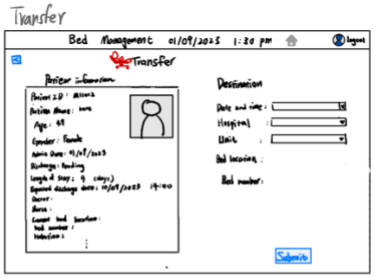


*Requirement 6
Requirement 7

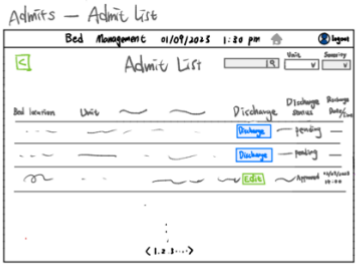
Transfer List



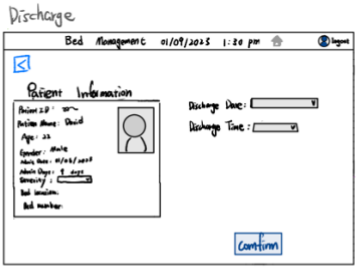
Transfer



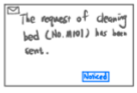
Admits – Admit List



Discharge

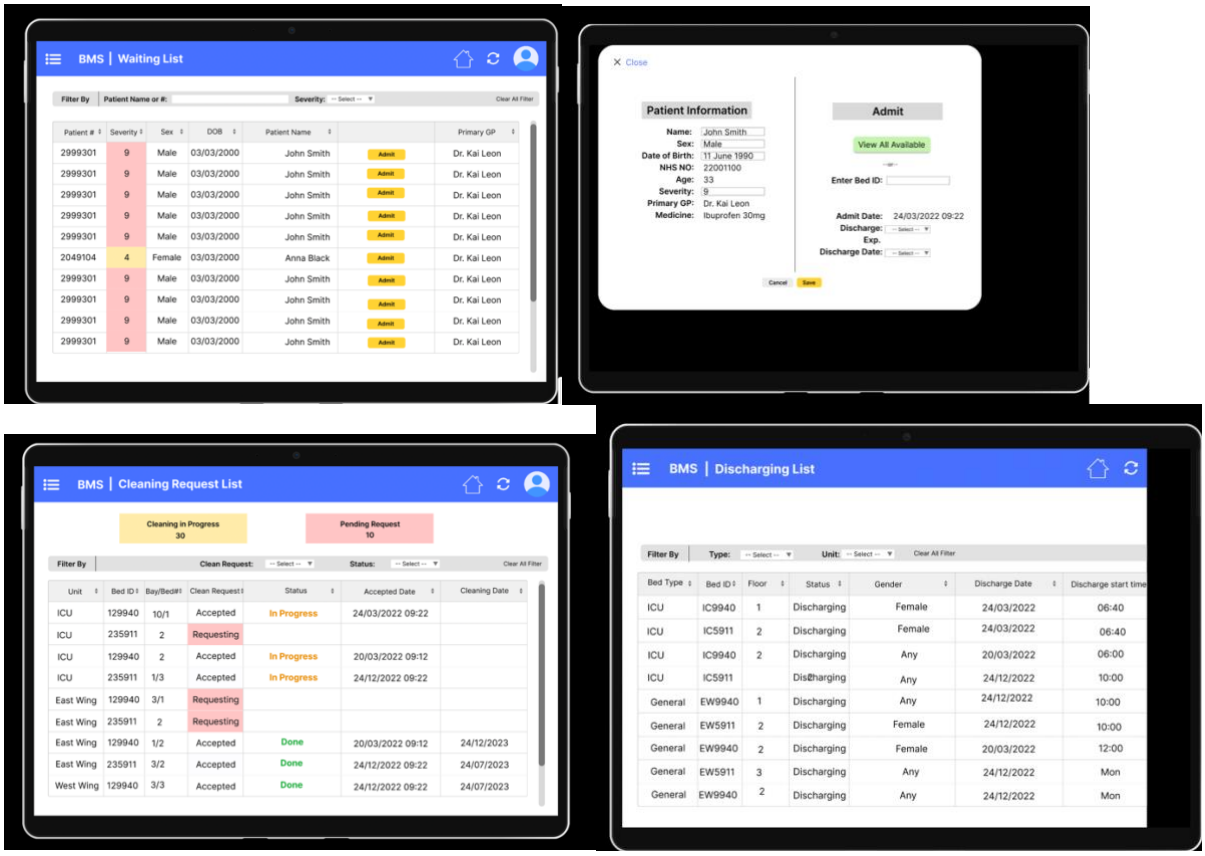


After clicking on the "confirm" button pop-up screen

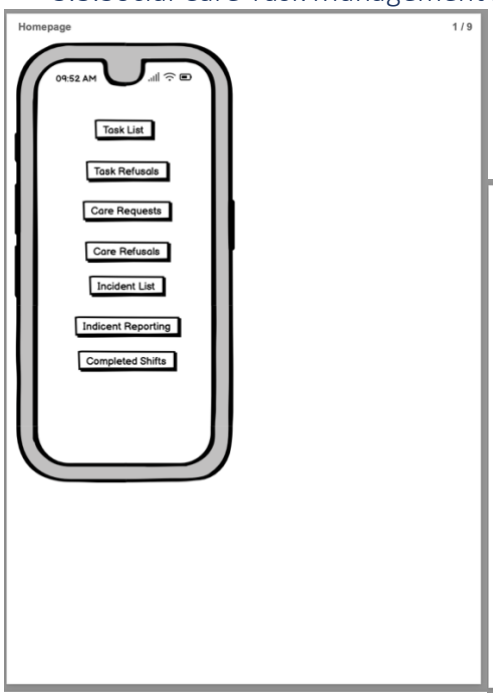


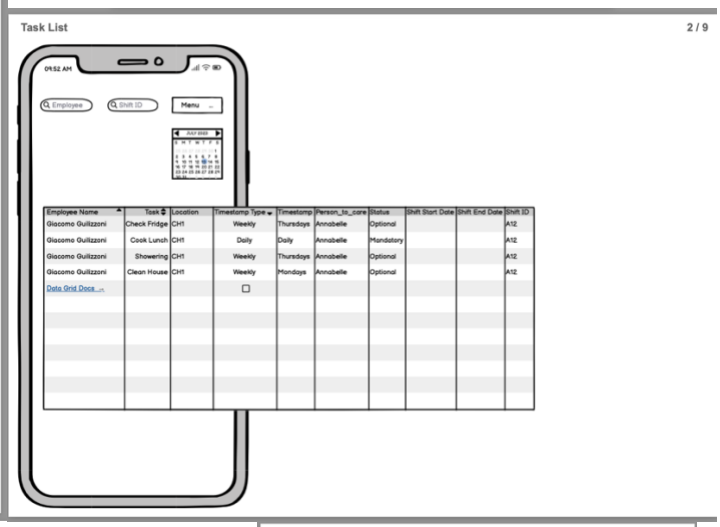
*Requirement 6
Requirement 7

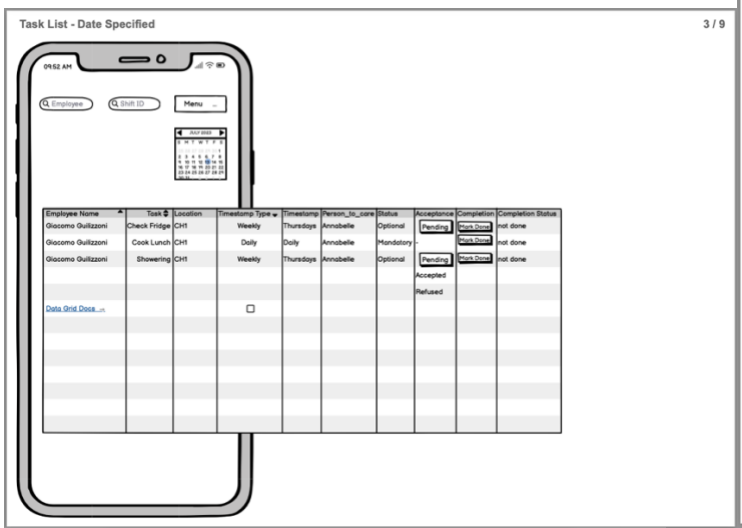
3.2.Bed Management Figma Prototype

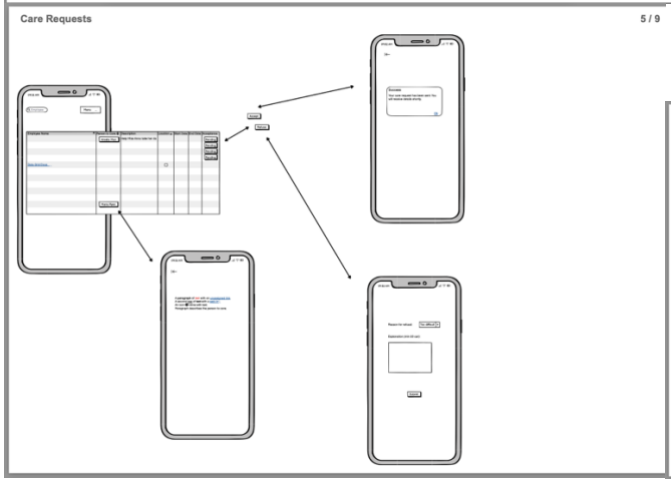

BMS | Waiting List
Filter By
Patient Name or #:
Severity:
Clear All Filter
Patient # | Severity | Sex | DOB | Patient Name | Primary GP
2999301 | 9 | Male | 03/03/2000 | John Smith | Admit | Dr. Kai Leon
2999301 | 9 | Male | 03/03/2000 | John Smith | Admit | Dr. Kai Leon
2999301 | 9 | Male | 03/03/2000 | John Smith | Admit | Dr. Kai Leon
2999301 | 9 | Male | 03/03/2000 | John Smith | Admit | Dr. Kai Leon
2999301 | 9 | Male | 03/03/2000 | John Smith | Admit | Dr. Kai Leon
2049104 | 4 | Female | 03/03/2000 | Anna Black | Admit | Dr. Kai Leon
2999301 | 9 | Male | 03/03/2000 | John Smith | Admit | Dr. Kai Leon
2999301 | 9 | Male | 03/03/2000 | John Smith | Admit | Dr. Kai Leon
2999301 | 9 | Male | 03/03/2000 | John Smith | Admit | Dr. Kai Leon
2999301 | 9 | Male | 03/03/2000 | John Smith | Admit | Dr. Kai Leon
Close
Patient Information
Name: John Smith
Sex: Male
Date of Birth: 11 June 1990
NHS NO: 22001100
Age: 33
Severity: 9
Primary GP: Dr. Kai Leon
Medicine: Ibuprofen 30mg
Admit
View All Available
Enter Bed ID:
Admit Date: 24/03/2022 09:22
Discharge:
Exp.
Discharge Date:
Cancel
Save
BMS | Cleaning Request List
Cleaning in Progress 30
Pending Request 10
Filter By
Clean Request:
Status:
Clear All Filter
Unit | Bed ID | Bay/Bed# | Clean Request | Status | Accepted Date | Cleaning Date
ICU | 129940 | 10/1 | Accepted | In Progress | 24/03/2022 09:22
ICU | 235911 | 2 | Requesting
ICU | 129940 | 2 | Accepted | In Progress | 20/03/2022 09:12
ICU | 235911 | 1/3 | Accepted | In Progress | 24/12/2022 09:22
East Wing | 129940 | 3/1 | Requesting
East Wing | 235911 | 2 | Requesting
East Wing | 129940 | 1/2 | Accepted | Done | 20/03/2022 09:12 | 24/12/2023
East Wing | 235911 | 3/2 | Accepted | Done | 24/12/2022 09:22 | 24/07/2023
West Wing | 129940 | 3/3 | Accepted | Done | 24/12/2022 09:22 | 24/07/2023
BMS | Discharging List
Filter By
Type:
Unit:
Clear All Filter
Bed Type | Bed ID | Floor | Status | Gender | Discharge Date | Discharge start time
ICU | IC9940 | 1 | Discharging | Female | 24/03/2022 | 06:40
ICU | IC5911 | 2 | Discharging | Female | 24/03/2022 | 06:40
ICU | IC9940 | 2 | Discharging | Any | 20/03/2022 | 06:00
ICU | IC5911 | | Discharging | Any | 24/12/2022 | 10:00
General | EW9940 | 1 | Discharging | Any | 24/12/2022 | 10:00
General | EW5911 | 2 | Discharging | Female | 24/12/2022 | 10:00
General | EW9940 | 2 | Discharging | Female | 20/03/2022 | 12:00
General | EW5911 | 3 | Discharging | Any | 24/12/2022 | Mon
General | EW9940 | 2 | Discharging | Any | 24/12/2022 | Mon

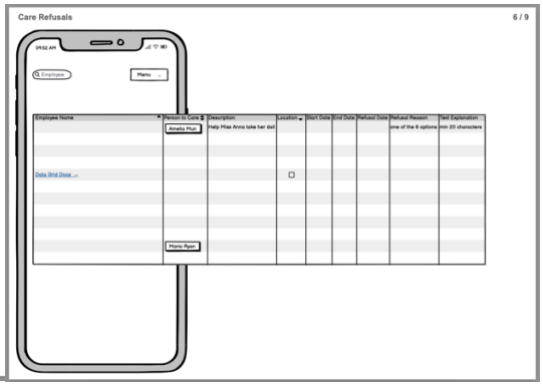
3.3.Social Care Task Management App Wireframe

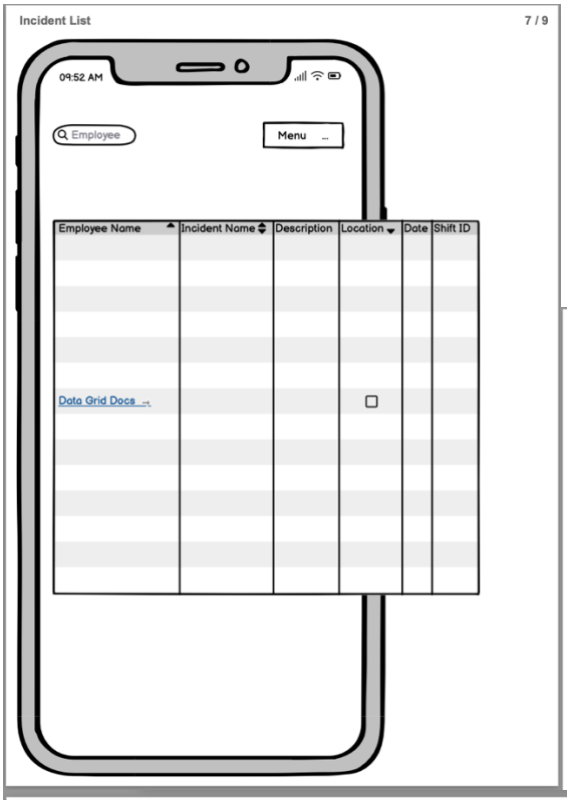


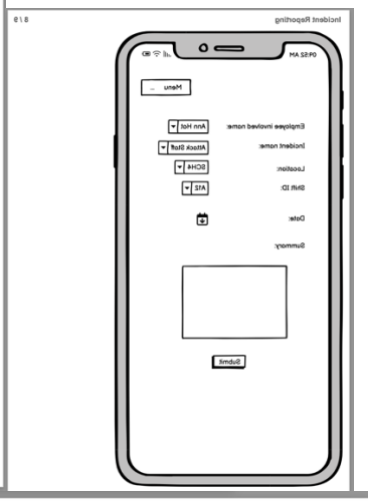


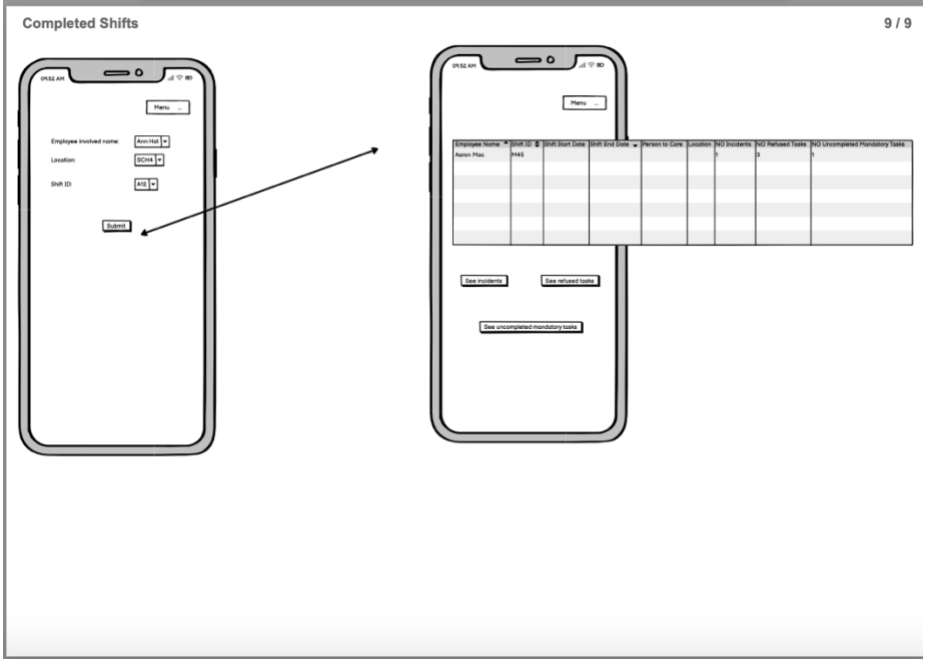




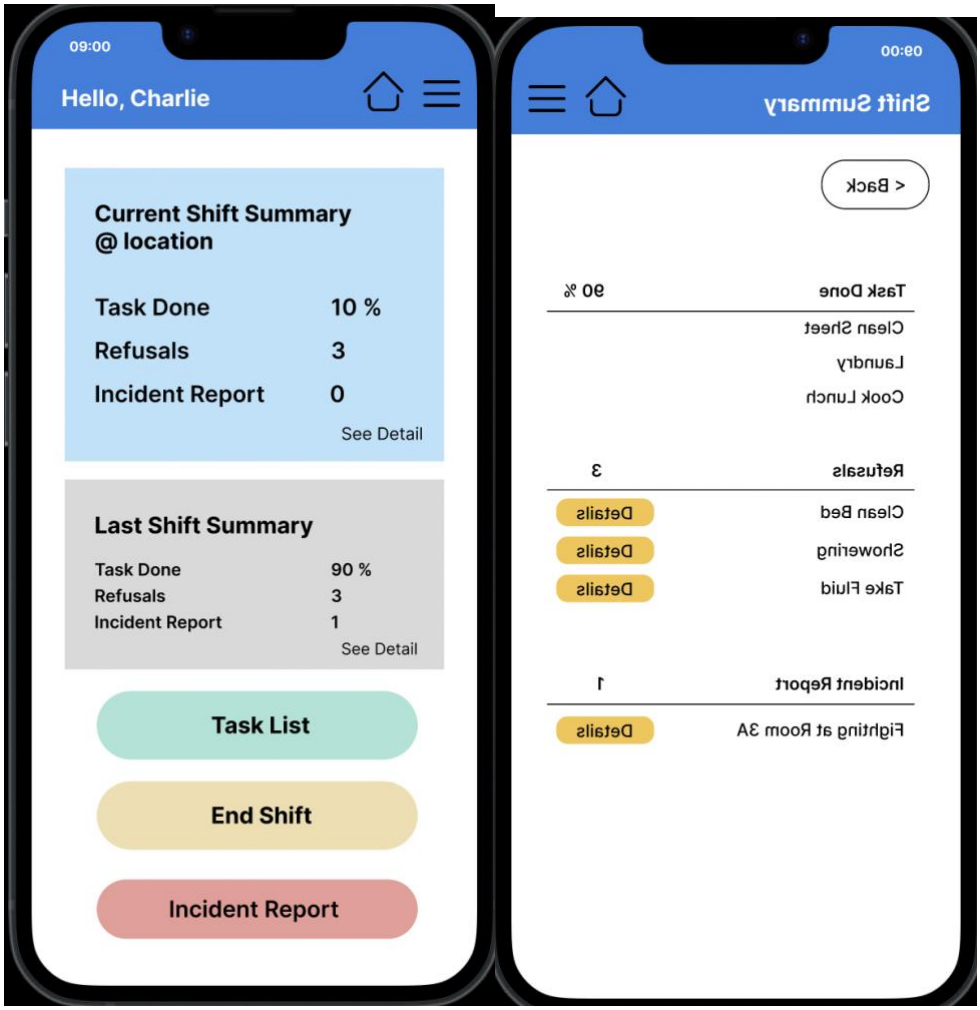


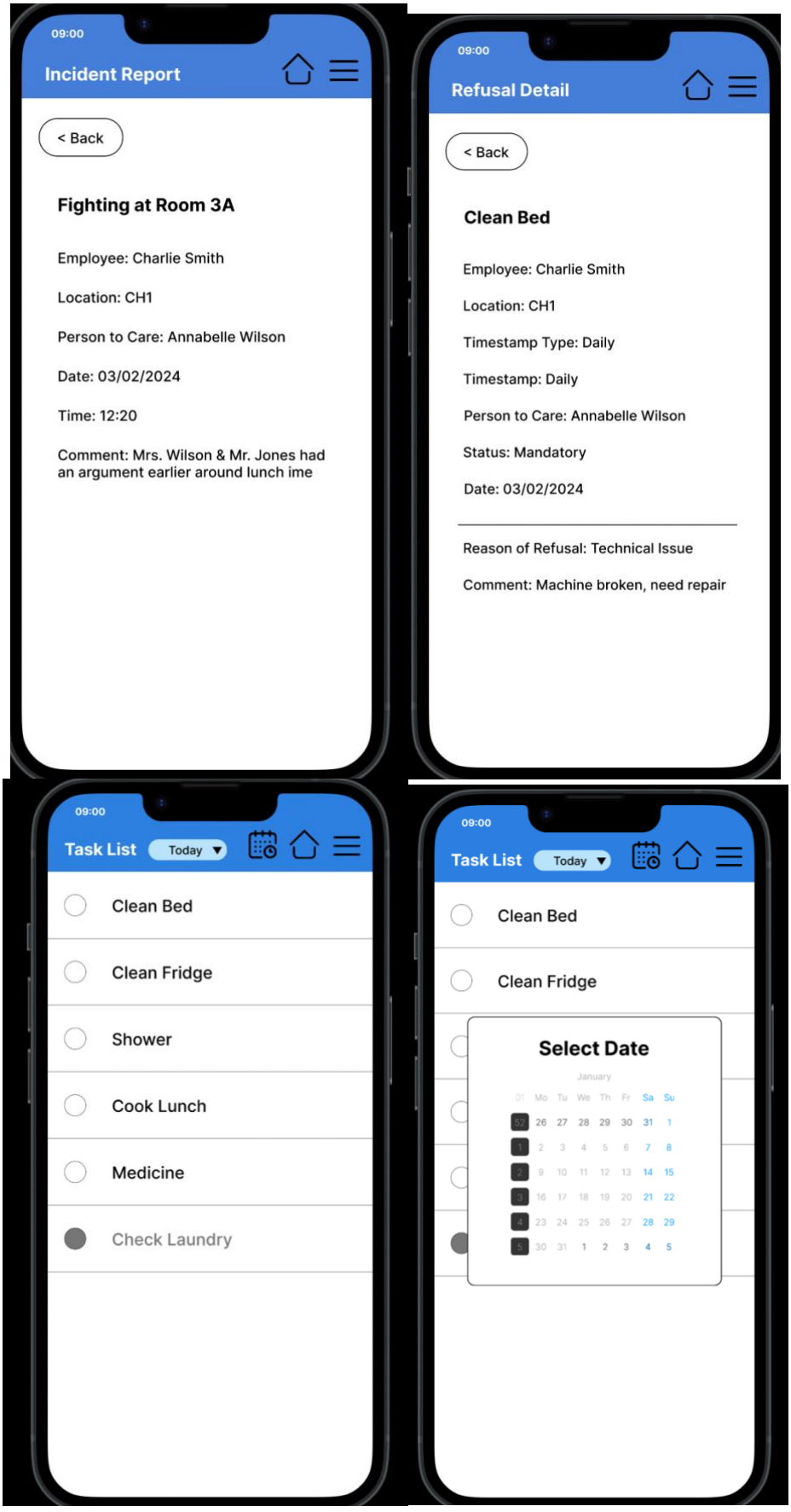

Incident List
7 / 9
09:52 AM
Employee
Menu ...
Employee Name
Incident Name
Description
Location
Date
Shift ID
Data Grid Docs →

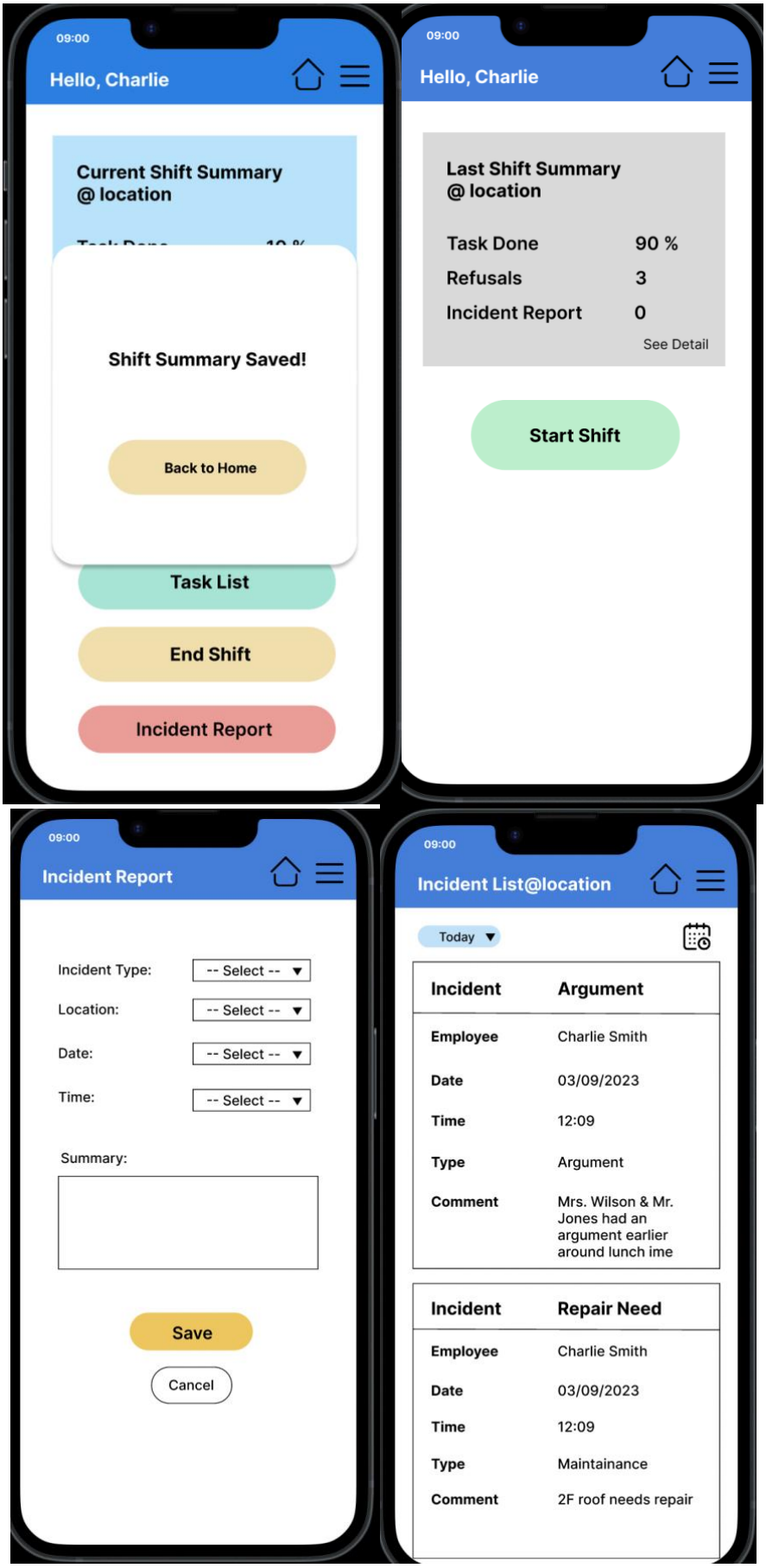


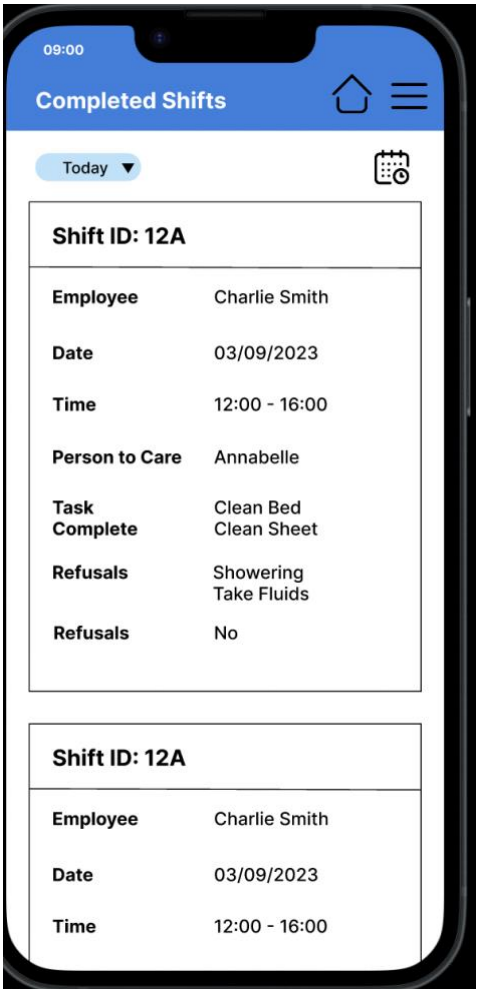

Completed Shifts
9 / 9
Menu ...
Employee involved name:
Location:
Shift ID:
Submit
Menu ...
Employee Name
Shift ID
Shift Start Date
Shift End Date
Person to Care
Location
NO Incidents
NO Refused Tasks
NO Uncompleted Mandatory Tasks
See incidents
See refused tasks
See uncompleted mandatory tasks

3.4.Social Care Task Management App Figma




09:00
Incident Report
< Back
Fighting at Room 3A
Employee: Charlie Smith
Location: CH1
Person to Care: Annabelle Wilson
Date: 03/02/2024
Time: 12:20
Comment: Mrs. Wilson & Mr. Jones had an argument earlier around lunch ime
09:00
Refusal Detail
< Back
Clean Bed
Employee: Charlie Smith
Location: CH1
Timestamp Type: Daily
Timestamp: Daily
Person to Care: Annabelle Wilson
Status: Mandatory
Date: 03/02/2024
Reason of Refusal: Technical Issue
Comment: Machine broken, need repair
09:00
Task List
Today
Clean Bed
Clean Fridge
Shower
Cook Lunch
Medicine
Check Laundry
09:00
Task List
Today
Clean Bed
Clean Fridge
Select Date


09:00
Hello, Charlie
Current Shift Summary
@ location
Shift Summary Saved!
Back to Home
Task List
End Shift
Incident Report
09:00
Hello, Charlie
Last Shift Summary
@ location
Task Done 90 %
Refusals 3
Incident Report 0
See Detail
Start Shift
09:00
Incident Report
Incident Type: -- Select --
Location: -- Select --
Date: -- Select --
Time: -- Select --
Summary:
Save
Cancel
09:00
Incident List@location
Today
Incident Argument
Employee Charlie Smith
Date 03/09/2023
Time 12:09
Type Argument
Comment Mrs. Wilson & Mr. Jones had an argument earlier around lunch ime
Incident Repair Need
Employee Charlie Smith
Date 03/09/2023
Time 12:09
Type Maintainance
Comment 2F roof needs repair


09:00
Completed Shifts
Today
Shift ID: 12A
Employee
Charlie Smith
Date
03/09/2023
Time
12:00 - 16:00
Person to Care
Annabelle
Task Complete
Clean Bed
Clean Sheet
Refusals
Showering
Take Fluids
Refusals
No
Shift ID: 12A
Employee
Charlie Smith
Date
03/09/2023
Time
12:00 - 16:00

3.5.E-Rostering App Employee Use Case Specifications

Name:	DisplayShift/Activities
ID:	UC2
Brief Description:	The system displays the shifts and activities for the logged in employee for a selected date.
Primary Actor(s):	Employee
Secondary Actor(s):	System
Preconditions:	Employee is logged in
Main Flow:	1.The use case starts when the employee selects “Schedule” from the home page. 2. The employee selects a date. 3. The system displays 2 options: Task and Activity 4. If the employee clicks Task, then 4.1.The system displays the tasks (shifts) for the selected date and hides the activities 5. If the employee clicks Activity, then 5.1. The system hides the tasks and displays the activities for the selected date
Postconditions:	None
Alternative Flows:	None

Name:	DisplayAvailableTimes
ID:	UC8
Brief Description:	The system displays the available times for the logged in employee in a calendar. Available times is showing the published shifts and not declined holidays.
Primary Actor(s):	Employee
Secondary Actor(s):	System
Preconditions:	Employee is logged in
Main Flow:	1.The use case starts when the employee selects “Available Times” from the home page. 2. The system displays a calendar with the available times. It shows the published shifts coloured by type and the not declined holidays. It shows the current month. 3. If the employee selects the “Forward” icon, then 3.1. The system displays the available times for the next month. 4. If the employee selects the “Previous” icon, then 4.1. The system displays the available times for the previous month.
Postconditions:	None
Alternative Flows:	None

Name:	DisplayClockRecords
ID:	UC9
Brief Description:	The system displays the clock records for the logged in employee in a table. It also shows the accumulated working hours.
Primary Actor(s):	Employee
Secondary Actor(s):	System
Preconditions:	Employee is logged in
Main Flow:	1.The use case starts when the employee selects “Clock in/out Records” from the Home Page. 2.The system displays a table with the clock records and a box with the accumulated worked hours. 3. If the employee selects a start date, then 3.1. The system selects a subset of the records with their date after the start date. 3.2. The system updates the worked hours. 4. If the employee selects an end date, then 4.1. The system selects a subset of the records with their date before the end date. 4.2. The system updates the worked hours.
Postconditions:	None
Alternative Flows:	None

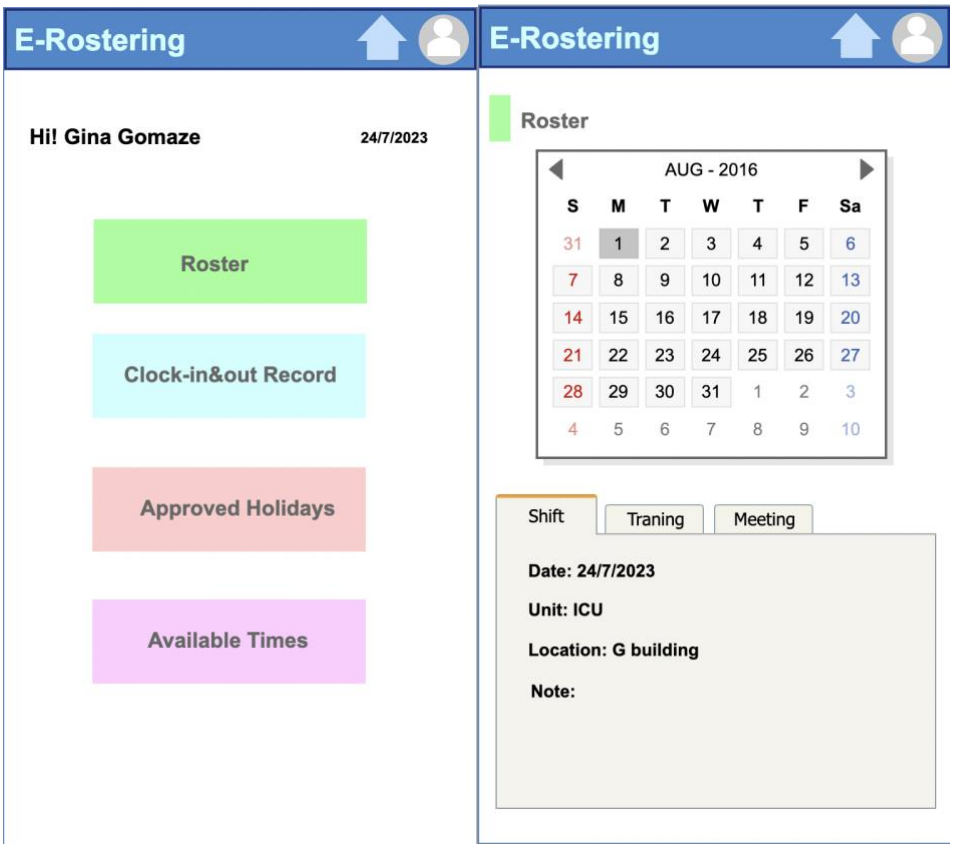
Name:	DisplayHolidays
ID:	UC10
Brief Description:	The system displays the holiday records for the logged in employee in a table.
Primary Actor(s):	System
Secondary Actor(s):	None
Preconditions:	Employee is logged in
Main Flow:	1.The use case starts when the employee selects “Holidays” from the home page. 2.The system displays a table with the holiday records. 3. If the Status is “Pending”, then 3.1. It is grey 4. If the status is “Declined”, then 4.1. It is red 5. If the status is “Accepted”, then 5.1. It is green
Postconditions:	None
Alternative Flows:	None

Name:	SendHolidayRequest
ID:	UC12
Brief Description:	Employee Sends a holiday request by selecting the dates.
Primary Actor(s):	Employee
Secondary Actor(s):	System
Preconditions:	Employee is logged in Employee is on “Holiday Records” page
Main Flow:	1.The use case starts when the employee selects “Send Holiday Request” 2. The system prompts the user the the request page and displays the buttons: Start Date, End Date, Cancel, Submit. 3. If the employee clicks “Cancel”, then 3.1. The system prompts the user to the Holiday Records page. 4. If the employee selects an end date that is before the start date, then 4.1. The system shows an error message and disabled the “Submit” button. 5. If the employee selects two dates already in the records, then 5.1. The system shows an error message and disabled the “Submit” button. 6. The employee submits the request.
Postconditions:	The records are added to the table, with the status “Pending”.
Alternative Flows:	None

Name:	ClockIn
ID:	UC13
Brief Description:	The employee uses an NFC tag which stores a location to clock in.
Primary Actor(s):	Employee
Secondary Actor(s):	System
Preconditions:	Employee is logged in.
Main Flow:	1.The use case starts when the employee selects the “Tag” icon from the header 2. The system displays 2 buttons: “Clock in” and “Clock out” 3. The user selects “Clock in” and is prompted to scan the NFC tag. 4. The user scans the tag.
Postconditions:	The records are added to the clock records, with the user id, the time, and the tag’s location.
Alternative Flows:	None

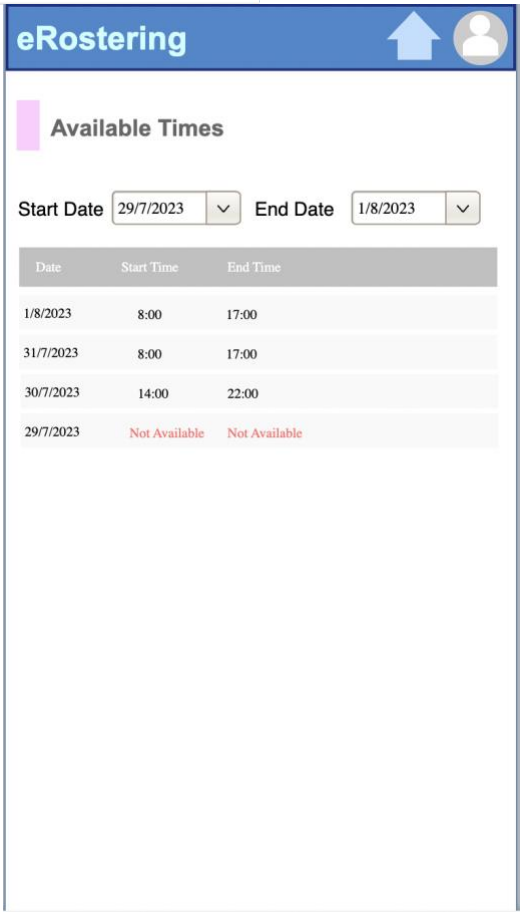
Name:	ClockOut
ID:	UC14
Brief Description:	The employee uses an NFC tag which stores a location to clock out.
Primary Actor(s):	Employee
Secondary Actor(s):	System
Preconditions:	Employee is logged in.
Main Flow:	1.The use case starts when the employee selects the “Tag” icon from the header 2. The system displays 2 buttons: “Clock in” and “Clock out” 3. The user selects “Clock out” and is prompted to scan the NFC tag. 4. The user scans the tag. 5. The system finds the row with the matching “Clock in”
Postconditions:	The records are added to the clock records, with the user id, the time, and the tag’s location.
Alternative Flows:	None

3.6.E-Rostering App Employees Wireframe

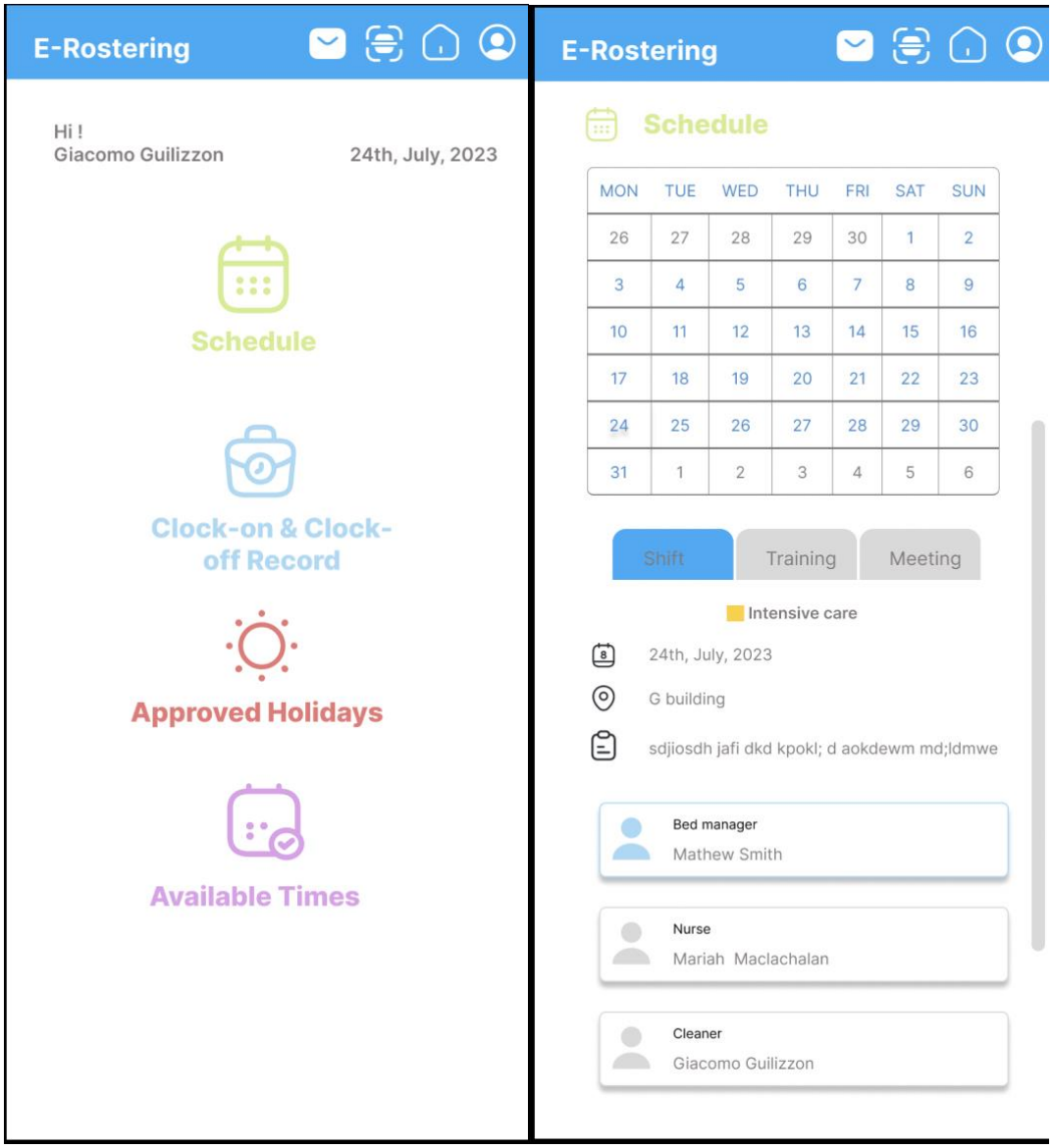


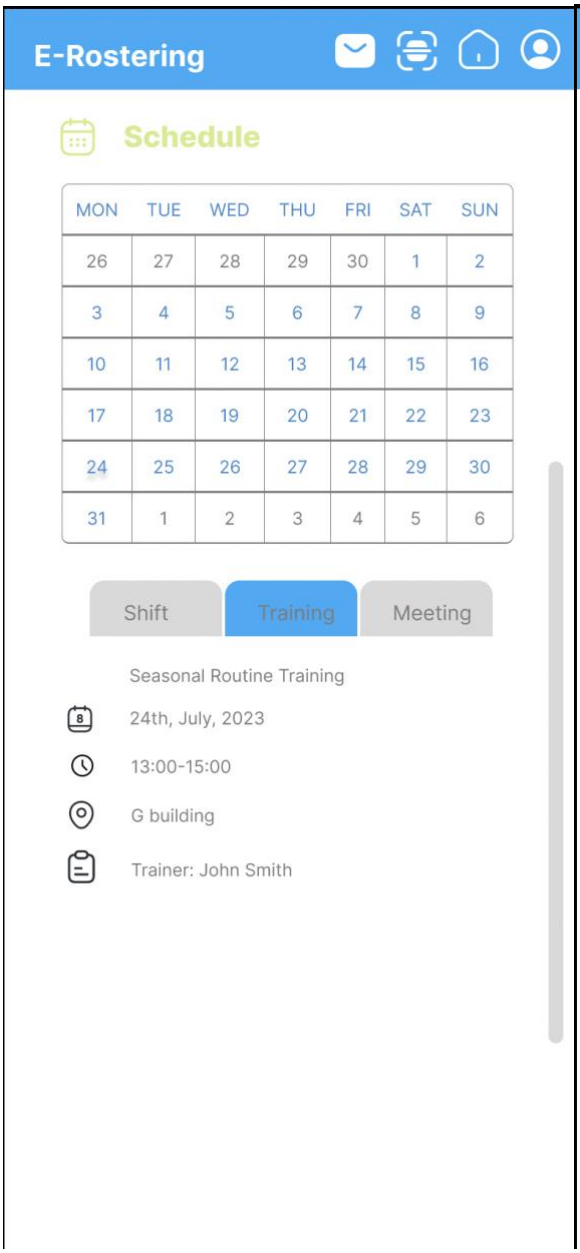

E-Rostering
Approved Holidays
Send a request
Holiday ID Employee ID First Name Last Name Start Date End Date Status
H-1001 EMP-0001 Richart Wang 29/7/2023 29/7/2023 Pending
H-1001 EMP-0001 Richart Wang 29/7/2023 29/7/2023 Pending
H-1001 EMP-0001 Richart Wang 29/7/2023 29/7/2023 Pending
H-1001 EMP-0001 Richart Wang 29/7/2023 29/7/2023 Pending

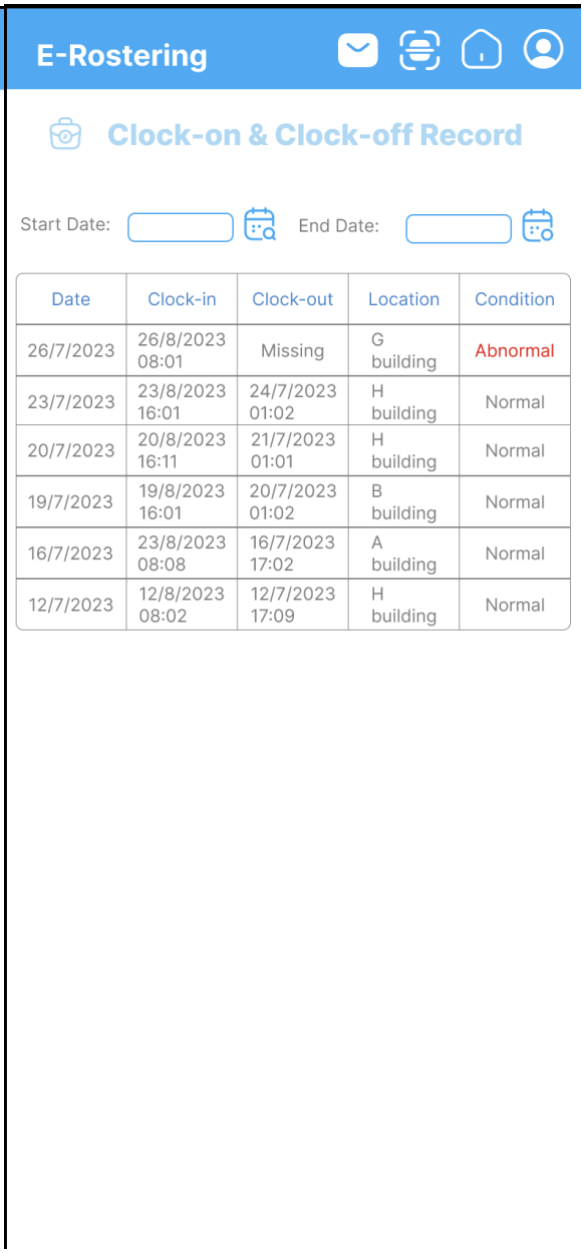

E-Rostering
Clock-in & Clock-out Record
Start Date 1/7/2023 End Date 4/7/2023
Clock ID Clock-in Clock-out Location Condition
C1001 1/7/2023 14:09:00 1/7/2023 22:09:00 Building H Normal
C1002 2/7/2023 13:01:00 1/7/2023 21:29:01 Building H Normal
C1003 3/7/2023 11:09:00 1/7/2023 29:00:00 Building H Normal
C1004 4/7/2023 14:09:00 1/7/2023 22:09:00 Building H Normal

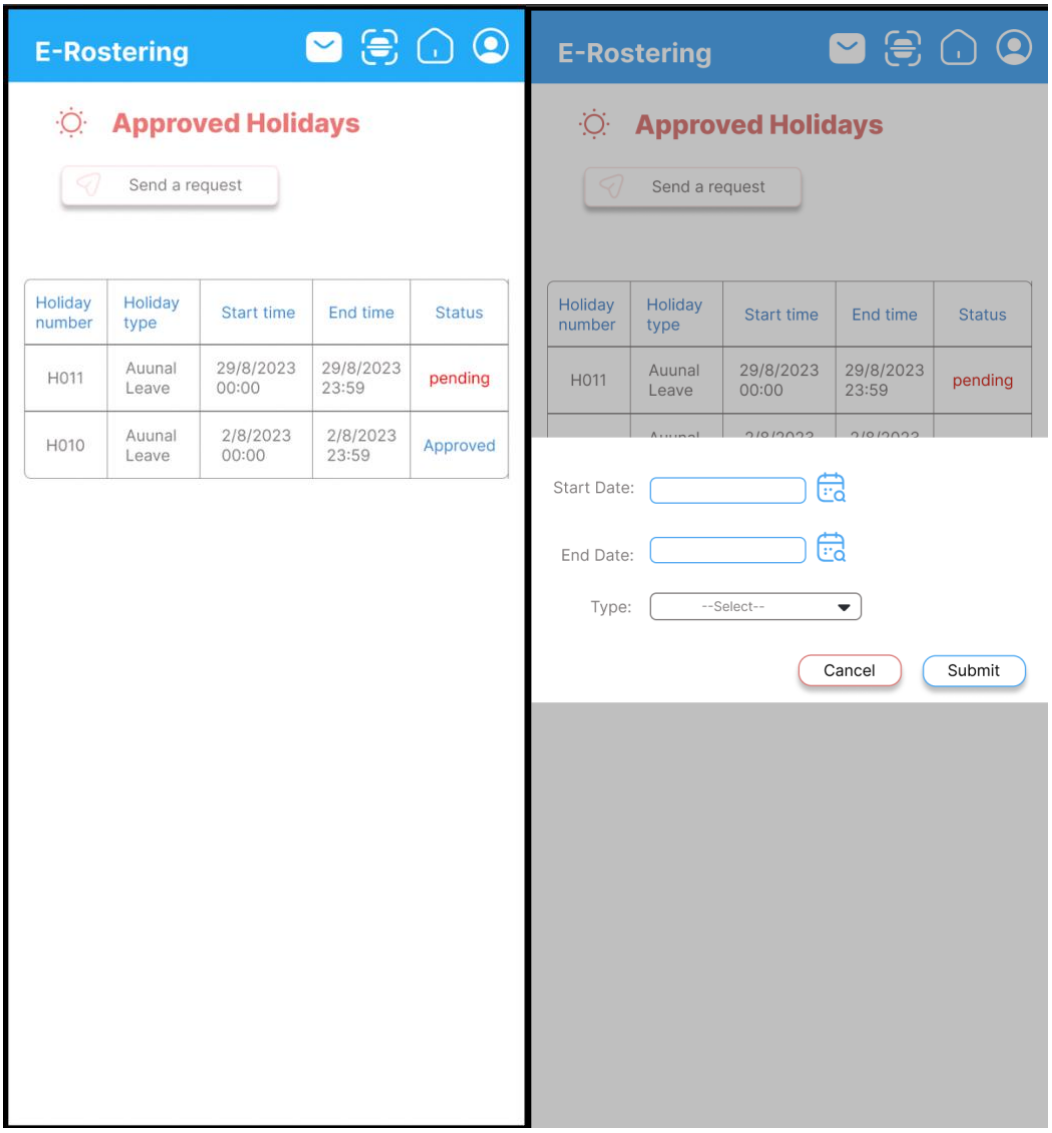

eRostering
Available Times
Start Date 29/7/2023 End Date 1/8/2023
Date Start Time End Time
1/8/2023 8:00 17:00
31/7/2023 8:00 17:00
30/7/2023 14:00 22:00
29/7/2023 Not Available Not Available

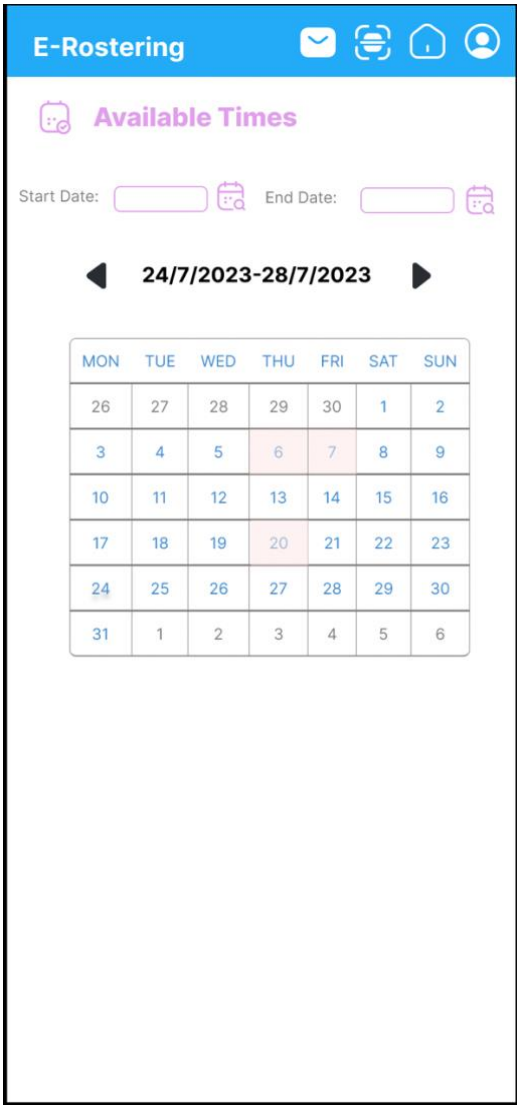
3.7.E-Rostering App Employees Figma




E-Rostering
Schedule
MON TUE WED THU FRI SAT SUN
26 27 28 29 30 1 2
3 4 5 6 7 8 9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
31 1 2 3 4 5 6
Shift
Training
Meeting
Seasonal Routine Training
24th, July, 2023
13:00-15:00
G building
Trainer: John Smith


E-Rostering
Clock-on & Clock-off Record
Start Date:
End Date:
Date Clock-in Clock-out Location Condition
26/7/2023 26/8/2023 08:01 Missing G building Abnormal
23/7/2023 23/8/2023 16:01 24/7/2023 01:02 H building Normal
20/7/2023 20/8/2023 16:11 21/7/2023 01:01 H building Normal
19/7/2023 19/8/2023 16:01 20/7/2023 01:02 B building Normal
16/7/2023 23/8/2023 08:08 16/7/2023 17:02 A building Normal
12/7/2023 12/8/2023 08:02 12/7/2023 17:09 H building Normal


E-Rostering
Approved Holidays
Send a request
Holiday number
Holiday type
Start time
End time
Status
H011
Auunal Leave
29/8/2023 00:00
29/8/2023 23:59
pending
H010
Auunal Leave
2/8/2023 00:00
2/8/2023 23:59
Approved
E-Rostering
Approved Holidays
Send a request
Holiday number
Holiday type
Start time
End time
Status
H011
Auunal Leave
29/8/2023 00:00
29/8/2023 23:59
pending
Start Date:
End Date:
Type:
--Select--
Cancel
Submit


E-Rostering
Available Times
Start Date:
End Date:
24/7/2023-28/7/2023
MON TUE WED THU FRI SAT SUN
26 27 28 29 30 1 2
3 4 5 6 7 8 9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
31 1 2 3 4 5 6

3. Test Results and Test Reports

For all the tests, OnTestCaseStart is set to Navigate(Home_Page), which means all tests start from the Home Page.

Holiday Request

- Send Valid Holiday Request

Step	Screen	Action
Count holidays for the logged in emoloyee	Home_Page	With({ theRecord: LookUp(Roster_Employee_Tables, Email = User().Email) }, Set(holidayBeforeTest, CountRows(Filter(Holiday_Tables, Employee_ID_Holiday.Employee_ID = theRecord.Employee_ID))))
Trace holidays before test	Home_Page	Trace("holidaysBeforeTest : " & holidayBeforeTest);
Go to Holidays	Home_Page	Select(Icon6)
Go to Holiday Request	Holidays	Select(Button2_3)

Generate 2 dates	Holiday_Request	Set(
		varRandomDate1,
		Date(
		2023,
		Rand()*12 + 1, // Random month
		Rand()*28 + 1 // Random day
		)
		);
		
		// Make varRandomDate2 be varRandomDate1 + 3 days
		Set(
		varRandomDate2,
		DateAdd(varRandomDate1, 3, TimeUnit.Days)
		);

Assert validity of dates	Holiday_Request	Set(varIsValid, varRandomDate1 <= varRandomDate2 && IsBlank(LookUp(Holiday_Tables, varRandomDate1 <= DateValue(End_Date_Holiday) && varRandomDate2 >= DateValue(Start_Date_Holiday) && LookUp(Roster_Employee_Tables, User().Email = Email).Employee_ID = Employee_ID_Holiday.Employee_ID && (HolidayStatus="Approved" || HolidayStatus="Pending")).Holiday_ID));
Assert	Holiday_Request	Assert(varIsValid, "Test failed: Invalid Dates")
Set 'HStart'.SelectedDate to a valid date	Holiday_Request	SetProperty(HStart.SelectedDate, varRandomDate1)
Select 'HStart'	Holiday_Request	Select(HStart)
Set 'HEnd'.SelectedDate to a valid date	Holiday_Request	SetProperty(HEnd.SelectedDate, varRandomDate2)
Select 'HEnd'	Holiday_Request	Select(HEnd)
Select submit button	Holiday_Request	Select(Button2_2)
Count rows after test	Submitted	With({ theRecord: LookUp(Roster_Employee_Tables, Email = User().Email) }, Set(holidayAfterTest, CountRows(Filter(Holiday_Tables, Employee_ID_Holiday.Employee_ID = theRecord.Employee_ID))))
Assert holiday records created	Submitted	Assert(holidayAfterTest = holidayBeforeTest + 1, "Holidays count incorrect. Expected : " &

		holidayBeforeTest + 1 & " Actual :" & holidayAfterTest)

- Send invalid request - Date 1 Larger - button disabled check

Step	Screen	Action
Go to Holidays	Home_Page	Select(Icon6)
Go to Holiday Request	Holidays	Select(Button2_3)
Set 'HStart'.SelectedDate to Date(2023,8,28)	Holiday_Request	SetProperty(HStart.SelectedDate, Date(2023,8,28))
Select 'HStart'	Holiday_Request	Select(HStart)
Set 'HEnd'.SelectedDate to Date(2023,8,26)	Holiday_Request	SetProperty(HEnd.SelectedDate, Date(2023,8,26))
Select 'HEnd'	Holiday_Request	Select(HEnd)
Assert “Submit” is disabled	Holiday_Request	Assert(Button2_2.DisplayMode = DisplayMode.Disabled, "Button2_2 should be disabled");

- Send invalid request - date 1 larger - check error message appears

Step	Screen	Action
Go to Holidays	Home_Page	Select(Icon6)
Go to Holiday Request	Holidays	Select(Button2_3)
Set 'HStart'.SelectedDate to Date(2023,8,28)	Holiday_Request	SetProperty(HStart.SelectedDate, Date(2023,8,28))
Select 'HStart'	Holiday_Request	Select(HStart)
Set 'HEnd'.SelectedDate to Date(2023,8,26)	Holiday_Request	SetProperty(HEnd.SelectedDate, Date(2023,8,26))
Select 'HEnd'	Holiday_Request	Select(HEnd)
Assert error message appears	Holiday_Request	Assert(Label20.Visible, "Error message should be visible");

- Send invalid request - Dates already taken - check button disabled

Step	Screen	Action
Go to Holidays	Home_Page	Select(Icon6)
Go to Holiday Request	Holidays	Select(Button2_3)
Generate 2 taken dates already taken	Holiday_Request	Set(varCurrentEmployeeID, LookUp(Roster_Employee_Tables, Email = User().Email).Employee_ID);

		Set(varInvalidHolidayRecord, First(Filter(Holiday_Tables, Employee_ID_Holiday.Employee_ID = varCurrentEmployeeID && (HolidayStatus="Approved"||HolidayStatus="Pending")))); Set(varInvalidDate1, varInvalidHolidayRecord.Start_Date_Holiday); Set(varInvalidDate2, varInvalidHolidayRecord.End_Date_Holiday);
Select 'HStart'	Holiday_Request	Select(HStart)
Set 'HStart'.SelectedDate	Holiday_Request	SetProperty(HStart.SelectedDate, varInvalidDate1)
Select 'HEnd'	Holiday_Request	Select(HEnd)
Set 'HEnd'.SelectedDate	Holiday_Request	SetProperty(HEnd.SelectedDate, varInvalidDate2)
Assert “Submit” is disabled	Holiday_Request	Assert(Button2_2.DisplayMode = DisplayMode.Disabled, "Button2_2 should be disabled");

- Send invalid request - Dates already taken - check error message visible

Step	Screen	Action
Go to Holidays	Home_Page	Select(Icon6)
Go to Holiday Request	Holidays	Select(Button2_3)

Generate 2 taken dates already taken	Holiday_Request	Set(
		  varCurrentEmployeeID,
		  LookUp(Roster_Employee_Tables, Email = User().Email).Employee_ID
		);
		Set(
		  varInvalidHolidayRecord,
		  First(
		  Filter(
		  Holiday_Tables,
		  Employee_ID_Holiday.Employee_ID = varCurrentEmployeeID
		  &&
		(HolidayStatus="Approved"||HolidayStatus="Pending")
		  )
		  )
		);
		
		Set(
		  varInvalidDate1,
		  varInvalidHolidayRecord.Start_Date_Holiday
		);
		
		Set(
		  varInvalidDate2,
		  varInvalidHolidayRecord.End_Date_Holiday
		);

Select 'HStart'	Holiday_Request	Select(HStart)
Set 'HStart'.SelectedDate	Holiday_Request	SetProperty(HStart.SelectedDate, varInvalidDate1)
Select 'HEnd'	Holiday_Request	Select(HEnd)
Set 'HEnd'.SelectedDate	Holiday_Request	SetProperty(HEnd.SelectedDate, varInvalidDate2)
Assert error message appears	Holiday_Request	Assert(Label20.Visible, "Error message should be visible");

Schedule

- Check Task is highlighted on click

Step	Screen	Action
Go to Schedule	Home_Page	Select(Icon4)
Select 'Task'	Schedule	Select(Task)
Check task is highlighted and activity not		Assert(Task.BorderColor = ColorValue("Blue") && Activity.BorderColor = ColorValue("Gray"),"Task should be highlighted")

- Stress test - check Activity is highlighted after multiple clicks

Step	Screen	Action
Go to Schedule	Home_Page	Select(Icon4)
Select 'Task'	Schedule	Select(Task)
Select 'Activity'	Schedule	Select(Activity)
Select 'Task'	Schedule	Select(Task)
Select 'Activity'	Schedule	Select(Activity)
Assert activity highlighted		Assert(Task.BorderColor = ColorValue("Gray") && Activity.BorderColor = ColorValue("Blue"),"Activity should be highlighted")

- Check the right elements are visbile for Task

Step	Screen	Action
Go to Schedule	Home_Page	Select(Icon4)
Select 'Task'	Schedule	Select(Task)
Check what elements are visible		Assert(Icon10.Visible && Icon11.Visible && Label20_2.Visible && Label20_3.Visible && !Gallery2.Visible,"Not the right elements are visible for Task")

- Check the right elements are visbile for Activity

Step	Screen	Action
Go to Schedule	Home_Page	Select(Icon4)
Select 'Activity'	Schedule	Select(Activity)
Assert		Assert(!Icon10.Visible && !Icon11.Visible && !Label20_2.Visible && !Label20_3.Visible && Gallery2.Visible,"Not the right elements are visible for Activity")

- Check correct data identified for a date and task

Step	Screen	Action
Go to Schedule	Home_Page	Select(Icon4)
Set 'Date'.SelectedDate to Date(2023,8,29)	Schedule	SetProperty(Date.SelectedDate, Date(2023,8,29))
Select 'Date'	Schedule	Select(Date)
Select 'Task'	Schedule	Select(Task)
Check if it matches the records	Schedule	Assert(Label20_3.Text="Some note", "Invalid data displayed: " & Label20_3.Text& " ,Expected: Some note ")

Holiday

- Check if holidays are for logged in employee

Step	Screen	Action
Go to Holiday	Home_Page	Select(Icon6)
Calculate expected count	Holiday	Set(varExpectedCount, CountRows(With({ theRecord: LookUp(Roster_Employee_Tables, Email = User().Email) }, Filter(Holiday_Tables, Employee_ID_Holiday.Employee_ID = theRecord.Employee_ID

		))));
Count rows in gallery	Holiday	Set(varDisplayedCount, CountRows(Gallery14.AllItems));
Assert		Assert(varExpectedCount = varDisplayedCount,"not logged in users data. Expected rows: ")

Clock in records

- logged in employees values in galery with applied date filters

Step	**Screen**	**Action**
Go to Clock Records	Home_Page	Select(Icon5)
Set 'StartDate'.SelectedDate to Date(2023,8,10)	Clock	SetProperty(StartDate.SelectedDate, Date(2023,8,10))
Select 'StartDate'	Clock	Select(StartDate)
Set 'EndDate'.SelectedDate to Date(2023,8,27)	Clock	SetProperty(EndDate.SelectedDate, Date(2023,8,27))
Select 'EndDate'	Clock	Select(EndDate)
Create variable for employee and selected dates	Clock	Set(varLoggedInEmployee, LookUp(Roster_Employee_Tables, Email = User().Email)); Set(varStartDate, StartDate.SelectedDate); Set(varEndDate, EndDate.SelectedDate);
// Go through each record in the galery	Clock	// Go through each record in the gallery and check if they match the conditions ForAll(Gallery1.AllItems, Assert(Employee_ID_clock.Employee_ID = varLoggedInEmployee.Employee_ID && ClockOn_Time >= varStartDate

		&& ClockOn_Time <= varEndDate, "Test Failed: Incorrect record displayed in the gallery!"& varLoggedInEmployee.Employee_ID & varStartDate & varEndDate))

- Total hours

Step	Screen	Action
Go to Clock Records	Home_Page	Select(Icon5)
Set 'StartDate'.SelectedDate to Date(2023,8,9)	Clock	SetProperty(StartDate.SelectedDate, Date(2023,8,9))
Select 'StartDate'	Clock	Select(StartDate)
Set 'EndDate'.SelectedDate to Date(2023,8,27)	Clock	SetProperty(EndDate.SelectedDate, Date(2023,8,27))
Select 'EndDate'	Clock	Select(EndDate)
Compute the expected hours	Clock	Set(varComputedTotalHours, Sum(ForAll(Gallery1.AllItems, DateDiff(ClockOn_Time, ClockOff_Time, TimeUnit.Hours)), Value // This extracts the computed hour value from each item));
Format expected hours	Clock	Set(varExpectedTotalHours, "Total: " & varComputedTotalHours & " hours");
Assert	Clock	Assert(varExpectedTotalHours = Label24.Text, "Test Failed: Total hours is incorrect!")

4. Code Listing – E-Rostering Employees

App

OnStart =

```
Set(_FirstDateOfMonth,Date(Year(Today()),Month(Today()),1));
ClearCollect(DaysWeek,{ShortName: "Mon"},{ShortName: "Tue"},{ShortName: "Wed"},
{ShortName: "Thu"}, {ShortName: "Fri"}, {ShortName: "Sat"}, {ShortName: "Sun"});
With(
   {
      theRecord: LookUp(Roster_Employee_Tables, Email = User().Email)
   },
   ClearCollect(
      colHourDifferences,
      ForAll(
         Filter(ClockInOffRecord_Tables, Employee_ID.Employee_ID =
theRecord.Employee_ID),
         {
            EmployeeID: Employee_ID.Employee_ID,
            ClockOnTime: cr7c7_clockontime,
            HourDifference: DateDiff(cr7c7_clockontime, cr7c7_clockofftime, TimeUnit.Hours)
         }
      )
   )
)
```

Home Page

Display User's Name

Text =

```
With(
   {
      theRecord: LookUp(Roster_Employee_Tables, Email = User().Email)
   },
   theRecord.FirstName & " " & theRecord.LastName
)
```

Schedule

Task and Activity Buttons

```
OnSelect = UpdateContext({ IsActivitySelected: false, IsTaskSelected: true })
```

```
BorderColor =  If(IsTaskSelected, ColorValue("Blue"), ColorValue("Gray"))
```

```
OnSelect = UpdateContext({ IsActivitySelected: true, IsTaskSelected: false})
```

```
BorderColor = If(IsActivitySelected, ColorValue("Blue"), ColorValue("Gray"))
```

Task Components

```
Visible = IsTaskSelected
```

Activity Components

```
Visible = IsActivitySelected
```

Task

```
Text = Text(
   First(
      Filter(
         Shift_Tables,
         Shift_Date = Date.SelectedDate &&
         cr7c7_status = "Published" &&
         Employee_ID_ShiftTable.Employee_ID = LookUp(
            Roster_Employee_Tables,
            Email = User().Email
         ).Employee_ID
      )
   ).Unit
)
```

```
Text =     First(
      Filter(
         Shift_Tables,
         Shift_Date = Date.SelectedDate &&
         cr7c7_status = "Published" &&
         Employee_ID_ShiftTable.Employee_ID = LookUp(
            Roster_Employee_Tables,
            Email = User().Email
         ).Employee_ID
      )
   ).Notes
)
```

Activity

```
Items = Filter(Activity_Tables, Meeting_ID in ForAll(Filter(AssignActivity_Tables,
Employee_ID_AssignActivity.Employee_ID = LookUp(Roster_Employee_Tables, Email =
```

```
User().Email).Employee_ID), Activity.Meeting_ID) &&
DateValue(Text(StartDateTime_Activity, "[$-en-US]yyyy-mm-dd"))  = Date.SelectedDate)

Text = Text(ThisItem.StartDateTime_Activity, "[$-en-US]hh:mm:ss AM/PM") & " - " &
Text(ThisItem.EndDateTime_Activity, "[$-en-US]hh:mm:ss AM/PM")

Text = ThisItem.Location_Activity

Text = ThisItem.Activity_Name

Text = ThisItem.Activity_Type
```

Clock In Records

Table

```
Items =     With(
   {
      theRecord: LookUp(Roster_Employee_Tables, Email = User().Email)
   },
   Sort(
      Filter(
         ClockInOffRecord_Tables,
         Employee_ID.Employee_ID = theRecord.Employee_ID
         && cr7c7_clockontime >= StartDate.SelectedDate
         && cr7c7_clockontime <= EndDate.SelectedDate
      ),
      cr7c7_clockontime, SortOrder.Ascending
   )
)

Text = ThisItem.'ClockOnTime '

Text = ThisItem.ClockOnLocation

Text = ThisItem.ClockOffTime

Text = ThisItem.'ClockOffLocation '
```

Total Hours

```
Text = "Total: "& Sum(
   Filter(
      colHourDifferences,
      ClockOnTime >= StartDate.SelectedDate && ClockOnTime <= EndDate.SelectedDate
   ),
   HourDifference
```

```
) & " hours"
```

Holiday

```
Items =

With(
    {
        theRecord: LookUp(Roster_Employee_Tables, Email = User().Email)
    },
    Sort(
        Filter(
            Holiday_Tables,
            Employee_ID_Holiday.Employee_ID = theRecord.Employee_ID
        ),
        Start_Date_Holiday, SortOrder.Ascending
    )
)

Text = ThisItem.cr7c7_name

Text = Text(DateValue(ThisItem.Start_Date_Holiday),"dd/mm/yyyy" )

Text = Text(DateValue(ThisItem.End_Date_Holiday),"dd/mm/yyyy")

Text = ThisItem.HolidayStatus

Color = If(
    ThisItem.HolidayStatus= "Approved",
     Color.Green,
    If(
        ThisItem.HolidayStatus = "Declined",
        Color.Red,
        Color.DimGrey
    )
)
```

Holiday Request

```
OnSelect =

With(
    {
        theRecord: LookUp(Roster_Employee_Tables, Email = User().Email)
    },
        Patch(
            Holiday_Tables,
```

```
        Defaults(Holiday_Tables),
        {
            Employee_ID_Holiday: theRecord,
            Start_Date_Holiday: HStart.SelectedDate,
            End_Date_Holiday: HEnd.SelectedDate,
            HolidayStatus: "Pending"
        }
    )
  );

Reset(HStart);
Reset(HEnd);
Navigate(Submitted);
```

Button Disabled

```
DisplayMode =

If(HStart.SelectedDate>HEnd.SelectedDate,DisplayMode.Disabled,If(IsBlank(LookUp(Holiday
_Tables,HStart.SelectedDate <= DateValue(End_Date_Holiday) && HEnd.SelectedDate >=
DateValue(Start_Date_Holiday) && LookUp(Roster_Employee_Tables, User().Email =
Email).Employee_ID = Employee_ID_Holiday.Employee_ID
&&(HolidayStatus="Approved"||HolidayStatus="Pending")).Holiday_ID),
DisplayMode.Edit,DisplayMode.Disabled))
```

Error Message

```
Visible =

Button2_2.DisplayMode = DisplayMode.Disabled
```

Available Times

Month text

```
Text =

Text(_FirstDateOfMonth, "mmmm, yyyy")
```

Arrows

```
OnSelect =

Set(_FirstDateOfMonth,Date(Year(_FirstDateOfMonth),Month(_FirstDateOfMonth)+1,1));
```

```
OnSelect =
```

```
Set(_FirstDateOfMonth,Date(Year(_FirstDateOfMonth),Month(_FirstDateOfMonth)-1,1));
```

Weekdays

```
Items = Calendar.WeekdaysShort()
```

```
Text = ThisItem.Value
```

Calendar

```
Items =
```

```
ForAll(Sequence(42),Value+_FirstDateOfMonth-Weekday(_FirstDateOfMonth))
```

```
Text = Day(ThisItem.Value)
```

Shift Calendar

```
Items =
Filter(Shift_Tables,Shift_Date=DateValue(DateAdd(ThisItem.Value-1,1,TimeUnit.Days))&&
'Status (cr7c7_status)' ="Published" &&Employee_ID_ShiftTable.Employee_ID=With(
   {
      theRecord: LookUp(Roster_Employee_Tables, Email = User().Email)
   },
   theRecord.Employee_ID
) )
```

```
Text = ThisItem.Shift_Type
```

```
Fill =
If(
   ThisItem.Shift_Type = 'Shift_Type (Shift_Tables)'.Night,
   RGBA(102, 178, 255, 0.5), //blue
   If(
      ThisItem.Shift_Type ='Shift_Type (Shift_Tables)'.Day,
      RGBA(255, 178, 102, 0.5),

         RGBA(0, 0, 0, 0)  // Default color if none of the conditions match

   )
)
```

Holiday Calendar

```
Items =
```

```
Filter(
   Holiday_Tables,
   Start_Date_Holiday < DateAdd(ThisItem.Value, 1, TimeUnit.Days) &&
   End_Date_Holiday >= ThisItem.Value &&
   Employee_ID_Holiday.Employee_ID = With(
   {
      theRecord: LookUp(Roster_Employee_Tables, Email = User().Email)
   },
   theRecord.Employee_ID
) &&
   HolidayStatus<>"Declined")

Text = ThisItem.HolidayStatus
```

Clock On/off

Clock On

```
OnSelect =

Set(varNFCtag, ReadNFC());
Set(userID,LookUp(Roster_Employee_Tables, Email = User().Email));


With(
   {
      theRecord: LookUp(Roster_Employee_Tables, Email = User().Email)
   },

Patch(
   ' ClockOnOffRecord_Tables',
   Defaults(' ClockOnOffRecord_Tables'),
   {
      Employee_ID_clock: theRecord,
      ClockOn_Location: Last(varNFCtag.NDEFRecords).Text,
     ClockOn_Time: Now(),
     Status_clockon:"Yes",
   Status_clockoff:"No"

   })
   );
```

Clock Off

```
OnSelect =

Set(varNFCtag, ReadNFC());
```

```
Set(userID, LookUp(Roster_Employee_Tables, Email = User().Email));
With(
   {
      theRecord: LookUp(Roster_Employee_Tables, Email = User().Email)
   },
With(
   { closestRecord: LookUp(
      ' ClockOnOffRecord_Tables',
      Employee_ID_clock.Employee_ID = theRecord.Employee_ID &&
      DateDiff(ClockOn_Time, Now(), TimeUnit.Hours) < 16 &&
         Status_clockoff = "No"
   ) },
   Patch(
      ' ClockOnOffRecord_Tables',
      closestRecord,
      {
         ClockOff_Location: Last(varNFCtag.NDEFRecords).Text,
         ClockOff_Time: Now(),
         Status_clockoff:"Yes"
      }
   )
)
);
```