Cormac Mattimoe

C16386071

Submitted in partial fulfillment of the requirements for the degree of:

B.Sc. in Business Computing

Technological University Dublin

Year 4

Supervised by – Farrah Higgins

This is an original work. All References and assistance are acknowledged.

Signed: Cormac Mattimoe

Date 30/04/2021

GitHub Repository: https://github.com/cormacmattimoe/BedMS-FYP.git

Contents

Acknowledgments

I'd like to take this opportunity to thank everyone who has helped me get to this point in my college career and in this project.

First and foremost, I'd like to express my sincere thanks to Farrah, my project supervisor, for her invaluable assistance during the entire final year project phase. I appreciate all of the encouragement and constructive criticism that helped me stay on track with this project and develop my preparation and time management skills.

Also, I would like to thank Jenny for her feedback as a second reader and for her role as a second reader she gave me great advice to enhance the complexity and UI of my app.

Finally, I'd like to express my gratitude to my family, girlfriend, and friends, as well as everyone else who assisted me in completing this project, for keeping me focused and inspiring me during this trying year. Especially to my parents, who have been extremely supportive throughout the year, considering the fact that I had to move back home and continue the year from there due to Covid.

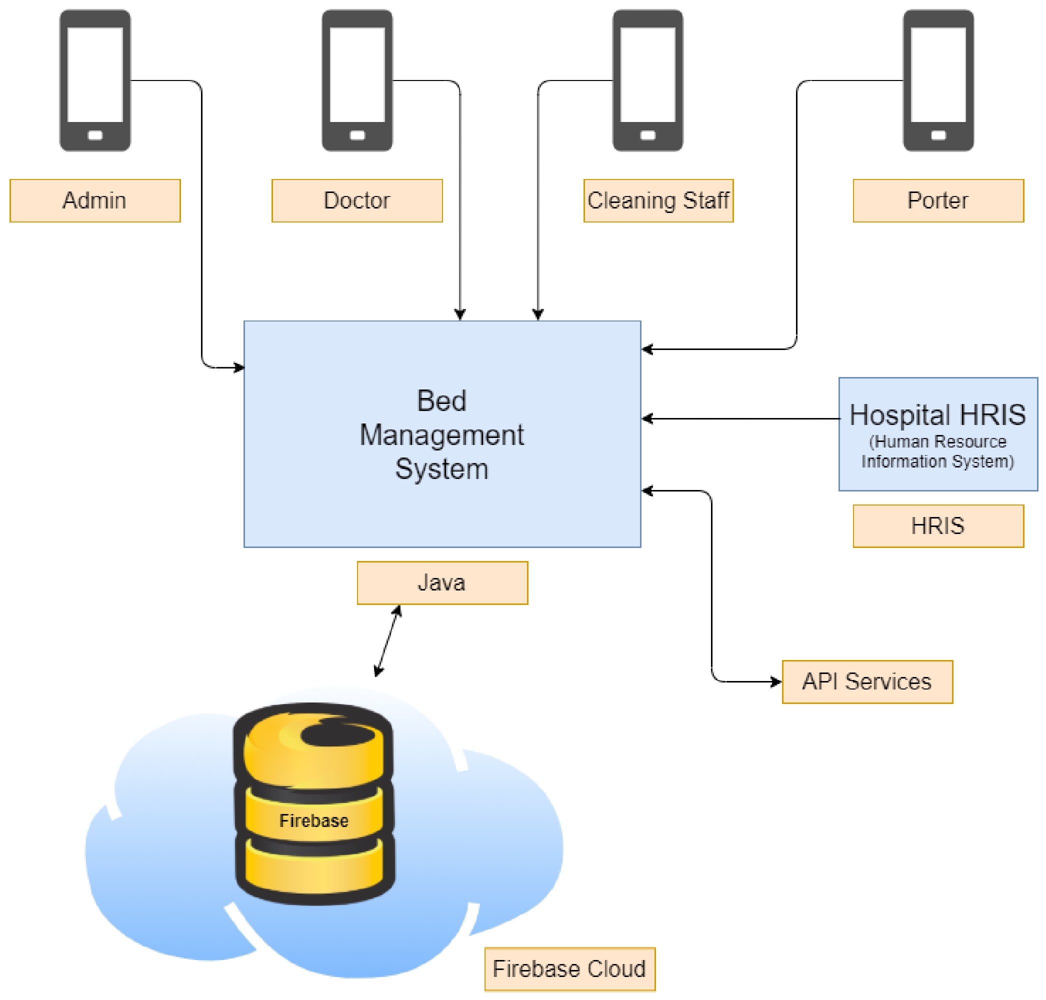
Abstract

Bed Management System will create an efficient solution for bed management in a modern hospital.

An already acute bottleneck is further exacerbated by the onset of the Covid crisis. My solution will create a system that simplifies and streamlines the allocation of a patient to an appropriate bed. The solution will bring substantial benefits to the patient, the medical staff, the hospital administration staff and the overall efficacy of the hospital.

The solution will utilise modern technology architecture including Cloud, Mobile technologies and provide deep analytics into the bed utilization and wait times.

The project will be run in an agile manner establishing a solid core architecture and expanding the business functionality of it over several sprints. This solution will utilise a number of different technologies best suited to the business requirement.



1. Introduction

1.1 Project Title: Bed Management System

This project delivers an efficient solution for bed management in a modern hospital. An already acute bottleneck is further exacerbated by the onset of the Covid crisis. My solution will create a system that simplifies and streamlines the allocation of a patient to an appropriate bed. The solution will bring substantial benefits to the patient, the medical staff, the hospital administration staff and the overall efficacy of the hospital.

1.2 Objectives of Project

My solution will focus on simplifying the bed management process by having clear real-time status on every-bed. It will provide information to doctors real-time about what beds are available so that new patients can be admitted.

To drive efficacy, the project will report on 3 key wait-times in the hospital:

1. The time between when a patient is admitted into the hospital and when they see their doctor.
2. The time delays from Decision to Admit (DTA) to the actual time the patient arrives at their bed on the ward.
3. The time the bed is unavailable between patients – from when the bed is vacated by a patient and marked Ready for Cleaning – to when it is made available again.

My solution is designed to allow extension to other key times and metrics.

1.3 Business Case

The 2020 COVID-19 pandemic has brought renewed attention to the capacity and efficacy of our health system. Over the past several years Ireland has bounced from one health crisis to the next with a flow of alarming headlines.

The availability of beds in a hospital system, and the efficient management of the pool of beds, are two critical elements of this problem.

Like the airline industry, one of the most critical parts of successful operations is "turnaround time": "When the plane is sitting, they are not making money". In the hospital scenario, equal focus can be placed on bed management: if beds are not being used, then patients may not be receiving the best treatment and the hospital is not operating effectively. "Managing a bed crisis" by Nick Egan (J3Accid Emerg Med 1999;16:145-146) concludes that ineffective bed management leads to the following issues:

- cause gridlock in A&E.
- increase waiting times in A&E.
- increase length of stay in A&E.
- increase morbidity.
- reduce staff availability.
- increase A&E costs.
- create a negative patient appreciation of A&E - "mud sticks".

Additionally, there is a huge demand on the Health Service to provide the best healthcare and people's expectations of successful and hassle-free treatment is very high.

Having worked as a porter in a local hospital and experienced the challenges first-hand, I decided to create a solution for this problem in my final year project.

My research into this problem at St. James Hospital in Dublin confirmed that this is indeed a critical problem and new solutions are needed. I discovered that St James has a mainframe system to help govern use and allocation of beds. However, the interface to this system is limited to fixed locations in the Nurse's station in the Ward, causing significant delays in accessing information and excess work for staff. The system is difficult to use leading to underuse and errors.

I also saw that smaller regional hospitals, including Our Lady's Hospital in Navan, have no such system and resort to managing bed occupancy via three sets of boards across the room recording all the bed occupancies and vacancies throughout the hospital. This type of manual solution is very time consuming and does not display in real time when a bed becomes available or occupied.

My solution tackles the above by reporting on how long a patient is waiting to see a doctor from time of admission. This metric has a huge impact on the patient experience – and having it clearly reported will help drive the delays down.

My solution also focuses on how to reduce the time from when the patient/client is documented by the doctor as being admitted (Decision to Admit (DTA) to the actual time the patient/client arrives at their bed on the ward.

This is where I see a tremendous opportunity to streamline information about beds and their occupancy, and make that information available to doctors, nurses, and administration staff in a modern simple solution.

The solution would be real-time and give the facility to look up bed availability and book a bed - as they are standing right beside the patient. Allowing the doctor to book a bed on a ward will save the endless phone calls and trips to wards to see which bed is available or not. This will reduce the time delays in organising a bed for their patient/client - resulting in better experience for the patient and lower workload for staff, and better utilisation of beds.

Bed cleaning is one of the most important aspects in the operations of a hospital especially during a Covid 19 pandemic. Keeping a good record of the beds which are being cleaned and the time that elapses from when a bed is “Available for Cleaning” to when it is cleaned and set to “Open” is important and will be included in this solution.

1.4 Overview of technologies used in Project.

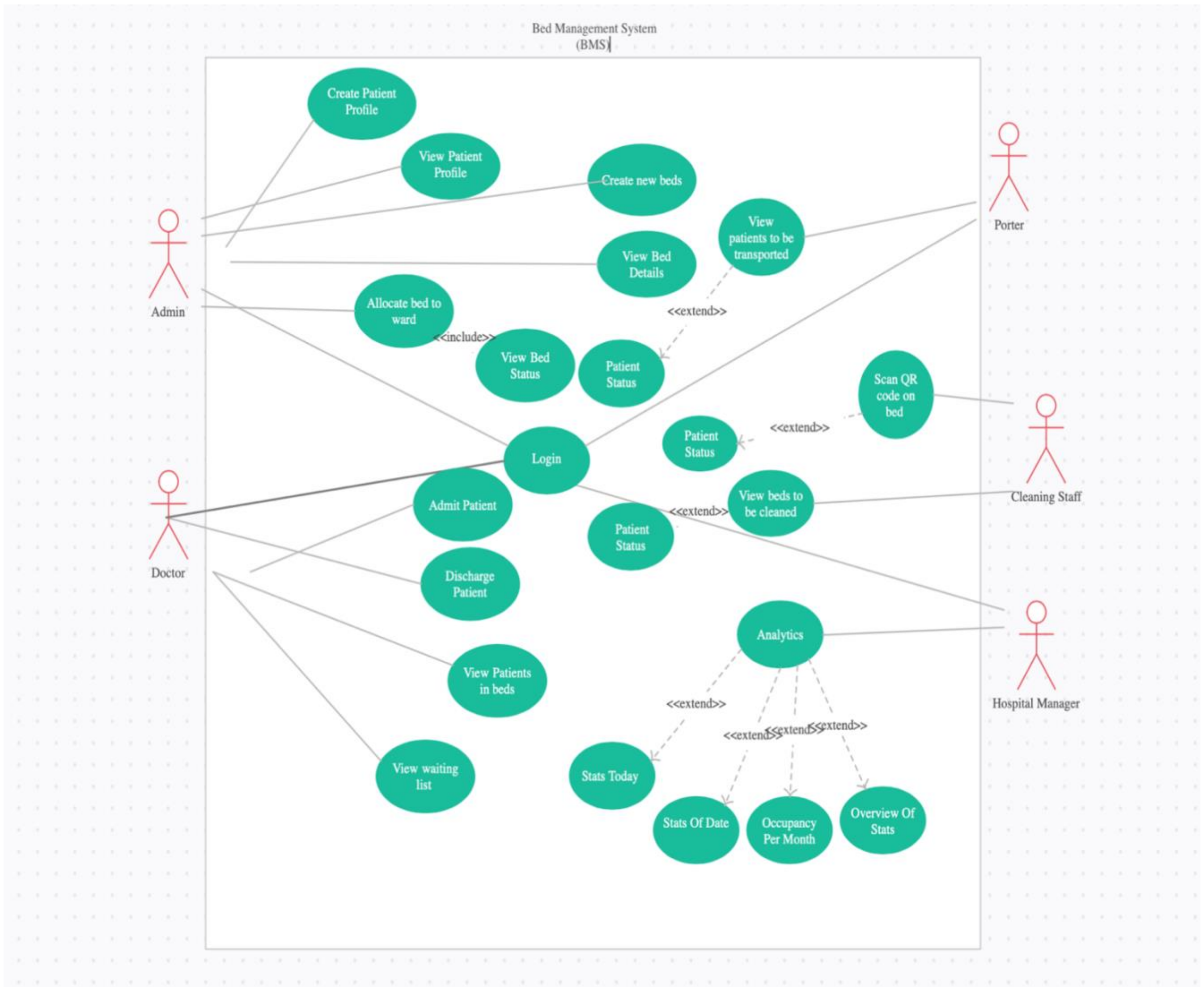
The solution will use a modern technology architecture including Cloud, Mobile technologies and provide deep analytics into the bed utilisation and wait times.

The project will be run in an agile manner establishing a solid core architecture and expanding the business functionality of it over several sprints.

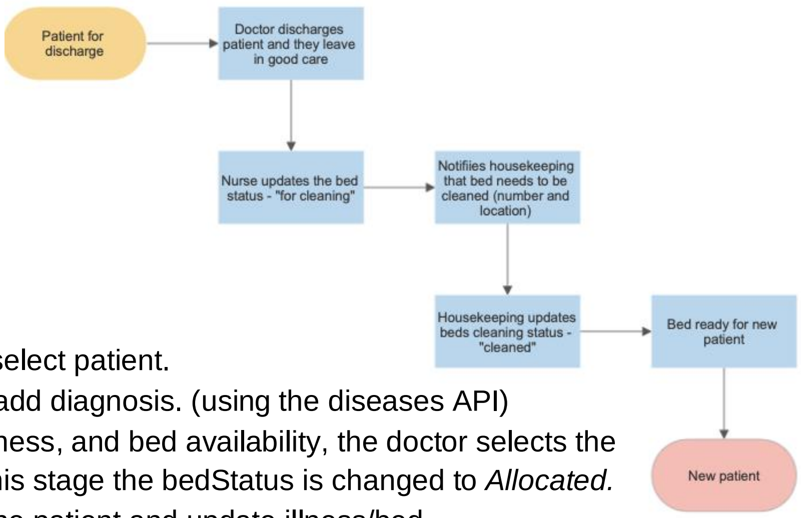
2. Requirements Capture and Analysis

1.1 Business Actors

UML Use Case Diagram

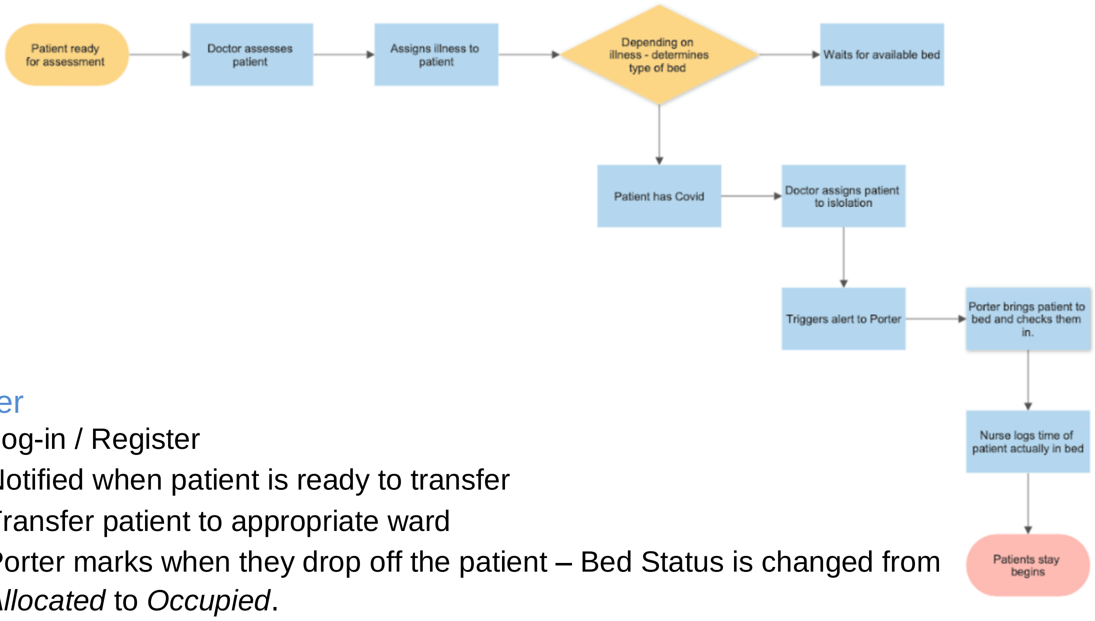


Functional requirements by Role


Patient for discharge
Doctor discharges patient and they leave in good care
Nurse updates the bed status - "for cleaning"
Notifiies housekeeping that bed needs to be cleaned (number and location)
Housekeeping updates beds cleaning status - "cleaned"
Bed ready for new patient
New patient

Doctor

- Log-in / Register
- Review Patient waiting list and select patient.
- Update patient information and add diagnosis. (using the diseases API)
- Depending upon the patient's illness, and bed availability, the doctor selects the most appropriate bed-type. At this stage the bedStatus is changed to *Allocated.*
- During stay, Doctor can check the patient and update illness/bed.
- Doctor discharges patient when the patient is well enough.
- Overview of beds capacity - by type and by ward.


Patient ready for assessment
Doctor assesses patient
Assigns illness to patient
Depending on illness - determines type of bed
Waits for available bed
Patient has Covid
Doctor assigns patient to isiolation
Triggers alert to Porter
Porter brings patient to bed and checks them in.
Nurse logs time of patient actually in bed
Patients stay begins

Porter

- Log-in / Register
- Notified when patient is ready to transfer
- Transfer patient to appropriate ward
- Porter marks when they drop off the patient – Bed Status is changed from *Allocated* to *Occupied*.

Hospital Manager

- Log-in / Register

Charts displaying:

- Stats : as of today.
- Stats : as of a certain date.
- Bed Occupancy Per Month.
- Overview of Key Wait-time stats.

MIS on wait-times:

- *View report on waitTime for*
  a. *patients waiting on porter*
  b. *bed waiting on cleaning*
  c. *people waiting to see doctor (if time permits)*

Bed Utilisation

- *Utilisation of beds over each month*
- *Overview of bed status for ward: By type By date*

Central Admin

- Log-in / Register

Patient Administration:

- Admits patient - gather their details. Patient then waits on doctor
- Update patient details

Employee Administration

- Manage Employees - add employee/ view employee
- Admin will enter new employees into the system
- Employee receives email and completes registration themselves.
- Show list of all employees and role.

Bed Administration

- Manage Beds :   add/ allocate / view beds
- View bed list by type and by ward
- Add new beds and how many they would like to add.
- Allocate bed to ward.

Security considerations

Registration

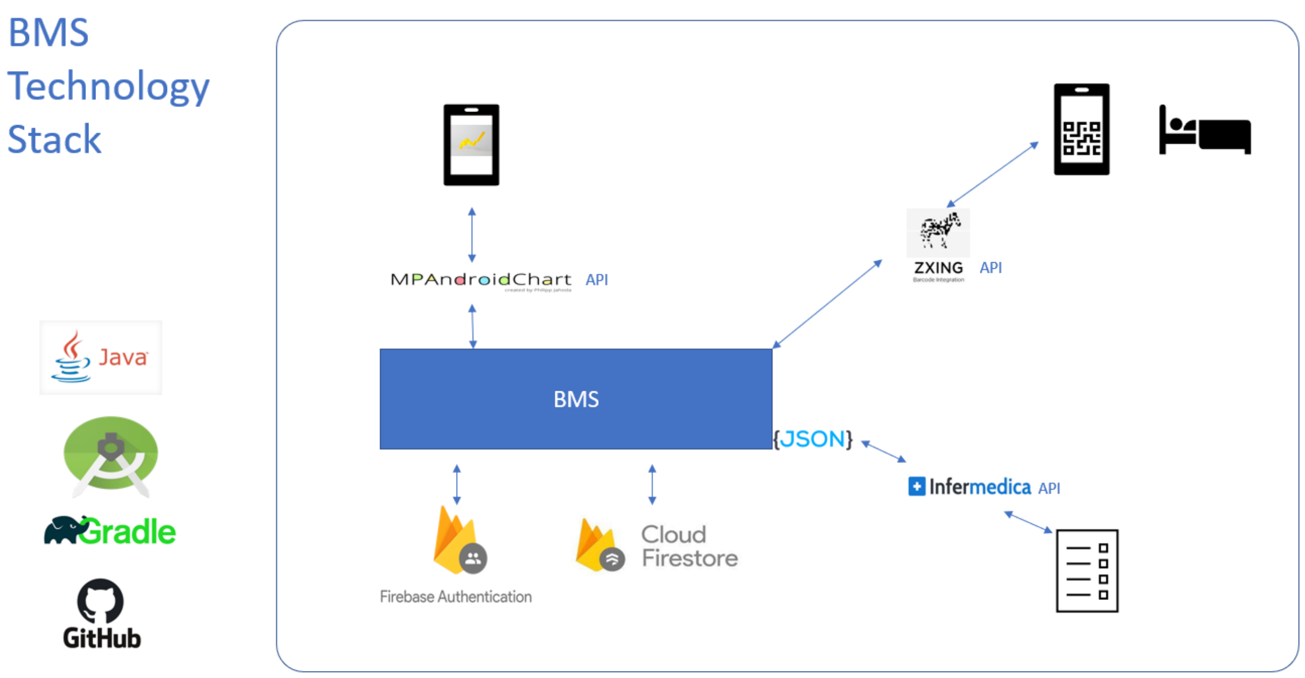
- Hospital Administration set up new employee details on the central system.
- Employee receives link to complete Registration online
- The employee clicks on the link to register
- Employee is asked for additional information
- They then set up secure valid password
- Once validated, the password is encrypted and stored in the database.
- Employee status is set to "registered "
- Date and time recorded

Login

- Employee asked for email and password
- At login - look up employee database for role, and link to appropriate screen
- Record date and time

3. Systems Design

3.1 System Architecture



3.2 Choice of Technologies

I used several technologies in the design and implementation of this application. The primary technologies used are listed below and explained. The main programming language I used was Java and the main database was Firebase.

Application Technologies

Android Studio

The BMS application was built solely in Android Studio using the Android SDK. Android Studio is a platform built off the IntelliJ IDEA and is the official IDE for Android Development. Android Studio offers a substantial suite of tools for application development, with built in Testing tools and Frameworks, a Gradle-based build system, an emulator and extensive Lint tools.

I carried out the testing of my application on my parents Android phones which offers much easier access to build in Android features such as using the camera. Testing on this phone in addition to using the emulator allowed me to carry out real world testing of the application.

I chose to use Android studio with Java as it was a framework and development model which I was quite familiar with both from my own personal experience as well as using it for the Mobile Application Development module.

I utilized several open-source Java plug-in libraries including:
- MP Android Charts
- Zxing Barcode Qr Reader
- Infermedica Disease API

Java

My app was written in Java. This is my primary language, which I first commenced learning from the beginning of this journey. In order to advance the complexities inside my app, I felt at ease using Java, which is commonly used in the Android development process.

JSON

JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. I used JSON as it allowed me to parse my object and create a list of conditions using Infermedica's API. As a result, I was able to send data between my app and different web services.

Zxing Barcode Reader

I used ZXING open-source tool to generate and capture QR Codes for the beds.

These are used by Admin staff when creating the bed, and by cleaning staff when cleaning the bed.

Google Firebase

Google Firebase is a mobile and web application development suite, that offers numerous features to help build applications fast, without managing infrastructure. I choose to use a few Firebase technologies in my application as they work well in harmony as it is a well-integrated single platform where the different products can share data and insights between each other.

Firebase Authentication, which allows users to register and login, and Cloud Firestore Database were the Firebase platforms I wanted to implement.

Firebase Authentication

Firebase Authentication offers an easy registration and sign-in with any platform. It provides an end-to-end identity solution supporting many types of sign in options such as email and password, Google sign in and Facebook log in.

Firebase Authentication offers comprehensive security features and advanced encryption of passwords meaning that not even the admin of the database can view a user's password.

Cloud Firestore

Cloud Firestore is a NoSQL database stored in the cloud. It allows for structuring and querying of data in the way in which a user chooses. Firestore allows for the building of truly serverless applications, with strong security and is easily linked up with Firebase Authenticator.

I decided to go with Cloud Firestore after quite a bit of testing and trailing of other technologies. Originally, I had planned to use MySQL for my application. I decided on Firestore as I felt it was best fit for my purpose. My application relies on continuous storing of quite a large amount of data on a secure cloud platform - and Firestore was the most appropriate tool for this and communicated seamlessly with the other Firebase products already implemented in the application.

MP Android Chart

This powerful library allowed me to include pie charts, line charts, and bar charts in my application to show data from various bed and patient statuses. MP Android Chart is a visual and interactive way to view data in various formats, such as decimal and percentages.

Medical diseases API

Infermedica is a powerful Medical Diagnosis suite.

I am using their API to check for patient illness and tested the functionality using Postman. This feature will ensure patient diagnosis is consistently applied. This ensures correct terminology used across the application and opens up areas for further automation.

Build Automation

Gradle is an advanced build toolkit, to automate and manage the build process, while allowing you to define flexible custom build configurations.

Version Control

Git and GitHub

I used GIT for local version control throughout the project and stored these on GitHub. Git and GitHub are very powerful tools which I found extremely useful in the development process to store my changes and develop new feature branches.

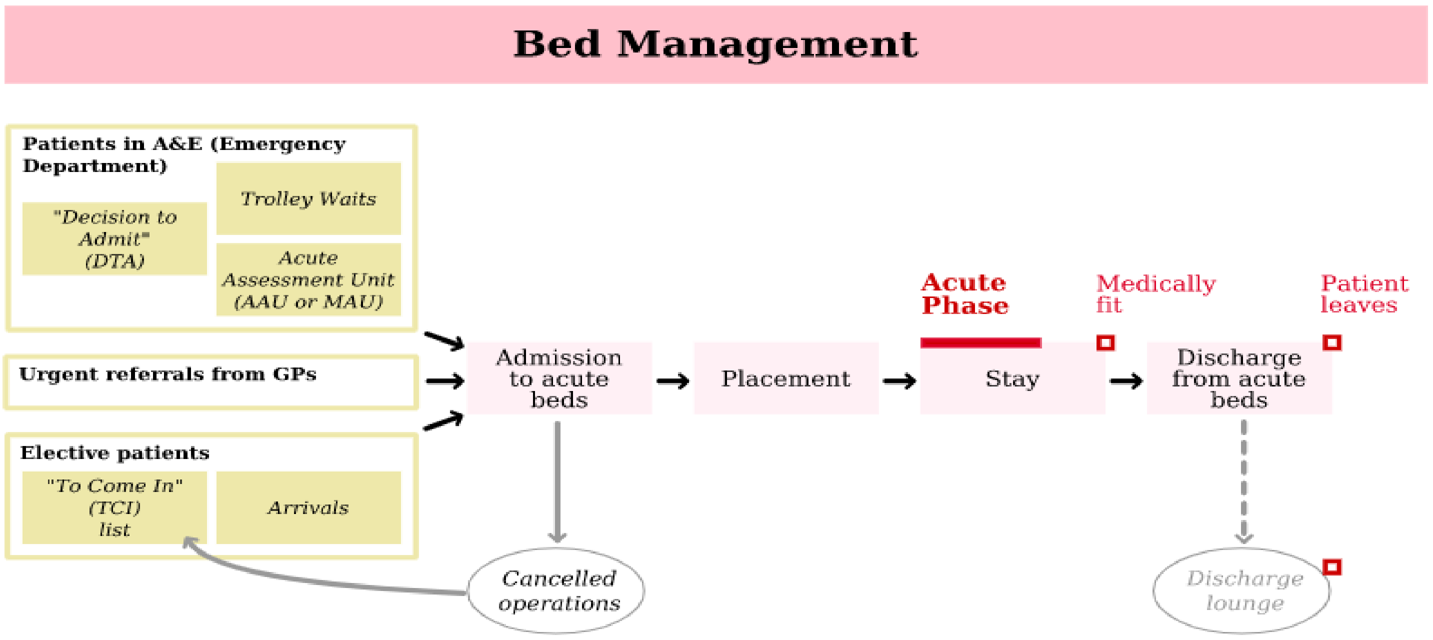
GitHub provided me with the capability to keep track of any improvements I made while also storing my code in a secure repository thanks to GitHub. In a situation where new releases were causing problems, it also ensured that I had regular access to previous versions. Since GitHub is cloud-based, it can be accessed from anywhere, which was extremely helpful when developing my app. Android Studio's Version Control Service (VCS) is a powerful but simple way to integrate GitHub with the development environment. VCS's features make managing and tracking code simple and relaxed.

3.3 Business Process Overview

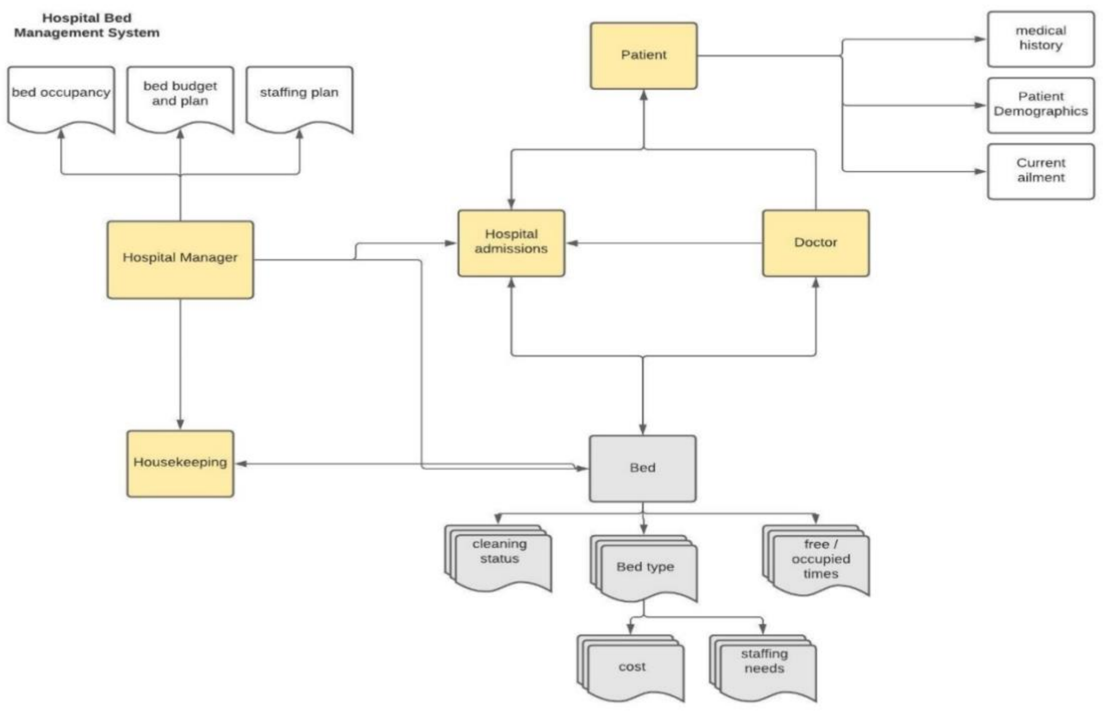
Below is the Process flow published by NHS UK for how they execute bed management.

This provides a good shell for the flows of my new system, which I will expand to add more detail on when a bed is available, the cleaning process, and increasing access to the data - for medical and administration staff.

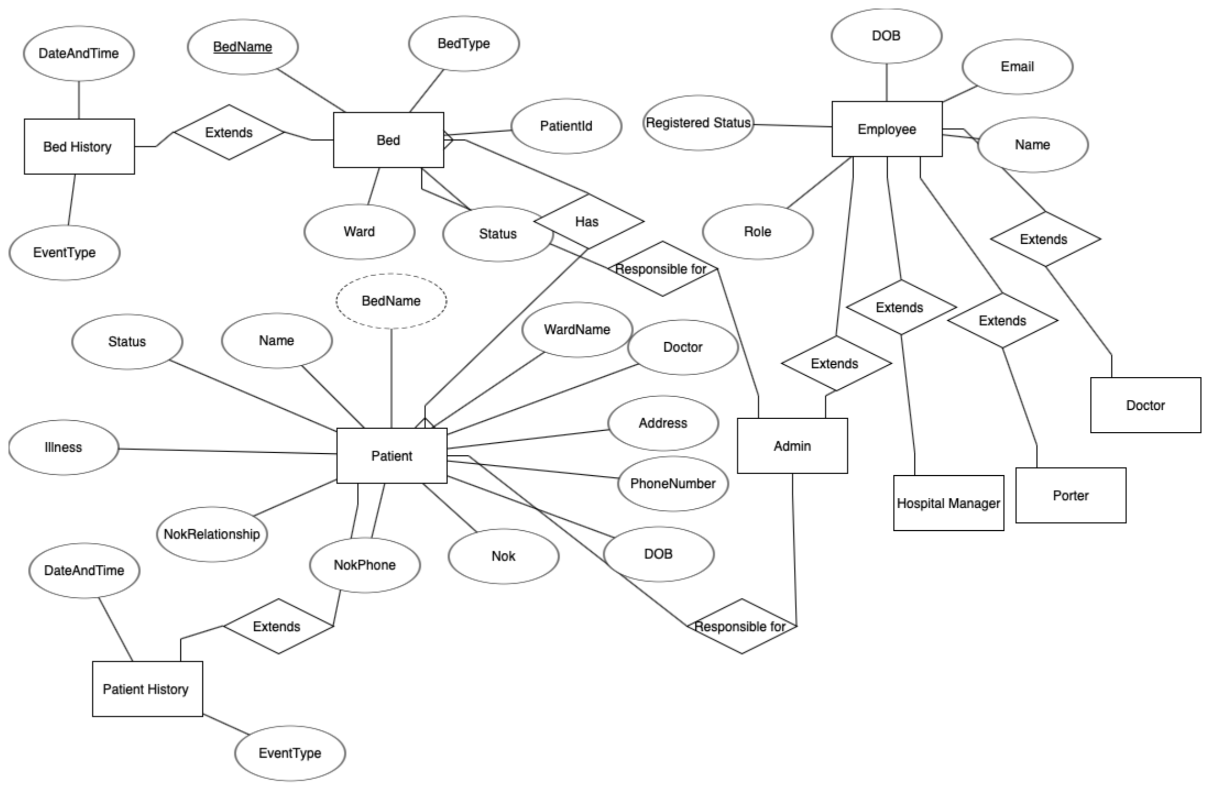
Process Flow for Bed Management (from NHS UK)



ERD of the main data elements and flows of the proposed system.



Entity Relationship Diagram



Bed Status and BedHistory Events.

Bed Status	**Bed History – EventType**
Open	Bed added
Open	Bed allocated to a ward
Allocated	Bed allocated – patient on way
Occupied	Patient in bed in ward
Waiting for cleaning	Bed ready for cleaning
Open	Bed is cleaned – ready for next patient

Database Table Design

Here is the design I used for each part of my application.

Employee	
id	Auto generated
name	String
Employee role	Cleaning Staff, Porter, Nurse, Administration, Doctor, Hospital Manager
Contact Details	contact-details

Employee Roles	
Role	role

Patient	
Id	Auto generated
Name	string
Date Of Birth	dd/mm/yyyy
Illness	Look up list of illnesses
Doctor	From Employee table
Bed Number	From Bed table
Date into hospital	dd/mm/yyyy
Time into Hospital	hh:mm:
Time admitted to bed - by Doctor	hh:mm
Actual Discharge Date	dd/mm/yyyy
Next of Kin	Names

Patient History	
Event	Event
Event Date	Date

Bed	
Bed-type	Look up table for bed-type
Status	Open, Ready-for-Cleaning, Cleaned, Allocated, Occupied
Number	Auto generated + Barcode.
Ward	Ward Name
Patient-ID	From Patient table
BedHistoryEvents	Look up for bedId

Bed History	
Event	Event
Event Date	Date

Ward	
WardName	name

3.4 Software Design

Cloud Firestore

Cloud Firestore is a flexible and highly scalable NoSQL database that is part of the Firebase ecosystem which is made by Google. The Cloud Firestore keeps all data in synchronisation across client apps through the use of real time listeners and offers offline support, so you can build responsive apps that work regardless of network latency or internet connectivity. Cloud Firestore also works seamlessly with other Firebase products such as Firebase Authentication.

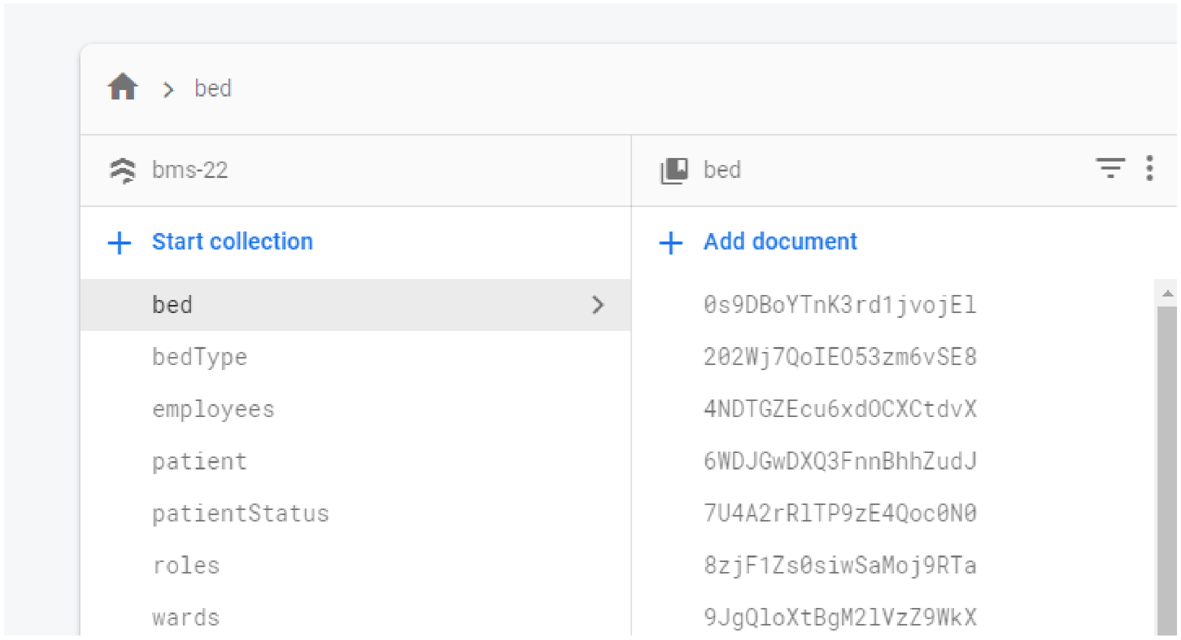
Cloud Firestore is a NoSQL database so instead of using tables and rows like relational databases, Cloud Firestore uses a series of Collections and Documents in order to store and structure the database. In Firestore you store data in documents that contain fields mapping to values. Documents are then stored in Collections, which are used to organise data and build queries. Documents support the storing of any data type from simple strings to nested collections. Cloud Firestore also has quite a flexible, efficient querying system, that can range from a shallow query such as getting all the documents in a database to deep nested querying such as finding a specific data item from a sub - collection. These queries can also be sorted, filtered and can have limits applied to them. As well as this Cloud Firestore offers a series of Realtime Listeners which can trigger events when a change happens in a database.

Security is a key requirement of my application and I chose Firestore for its strong and test authentication as well as its encryption capabilities. I also found Firestore easy to use and iterated with several differ database designs with little challenge. Initially I created my own keys for “Patient”, “ Employee” and “Bed”. Then I used the auto-generation feature and that was much simpler.

The Firebase Cloud Firestore is stored in the Google cloud and is interactable through the Firebase console through a web browser, this console is linked to my Google account.

BMS – Firestore Implementation:

Bed: Below we see the main page of the database, on the left is the collections used and, on the right, the specific bedIDs created.



bed

bms-22

Start collection

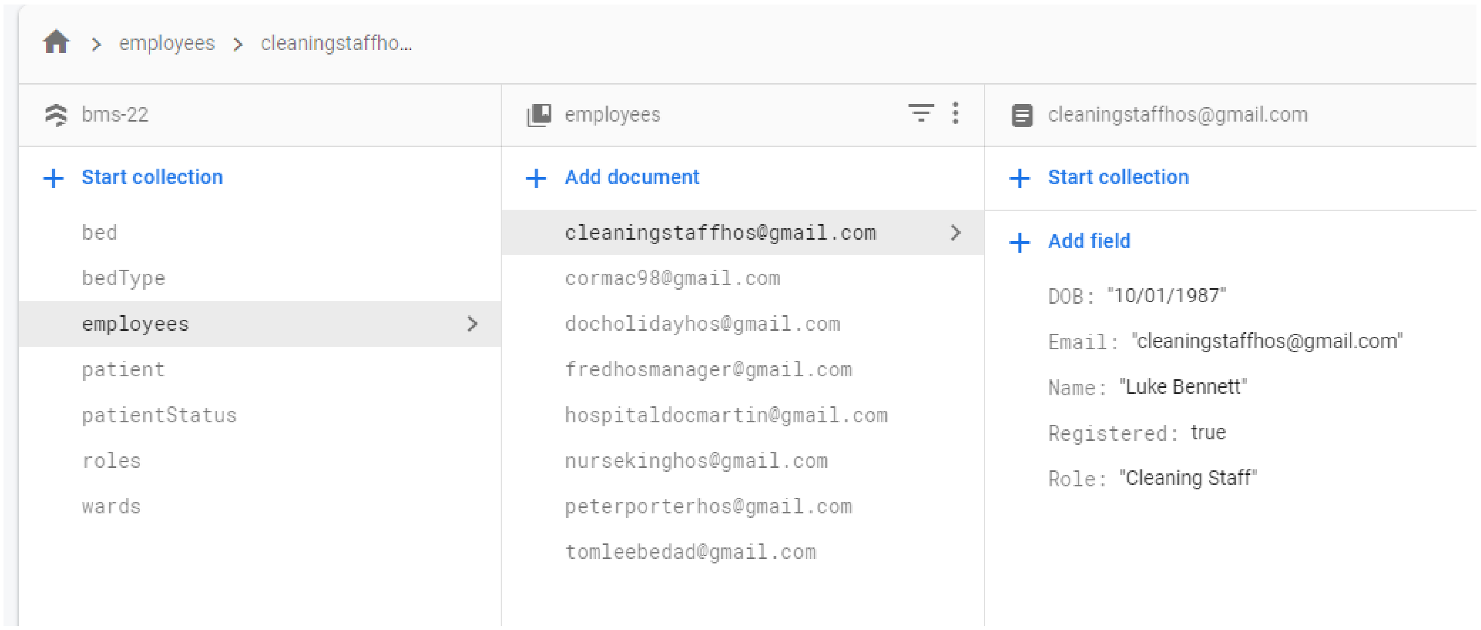
- bed
- bedType
- employees
- patient
- patientStatus
- roles
- wards

bed

Add document

- 0s9DBoYTnK3rd1jvojEl
- 202Wj7QoIE053zm6vSE8
- 4NDTGZEcu6xdOCXCtdvX
- 6WDJGwDXQ3FnnBhhZudJ
- 7U4A2rRlTP9zE4Qoc0N0
- 8zjF1Zs0siwSaMoj9RTa
- 9JgQloXtBgM2lVzZ9WkX

Employee: here I have the Employee collection with Luke Bennett (cleaner) details showing.



employees > cleaningstaffho...

bms-22

Start collection

- bed
- bedType
- employees
- patient
- patientStatus
- roles
- wards

employees

Add document

- cleaningstaffhos@gmail.com
- cormac98@gmail.com
- docholidayhos@gmail.com
- fredhosmanager@gmail.com
- hospitaldocmartin@gmail.com
- nursekinghos@gmail.com
- peterporterhos@gmail.com
- tomleebedad@gmail.com

cleaningstaffhos@gmail.com

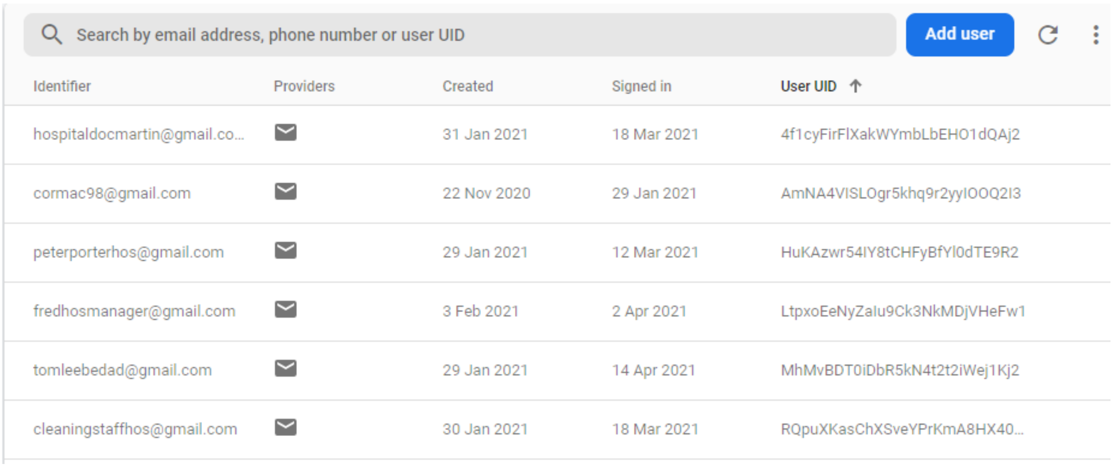
Start collection

Add field

DOB: "10/01/1987"
Email: "cleaningstaffhos@gmail.com"
Name: "Luke Bennett"
Registered: true
Role: "Cleaning Staff"

The ID for the Employee object is not auto generated but instead linked to Firebase Authentication. One can see this below with the account "cleaningstaffhos@mail.com", the User ID in the Cloud Firestore is linked to the User UID generated from Firebase Authentication.

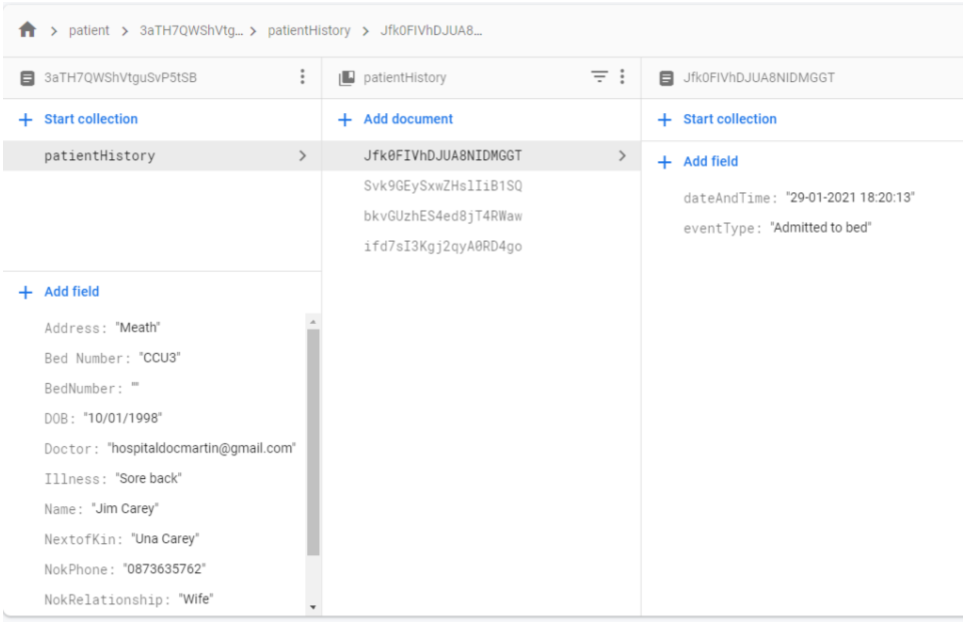
When a user logs into the account for the first time a new document is added to the database.



Identifier	Providers	Created	Signed in	User UID ↑
hospitaldocmartin@gmail.co...	✉	31 Jan 2021	18 Mar 2021	4f1cyFirFlXakWYmbLbEHO1dQAj2
cormac98@gmail.com	✉	22 Nov 2020	29 Jan 2021	AmNA4VISLOgr5khq9r2yyIOOQ2I3
peterporterhos@gmail.com	✉	29 Jan 2021	12 Mar 2021	HuKAzwr54IY8tCHFyBfYl0dTE9R2
fredhosmanager@gmail.com	✉	3 Feb 2021	2 Apr 2021	LtpxoEeNyZalu9Ck3NkMDjVHeFw1
tomleebedad@gmail.com	✉	29 Jan 2021	14 Apr 2021	MhMvBDT0iDbR5kN4t2t2iWej1Kj2
cleaningstaffhos@gmail.com	✉	30 Jan 2021	18 Mar 2021	RQpuXKasChXSveYPrKmA8HX40...

The User object contains an email address, first name, last name and an ID. I felt there was no need for a username as the email must be unique and that is what is used to register and sign in.

Patient History: Below I have Patient "Jim Carey" who was "Admitted to bed" on 29/Jan/2021 @ 18:20

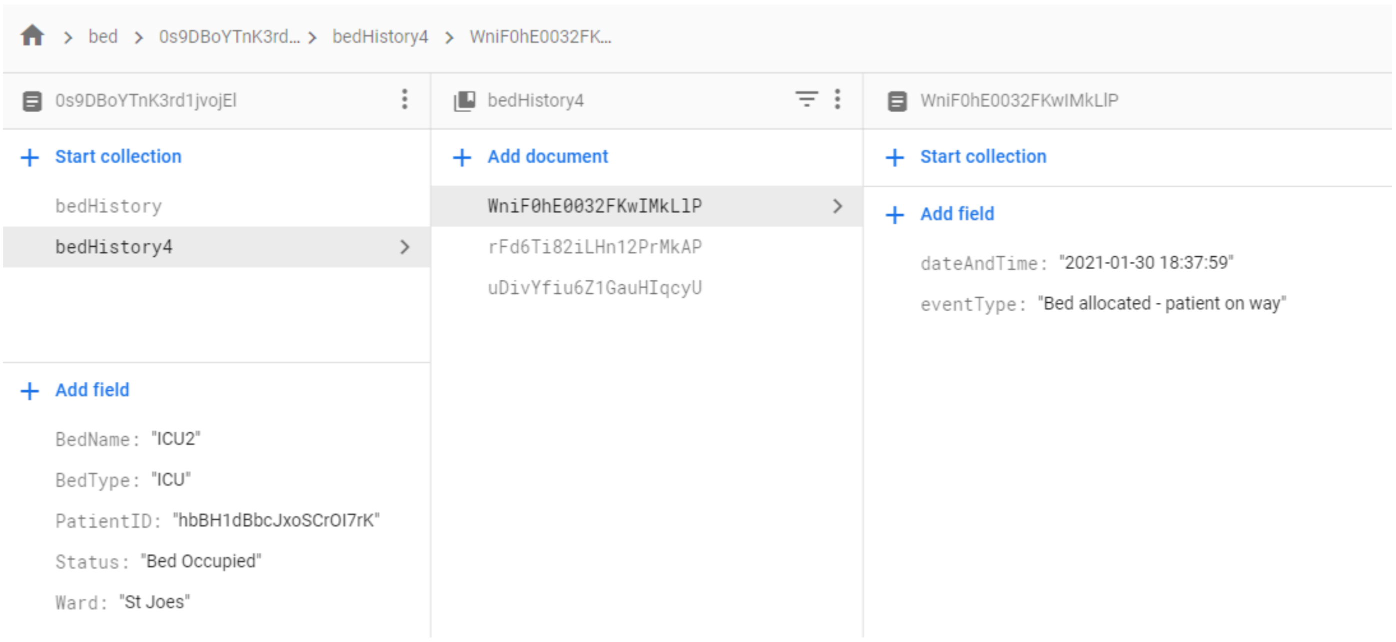


```
patientHistory                Jfk0FIVhDJUA8NIDMGGT
                              Svk9GEySxwZHslIiB1SQ
                              bkvGUzhES4ed8jT4RWaw
                              ifd7sI3Kgj2qyA0RD4go

dateAndTime: "29-01-2021 18:20:13"
eventType: "Admitted to bed"

Address: "Meath"
Bed Number: "CCU3"
BedNumber: ""
DOB: "10/01/1998"
Doctor: "hospitaldocmartin@gmail.com"
Illness: "Sore back"
Name: "Jim Carey"
NextofKin: "Una Carey"
NokPhone: "0873635762"
NokRelationship: "Wife"
```

Bed History: below I have the full information for bed:

0s9DBoYTnK3rd1jvojEl and /bedHistory/ WniF0hE0032FKwIMkLIP.

This is an ICU bed – which is "Allocated – patient on Way"



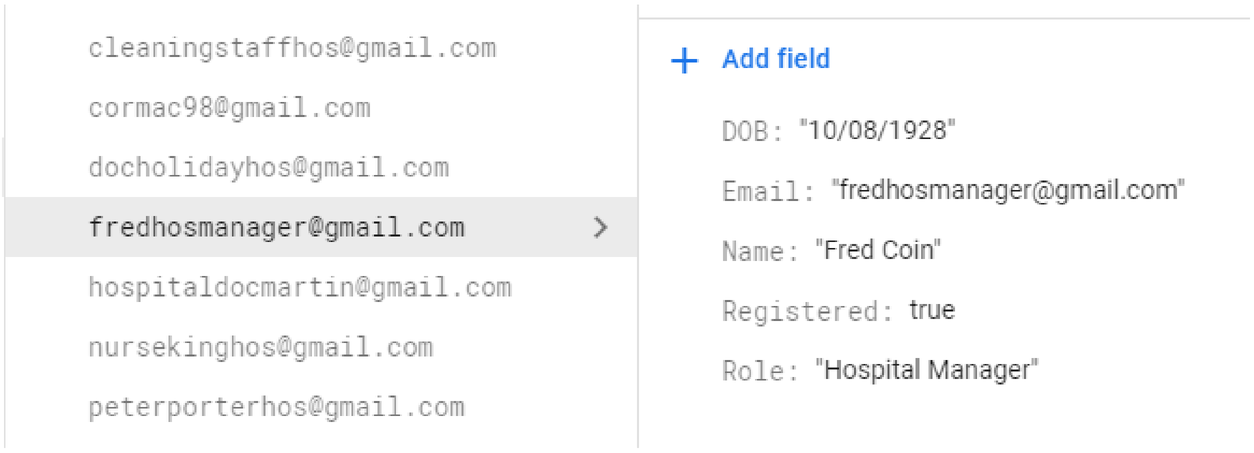
```
dateAndTime: "2021-01-30 18:37:59"
eventType: "Bed allocated - patient on way"
```

```
BedName: "ICU2"
BedType: "ICU"
PatientID: "hbBH1dBbcJxoSCrOI7rK"
Status: "Bed Occupied"
Ward: "St Joes"
```

BMS security and authentication design

Only registered employees can use the app. The Admin sets up new users in the BMS system with Status set to "awaiting registration '.

Firebase Authentication then emails the employee with a link which they use to complete registration and set password. Their status is then set to “registered”



```
cleaningstaffhos@gmail.com
cormac98@gmail.com
docholidayhos@gmail.com
fredhosmanager@gmail.com
hospitaldocmartin@gmail.com
nursekinghos@gmail.com
peterporterhos@gmail.com
```

```
DOB: "10/08/1928"
Email: "fredhosmanager@gmail.com"
Name: "Fred Coin"
Registered: true
Role: "Hospital Manager"
```

Registration

The employee clicks on the link in their email to register . The Employee is asked for additional information [manager and DOB] to verify themselves. They then set up secure password - Password validated against rules :

- more than 6 characters long
- No spaces
- Not same as previous password

Once valid the password is valid it's encrypted and stored in Firebase.

Login

Employee asked for email and password
At login - look up employee database for role, and link to appropriate screen for that role.

Access rights to the specific objects:

CRUD - Create, Read, Update, Delete

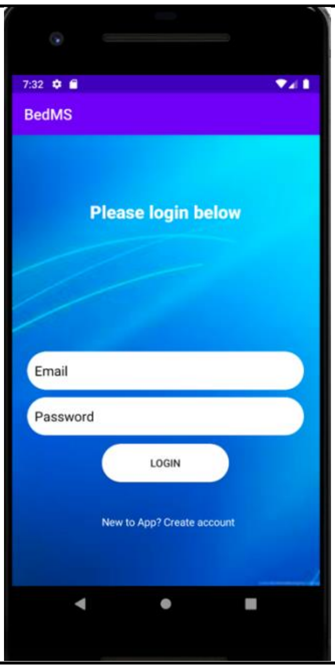
Doctor	Bed:	R, U
	Waiting List:	R, U
	Patient	CRUD
	Illness	R
Nurses	Bed:	RU
	Waiting List:	RU
	Patient	CRU
	Illness	R
Porters	Bed:	R
	Waiting List:	None
	Patient	R
	Illness	none
Housekeeper	Bed:	RU
	Waiting List:	none
	Patient	R
	Illness	none
Admissions	Bed:	RU
	Waiting List:	RU
	Patient	CRU
	Illness	
	Bed information	
Hospital Manager	Bed:	CRUD
	Waiting List:	CRUD
	Patient	CRU
	Illness	R
Patient	Bed:	None
	Waiting List:	None
	Patient	None
	Illness	None
	Bed information	

3.5 User Interface Design

Application users will open on the login screen where they are prompted to log in using their email and password.

No Register feature is available for security.

The login and authentication are powered by Firebase Authenticator and create a new session which retrieves user role using the getCurrentUser() method and hands control to the opening screen for their role.



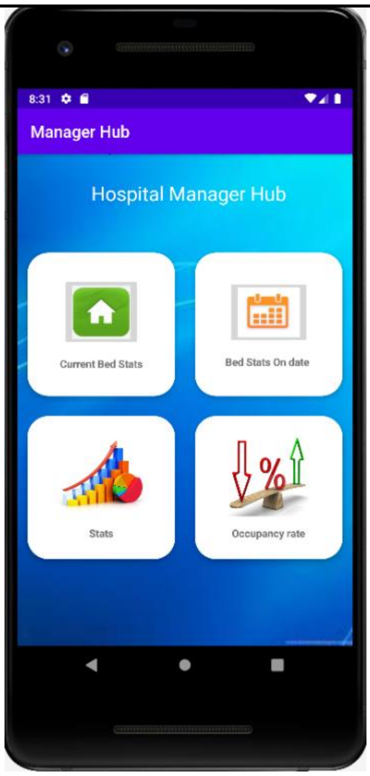
Hospital Manager Landing Page.

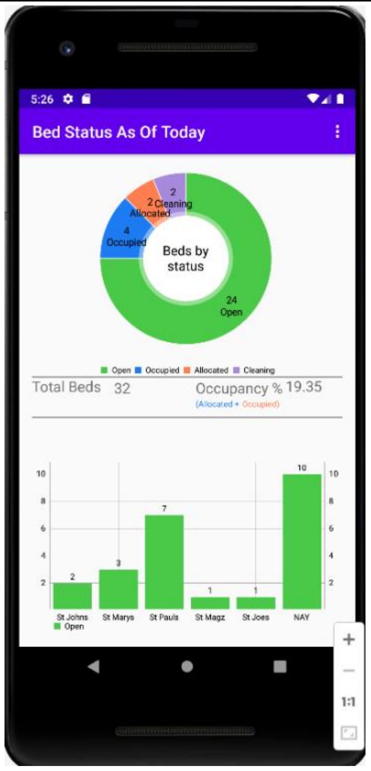
It shows the Bed Status for today.

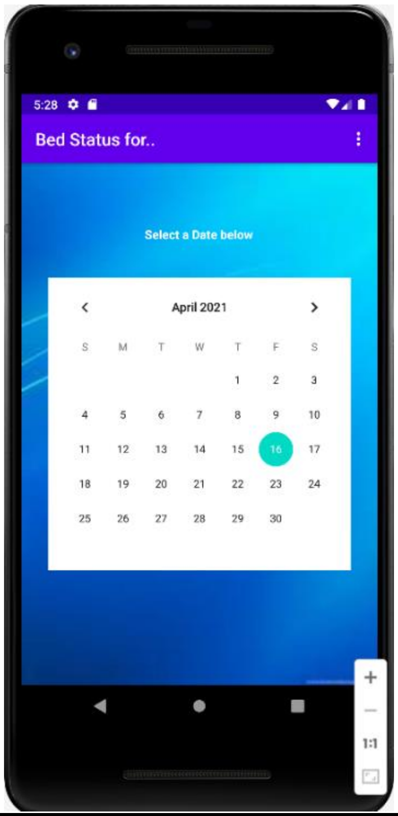
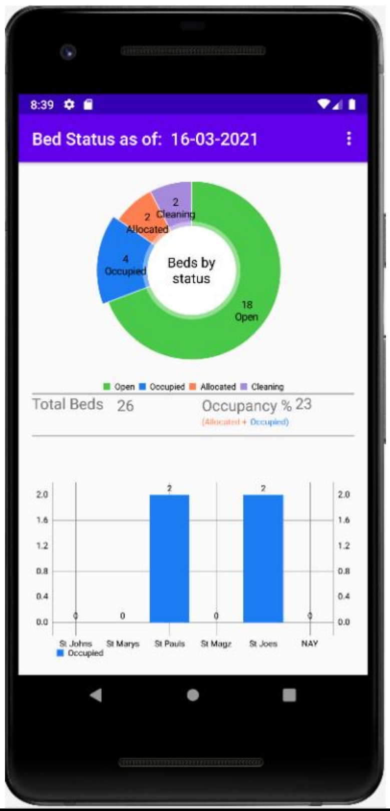
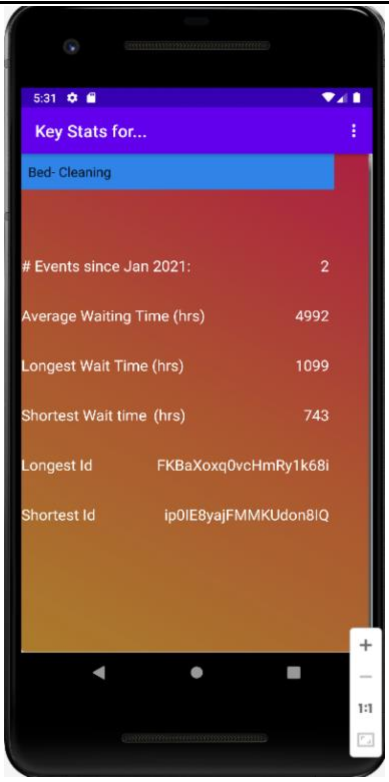
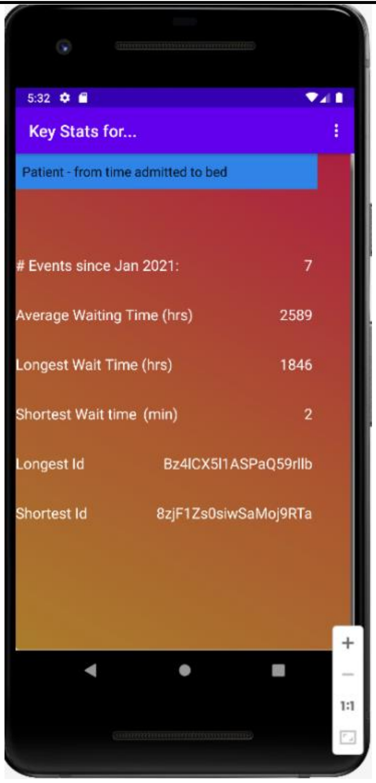
It allows manager to :

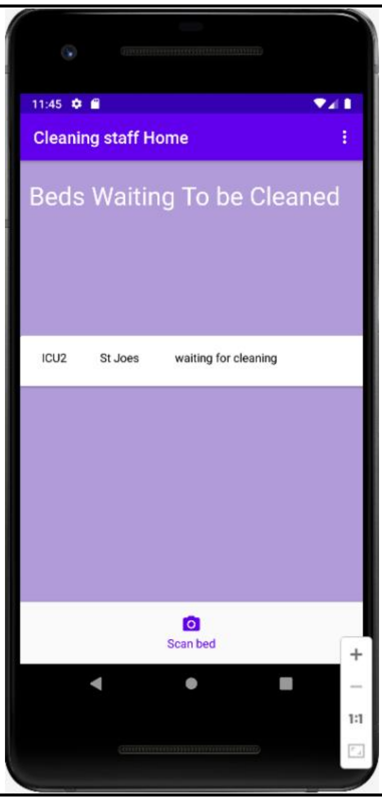
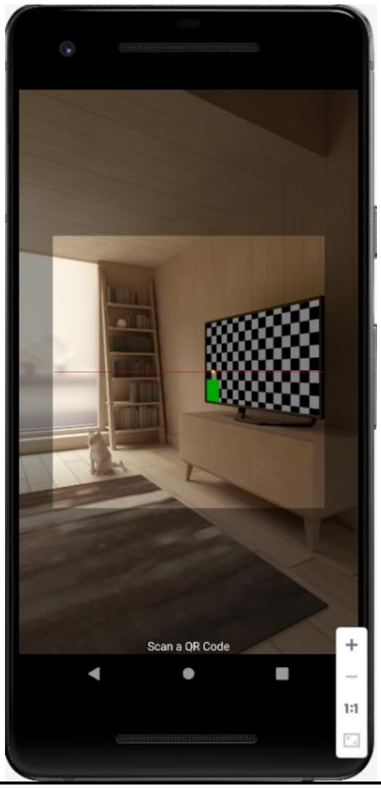
- See Bed Status for a date

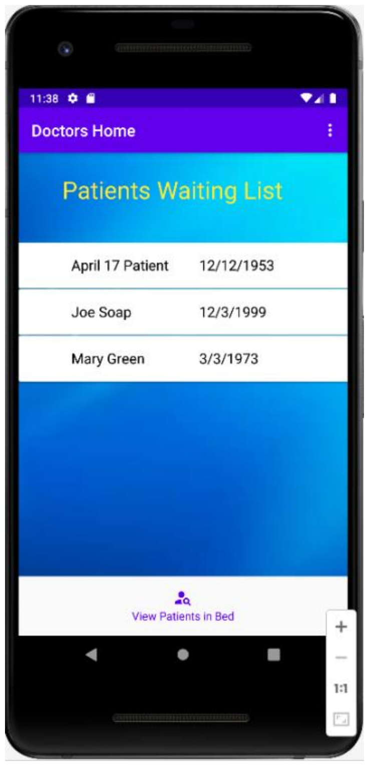
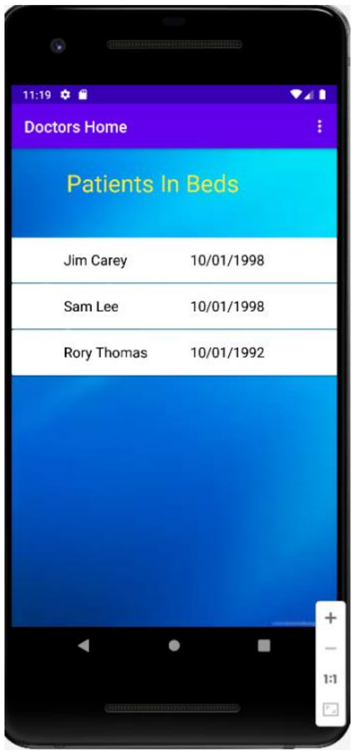
- See Wait Times
  - Bed Cleaning
  - Patient arrival at Ward
- See Occupancy Per Day
- See Occupancy per Month

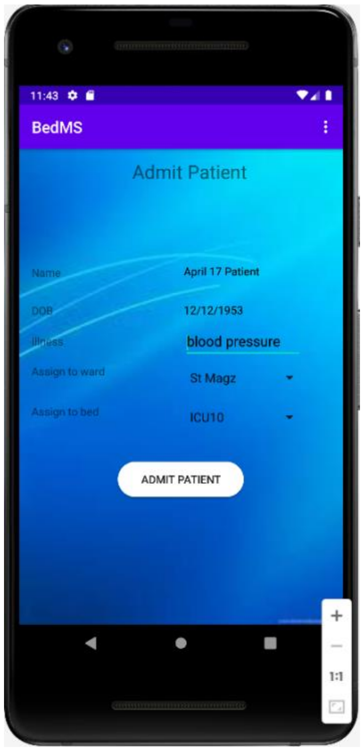
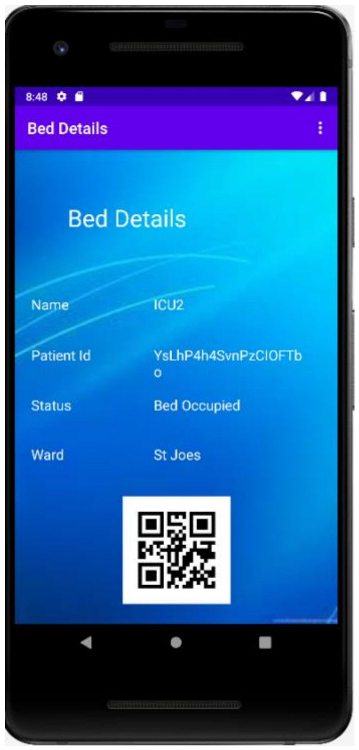


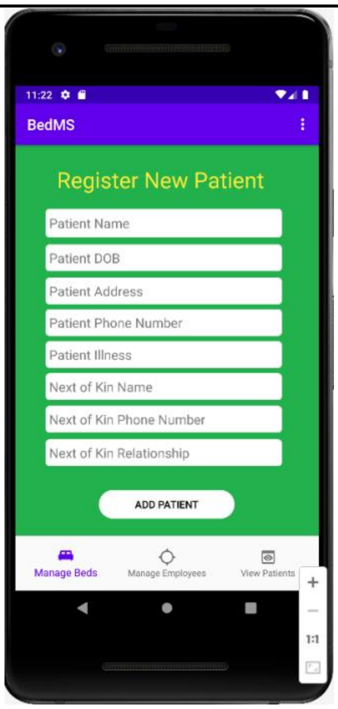
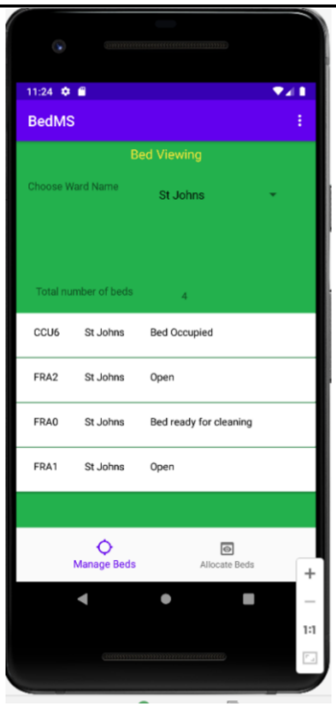


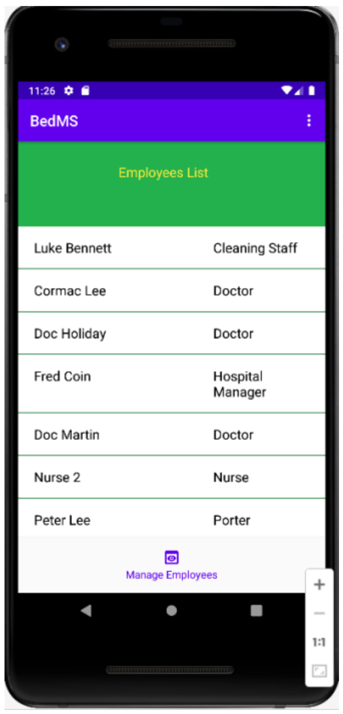
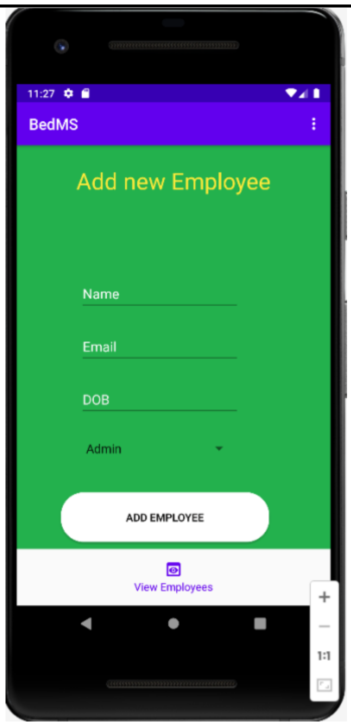
Hospital Manager Stats as of date….. • Number of beds • Occupancy • Beds by Status • Beds by Ward	 5:28 Bed Status for.. Select a Date below April 2021 S M T W T F S 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30  8:39 Bed Status as of: 16-03-2021 2 Cleaning 2 Allocated 4 Occupied Beds by status 18 Open Open Occupied Allocated Cleaning 2.0 1.6 1.2 0.8 0.4 0.0 St Johns St Marys St Pauls St Magz St Joes NAY Occupied Total Beds 26 Occupancy % 23 (Allocated + Occupied)
Hospital Manager. Key Stats Page Wait Times for : • Bed Cleaning • Patient transfer to bed	 5:31 Key Stats for... Bed- Cleaning # Events since Jan 2021: 2 Average Waiting Time (hrs) 4992 Longest Wait Time (hrs) 1099 Shortest Wait time (hrs) 743 Longest Id FKBaXoxq0vcHmRy1k68i Shortest Id ip0IE8yajFMMKUdon8IQ  5:32 Key Stats for... Patient - from time admitted to bed # Events since Jan 2021: 7 Average Waiting Time (hrs) 2589 Longest Wait Time (hrs) 1846 Shortest Wait time (min) 2 Longest Id Bz4ICX5I1ASPaQ59rIlb Shortest Id 8zjF1Zs0siwSaMoj9RTa

Cleaning staff – can see the waiting list of beds to be cleaned. They can access the bed using the Scanning utility – to quickly scan bedCode, mark bed as being cleaned and set status to Open and ready for next patient.	 11:45 Cleaning staff Home Beds Waiting To be Cleaned ICU2 St Joes waiting for cleaning 

Doctor Screens View Waiting List of patients in A&E waiting to be seen. Patients in Beds waiting to be seen	 11:38 Doctors Home Patients Waiting List April 17 Patient 12/12/1953 Joe Soap 12/3/1999 Mary Green 3/3/1973  11:19 Doctors Home Patients In Beds Jim Carey 10/01/1998 Sam Lee 10/01/1998 Rory Thomas 10/01/1992

Doctor Screens (cont.d) View and update Patient details Admit or Discharge patient	 11:43 BedMS Admit Patient Name: April 17 Patient DOB: 12/12/1953 Illness: blood pressure Assign to ward: St Magz Assign to bed: ICU10 ADMIT PATIENT  8:48 Bed Details Bed Details Name: ICU2 Patient Id: YsLhP4h4SvnPzClOFTbo Status: Bed Occupied Ward: St Joes

Admin Screens Registering new patient Plus, main menu for - Employee Admin - Bed Admin - Patient Admin View's patients in Ward	 11:22 BedMS Register New Patient Patient Name Patient DOB Patient Address Patient Phone Number Patient Illness Next of Kin Name Next of Kin Phone Number Next of Kin Relationship ADD PATIENT Manage Beds · Manage Employees · View Patients  11:24 BedMS Bed Viewing Choose Ward Name: St Johns Total number of beds: 4 CCU6 · St Johns · Bed Occupied FRA2 · St Johns · Open FRA0 · St Johns · Bed ready for cleaning FRA1 · St Johns · Open Manage Beds · Allocate Beds

Admin Screens (cont.d) View and Add employees	 

11:26

BedMS

Employees List

Luke Bennett	Cleaning Staff
Cormac Lee	Doctor
Doc Holiday	Doctor
Fred Coin	Hospital Manager
Doc Martin	Doctor
Nurse 2	Nurse
Peter Lee	Porter

Manage Employees

11:27

BedMS

Add new Employee

Name

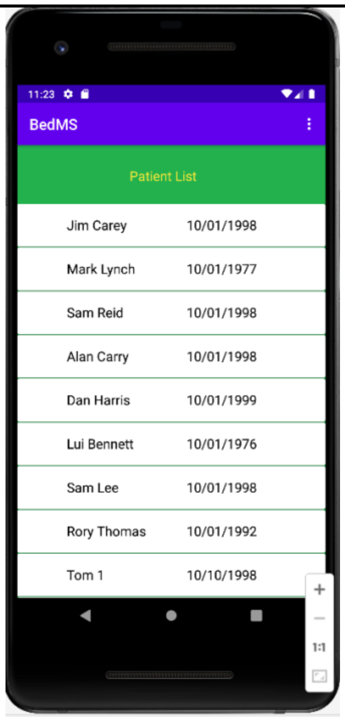
Email

DOB

Admin

ADD EMPLOYEE

View Employees

Admin Screens (cont.d) View list of patients	

11:23

BedMS

Patient List

Jim Carey	10/01/1998
Mark Lynch	10/01/1977
Sam Reid	10/01/1998
Alan Carry	10/01/1998
Dan Harris	10/01/1999
Lui Bennett	10/01/1976
Sam Lee	10/01/1998
Rory Thomas	10/01/1992
Tom 1	10/10/1998

3.6 Issues and Resolutions

List of possible topics here:

1. Use of Firebase Authorization
2. Charting and graphics tool
3. API connection for Illness verification
4. Barcode technology
5. Test data

Firebase Authorization

I initially struggled with Authorizations – and tended to overcomplicate the design by trying to generate my own UserID. Once I relied on Firebase to assign the ID to the person and used their email as the main identifier – it was quite straightforward.

Charting Solutions and graphics tool

I looked at several charting tool solutions. I wanted something that I could build into the app – not a separate BI solution. I looked at Power BI, Tableaux, and Domo. I discounted all three because of cost, and that they preferred to run from a separate data repository.

After some research I discovered MP Android Charts from the open-source community. The solution is used to present the main stats from the application and after an initial learning curve, it works well for me. I feel that the quality of the graphics could be improved, and the level of documentation on how to use the main variables needs refinement.

Overall, I am pleased with the tool – it enabled me to generate interactive graphs and is one of the core parts of my app.

API connection for Illness verification

I discovered an open API from medical technology company Infermedica which I was able to secure a free version for use. This allows me to verify the illnesses that a person can have and apply these consistently across the application.

There was no support available for the API and the protocol for authorization was complicated to implement.

This API could be further utilized as it provides additional functions:

- Natural Language Processing on symptoms entered.
- And a probabilistic diagnosis for the patient.

I decided not to implement these functions – they are unnecessary as my patients are already in hospital and being seen by a doctor.

Barcode technology

I used ZXING QR and barcode solution. This came with an in-built simulator – which took a lot of time to master. I went for a simple solution – the barcode is generated for the bedName as needed by the Admin or Cleaning staff, and the switch from barcode to bedName was successfully implemented.

Test data

The solution is best demonstrated using lots of test data. I have tested the individual parts of the solutions well – and am confident of the quality of my software – but do not have the large volumes of test data – generated over 6+ months - that would showcase the solution at its best. The areas where this manifests itself most is in the Management reporting for “Wait Time” and “Occupancy”. The “Wait Times” for some patients or for bed cleaning is several hundred hours long – which is inaccurate – as I have not completed their processing. This would not arise in everyday life.

To have generated this data - I would have had to use the app every day – as people would do of the real world – to generate the data needed. As the data is heavily interrelated – patient, PatientHistory, Bed, BedHistory – it would have been challenging to have generated test data outside of the application.

4. Implementation

4.1 Overview of BMS Implementation

The design of my BMS was built around three main objects:

- Bed
- Patient
- Employee

I began by developing simple CRUD applications for these elements. I explored MySQL for the database but chose Firestore because it had a lot of features that were important to my app's needs, including authorization. There was a learning curve about how to develop an acceptable model and communicate with the database for querying and data manipulation since it was my first experience using a NoSQL database. Firebase uses collection and documents as their main way of interacting with the app and database.

As I developed the process flows, I decided to create two additional Firebase stores:

- BedHistory – a sub-collection of Bed which held the historical events and dates.
- PatientHistory – similar for Patient.

I used Java for all my logic within Android App Studio. I found Android Studio had a significant learning curve. My Java design consisted of the following Model Classes:

- Bed
- Patient
- Doctor
- Admin
- Hospital Manager
- Cleaning Staff
- Porter

Looking back, I could have implemented one primary Model class for Employee and then extended it for each employee group.

My main Java Classes addressed the business functionality for that process. I believe I could have modularised the code more – and created more common utilities that could have been used across the application. I used two open-source applications – for charting and QR code reading.

4.2 Issues and Resolutions with Implementation

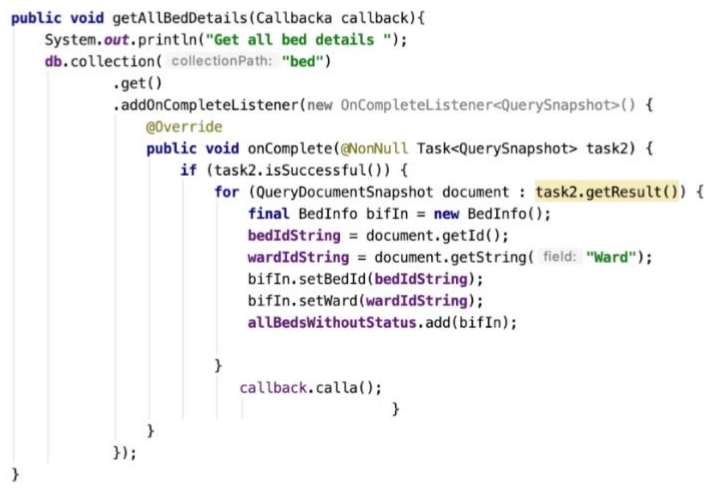
There were a number of areas that presented challenges to me including :

2. Read data from Firebase Firestore synchronously from my Java code.
3. Testing the application
4. Software design and use of Patterns
5. Project Management and estimation
6. Firebase Database design for BedHistory.

1. Reading Firestore synchronously from Java

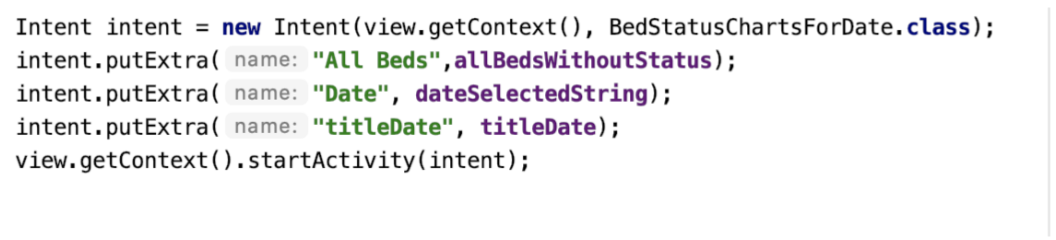
Querying Firestore from Java generates a non-synchronous event where the OnCompleteListener is used. When a database access is executed – as below – Java executes the call to the database – but continues to execute the main line of code. The onCompleteListener notifies the calling thread once the results are back. In the stats part of the app - where I needed lots of database access – this was a continuing challenge to overcome. I did two things to address this issue :

I. I adjusted the design to build in steps that allowed the database time to respond in a seamless way to the user.
II. I also built in a Callback functionality to address this delay where I could not change the User flow – see the example below.



```
public void getAllBedDetails(Callbacka callback){
    System.out.println("Get all bed details ");
    db.collection( collectionPath: "bed")
            .get()
            .addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
                @Override
                public void onComplete(@NonNull Task<QuerySnapshot> task2) {
                    if (task2.isSuccessful()) {
                        for (QueryDocumentSnapshot document : task2.getResult()) {
                            final BedInfo bifIn = new BedInfo();
                            bedIdString = document.getId();
                            wardIdString = document.getString( field: "Ward");
                            bifIn.setBedId(bedIdString);
                            bifIn.setWard(wardIdString);
                            allBedsWithoutStatus.add(bifIn);

                        }
                           callback.calla();
                                             }
                }
            });
}
```

III. I use the Intent Function to pass data from one step in the process to the next, eliminating the need to go to the database in that situation.



```
Intent intent = new Intent(view.getContext(), BedStatusChartsForDate.class);
intent.putExtra( name: "All Beds",allBedsWithoutStatus);
intent.putExtra( name: "Date", dateSelectedString);
intent.putExtra( name: "titleDate", titleDate);
view.getContext().startActivity(intent);
```

IV. This problem was exacerbated when retrieving the stats by bed, by status, by day for the month. Here I tried something new and I cached all the bed details at the start of the query when opening the splash screen. This reduced my database calls dramatically as I have all data elements in memory for each day. Looking across the application this is something I could have implemented for other queries to reduce the database transactions. This works because I have a small database. If my database was large this approach would need to be tested for performance.

2. Testing the application

I have recently learned how to use the Test capability of Android App Studio to assist my testing. Prior to that I was testing my code by going in from UI into the application. This was very time consuming and slow. More recently, I have used the tool to help test individual methods and classes, and this shows the coverage of testing undertaken. Using the Tool to test specific methods and classes has been a great boost to quality and speed.

3. Software Design and use of Patterns.

Since beginning the development of this solution, I have developed my understanding of Software Patterns. I have used patterns in this solution including:

- **Command** – used for getting bed history.
- **Adapter –** using Adapter for managing my interface.

I now see additional areas where I could have enhanced the solution using more Design patterns. These include:

- The Builder Pattern could have been used to create bed-history for each bed and Patient history for each patient.
- The State Pattern could have been used to determine the status of the bed.
- I could have brought all database access into the Bed, and Patient, and Employee Objects – and used an Observer pattern to kick off database updates once the core object was altered.

4. Project Management.

I have found that I tended to underestimate the time required to complete a task. This project has given me a much better understanding of function estimation and how to account for unknowns at the planning stage.

I have learned many lessons while developing my first full stack application. To begin, rather than using a word document to log sprints and task in hand I would use a planning tool like Jira. Jira would have provided me with a more detailed picture of the project's success, problems that were unresolved, and my accomplishments. I would also have spent more time finding different solutions and libraries to use when working out my project.

5. Firebase Database design for BedHistory.

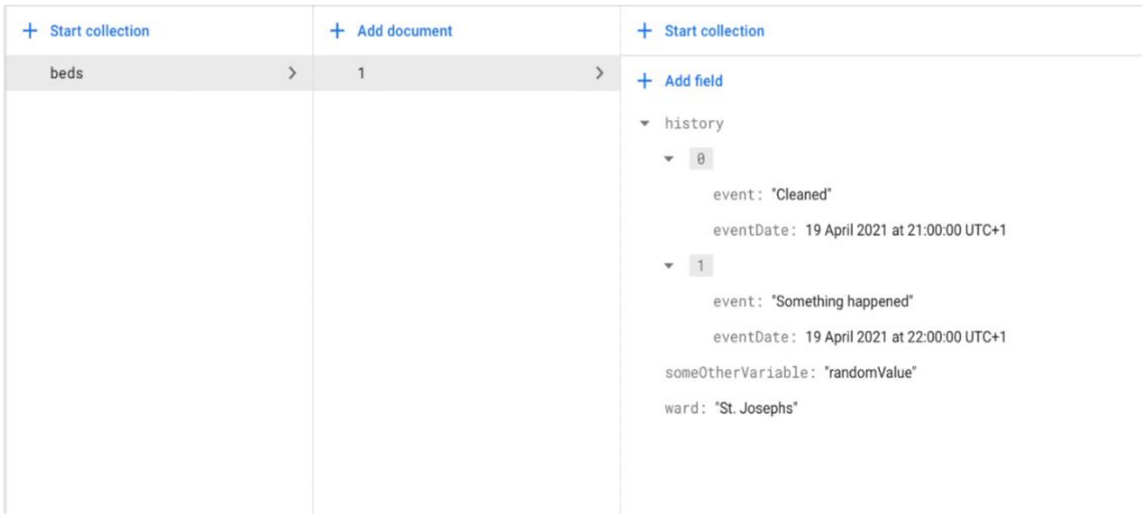
I implemented Bed in a collection, and BedHistory in a sub-collection.

This worked well for the major components of my application. I created a common routine “UpdateBedHistory” which is called when an event happens to a bed. I store the time of the event using a timestamp.

However, when I want to report on stats for Wait-Time or Utilisation I see areas where I can improve my design. Wait-Time calculations require that I read back through all events for each bed and check the difference for relevant events between the time of that event and the next event (or current time if there are no more events). When looking at one dimension – example, bed status or Ward. But my stats quickly expanded to multi-dimension – reporting stats by bed, by Status, by Ward, by date, and by month. All these additional dimensions dramatically increase the database accesses required.

I therefore would recommend altering the database design to the following – where BedHistory is stored as an array within Bed. That would significantly reduce the database calls and increase the processing speed of the application.

Recommended Bed/BedHistory Design.



```
history
  0
    event: "Cleaned"
    eventDate: 19 April 2021 at 21:00:00 UTC+1
  1
    event: "Something happened"
    eventDate: 19 April 2021 at 22:00:00 UTC+1
someOtherVariable: "randomValue"
ward: "St. Josephs"
```

6. Test Plan

Test Cases

Test Case	Test Scenario	Expected Result	Test Result: Pass/Fail
Launch the App	Splash Screen appears.	Splash Screen is displayed and opens on the login/register screen.	PASS
Register New User	User attempts to register account with unique email address.	User account is created, and a user object is added to the database.	PASS
Register New User	User attempts to register account with existing email address	User account is not created, and user is alerted that email address already exists	PASS
Register New User	When creating an account and leave certain fields blank.	User receives error message.	PASS
Register New User	Attempt to build a user with an email address that isn't properly formatted or a password that is less than seven characters long.	User receives error message.	PASS
Login	The ability for a user to log into their account.	User is logged in successfully.	PASS
Login	Attempt to login to their account while leaving a field empty.	User receives error message.	PASS
Login	Attempt to login to account with invalid username or password.	User receives error message.	PASS

Admit Patient	Attempt to admit a new patient by filling in all appropriate fields.	A new patient is added to the database and added to the list of existing patients.	PASS
Admit Patient	Attempt to insert a patient which are missing mandatory field.	A new patient is not added to the database and user is notified to fill in mandatory fields.	PASS
View All Patients	Attempt to view all patients details which were being retrieve the database.	All patient details display.	PASS
Create Bed	Attempt to admit a new bed choosing the type of bed and enter quantity amount.	New beds are added to the database with correct data as fields and correct quantity of beds added.	PASS
View all Bed	Attempt to view all bed details which were being retrieve the database.	All bed details display.	PASS
Allocate Bed to Ward	Attempt to assign a bed to a ward by using the drop down to choose a bed and ward.	Bed is now allocated to a certain ward.	PASS
View Beds Inventory	Attempt to view all beds from inventory with all the status.	All beds from the database display with details.	PASS
View Beds	Attempt to view all beds by ward and show beds statuses.	All beds display with details.	PASS
Create Employee	Attempt to admit a new employee by filling in all appropriate fields.	A new employee is added to the database and added to the list of existing patients.	PASS
View Employees	Attempt to view all employees from inventory.	All employees from the database display with details.	PASS
Doctor – logged in and waiting list shows up	The ability for a user to log into doctor account and doctors waiting list shows up.	Doctor waiting list of patients shows up.	PASS
View Patient	The ability to select a patient	Select patient and show patient	PASS

details.	from the waiting list and view patient details.	details.	
Admit Patient to a certain bed.	The ability to admit a patient to a certain bed and ward.	Patient Id added to the bed.	PASS
Discharge Patient	The ability to discharge a certain patient.	Patient status in database changes to discharged.	PASS
Cleaning staff – logged in and waiting list shows up	The ability for a user to log into cleaning staff account and bed waiting list shows up.	Cleaning staff waiting list of beds shows up.	PASS
Scan Qr Code of bed	Ability to scan a Qr code and display details of bed.	Data of bed shows up on screen when qr code is scanned.	PASS
View Bed details.	The ability to select a bed from the waiting list and view bed details.	Cleaning staff waiting list of beds shows up.	PASS
Mark bed as cleaned.	The ability to mark a specific bed as cleaned and ready for new patient.	Bed status in database changes to cleaned and then to open.	PASS
Hospital Manager – logged in and home page shows up	The ability to log in and display the hospital manager hub.	Hospital Manager shows up.	PASS
View stats as of today	The ability to show stats as of today.	Pie chart and bar chart representing data as requested.	PASS
View stats as of particular date.	The ability to choose a date and display stats.	When date selected from calendar stats show up on next screen as pie and bar chart.	PASS
View occupancy	The ability to choose by the	Line chart shows up with stats	PASS

Per Month	month the occupancy.	ranging from per month.	
View overview of waiting time stats	The ability to choose waiting time stats and use drop down to change type of waiting times.	Data of specific waiting time types of display.	PASS
Logout	Attempt to log out of their user’s account.	Log out successfully and the Login page is shown.	PASS

7. Future Development of the Project

There is massive potential for enhancement and development of this concept – and the application of technology to healthcare is the next big wave. Solutions like this will be needed to enhance the patient experience and drive maximum efficiency in the resources available.

When I started this project, I needed to learn many things. I was unfamiliar with the detailed processes that hospitals use for:

- Employee management
- Bed management
- Patient management
- And how these all interact.

This project has shone a light on these processes – and I believe could be developed to go much deeper into each process involved - this program is superficial in places and would need deeper analysis of the processes involved. It could also be developed to better accommodate the needs of each actor in their daily working lives.

More can be done in the app to benefits from modern technologies including AI, and IoT. Some examples include:

- Like the airline industry – predictive modeling can be used to recommend the numbers of beds and staff required for the time of year.
- Linking geo-spatial solutions to reduce the travel-time for staff – allowing a member of staff to see what needs to be done near to where they are, thus reducing travel time.
- AI can be used to help prioritise actions – looking at the medical issues on hand and the corresponding geo-spatial information to help decide what to do next.
- AI modelling can be used to predict how long a patient will be in a bed – using their age and prognosis.

The area which I believe is most valuable from this solution is the ability to shine a focus on key metrics in the system. The two which I have delivered – “Wait Time for Bed Cleaning” and “Wait Time from when patient admitted to their arrival in the bed” bring a valuable focus on the operating model of the Hospital and help increase the patient experience.
Once the metric is refined, staff can set thresholds and goals. Then, should a specific event not occur within its threshold, alerts and report can be created. Likewise, rewards and incentives can be applied where goals and targets are met. These are vital operational metrics for a modern hospital but would need to be surrounded with other metrics like staff feedback, patient feedback, and patient care outcomes.

8. Conclusion

To summarise, final year was both demanding and rewarding. I have learned a lot from this project – and have grown considerably in confidence and technical abilities, all of which will benefit me when I graduate.

I learned how to cope with a variety of project roadblocks and unforeseen challenges, one of which was the COVID-19 pandemic. The college's closure added to the pressure because support was no longer available when you needed it. Yet I am delighted with how I persevered, and I'm pleased with the outcome.

Given the variety of technology present, the project's initial launch was a little intimidating. I had a steep learning curve ahead of me because I had no prior experience designing a completely functioning application. I overcome this and now I have respect and admiration of developers in the job force who face these problems day in and day out. The amount of effort and passion that goes into creating an app is taken for granted by Android and iOS users.

From a business standpoint, I believe I have filled a consumer need for a simple, user-friendly Bed Management System. I've shown the app to hospital personnel at my local hospital, and they've given it a thumbs up.

Overall, I enjoyed developing Bed Management System, and I feel it was an important step in preparing me for the technical working world that awaits me.

9. References

1. Firebase. 2020. *Cloud Firestore | Firebase*. [online] Available at: <https://firebase.google.com/docs/firestore> [Accessed 28 April 2020].
2. Firebase. 2020. *ML Kit For Firebase*. [online] Available at: <https://firebase.google.com/docs/ml- kit> [Accessed 29 April 2020].
3. Google Developers. 2020. *Overview | Places API | Google Developers*. [online] Available at: <https://developers.google.com/places/web-service/intro> [Accessed 30 April 2020].
4. Google Developers. 2020. *Overview | Maps SDK For Android | Google Developers*. [online] Available at: <https://developers.google.com/maps/documentation/android-sdk/intro> [Accessed 31 April 2020].
5. Android Studio Documentation [online] Available at: <https://developer.android.com/docs> [Accessed 1 May 2020].
6. “Managing a bed crisis” by Nick Egan (J3Accid Emerg Med 1999;16:145-146)