Operating Room Scheduling Application

Scott Mackenzie

A thesis submitted to Auckland University of Technology in partial fulfilment of the requirements for the degree of Master of Computer and Information Sciences

School of Engineering, Computer and Mathematical Sciences
Auckland University of Technology

2018

Abstract

This project explores how the use of a purpose-built software can improve visibility and usage in operating rooms in Auckland City Hospital. Using a design science approach an Angular application was developed as a replacement for an already existing manual system. It was found that the application developed increased visibility of information as well as saves time when reporting compared to the existing system. Although the software was completed as planned, for various reasons the client evaluation in normal use could not take place in the time frame of this masters. Further live testing is recommended in order to gain quantitative data in regards to increase in session usage.

Contents

List of Figures

Attestation of Authorship

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person (except where explicitly defined in the acknowledgements), nor material which to a substantial extent has been submitted for the award of any other degree or diploma of a university or other institution of higher learning.

Scott Mackenzie

Date

Acknowledgements

I would like to acknowledge the expertise and support provided by my supervisory team of Associate Professor David Parry and Associate Professor Stephen Reay. Their guidance and advice has made this process immensely easier.

I would like to acknowledge the production planning team from ADHB and the DHW Lab. Without them this research would never have become.

My Mother and her partner Vic, for allowing me to move back home for 6 months while I studied and for dealing with my late nights at the computer.

And finally my partner Cécile who waited patiently (at times) in another country for me to finish. Merci pour tout

Chapter 1. Introduction

In 2014 the Auckland District Health Board introduced a new operating room weekly production planning (SCRUM (Sutherland & Schwaber, 2013)) process. This new process' performance improvements were measured against the previous year showing much improvement. When measured against 2013 an additional 191 operating room sessions were run, a 0.6% improvement representing 1.4 million dollars of additional capacity (*OR SCRUM Overview for Surgical Board Meeting*, 2014).

With on average session usage varying between 95 and 98 percent week by week the production planning team at Auckland Hospital predicted even more improvement could be made to the process by implementing a purpose-built software.

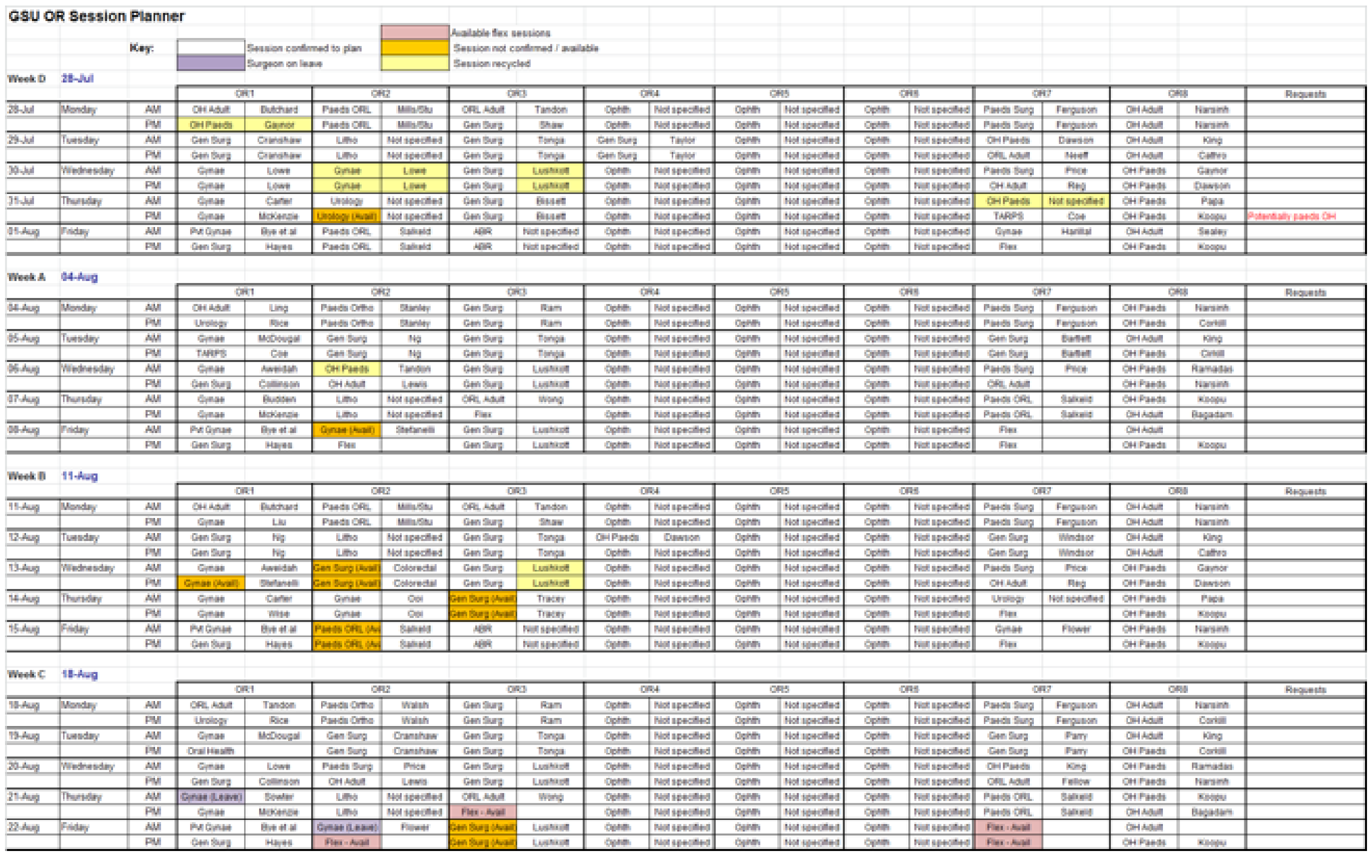
1.1 Auckland Hospital case study

In order to understand the research area the existing system for scheduling operating rooms in Auckland Hospital must be understood. Auckland District Health Board has an established set of processes it uses (*Operating Room Weekly Production Planning (SCRUM) Overview*, 2014) which will be explained in depth below:

1.1.1 Operating room session planner

Before the scheduling process begins an operating room session planner is created. A session in this context is either a morning or afternoon block of time which is allocated to either an individual surgeon or surgical service. This planner is intended to be the primary source of reference for all staff wanting an understanding of planned operating room sessions for both surgical services and surgeons. All operating room sessions across Auckland Hospital are included in this schedule which is forecast approximately six months in advance.

The planner is also used to provide visibility of surgeon leave, sessions available and requests for additional operating room sessions using predetermined colour blocks as shown in Figure 1. It is readily available as a live spreadsheet through the Auckland District Health Board intranet.



GSU OR Session Planner

Key:

⬜	Session confirmed to plan
🟪	Surgeon on leave
🟥	Available flex sessions
🟧	Session not confirmed / available
🟨	Session recycled

Week D 28-Jul

			OR1		OR2		OR3		OR4		OR5		OR6		OR7		OR8		Requests
28-Jul	Monday	AM	OH Adult	Butchard	Paeds ORL	Mills/Stu	ORL Adult	Tandon	Ophth	Not specified	Ophth	Not specified	Ophth	Not specified	Paeds Surg	Ferguson	OH Adult	Narsinh	
		PM	OH Paeds	Gaynor	Paeds ORL	Mills/Stu	Gen Surg	Shaw	Ophth	Not specified	Ophth	Not specified	Ophth	Not specified	Paeds Surg	Ferguson	OH Adult	Narsinh	
29-Jul	Tuesday	AM	Gen Surg	Cranshaw	Litho	Not specified	Gen Surg	Tonga	Gen Surg	Taylor	Ophth	Not specified	Ophth	Not specified	OH Paeds	Dawson	OH Adult	King	
		PM	Gen Surg	Cranshaw	Litho	Not specified	Gen Surg	Tonga	Gen Surg	Taylor	Ophth	Not specified	Ophth	Not specified	ORL Adult	Neeff	OH Adult	Cathro	
30-Jul	Wednesday	AM	Gynae	Lowe	Gynae	Lowe	Gen Surg	Lushkott	Ophth	Not specified	Ophth	Not specified	Ophth	Not specified	Paeds Surg	Price	OH Paeds	Gaynor	
		PM	Gynae	Lowe	Gynae	Lowe	Gen Surg	Lushkott	Ophth	Not specified	Ophth	Not specified	Ophth	Not specified	OH Adult	Reg	OH Paeds	Dawson	
31-Jul	Thursday	AM	Gynae	Carter	Urology	Not specified	Gen Surg	Bissett	Ophth	Not specified	Ophth	Not specified	Ophth	Not specified	OH Paeds	Not specified	OH Paeds	Papa	
		PM	Gynae	McKenzie	Urology (Avail)	Not specified	Gen Surg	Bissett	Ophth	Not specified	Ophth	Not specified	Ophth	Not specified	TARPS	Coe	OH Paeds	Koopu	Potentially paeds OH
01-Aug	Friday	AM	Pvt Gynae	Bye et al	Paeds ORL	Salkeld	ABR	Not specified	Ophth	Not specified	Ophth	Not specified	Ophth	Not specified	Gynae	Harillal	OH Adult	Sealey	
		PM	Gen Surg	Hayes	Paeds ORL	Salkeld	ABR	Not specified	Ophth	Not specified	Ophth	Not specified	Ophth	Not specified	Flex		OH Paeds	Koopu	

Week A 04-Aug

			OR1		OR2		OR3		OR4		OR5		OR6		OR7		OR8		Requests
04-Aug	Monday	AM	OH Adult	Ling	Paeds Ortho	Stanley	Gen Surg	Ram	Ophth	Not specified	Ophth	Not specified	Ophth	Not specified	Paeds Surg	Ferguson	OH Paeds	Narsinh	
		PM	Urology	Rice	Paeds Ortho	Stanley	Gen Surg	Ram	Ophth	Not specified	Ophth	Not specified	Ophth	Not specified	Paeds Surg	Ferguson	OH Paeds	Corkill	
05-Aug	Tuesday	AM	Gynae	McDougal	Gen Surg	Ng	Gen Surg	Tonga	Ophth	Not specified	Ophth	Not specified	Ophth	Not specified	Gen Surg	Bartlett	OH Adult	King	
		PM	TARPS	Coe	Gen Surg	Ng	Gen Surg	Tonga	Ophth	Not specified	Ophth	Not specified	Ophth	Not specified	Gen Surg	Bartlett	OH Paeds	Corkill	
06-Aug	Wednesday	AM	Gynae	Aweidah	OH Paeds	Tandon	Gen Surg	Lushkott	Ophth	Not specified	Ophth	Not specified	Ophth	Not specified	Paeds Surg	Price	OH Paeds	Ramadas	
		PM	Gen Surg	Collinson	OH Adult	Lewis	Gen Surg	Lushkott	Ophth	Not specified	Ophth	Not specified	Ophth	Not specified	ORL Adult		OH Paeds	Narsinh	
07-Aug	Thursday	AM	Gynae	Budden	Litho	Not specified	ORL Adult	Wong	Ophth	Not specified	Ophth	Not specified	Ophth	Not specified	Paeds ORL	Salkeld	OH Paeds	Koopu	
		PM	Gynae	McKenzie	Litho	Not specified	Flex		Ophth	Not specified	Ophth	Not specified	Ophth	Not specified	Paeds ORL	Salkeld	OH Adult	Bagadam	
08-Aug	Friday	AM	Pvt Gynae	Bye et al	Gynae (Avail)	Stefanelli	Gen Surg	Lushkott	Ophth	Not specified	Ophth	Not specified	Ophth	Not specified	Flex		OH Adult		
		PM	Gen Surg	Hayes	Flex		Gen Surg	Lushkott	Ophth	Not specified	Ophth	Not specified	Ophth	Not specified	Flex		OH Paeds	Koopu	

Week B 11-Aug

			OR1		OR2		OR3		OR4		OR5		OR6		OR7		OR8		Requests
11-Aug	Monday	AM	OH Adult	Butchard	Paeds ORL	Mills/Stu	ORL Adult	Tandon	Ophth	Not specified	Ophth	Not specified	Ophth	Not specified	Paeds Surg	Ferguson	OH Adult	Narsinh	
		PM	Gynae	Liu	Paeds ORL	Mills/Stu	Gen Surg	Shaw	Ophth	Not specified	Ophth	Not specified	Ophth	Not specified	Paeds Surg	Ferguson	OH Adult	Narsinh	
12-Aug	Tuesday	AM	Gen Surg	Ng	Litho	Not specified	Gen Surg	Tonga	OH Paeds	Dawson	Ophth	Not specified	Ophth	Not specified	Gen Surg	Windsor	OH Adult	King	
		PM	Gen Surg	Ng	Litho	Not specified	Gen Surg	Tonga	Ophth	Not specified	Ophth	Not specified	Ophth	Not specified	Gen Surg	Windsor	OH Adult	Cathro	
13-Aug	Wednesday	AM	Gynae	Aweidah	Gen Surg (Avail)	Colorectal	Gen Surg	Lushkott	Ophth	Not specified	Ophth	Not specified	Ophth	Not specified	Paeds Surg	Price	OH Paeds	Gaynor	
		PM	Gynae (Avail)	Stefanelli	Gen Surg (Avail)	Colorectal	Gen Surg	Lushkott	Ophth	Not specified	Ophth	Not specified	Ophth	Not specified	OH Adult	Reg	OH Paeds	Dawson	
14-Aug	Thursday	AM	Gynae	Carter	Gynae	Ooi	Gen Surg (Avail)	Tracey	Ophth	Not specified	Ophth	Not specified	Ophth	Not specified	Urology	Not specified	OH Paeds	Papa	
		PM	Gynae	Wise	Gynae	Ooi	Gen Surg (Avail)	Tracey	Ophth	Not specified	Ophth	Not specified	Ophth	Not specified	Flex		OH Paeds	Koopu	
15-Aug	Friday	AM	Pvt Gynae	Bye et al	Paeds ORL (Av	Salkeld	ABR	Not specified	Ophth	Not specified	Ophth	Not specified	Ophth	Not specified	Gynae	Flower	OH Paeds	Narsinh	
		PM	Gen Surg	Hayes	Paeds ORL (Av	Salkeld	ABR	Not specified	Ophth	Not specified	Ophth	Not specified	Ophth	Not specified	Flex		OH Paeds	Koopu	

Week C 18-Aug

			OR1		OR2		OR3		OR4		OR5		OR6		OR7		OR8		Requests
18-Aug	Monday	AM	ORL Adult	Tandon	Paeds Ortho	Walsh	Gen Surg	Ram	Ophth	Not specified	Ophth	Not specified	Ophth	Not specified	Paeds Surg	Ferguson	OH Adult	Narsinh	
		PM	Urology	Rice	Paeds Ortho	Walsh	Gen Surg	Ram	Ophth	Not specified	Ophth	Not specified	Ophth	Not specified	Paeds Surg	Ferguson	OH Adult	Corkill	
19-Aug	Tuesday	AM	Gynae	McDougal	Gen Surg	Cranshaw	Gen Surg	Tonga	Ophth	Not specified	Ophth	Not specified	Ophth	Not specified	Gen Surg	Parry	OH Adult	King	
		PM	Oral Health		Gen Surg	Cranshaw	Gen Surg	Tonga	Ophth	Not specified	Ophth	Not specified	Ophth	Not specified	Gen Surg	Parry	OH Paeds	Corkill	
20-Aug	Wednesday	AM	Gynae	Lowe	Paeds Surg	Price	Gen Surg	Lushkott	Ophth	Not specified	Ophth	Not specified	Ophth	Not specified	OH Paeds	King	OH Paeds	Ramadas	
		PM	Gen Surg	Collinson	OH Adult	Lewis	Gen Surg	Lushkott	Ophth	Not specified	Ophth	Not specified	Ophth	Not specified	ORL Adult	Fellow	OH Paeds	Narsinh	
21-Aug	Thursday	AM	Gynae (Leave)	Sowter	Litho	Not specified	ORL Adult	Wong	Ophth	Not specified	Ophth	Not specified	Ophth	Not specified	Paeds ORL	Salkeld	OH Paeds	Koopu	
		PM	Gynae	McKenzie	Litho	Not specified	Flex - Avail		Ophth	Not specified	Ophth	Not specified	Ophth	Not specified	Paeds ORL	Salkeld	OH Adult	Bagadam	
22-Aug	Friday	AM	Pvt Gynae	Bye et al	Gynae (Leave)	Flower	Gen Surg (Avail)	Lushkott	Ophth	Not specified	Ophth	Not specified	Ophth	Not specified	Flex - Avail		OH Adult		
		PM	Gen Surg	Hayes	Flex - Avail		Gen Surg (Avail)	Lushkott	Ophth	Not specified	Ophth	Not specified	Ophth	Not specified	Flex - Avail		OH Paeds	Koopu	

Figure 1: Operating room session planner

1.1.2 Process overview

The existing SCRUM process used has seven stages identified in Figure 2.

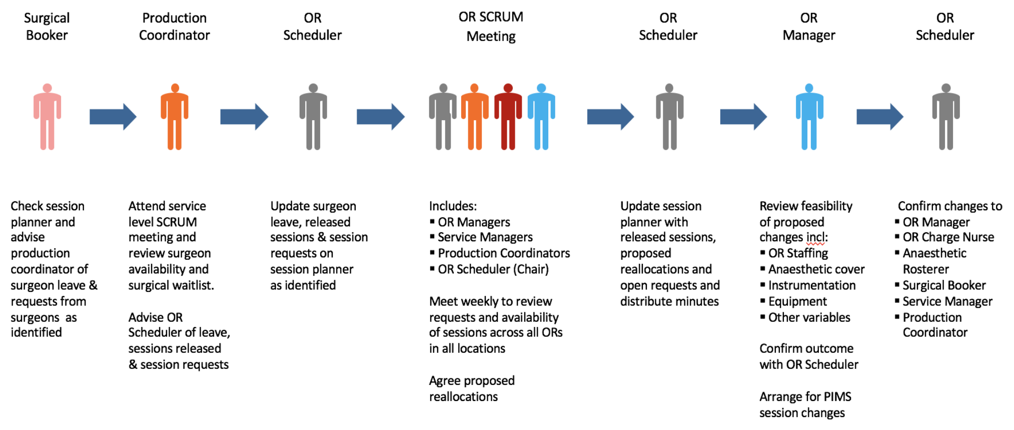

Surgical Booker
Production Coordinator
OR Scheduler
OR SCRUM Meeting
OR Scheduler
OR Manager
OR Scheduler
Check session planner and advise production coordinator of surgeon leave & requests from surgeons as identified
Attend service level SCRUM meeting and review surgeon availability and surgical waitlist.
Advise OR Scheduler of leave, sessions released & session requests
Update surgeon leave, released sessions & session requests on session planner as identified
Includes:
▪ OR Managers
▪ Service Managers
▪ Production Coordinators
▪ OR Scheduler (Chair)
Meet weekly to review requests and availability of sessions across all ORs in all locations
Agree proposed reallocations
Update session planner with released sessions, proposed reallocations and open requests and distribute minutes
Review feasibility of proposed changes incl:
▪ OR Staffing
▪ Anaesthetic cover
▪ Instrumentation
▪ Equipment
▪ Other variables
Confirm outcome with OR Scheduler
Arrange for PIMS session changes
Confirm changes to
▪ OR Manager
▪ OR Charge Nurse
▪ Anaesthetic Rosterer
▪ Surgical Booker
▪ Service Manager
▪ Production Coordinator

Figure 2: Operating room SCRUM process overview. Retrieved from *Operating Room Weekly Production Planning (SCRUM) Overview* (2014)

1. A surgical booker compares the master operating room session planner against their own information. Surgical leave and requests from surgeons or services are at this point communicated to the production coordinator.

2. The production coordinator attends SCRUM meetings at a service level where surgeon availability as well as a review of surgical wait lists are considered. The production coordinator must then inform the operating room scheduler of any changes to surgeon leave, sessions to be released and sessions to be requested for their service.

3. Before the weekly operating room SCRUM meeting the operating room scheduler must update surgeon leave, released sessions as well as session requests as communicated by the production coordinators.

4. Weekly a SCRUM meeting is held with operating room managers, service managers, production coordinators and the operator room scheduler in attendance. The objective of this meeting is to review all requests and the availability of all operating room sessions in Auckland Hospital. The outcome of this meeting is the acceptance or rejection of proposed reallocation of sessions.

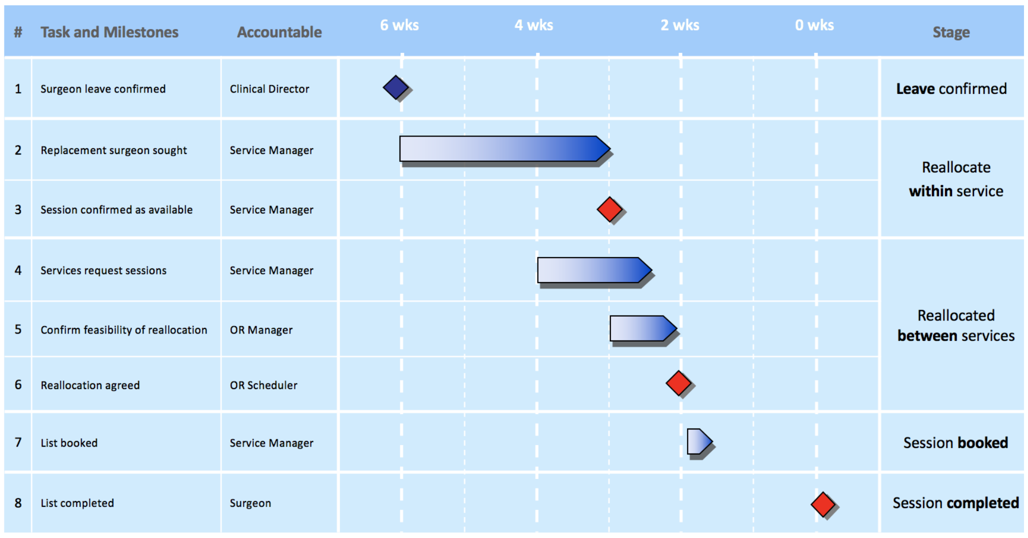
5. The operating room scheduler will reflect any changes confirmed during the SCRUM meeting by updating the operating room session planner. Minutes showing this changes are then distributed to all parties.

6. The operating managers must review the feasibility of any changes. Operating room staffing, anaesthetist cover, equipment and instrumentation must all be considered.

If the changes are feasible the operating room manager must then confirm with the operating room scheduler.

7. Finally the operating room scheduler must confirm the changes to the operating room manager, operating room charge nurse, anaesthetist rosterer, surgical booker, service manager and production coordinator.

The expected timing of the above seven stages are shown in Figure 3.



#	Task and Milestones	Accountable	Stage
1	Surgeon leave confirmed	Clinical Director	**Leave** confirmed
2	Replacement surgeon sought	Service Manager	Reallocate **within service**
3	Session confirmed as available	Service Manager	
4	Services request sessions	Service Manager	Reallocated **between** services
5	Confirm feasibility of reallocation	OR Manager	
6	Reallocation agreed	OR Scheduler	
7	List booked	Service Manager	Session **booked**
8	List completed	Surgeon	Session **completed**

Figure 3: Weekly operating room production planning time-frames . Retrieved from *Operating Room Weekly Production Planning (SCRUM) Overview* (2014)

1.1.3 Business rules

To accompany the SCRUM process a list of business rules are used:

1. A surgeon being on leave does not indicate that a session is available to be reallocated.

2. A session where the assigned surgeon is on leave must be confirmed as being used or released by the service a minimum of three weeks prior to its occurrence. Any session not confirmed which is less than three weeks away will be offered for reallocation as part of the operating room SCRUM process.

3. Once a session has been confirmed as released for reallocation by a service it can be claimed back by the original service as long as there are no other requests on it.

4. The reallocation of any session is at the discretion of the operating room manager based on the ability to provide staff and other resources.

5. Where multiple services request the same session then the priority rules will be used to determine the allocation.

6. The operating room scheduler is the point of contact for confirming availability of sessions.

If a vacant session is requested by multiple services the following priority rules are used in order to determine which service receives the session:

1. The clinical priority of the case(s) intended for the requested session.
2. The overall level of ESPI (Elective Services Patient Flow Indicators) 5 risk for each service assuming that both services intend to use the session for completion of an ESPI 5 case.
3. The overall progress by service against planned discharges.
4. The timing of the submitted request. Priority given to services submitting requests more than two weeks in advance over requests received within two weeks.
5. The ability of the operating room to complete the planned sessions proposed by each service.
6. The level of session usage of each service and the previous record of services in releasing sessions.

1.1.4 Discussion

As can be seen in the above descriptions, there are three key components to the operating room scheduling process at Auckland Hospital:

1. The operating room session planner which is the single point of reference for all planned operating room sessions.
2. The operating room scheduler who is the primary point of coordination for all changes and requests to regular operating room allocations as well as responsible for keeping the session planner up to date.
3. The operating room SCRUM meeting which is the primary forum for discussion of requests and reallocation of sessions.

In reference to Figure 2 the operating room scheduler is involved in six of the seven stages which represents a large bottleneck. The accuracy of the session planner is directly connected to the timely fashion in which it is updated by the operating room scheduler.

Whilst working directly with the operating room scheduler it is proposed that an updated process involving a purpose-built software would alleviate this bottleneck as well as increasing the accuracy and visibility of the session planner.

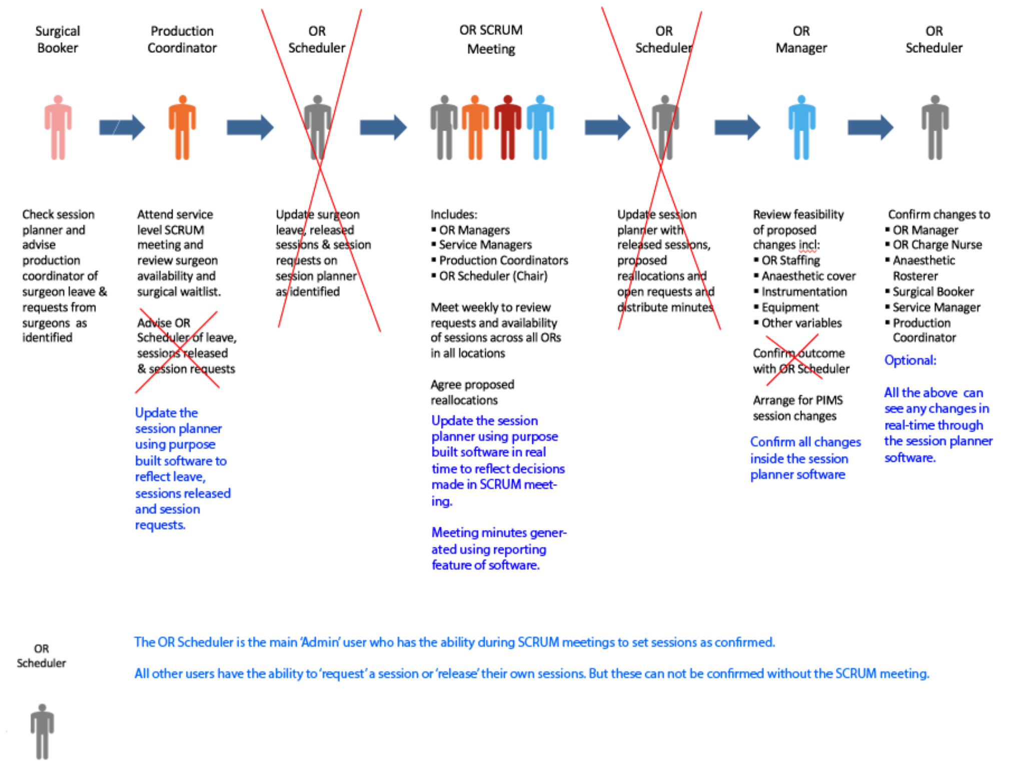

Surgical Booker
Production Coordinator
~~OR Scheduler~~
OR SCRUM Meeting
~~OR Scheduler~~
OR Manager
OR Scheduler
Check session planner and advise production coordinator of surgeon leave & requests from surgeons as identified
Attend service level SCRUM meeting and review surgeon availability and surgical waitlist.
~~Advise OR Scheduler of leave, sessions released & session requests~~
Update the session planner using purpose built software to reflect leave, sessions released and session requests.
~~Update surgeon leave, released sessions & session requests on session planner as identified~~
Includes:
▪ OR Managers
▪ Service Managers
▪ Production Coordinators
▪ OR Scheduler (Chair)
Meet weekly to review requests and availability of sessions across all ORs in all locations
Agree proposed reallocations
Update the session planner using purpose built software in real time to reflect decisions made in SCRUM meeting.
Meeting minutes generated using reporting feature of software.
~~Update session planner with released sessions, proposed reallocations and open requests and distribute minutes~~
Review feasibility of proposed changes incl:
▪ OR Staffing
▪ Anaesthetic cover
▪ Instrumentation
▪ Equipment
▪ Other variables
~~Confirm outcome with OR Scheduler~~
Arrange for PIMS session changes
Confirm all changes inside the session planner software
Confirm changes to
▪ OR Manager
▪ OR Charge Nurse
▪ Anaesthetic Rosterer
▪ Surgical Booker
▪ Service Manager
▪ Production Coordinator
Optional:
All the above can see any changes in real-time through the session planner software.
OR Scheduler
The OR Scheduler is the main 'Admin' user who has the ability during SCRUM meetings to set sessions as confirmed.
All other users have the ability to 'request' a session or 'release' their own sessions. But these can not be confirmed without the SCRUM meeting.

Figure 4: Proposed new operating room scheduling process

The proposed new operating room scheduling process seen in Figure 4 removes much of the need for the operating room scheduler to be used as a middleman. No longer does the accuracy of the session planner rely on one person.

The new process involves four main steps and one optional step as follows:

1. The surgical booker compares the master operating room session planner found on the purpose-built software against their own information. Surgical leave and requests from surgeons or services are at this point updated on the purpose-built software.

2. The production coordinator attends SCRUM meetings at a service level where surgeon availability as well as a review of surgical wait lists are considered. The production coordinator then reflects changes to surgeon leave, sessions to be released and sessions to be requested by updating the purpose-built software.

3. Weekly a SCRUM meeting is held with operating room managers, service managers, production coordinators and the operator room scheduler in attendance. The objective of this meeting is to review all requests and the availability of all operating room sessions in Auckland Hospital. The outcome of this meeting is the acceptance or rejection of proposed reallocation of sessions.

   All outcomes are recorded in real time on the purpose-built software. A report of all changes (minutes) is then distributed.

4. The operating managers must review the feasibility of any changes. Operating room staffing, anaesthetist cover, equipment and instrumentation must all be considered. If the changes are feasible the operating room manager can confirm through the purpose-built software.

5. A final optional stage is for the operating room scheduler to inform the operating room manager, operating room charge nurse, anaesthetist rosterer, surgical booker, service manager and production coordinator of all changes. This stage can be replaced by simply looking at the up to date session planner in the purpose-built software.

This proposed process cuts all need for the operating room scheduler to act as the middle man between involved parties. It puts responsibility to multiple parties to ensure the single source of truth (the session planner) is kept current. By doing this the visibility of changes is increased. An example being a surgical booker directly indicates through the purpose-built software a surgeon will be on leave on a specific date, this is instantly available information to all reviewing the session planner through the software. The current process, the surgical booker emails the operating room scheduler of a surgeon being on leave, the operating room booker would then update the session planner spreadsheet as soon as possible. Only after this would the leave be indicated on the session planner spreadsheet hosted on the intranet.

All business rules from the original process remain the same except for a variation to business rule 6. It should now read: The session planner inside the purpose-built software is the point of reference for confirming availability of operating room sessions.

1.2 Hypothesis and research questions

It is hypothesised that the proposed operating room scheduling process can increase operating room utilisation by increasing visibility of information through a purpose-built software.

Research Question: Can the existing operating room scheduling process in Auckland Hospital be improved by implementing a purpose-built software?

Chapter 2 Literature Review

2.1 Introduction

This chapter is split into several sections that review the literature around specific aspects in relation to operating room scheduling. Section 2.2 is going to discuss the phases involved in the scheduling of operating rooms, pinpointing the phases of particular interest to this research. Section 2.3 will identify the common problems surrounding operating room scheduling, this will give valuable insight towards which areas deserve the most focus when working to develop a more efficient and effective operating room booking system. Section 2.4 introduces existing systems used for the scheduling of operating rooms. By reviewing existing systems it can be identified how effective they are when dealing with the problems raised in section 2.3, which in turn gives valuable insight for the development of a new system.

2.2 Scheduling phases

Before discussing the levels involved when scheduling operating rooms it is important to understand the following resource management strategies when allocating operating rooms:

- **Open Scheduling**
  Open scheduling is often referred to as a first come first serve strategy. No operating resources are allocated and an empty schedule is filled with surgical cases in order of requested. This strategy heavily favours surgeons who have the ability to scheduling well in advance due to predictability. But is very unfavourable toward specialities who do not have that degree of certainty (Patterson, 1996).

- **Block Scheduling**
  ”Block booking is essentially a reservation. Let’s say you are going to dinner on Saturday night, and you don’t want to wait for a table. You call ahead and reserve-block book-a table. Your evening is pretty well planned now. You don’t have to worry about rushing around or getting to the restaurant early-you are taken care

of." (Earnhart, 2003).
Block scheduling is when a block of time is allocated to either a surgeon or group of surgeons. This block is then "owned" by them and in the true definition of black scheduling would never be released (Miller et al., 2014). This strategy provides surgeons with a schedule that is predictable so they can book individual sessions in this block. It's downfall is if the session not going to be used, utilisation suffers. The reallocation of block time last minute can raise difficult issues especially with two competing groups requesting that session, on top of staffing and resource availability.

- **Modified Block Scheduling**
  Modified block scheduling is a modification of both the above strategies. There are two main modifications that are commonly employed (Patterson, 1996). The first is to block a share of allocation while leaving the remainder open and the second is to have an agreed upon time where unused blocks must be released and made available to other surgeons or groups. This method balances the needs of specialities where some can not book as far ahead as others. Although it does mean an efficient and visible way of releasing sessions must be employed in order to maximise utilisation.

To keep the review scoped to the current resource management strategy used at Auckland Hospital from here onward it is assumed a modified block scheduling strategy is used.

Abedini, Li, and Ye (2017) identify three main levels involved in the scheduling of operating rooms:

1. **Strategic:** Using historical data and forecasting, operating room blocks are allocated to specific surgery groups. This phase generally has a horizon of one year (Blake & Carter, 2002).

2. **Tactical:** Development of a master surgical schedule is the main outcome from this phase. This schedule outlines the number and type of available operating rooms, hours the rooms are open as well as the surgeons or surgical groups that the blocks are allocated too. Time horizon of this phases is typically one to three months (Guerriero & Guido, 2011).

3. **Operational:** This phase involves the scheduling at a much more detailed level such as assigning each case to specific operating rooms, start and end times of scheduled cases and reservation of specialised equipment. This schedule can be modified to cater to unexpected events such as cancellations, surgery going under or over allocated times as well as the inclusion of urgent cases. This phase is completed on a daily basis (Guerriero & Guido, 2011).

The scope of this project is mainly concerned with the tactical level and to an extent the strategic level. But as the three levels are dependent on each other (the outcomes from each being the input in to the next phase) it is important to have a thorough understanding of the whole process.

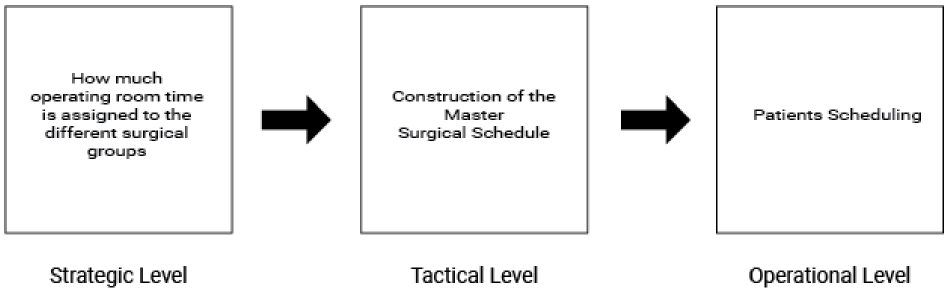

How much operating room time is assigned to the different surgical groups
Construction of the Master Surgical Schedule
Patients Scheduling
Strategic Level
Tactical Level
Operational Level

Figure 5: Hierarchical decision levels. Adapted from Guerriero and Guido (2011)

2.2.1 Strategic level

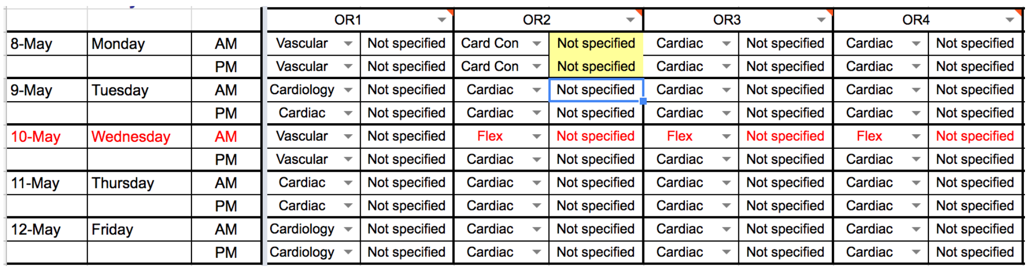
The goal of the strategic level is to determine the surgeries that are to be performed as well as the medical staffing. This allows the allocation of resources which is needed in order to ascertain costing involved. This furthermore allows decisions to made in order to maximise profit or minimise costs depending on business strategy (Blake & Carter, 1997).

Strategic planning is mostly based on historical data and forecasting with a horizon of one or more years (Guerriero & Guido, 2011). There is conflicting reports in the literature about the effectiveness of this process. Masursky, Dexter, OLeary, Applegeet, and Nussmeier (2008) studied three hospitals who produce long term forecasts in an attempt to predict long term changes in operating room usage using the changes to local population. Their findings showed that these forecasts were highly inaccurate. It was noted that a more accurate method is to save billing and operating room time data display it graphically year by year and model using corresponding forecasting methods to show changes. A study by Dexter, Macario, Qian, and Traub (1999) tested the validity of a common forecasting method in which the average of the most recent year's total hours of elective cases is used to predict future usage of operating room time. Their results showed that this method provided adequate results.

There have been numerous studies in which improvements have been attempted to be made at the strategic level. It was discovered by VanBerkel and Blake (2007) that a significant bottle neck in wait times was caused by resource allocation. Beds in particular were identified as a key resource causing problems. They observed through simulation that redistribution of beds at the time of strategic level planning operating room time was better utilised.

2.2.2 Tactical level

Guerriero and Guido (2011) state that it is accepted that most elective surgeries do not differ over several weeks. As such the tactical level is dedicated to developing a master surgical schedule. A master surgical schedule is constructed to define which surgical group or surgeon is allocated to individual time blocks per operating room per day as seen in Figure 6. The horizon of the planning period is predetermined by the planning team involved commonly between one month to one year.



			OR1		OR2		OR3		OR4	
8-May	Monday	AM	Vascular	Not specified	Card Con	Not specified	Cardiac	Not specified	Cardiac	Not specified
		PM	Vascular	Not specified	Card Con	Not specified	Cardiac	Not specified	Cardiac	Not specified
9-May	Tuesday	AM	Cardiology	Not specified	Cardiac	Not specified	Cardiac	Not specified	Cardiac	Not specified
		PM	Cardiac	Not specified	Cardiac	Not specified	Cardiac	Not specified	Cardiac	Not specified
10-May	Wednesday	AM	Vascular	Not specified	Flex	Not specified	Flex	Not specified	Flex	Not specified
		PM	Vascular	Not specified	Cardiac	Not specified	Cardiac	Not specified	Cardiac	Not specified
11-May	Thursday	AM	Cardiac	Not specified	Cardiac	Not specified	Cardiac	Not specified	Cardiac	Not specified
		PM	Cardiac	Not specified	Cardiac	Not specified	Cardiac	Not specified	Cardiac	Not specified
12-May	Friday	AM	Cardiology	Not specified	Cardiac	Not specified	Cardiac	Not specified	Cardiac	Not specified
		PM	Cardiology	Not specified	Cardiac	Not specified	Cardiac	Not specified	Cardiac	Not specified

Figure 6: Example master surgical schedule covering four operating rooms over five days

The master surgical schedule is constructed taking in to account historical data as well as actual demand from waiting lists and requests for surgeries. The master surgical schedule construction is an iterative process with input and negotiation from various parties. It is likely that a request from one surgical group would directly effect another thus each request has to be considered, accommodated (if possible), and finally reflected on the master schedule. This process is repeated as often as necessary.

Of particular note to this research is that a base master schedule is often reused over extended amounts of time (Kharraja, Albert, & Chaabane, 2006). The consequence of this is that it needs to be constantly re-optimized to reflect various scenarios. Aspects such as availability of nurses, surgeons and other medical staff as well as infrequent events (shutdowns / public holidays) can in effect cause major changes to the normal schedule thus these need to be identified early with suitable visibility to all involved.

At this stage of the process many key decisions such as the hiring of new staff, increases in ward usage as well as what surgical groups should be provided with increased resources are made. One such technique of making these decisions is to us the data envelopment analysis technique. Oneill and Dexter (2004) show this techniques potential as a tool for operating room managers decision making in regards to increasing resource allocations all based around data collected at this stage. Multi-factor efficiency and non-radial super-efficiency have also being investigated with promising results (O'neill, 2005). The strength of these techniques is directly dependent around the accuracy of the available data thus it is imperative that regardless of process used, data should be current and valid.

2.2.3 Operational level

The final level of the process is the operational level. This levels main purpose is to create a schedule for the day of surgery. Cardoen, Demeulemeester, and Beliën (2009) further split this level in to two distinct steps: an assignment step and a sequencing step.
The assignment step involves a discussion between the patient and the surgeon. This discussion is influenced by many factors such as free capacity availability in the surgeons allocated blocks as well as patient and surgeon preferences. The result of this stage is the patient and surgeon are aware of the day that surgery will happen but they will not know specific timings, this is to be decided in the sequencing step.
The sequencing step generally happens the day before surgery. The theatre manager at this time is aware of all the patients who are to be operated on the following day. There may be a preferred sequence by both surgeons and nurses but in order to resolve conflicts as well as balance limited resources this may not be possible. Once a sequence has been determined the patients will be contacted to inform them of surgery starting times.
Although this level is out of scope for the project input from the strategic and tactical levels are direct inputs. The theatre manager needs to be aware as soon as possible to changes in allocated time blocks as well as changes to speciality or surgeon allocations in order the efficiently and correctly produce daily schedules.

2.3 Operating room scheduling problems

The single largest cost when delivering surgical care is the operating room (Macario, Vitez, Dunn, & McDonald, 1995), with a significant portion of these costs being allocated to surgical staffing (Dexter, Macario, Traub, Hopwood, & Lubarsky, 1999). This highlights the need for efficient and robust systems to maximise labour, productivity and to reduce wastage in operating room resources.

The scheduling of operating rooms is a complex problem, with the organisation of various stakeholders, resources and services having to be taken in to consideration. Surgical and nursing staff, anaesthesiologist, medical equipment, and recovery beds in surgical wards all have to be taken in to consideration when scheduling operating room sessions. Li, Rafaliya, Baki, and Chaouch (2017) identify that a well-designed system should be concerned with the welfare of all involved parties and should prioritise the allocation of available resources in an efficient and effective manner.

2.3.1 The operating room

Operating rooms are scarce and expensive (Stepaniak & Pouwels, 2017). The demand for operating room capacity in some specialisations is beyond the supply, which can have negative side effects such as long delays in surgery for the patient, emotional stress for the patient and family along with the possibility of transfer to other hospitals resulting in lost revenue. Stepaniak and Pouwels (2017) identify that these issues can have serious implications on the running of a hospital, making it of the best interests for hospital management to optimise usage while taking in to account patient waiting times. The above issues are commonly referred to in the attempts for improved operating room scheduling systems which will be discussed further in Section 2.4.

Typically operating rooms will be allocated by speciality to the available operating rooms. The variability around types of surgeries as well as the fluctuations involved in the demand and duration of these surgeries makes regular schedules very hard to create (Li et al., 2017).

On top of all these issues hospital management has to efficiently schedule at a day to day level ensuring a balance between over-utilisation and under-utilisation. Strum, Vargas, and May (1999) define under-utilisation as time spent during scheduled operational hours that an operating room is not used and over-utilisation is defined as time used by scheduled cases past the end of scheduled operational hours. Dexter and Traub (2002) explain it well when they say ”Hypothetically, a service could be allocated two operating rooms for the day but perform all of its cases in one operating room. There could be both unused time in one and overtime in the other.”.

2.3.2 The operating room staffing

There is a magnitude of staffing involved in the use of operating rooms all with different goals and expectations. In a block scheduling strategy a theatre manager or charge nurse has to ensure that surgeons, nursing staff, anaesthesiologists are all aware of block bookings where they are required. Even after this the smooth running of the existing schedule is reliant on the efficiency of staff.
”If surgeons are not using their blocks effectively, they’re tying up resources that could be used by other surgeons and groups, this is why its so important to have a block management system that details procedures holders must follow if they need to make changes to their blocks.” (Sadler, 2015). Any delays in procedures have to be made up at a future time which will normally include overtime adding to an already high staff costing.

2.3.3 Cancellations

Cancellations are unfortunately an aspect that has to be dealt with regularly. A study by Appavu, Al-Shekaili, Al-Sharif, and Elawdy (2016) investigated reasons for operating room cancellations. Their findings showed that 26% of surgeries were cancelled with 63% of those being due to patient no shows, 17% due to surgical reasons and only 2% of cancellations due to operating room reasons.
If there is no similar and ready surgeries that can be pushed forward at the time this means

the operating room will be underutilised. Other than improvements to how patients are reminded and informed about bookings not much can be done from a planning perspective around this area. It is important to have a platform to identify reasons why this has occurred so further research and work can be done around this area.

2.4 Existing operating room scheduling systems

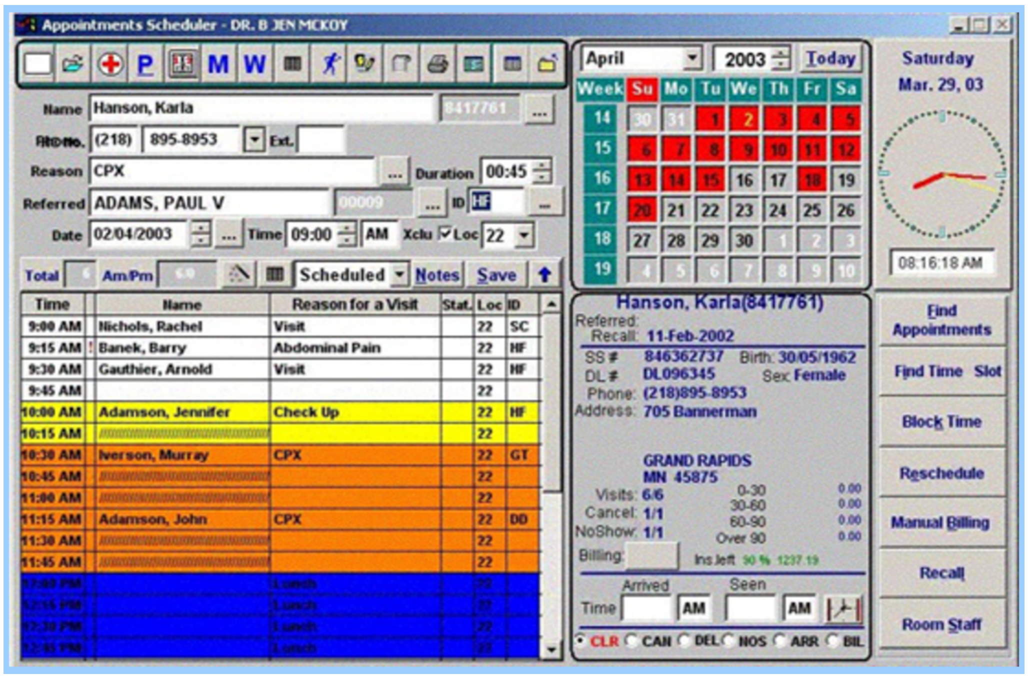
The most common method to scheduling is to follow the above planning levels with everything recorded in spreadsheets. In fact there is very little in terms of software solutions for this particular problem which further validates a need for more exploration.

At individual levels many studies have applied mathematical models in order to optimise either profits or utilisation. Yet this pure output does not translate well in to use as it is still in either a spreadsheet or graph. They are also rigid, meaning there is very little room for change or exceptional circumstances.

At an operational level one commercial software was found (*Operating Room Scheduling*, n.d.). It claims it's features as:

- Easy scheduling of operating rooms
- Print outs of staff schedules for surgery bookings
- Colour coded scheduling to view at a glance
- Doctor's day-list for chart tracking
- Multi-room scheduling

The software although very dated seems sufficient for a doctor who just wants to know their own schedule but nothing is mentioned in regards to an overall view for a whole operating theatre which excludes it from a tool that would be useful in this research.


Appointments Scheduler - DR. B JEN MCKOY

Name: Hanson, Karla 8417761
Phone No.: (218) 895-8953 Ext.
Reason: CPX Duration: 00:45
Referred: ADAMS, PAUL V 00009 ID: HF
Date: 02/04/2003 Time: 09:00 AM Xclu ✓Loc 22

Total 6 Am/Pm 6/0 Scheduled Notes Save

Time	Name	Reason for a Visit	Stat.	Loc	ID
9:00 AM	Nichols, Rachel	Visit		22	SC
9:15 AM	Banek, Barry	Abdominal Pain		22	HF
9:30 AM	Gauthier, Arnold	Visit		22	HF
9:45 AM				22	
10:00 AM	Adamson, Jennifer	Check Up		22	HF
10:15 AM				22	
10:30 AM	Iverson, Murray	CPX		22	GT
10:45 AM				22	
11:00 AM				22	
11:15 AM	Adamson, John	CPX		22	DD
11:30 AM				22	
11:45 AM				22	
12:00 PM		Lunch		22	
12:15 PM		Lunch		22	
12:30 PM		Lunch		22	
12:45 PM		Lunch		22	

April 2003 Today

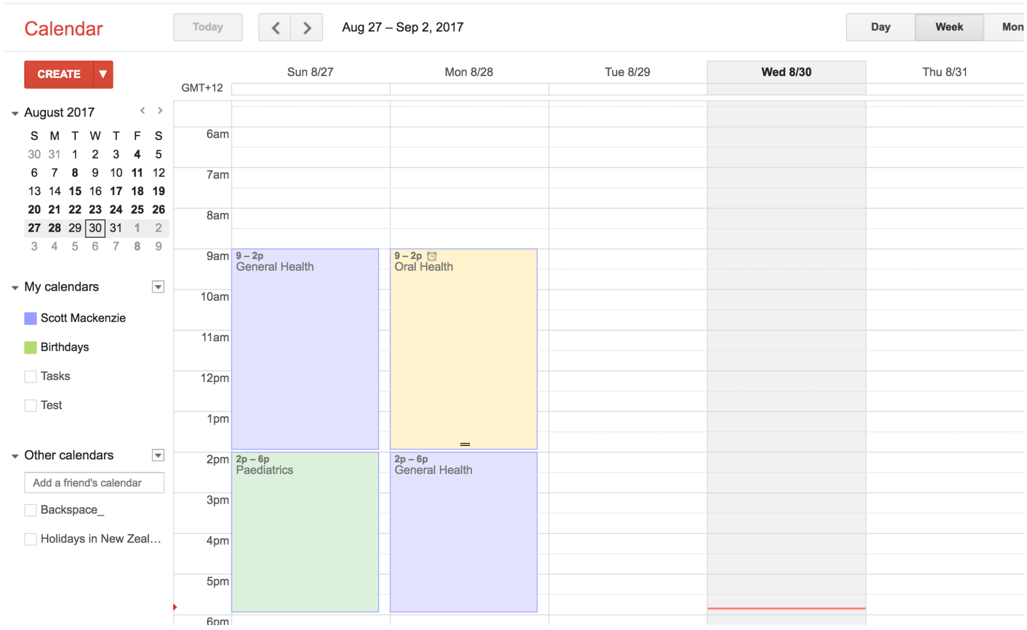
Week	Su	Mo	Tu	We	Th	Fr	Sa
14	30	31	1	2	3	4	5
15	6	7	8	9	10	11	12
16	13	14	15	16	17	18	19
17	20	21	22	23	24	25	26
18	27	28	29	30	1	2	3
19	4	5	6	7	8	9	10

Saturday Mar. 29, 03
08:16:18 AM

Hanson, Karla(8417761)
Referred:
Recall: 11-Feb-2002
SS #: 846362737 Birth: 30/05/1962
DL #: DL096345 Sex: Female
Phone: (218)895-8953
Address: 705 Bannerman
GRAND RAPIDS
MN 45875
Visits: 6/6 0-30 0.00
Cancel: 1/1 30-60 0.00
NoShow: 1/1 60-90 0.00
Over 90 0.00
Billing: Ins left 90 % 1237.19
Arrived Seen
Time AM AM
CLR CAN DEL NOS ARR BIL

Figure 7: Daily schedule software. Retrieved from *Operating Room Scheduling* (n.d.)

The closest software found to what is required would be a general organisation calendar such as Google Calendar (*Google Calendar*, n.d.). Although these tools do not offer anywhere near the sophistication needed to properly function an operating room scheduling system, many of the visual element could be adapted to portray information clearly.


Calendar Today Aug 27 – Sep 2, 2017 Day Week Month

CREATE

August 2017

S	M	T	W	T	F	S
30	31	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	7	8	9

My calendars: Scott Mackenzie, Birthdays, Tasks, Test
Other calendars: Add a friend's calendar, Backspace_, Holidays in New Zeal…

GMT+12 Sun 8/27 Mon 8/28 Tue 8/29 Wed 8/30 Thu 8/31

Sun 8/27: 9 – 2p General Health; 2p – 6p Paediatrics
Mon 8/28: 9 – 2p Oral Health; 2p – 6p General Health

Figure 8: A demonstration of how Google Calendar could show block bookings. Retrieved from *Google Calendar* (n.d.)

Chapter 3. Research Design

3.1 Introduction

The aim of this research is to improve and extend upon an existing system for the scheduling and usage of operating rooms in Auckland Hospital. It is hypothesised that through the use of a purpose-built software this system can improve efficiency, clarity and work flow in comparison to it's tedious manual counterpart. To test this hypothesis a software prototype must be designed and developed in order to demonstrate it's usage. Therefore any research methodology used must include the development of software as well as the evaluation of its use.

Both Action Research and Design Science were initially investigated as potential methodologies applicable to this research. Design Science was ultimately decided upon largely due to the explanation by Papas et al. (2012) around the definition of success to each methodology. It is explained that Action Research's definition of success is the improvement of practice being observed whereas Design Science's philosophy implies that the research is a success if the improvement of practice is observed by the use of an artefact. This suggests that the use of an artefact is not key to Action Research. By using a Design Science approach the artefact is key to the research allowing focus to be on the design, development and evaluation of the hypothesised purpose-built software.

Immediately preceding the Design Science methods a literature review has been conducted in order to identify the background and current knowledge in the problem space, this also allowed for a more overall understanding of the processes to which this studies focus area is just a small part. Following this the Design Science process which will be explained in more depth below.

3.2 Design Science

The main purpose of design science is achieving knowledge and understanding of a problem by building and using an artefact (Von Alan, March, Park, & Ram, 2004). Von Alan et al. (2004) present a set of guidelines of which design science research should address:

1. **Design as an artefact:** The research must produce an innovative and purposeful artefact for the specific problem domain.
2. **Problem relevance:** An artefact must either solve a problem that has not yet been solved or it should be an improved solution on an existing process.
3. **Design evaluation:** The quality and utility of the artefact must be demonstrated in regards to the problem area. These evaluations must be done using well-executed methods.
4. **Research contributions:** Well executed Design Science research must provide clear and verifiable contributions to the problem space.
5. **Research rigour:** Design Science research is reliant upon rigorous design, development and evaluation of the designed artefact.
6. **Design as a search process:** The search for an effective artefact requires utilising available means to reach desired ends while satisfying laws in the problem environment.
7. **Communication of research:** The research must be presented in a way that is meaningful to both management and technology orientated audiences.

The Design Science methodology process model which has been chosen for this research was designed by Papas et al. (2012). It has six key activities as shown in Figure 9 all of which are designed to address the guidelines above.

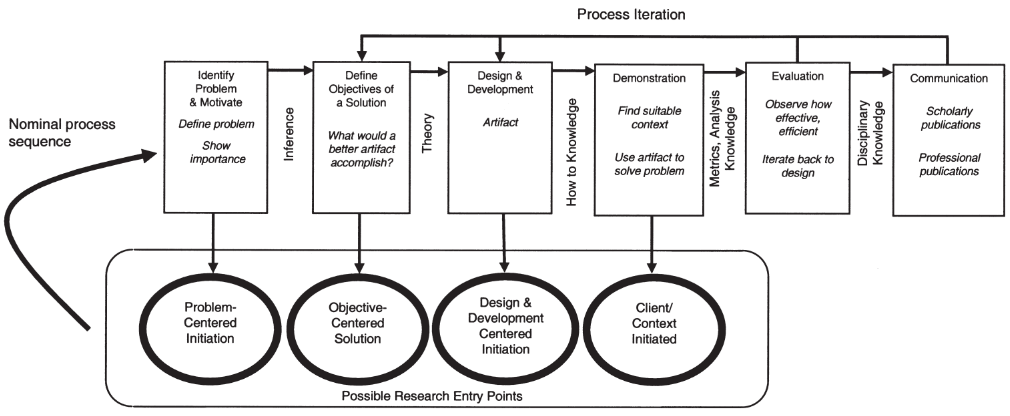

Process Iteration
Nominal process sequence
Identify Problem & Motivate
Define problem
Show importance
Inference
Define Objectives of a Solution
What would a better artifact accomplish?
Theory
Design & Development
Artifact
How to Knowledge
Demonstration
Find suitable context
Use artifact to solve problem
Metrics, Analysis Knowledge
Evaluation
Observe how effective, efficient
Iterate back to design
Disciplinary Knowledge
Communication
Scholarly publications
Professional publications
Problem-Centered Initiation
Objective-Centered Solution
Design & Development Centered Initiation
Client/ Context Initiated
Possible Research Entry Points

Figure 9: Design Science methodology model. Retrieved from Papas et al. (2012)

3.2.1 Identify the problem and motivation

By identifying the problem you can justify the value of a solution. The problem definition will be constantly referred too when developing an artefact as the key output of this research should be an artefact which solves the original problem. By identifying a satisfactory solution to the problem the researcher and audience are motivated to pursue it. By clearly identifying the problem it shows the researchers understanding of the area.

For this research the problem and motivation has been explained in depth in Chapter 1, while the targeted literature review in Chapter 2 shows a deeper understanding of the processes involved.

3.2.2 Define the objectives of a solution

With an understanding of what is possible in the subject area objectives for a satisfactory solution may be outlined. Objectives can be both qualitative or quantitative. By identifying these objectives it is possible to measure success of the project in comparison to alternative methods or solutions which have been discussed in Chapter 1 and Chapter 2.

For this research objectives have been portrayed as 'user stories' (Adolph, Cockburn, & Bramble, 2002). User stories are a popular tool in software development for modelling requirements. They allow a list of actions to be completed with an expected outcome to be described. By using these user stories as test cases the effectiveness of solutions can be measured.

The user stories for this research can be found in Chapter 4.

3.2.3 Design and Development

This activity involves the development of an artefact. For this research the artefact is in the form of a purpose-built software. The architecture decisions as well as the software development methods used to develop this software to meet functionality described through the use cases are key parts of this activity and are described in depth in Chapter 4.

3.2.4 Demonstration and Evaluation

Demonstration of the software solving the initial problems described is essential. For this research this is done through the software being used live side by side with the existing manual system. Using the results of these demonstrations features that need more work as they do not satisfy the use cases can be identified starting an iterative process by returning to the Design and Development activity. These evaluations are analysed in Chapter 6.

3.2.5 Communication

A key guideline to Design Science research is the communication of research. Research has to be presented for both management and technology orientated audiences. This communication has to identify the importance of the research, the artefact, its utility and novelty, the rigour used in its design and its effectiveness in use.

For this research this thesis as well as the software artefact is to communicate all the above.

Chapter 4. System Design

4.1 Requirements gathering

As a means of gathering requirements a number of objectives which are essential to the success of the research were listed as user stories. User stories are short descriptions that detail functionality as told by the user's perspective. They usually follow a template such as:

As a [user], I want [outcome] so that [reason].

This is then followed by acceptance criteria as a way to confirm the requirement has been met. It should be noted that no where in the user story is any detail to how the user story should be done.

User stories were chosen opposed to traditional requirement documents as a means to capture the high level requirements without limiting development to specifics. Traditional requirements documents detail how the software should and will act. Bartlett (2016) shows these differences well using a basic e-commerce site as an example:

- As a *customer*, I want to be able *to view the items in my cart* so that *I know for sure what Im purchasing*.

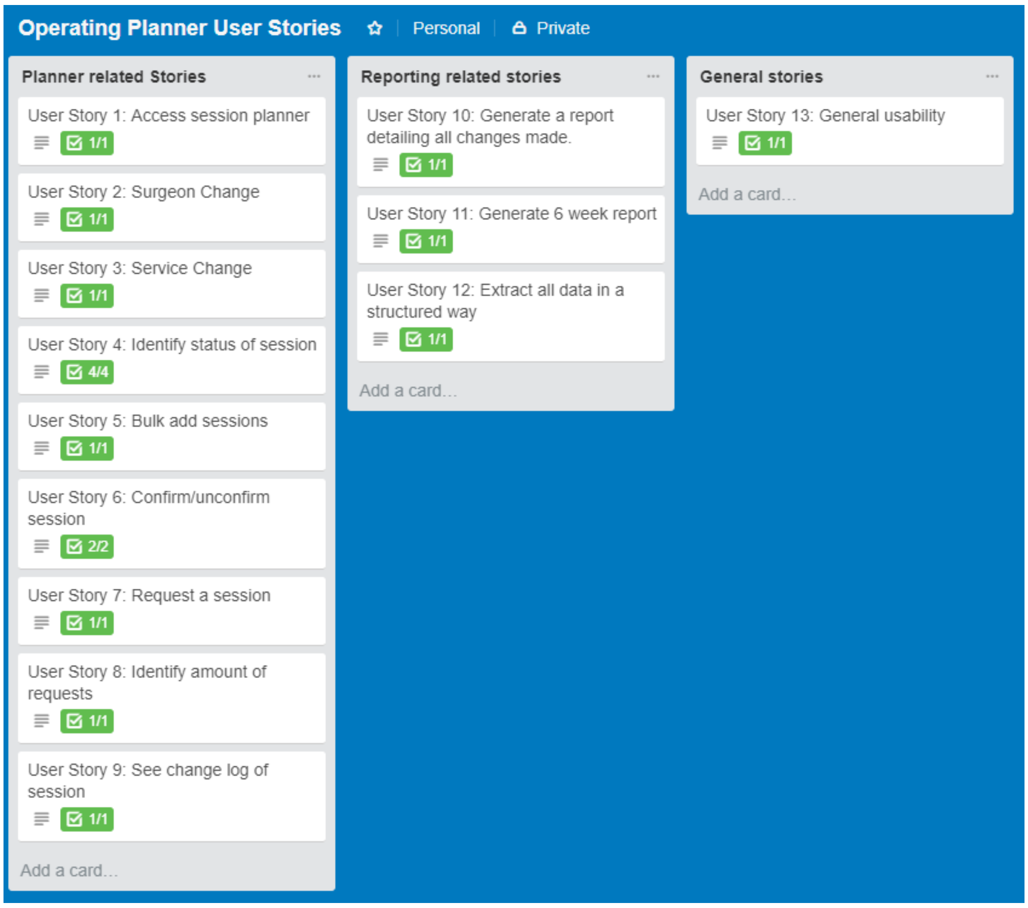
As can be seen, this user story does not include any specific details on implementation which allows more discussion and more flexibility during development. Whereas in contrast a requirements document may list requirements as such:

- Display the name of each item in the shopping cart.
- Display the quantity of each item in the shopping cart.
- Allow the user to remove any items in the shopping cart.

This is much more rigid way of approaching which was decided against in order to allow change throughout.

4.1.1 User Story Format

In order to have the user stories readily available Trello (Atlassian, 2016) was used. Trello is a web based application where you can create a board and add individual cards which can be used to simulate a Kan-ban board (Gross & McInnis, 2003).



Operating Planner User Stories | Personal | Private

Planner related Stories

- User Story 1: Access session planner — 1/1
- User Story 2: Surgeon Change — 1/1
- User Story 3: Service Change — 1/1
- User Story 4: Identify status of session — 4/4
- User Story 5: Bulk add sessions — 1/1
- User Story 6: Confirm/unconfirm session — 2/2
- User Story 7: Request a session — 1/1
- User Story 8: Identify amount of requests — 1/1
- User Story 9: See change log of session — 1/1

Add a card...

Reporting related stories

- User Story 10: Generate a report detailing all changes made. — 1/1
- User Story 11: Generate 6 week report — 1/1
- User Story 12: Extract all data in a structured way — 1/1

Add a card...

General stories

- User Story 13: General usability — 1/1

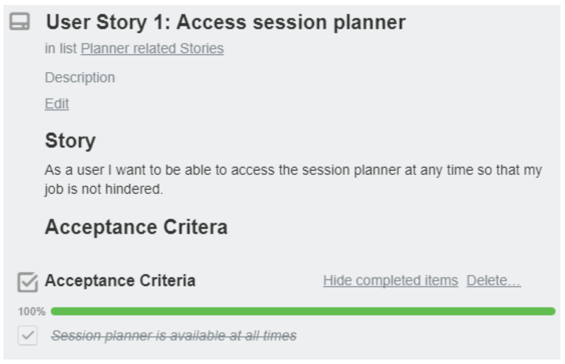
Add a card...

Figure 10: User stories shown in Kan-ban board style using Trello

The individual cards are styled so that a lot of information can be gathered with just a quick glance. They are numbered and named so that their stories can easily be identified and referenced. An indication to the amount of acceptance criteria that have been met is also present in this view. If more information is needed the individual cards can be expanded where the user story and acceptance criteria can be seen in full as well as any other comments anyone may have left.

4.1.2 User Stories

The following are high level user stories in which were identified crucial to the projects success. These provide the basis for the rest of the systems design as well as a benchmark for the systems evaluation.



User Story 1: Access session planner

in list Planner related Stories

Description

Edit

Story

As a user I want to be able to access the session planner at any time so that my job is not hindered.

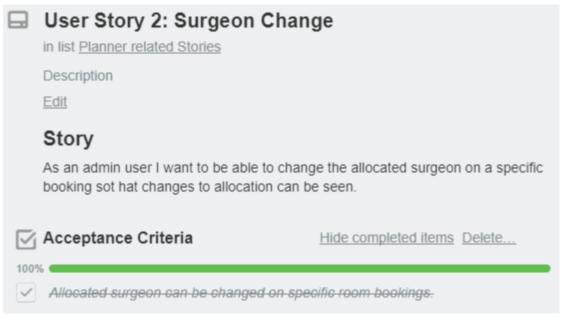
Acceptance Critera

Acceptance Criteria Hide completed items Delete...

100%

✓ ~~Session planner is available at all times~~

Figure 11: User Story 1



User Story 2: Surgeon Change

in list Planner related Stories

Description

Edit

Story

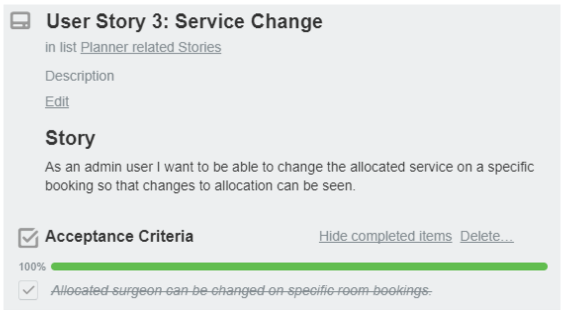
As an admin user I want to be able to change the allocated surgeon on a specific booking sot hat changes to allocation can be seen.

Acceptance Criteria Hide completed items Delete...

100%

✓ ~~Allocated surgeon can be changed on specific room bookings.~~

Figure 12: User Story 2



User Story 3: Service Change

in list Planner related Stories

Description

Edit

Story

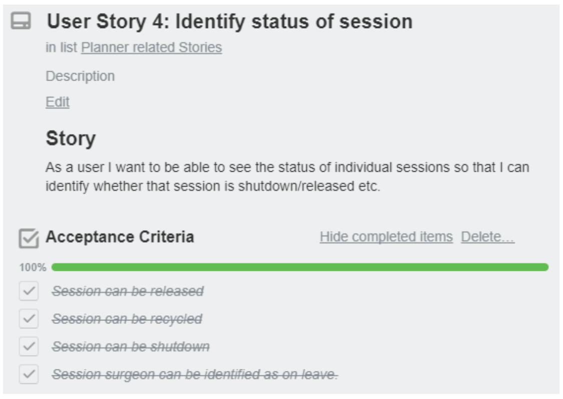
As an admin user I want to be able to change the allocated service on a specific booking so that changes to allocation can be seen.

Acceptance Criteria Hide completed items Delete...

100%

✓ ~~Allocated surgeon can be changed on specific room bookings.~~

Figure 13: User Story 3



User Story 4: Identify status of session

in list Planner related Stories

Description

Edit

Story

As a user I want to be able to see the status of individual sessions so that I can identify whether that session is shutdown/released etc.

Acceptance Criteria Hide completed items Delete...

100%

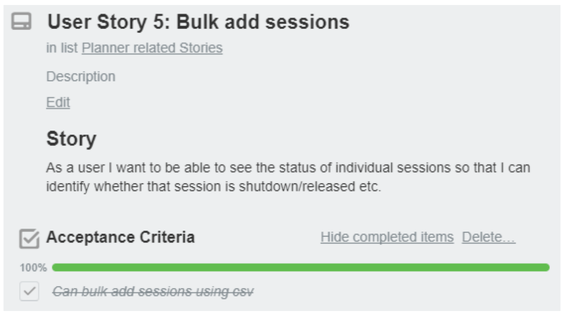
✓ ~~Session can be released~~

✓ ~~Session can be recycled~~

✓ ~~Session can be shutdown~~

✓ ~~Session surgeon can be identified as on leave.~~

Figure 14: User Story 4



User Story 5: Bulk add sessions

in list Planner related Stories

Description

Edit

Story

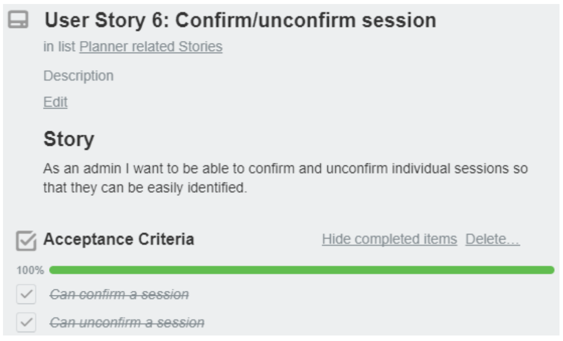
As a user I want to be able to see the status of individual sessions so that I can identify whether that session is shutdown/released etc.

Acceptance Criteria Hide completed items Delete...

100%

✓ ~~Can bulk add sessions using csv~~

Figure 15: User Story 5



User Story 6: Confirm/unconfirm session

in list Planner related Stories

Description

Edit

Story

As an admin I want to be able to confirm and unconfirm individual sessions so that they can be easily identified.

Acceptance Criteria Hide completed items Delete...

100%

✓ ~~Can confirm a session~~

✓ ~~Can unconfirm a session~~

Figure 16: User Story 6

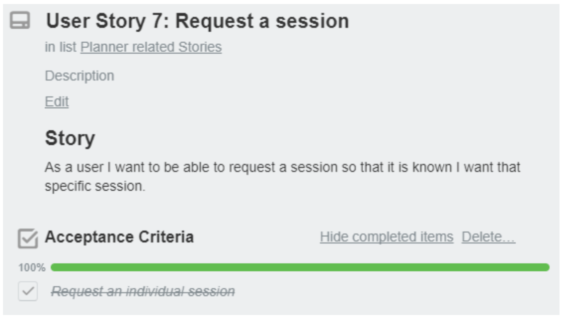

User Story 7: Request a session
in list Planner related Stories
Description
Edit
Story
As a user I want to be able to request a session so that it is known I want that specific session.
Acceptance Criteria Hide completed items Delete...
100%
~~Request an individual session~~

Figure 17: User Story 7

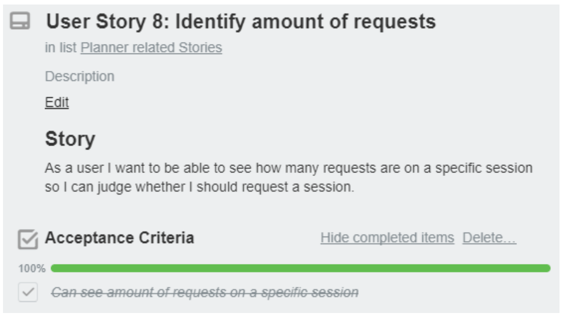

User Story 8: Identify amount of requests
in list Planner related Stories
Description
Edit
Story
As a user I want to be able to see how many requests are on a specific session so I can judge whether I should request a session.
Acceptance Criteria Hide completed items Delete...
100%
~~Can see amount of requests on a specific session~~

Figure 18: User Story 8

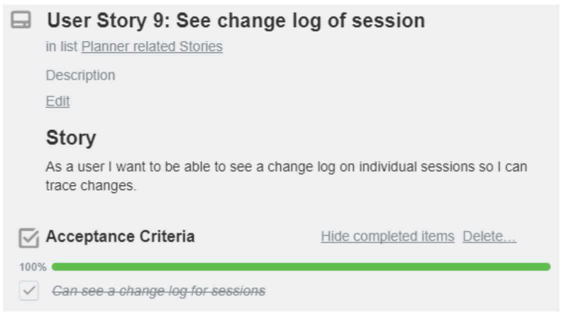

User Story 9: See change log of session
in list Planner related Stories
Description
Edit
Story
As a user I want to be able to see a change log on individual sessions so I can trace changes.
Acceptance Criteria Hide completed items Delete...
100%
~~Can see a change log for sessions~~

Figure 19: User Story 9

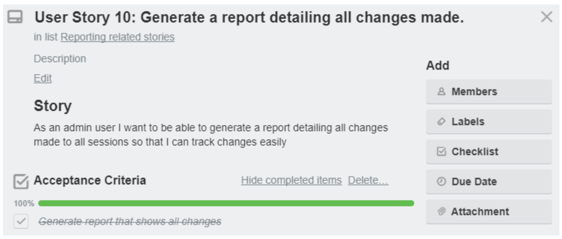

User Story 10: Generate a report detailing all changes made.
in list Reporting related stories
Description
Edit
Story
As an admin user I want to be able to generate a report detailing all changes made to all sessions so that I can track changes easily
Acceptance Criteria Hide completed items Delete...
100%
~~Generate report that shows all changes~~

Figure 20: User Story 10

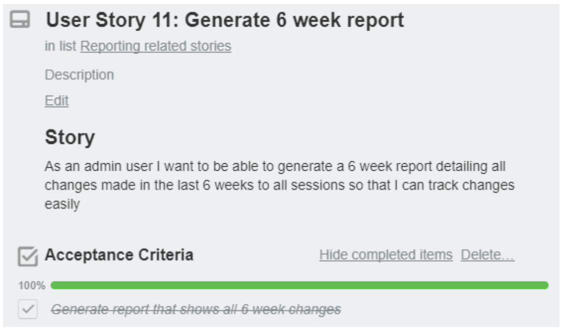

User Story 11: Generate 6 week report
in list Reporting related stories
Description
Edit
Story
As an admin user I want to be able to generate a 6 week report detailing all changes made in the last 6 weeks to all sessions so that I can track changes easily
Acceptance Criteria Hide completed items Delete...
100%
~~Generate report that shows all 6 week changes~~

Figure 21: User Story 11

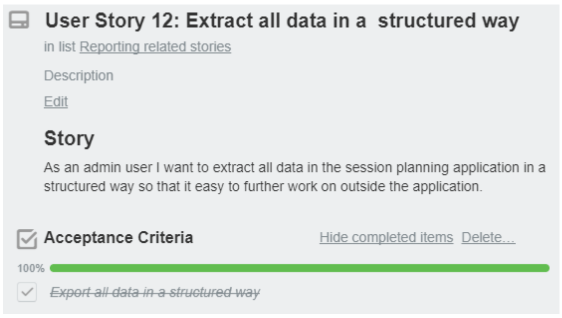

User Story 12: Extract all data in a structured way
in list Reporting related stories
Description
Edit
Story
As an admin user I want to extract all data in the session planning application in a structured way so that it easy to further work on outside the application.
Acceptance Criteria Hide completed items Delete...
100%
~~Export all data in a structured way~~

Figure 22: User Story 12

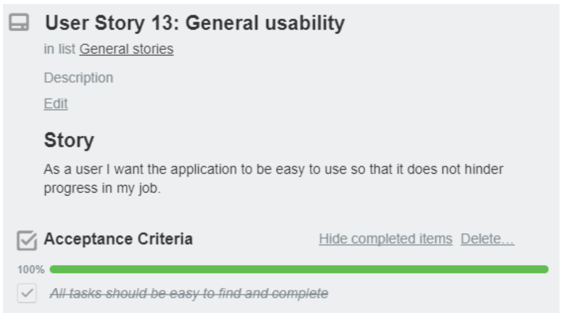

User Story 13: General usability
in list General stories
Description
Edit
Story
As a user I want the application to be easy to use so that it does not hinder progress in my job.
Acceptance Criteria Hide completed items Delete...
100%
~~All tasks should be easy to find and complete~~

Figure 23: User Story 13

4.2 Technologies

4.2.1 Front-End

A large aspect of this project involves the clear and accurate display of information to the screen. To facilitate this a suitable front-end JavaScript framework needs to be chosen. The reasons for using an existing framework as an alternative to vanilla JavaScript are numerous:

There is a common boastful saying along the lines of I can write your 100 lines of code in your preferred language in a few lines using my preferred language. Using a front-end framework this is often a truth. This leads to more efficiency in the writing of code as well as increased readability of code purely through the more concise nature. The communities involved are very large. They have a lot of people spending a lot of time in the development and usage of these frameworks. It makes sense to leverage this knowledge and to bypass many of the pain points others have experienced. Information and interaction with experts is readily available through platforms such as Stack Overflow. Three frameworks have been chosen for comparison in this project to ensure the best tool for the job is chosen. Angular, React and Vue are chosen because of their reputations as being on the leading edge. The factors to take in to consideration for this comparison are their community size, their likelihood of longevity, learning curve involved, performance expectations and completeness of development life cycle.

The Frameworks

Angular is a TypeScript based JavaScript framework maintained by Google. It is advertised as a framework that can be used to develop across all platforms by reusing code across all deployment targets such as web and mobile. Speed and performance is emphasised using Web Workers and server-side rendering with tooling available through a wide range of IDEs with the ability to extend existing simple declarative templates (Google, 2018). Components in Angular are used using a template structure separating view from logic.

React is A JavaScript library for building user interfaces (Facebook, 2018). React was developed and is maintained by Facebook. It has many high profile users such as Airbnb, Uber, Reddit, PayPal and many more (Facebook, 2017). It employs declarative views to make code easy to read and debug as well as a component-based system. Component logic is written in JavaScript using React instead of templates, this enables data to be passed through the application whilst keeping state out of the DOM. Vue is different to the two previous frameworks as in it was originally developed by a single person, Evan You who is a ex-Google employee (You, 2018). Since release it has been adopted by many large users such as Alibaba, Baidu, Expedia and Nintendo (Nifty Software E.U, 2018). It is advertised as being approachable, versatile, performant, maintainable and testable (You, 2014). The core library of Vue is focused purely on the View layer much like React. Again much like the above frameworks it uses declarative views and is component based. All three are covered by the MIT License.

Community Size

Community size and number of contributors is an important metric to consider when determining which framework to use. This determines the ease in which to get help as well as the amount of resources available to the developer. A more active community is generally going to be more beneficial than a less active one. Values that are going to be taken in to account are: Stack Overflow question numbers, Stack Overflow being the largest and most trusted community for developers (Stack Overflow, 2016). As well as GitHub contributor numbers which will indicate the amount of developers actively working on the platform. Angulars core framework repository has had 574 contributors for a total of 9,437 commits on GitHub. In total 93,929 questions have been tagged in Stack Overflow with Angular. Reacts core repository has had 1,163 contributors for a total of 9,615 commits on GitHub. In total 71,652 questions have been tagged with React in Stack Overflow. Vue has had 165 contributors for a total of 2,488 commits on GitHub. In total Vue has had 13,208 questions tagged in Stack Overflow.

Both Angular and React have very similar numbers of commits in GitHub whilst React has over double the number of contributors. In terms of questions asked on Stack Overflow Angular has a fairly large lead over React with Vue being much lower in all metrics.

Longevity and stability

It is important for ongoing support that the framework chosen doesn't disappear in the near future. The investment involved particularly in development time is a large one which shouldn't need to be repeated for a substantial amount of time. As of 2017 the Angular team has announced long-term-supporting version starting with Angular 4. One major update will occur every six months and these will be supported for at least the following year with bug fixes and important patches. React APIs are very stable and are used by over 20,000 components at Facebook alone, as such they are very reluctant to change any public APIs or behaviour. Vue has a planned release in March 2018 and has recently hired many new developers as part of ongoing expansion efforts (Ramsey, 2018). This indicates that a road map for ongoing support is intended yet there is no formal confirmation. The sheer volume of large users for React make it unlikely to go away any time soon, and the Angular team have vocally committed to ongoing releases (Minar, 2016). Vue seems to have intentions of ongoing support but nothing in concrete. It should also be noted in the case of React and Vue which are predominantly view only frameworks, that you are often reliant on many other libraries in which also need upkeep.

Learning curve

The learning curve involved in adopting any of these frameworks is an aspect that needs to be taken in to consideration. As well as weighing up the benefits to learning new techniques or technologies on top of learning already involved through the new framework.

TypeScript vs ES6 vs ES5

React uses JavaScript ES6, Vue uses ES5 or ES6 whilst Angular uses TypeScript. TypeScript is a super set of JavaScript and essentially a new language. It offers all features from JavaScript ES6 as well as static types and decorators which are useful for code intelligence tools, debugging and refactoring. This is particularly useful for large projects in which many different people work on. ES6 or ECMAScript 2015 is the latest version of JavaScript and includes many new features such as function shorthand arrows and classes. TypeScript although a new language is very recognisable to developers in languages such as Java as well as more traditional JavaScript developers.

Templates

Angular and Vue both use a very traditional approach of separating user interface template HTML from JavaScript whereas React uses JSX which mixes the JavaScript and HTML together. Angular templates are HTML with special angular keywords which enable you to inject JavaScript. This means you can still have a specialised design team without the need for them to learn JavaScript. Reacts templating style can best be described as injecting HTML into JavaScript. This is a big change from traditional approaches. Vue uses a single file components approach (Vue.js, 2018) in which the template scripts and JavaScript are all in the same file.

Performance

In a variety of metrics Angular 4, React and Vue are relatively comparable in which there will be no noticeable difference in performance (Krause, 2017).

Completeness of development life-cycle

Although referred to as frameworks React and Vue are much closer to libraries. They only cover the user interface component side of development which means you have a lot more decisions to make on the architecture of your application by integrating further libraries. Angular on the other hand is a fully functioning framework encompassing the entire life-cycle of the application. It has strict rules and guidelines with no need for further libraries. This comes with a higher learning curve for the core framework but in turn removes the need to research further libraries and the added complexity that comes with integrating additional libraries.

Decision

After comparing all the frameworks, the decision was made to develop using Angular. This decision was made for the following reasons:

- The use of TypeScript. This increases productivity and test-ability by being type safe.
- A traditional separation of HTML view from logic gives a more familiar feel.
- The community is very large and active with guaranteed ongoing support backed by large companies.
- Strong performance emphasis.
- Completeness of the framework. Only having one framework to worry about which is well documented reduces complexity which would have been introduced by adding libraries.

4.2.2 Back-End

The back-end of this application is split in to two main parts: the API and the database. These two parts work directly with each other to deliver data to the front-end which is then shown to the screen. The only requirements of them are:

- The data is returned to the application in a timely manner so not to deliver a bad user experience.
- Large amounts of requests can be made without disrupting the connection between front and back-ends.
- Data is easily structured in a way that reports can be generated easily.

REST API

A REST API is ”An API which defines a set of functions which developers can perform requests and receive responses via HTTP protocols such as GET and POST” (Deering, 2012). One of the conditions of being a RESTful system is that the client (front-end) and server (back-end) are independent of each other (Erl, Carlyle, Pautasso, & Balasubramanian, 2012), meaning that both the client or server can be replaced without having to change any underlying functionality. This condition was particularly important in regards to this project as the back-end had to be completely rebuilt late in to development.

REST API Version 1

The original REST API chosen for this project was a product by Cdata (CData Software, Inc., 2018). This product allowed an API to be built with just a few clicks which saved a lot of time that could be used for development in other aspects of the application.

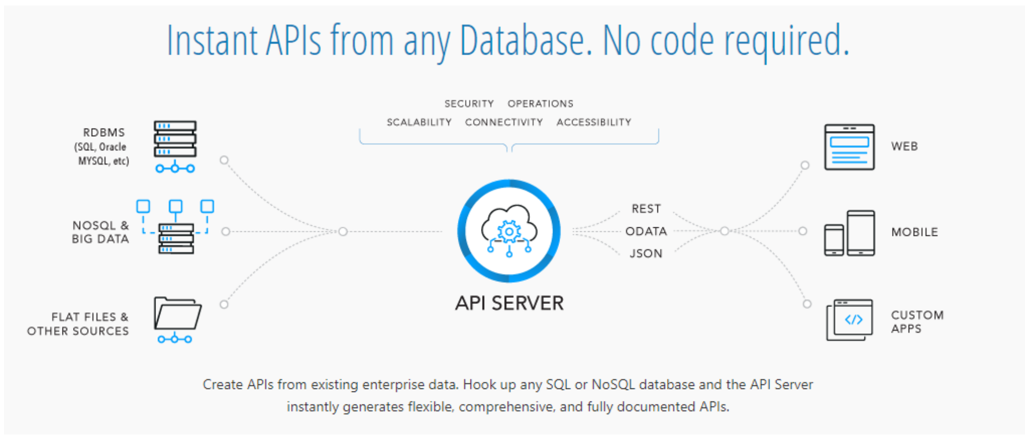
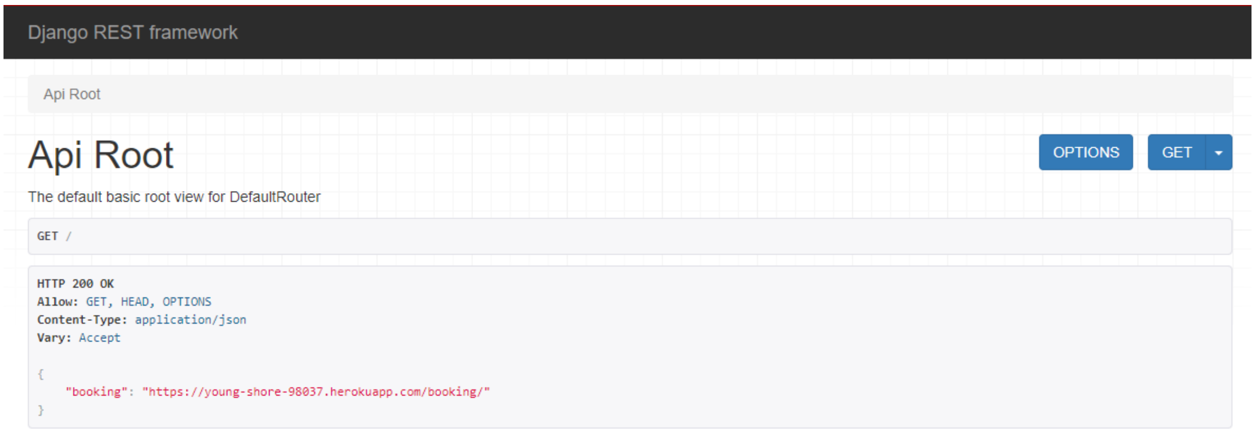

Instant APIs from any Database. No code required.
SECURITY
OPERATIONS
SCALABILITY
CONNECTIVITY
ACCESSIBILITY
RDBMS (SQL, Oracle MYSQL, etc)
NOSQL & BIG DATA
FLAT FILES & OTHER SOURCES
API SERVER
REST
ODATA
JSON
WEB
MOBILE
CUSTOM APPS
Create APIs from existing enterprise data. Hook up any SQL or NoSQL database and the API Server instantly generates flexible, comprehensive, and fully documented APIs.

Figure 24: Potential architectures using Cdata API Server. Retrieved from (CData Software, Inc., 2018)

This initial API allowed data to be retrieved from the applications database and for Angular services to be made and then injected in to the application using the documentation provided (CData Software, Inc., 2017). Unfortunately there was an undocumented limit to the amount of requests that could be made using the basic version of the product. This meant it was no longer suitable for the application as requests would be denied frequently providing a poor experience to the user.

REST API Version 2

Fortunately a suitable alternative existed in the form of django REST framework (Christie, 2018). Using this framework was considerably more work to set up but it did allow for more flexibility as well as a web browsable feature which is helpful for debugging.



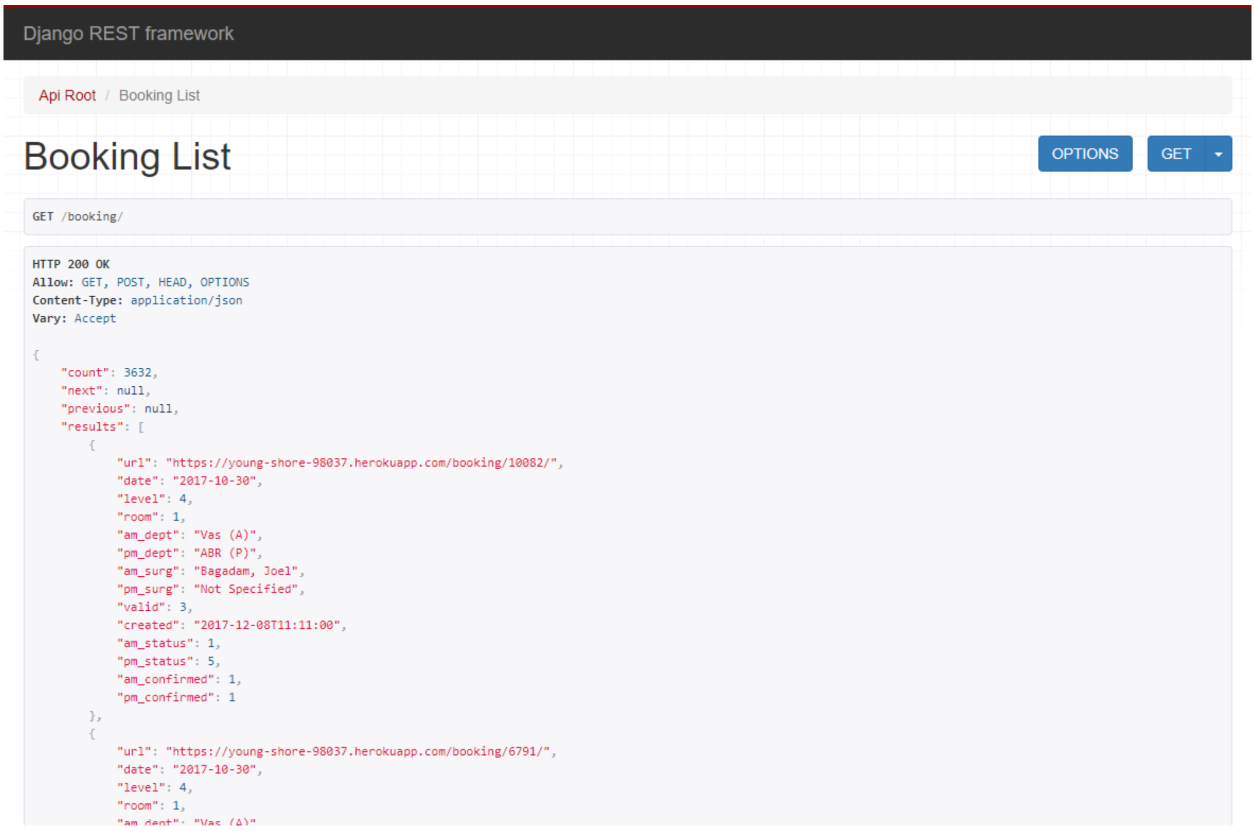
Api Root

The default basic root view for DefaultRouter

```
GET /
```

```
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "booking": "https://young-shore-98037.herokuapp.com/booking/"
}
```

Figure 25: The root API showing available calls.

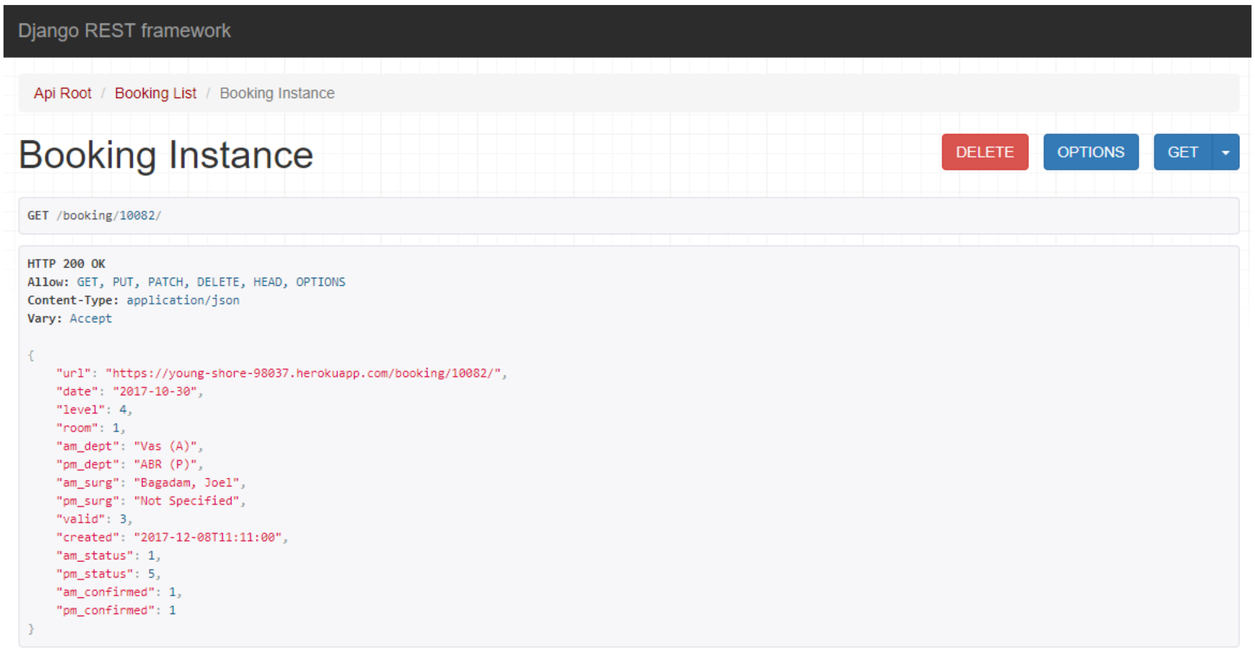


Booking List

```
GET /booking/
```

```
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "count": 3632,
    "next": null,
    "previous": null,
    "results": [
        {
            "url": "https://young-shore-98037.herokuapp.com/booking/10082/",
            "date": "2017-10-30",
            "level": 4,
            "room": 1,
            "am_dept": "Vas (A)",
            "pm_dept": "ABR (P)",
            "am_surg": "Bagadam, Joel",
            "pm_surg": "Not Specified",
            "valid": 3,
            "created": "2017-12-08T11:11:00",
            "am_status": 1,
            "pm_status": 5,
            "am_confirmed": 1,
            "pm_confirmed": 1
        },
        {
            "url": "https://young-shore-98037.herokuapp.com/booking/6791/",
            "date": "2017-10-30",
            "level": 4,
            "room": 1,
```

Figure 26: Showing GET call of all bookings.



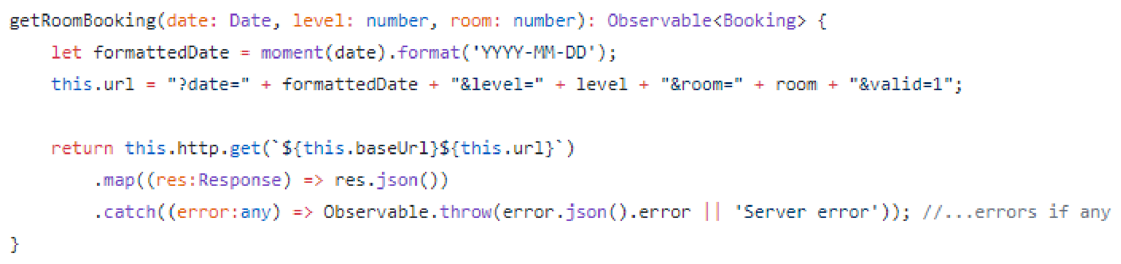
```
GET /booking/10082/
```

```
HTTP 200 OK
Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "url": "https://young-shore-98037.herokuapp.com/booking/10082/",
    "date": "2017-10-30",
    "level": 4,
    "room": 1,
    "am_dept": "Vas (A)",
    "pm_dept": "ABR (P)",
    "am_surg": "Bagadam, Joel",
    "pm_surg": "Not Specified",
    "valid": 3,
    "created": "2017-12-08T11:11:00",
    "am_status": 1,
    "pm_status": 5,
    "am_confirmed": 1,
    "pm_confirmed": 1
}
```

Figure 27: Showing GET call of an individual booking.

Using the available documentation it was simple to mimic the API calls that were created in version 1 of the API and simply changing the end-point in the Angular service confirming the applications truly separate client-server relationship.

The nature of the django REST framework makes it easy to further filter requests as needed such as the following method:

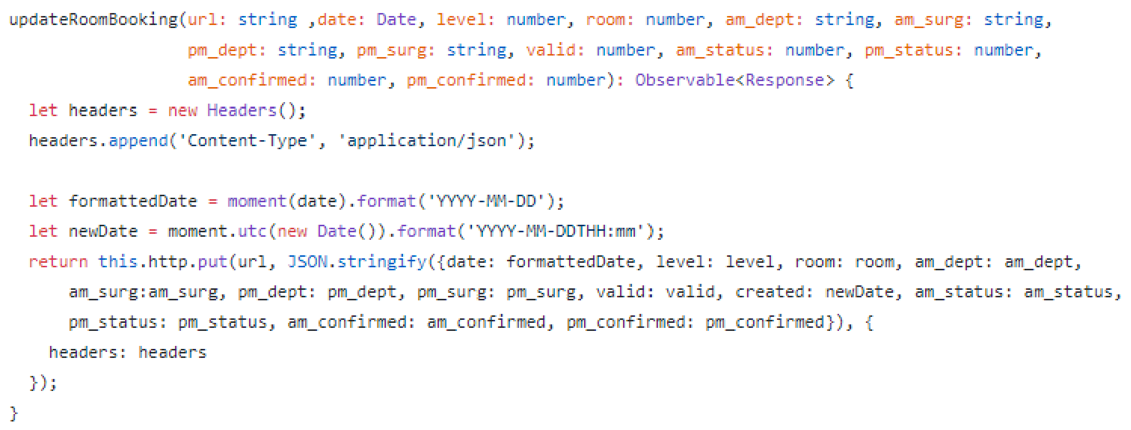


```
getRoomBooking(date: Date, level: number, room: number): Observable<Booking> {
    let formattedDate = moment(date).format('YYYY-MM-DD');
    this.url = "?date=" + formattedDate + "&level=" + level + "&room=" + room + "&valid=1";

    return this.http.get(`${this.baseUrl}${this.url}`)
        .map((res:Response) => res.json())
        .catch((error:any) => Observable.throw(error.json().error || 'Server error')); //...errors if any
}
```

Figure 28: Method for the request of specific booking.

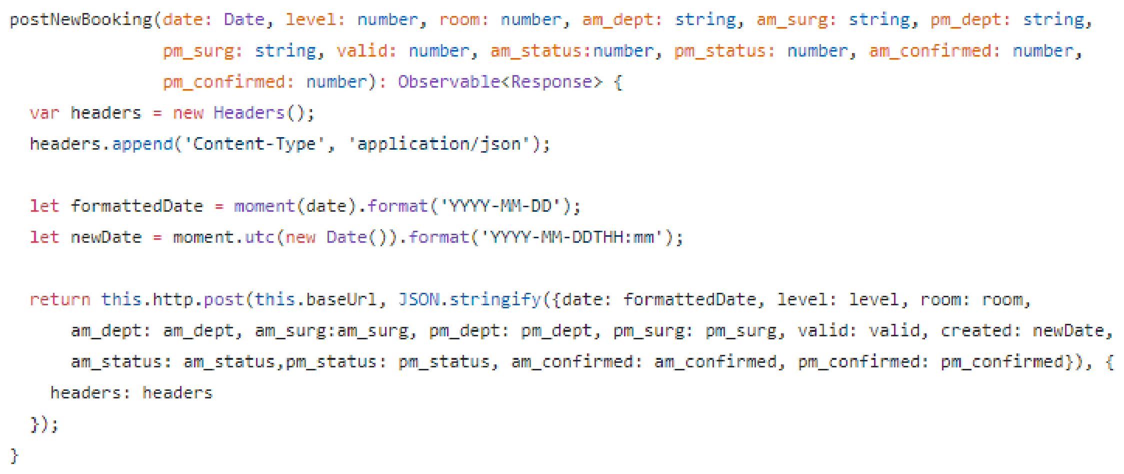
The above code snippet shows a request for a specific room's bookings on a specific date. The given parameters are appended on to the default endpoint URL and that specific booking object is then returned for further manipulation or display. This type of call is readily available with added parameters or even less. For example if all bookings on a certain date where requested you would just omit the level and room parameters.

In addition to the various GET methods there is also PUT and POST methods. PUT to update bookings and POST to insert new bookings as per HTTP Protocol standards (Belshe, Thomson, & Peon, 2015).



```
updateRoomBooking(url: string ,date: Date, level: number, room: number, am_dept: string, am_surg: string,
                  pm_dept: string, pm_surg: string, valid: number, am_status: number, pm_status: number,
                  am_confirmed: number, pm_confirmed: number): Observable<Response> {
  let headers = new Headers();
  headers.append('Content-Type', 'application/json');

  let formattedDate = moment(date).format('YYYY-MM-DD');
  let newDate = moment.utc(new Date()).format('YYYY-MM-DDTHH:mm');
  return this.http.put(url, JSON.stringify({date: formattedDate, level: level, room: room, am_dept: am_dept,
      am_surg:am_surg, pm_dept: pm_dept, pm_surg: pm_surg, valid: valid, created: newDate, am_status: am_status,
      pm_status: pm_status, am_confirmed: am_confirmed, pm_confirmed: pm_confirmed}), {
    headers: headers
  });
}
```

Figure 29: Method for updating a specific booking.



```
postNewBooking(date: Date, level: number, room: number, am_dept: string, am_surg: string, pm_dept: string,
               pm_surg: string, valid: number, am_status:number, pm_status: number, am_confirmed: number,
               pm_confirmed: number): Observable<Response> {
  var headers = new Headers();
  headers.append('Content-Type', 'application/json');

  let formattedDate = moment(date).format('YYYY-MM-DD');
  let newDate = moment.utc(new Date()).format('YYYY-MM-DDTHH:mm');

  return this.http.post(this.baseUrl, JSON.stringify({date: formattedDate, level: level, room: room,
      am_dept: am_dept, am_surg:am_surg, pm_dept: pm_dept, pm_surg: pm_surg, valid: valid, created: newDate,
      am_status: am_status,pm_status: pm_status, am_confirmed: am_confirmed, pm_confirmed: pm_confirmed}), {
    headers: headers
  });
}
```

Figure 30: Method for creating a new booking

Database

Both of the above APIs both were not reliant on the type of database used. An SQL based database was chosen due to the familiarity and ease of use. All data is stored on one simple table which makes it very easy to initially upload as well as copy. PostgreSQL was chosen as the database type due to the ease of integrating with Heroku which will be discussed more in 4.2.3.

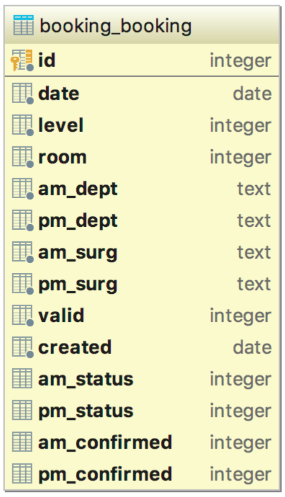


Figure 31: SQL Booking table diagram

The above diagram shows the fields present in the booking table which the REST API pulls from.

`date`: The date the booking is on.
`level`: The level the booking is on.
`room`: The room the booking is on.
`am_dept`: The department that has the booking in the AM session.
`pm_dept`: The department that has the booking in the PM session.
`am_surg`: The surgeon that has the booking in the AM session.
`pm_surg`: The surgeon that has the booking in the PM session.
`valid`:

- A `1` value in this field shows a valid booking.
- A `0` indicates another row has superseded this one.

`created`: The the date this row was created.
`am_status`:

- A `0` value in this field indicates the AM booking has not been changed from the original planned session.
- A `1` indicates that the session has been recycled.
- A `3` indicates the session has been shutdown.
- A `4` indicates the surgeon is on leave.
- A `5` indicates this session has requests against it.

`pm_status`:

- A `0` value in this field indicates the PM booking has not been changed from the original planned session.
- A `1` indicates that the session has been recycled.
- A `3` indicates the session has been shutdown.
- A `4` indicates the surgeon is on leave.
- A `5` indicates this session has requests against it.

`am_confirmed`

- A `1` value in this field shows a confirmed AM session.
- A `0` value in this field shows a unconfirmed AM session.

`pm_confirmed`

- A `1` value in this field shows a confirmed PM session.
- A `0` value in this field shows a unconfirmed PM session.

Using a relatively small table, all the needed data to display as well as to report is included in a very small database size.

4.2.3 Hosting

Cloud based hosting is ideal for this software due to the ease of deployment and the up time being as close to 100% as possible allowing it to be accessed anywhere, anytime as long as the user has an internet connection. Heroku (Salesforce, 2018) was chosen to be the cloud provider. By using Heroku, applications can be run directly from GitHub repositories which cuts deployment time greatly as well as being able to rollback the application to any commit version that exists on the application repository. Heroku also has pre-built PostgreSQL environments making the hosting of existing databases as easy as copying over the existing development database.

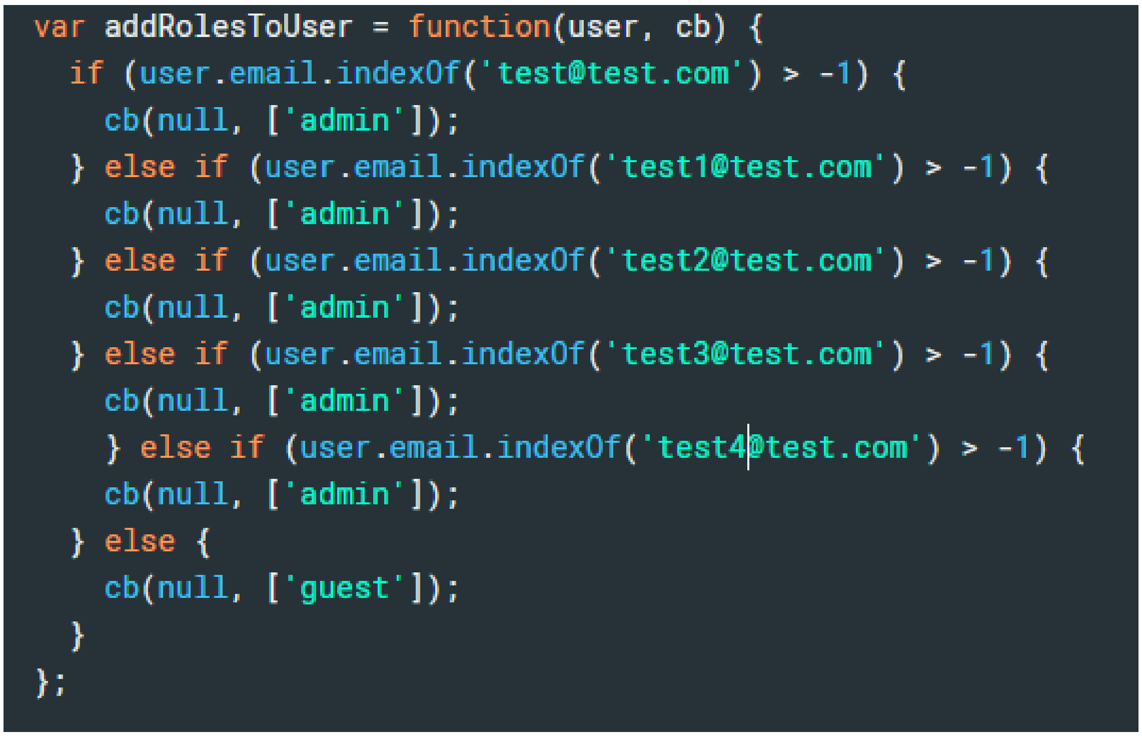
By hosting the Angular application, django-REST API and the PostgreSQL database on the same platform it allows simple building scripts to be run making deployment easy.

4.2.4 Login and Authentication

Although not as a such a security issue as there is no personal information of patients or of any others present in the application, it is still valuable to be able to identify who is using the application and even more importantly who is making changes so that they can be logged.

Schneier's law (Schneier, 2009) states "Anyone can invent an encryption algorithm they themselves can't break; it's much harder to invent one that no one else can break" This is the basis to why it's advised to not create your own encryption and to use a well tested alternate.

auth0 (auth0 inc., 2018) was chosen as the login service as it has an easy to use API. Custom code can be written inside this service in order to categorise users into guests, admin, normal users. As certain features are locked behind being an admin user this is essential.



```
var addRolesToUser = function(user, cb) {
  if (user.email.indexOf('test@test.com') > -1) {
    cb(null, ['admin']);
  } else if (user.email.indexOf('test1@test.com') > -1) {
    cb(null, ['admin']);
  } else if (user.email.indexOf('test2@test.com') > -1) {
    cb(null, ['admin']);
  } else if (user.email.indexOf('test3@test.com') > -1) {
    cb(null, ['admin']);
    } else if (user.email.indexOf('test4@test.com') > -1) {
    cb(null, ['admin']);
  } else {
    cb(null, ['guest']);
  }
};
```

Figure 32: Custom rule used to set role type.

This login was implemented in to a simple Angular service which will request from the API whether a user is authenticated and will receive back a response with either an error or a JSON object which has the users details. This is where their name and role type can be retrieved and further used in the application.

4.3 System Architecture

Pulling all of the above in to a simple high level system architecture diagram the clear separation of pieces can be seen.

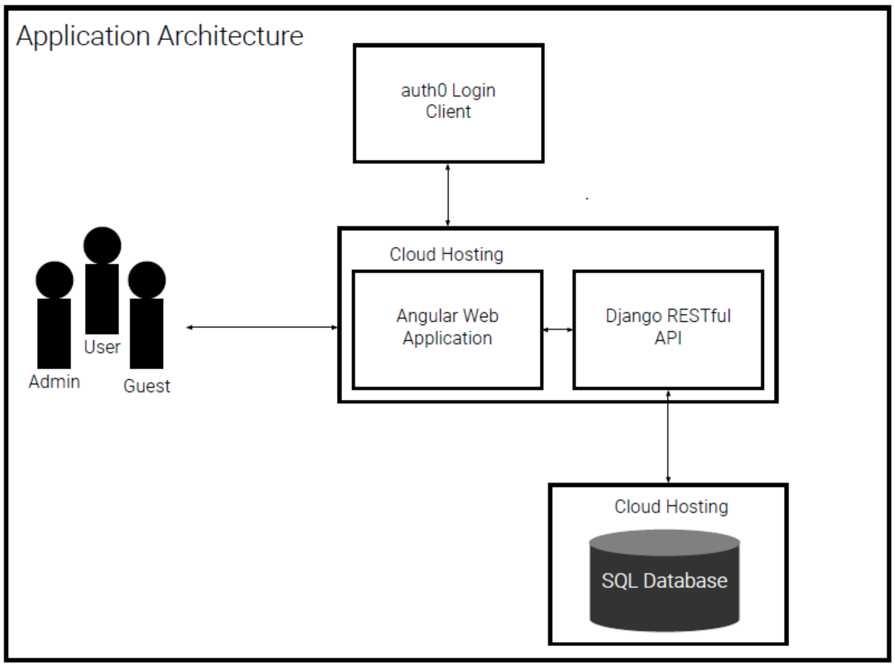


Figure 33: High level overview of system architecture.

Chapter 5. System Development

5.1 Planner Component

The planner component is the overarching container consisting of the calendar, booking and header components.

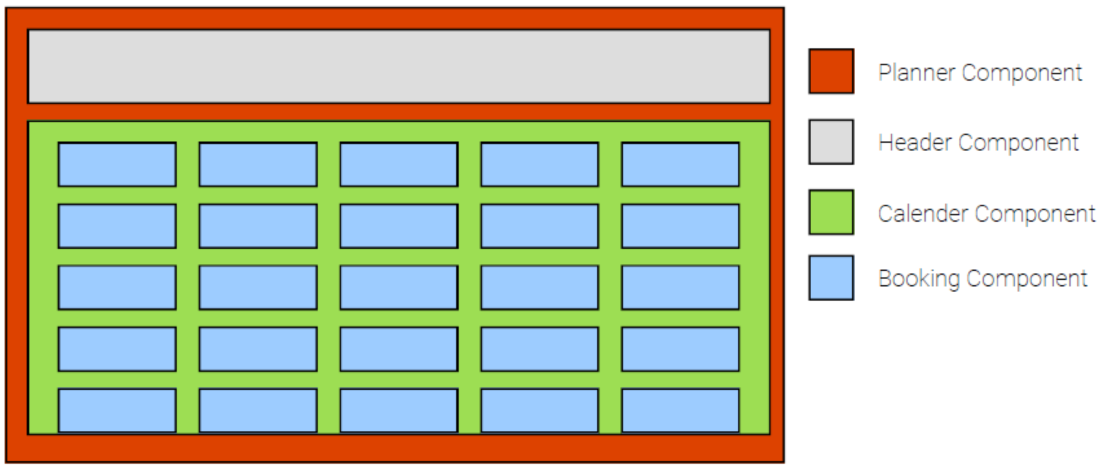


Figure 34: Planner component container showing sub components.

5.1.1 Planner-Header Component

The header component contains all the controls for the change of date and floor level that the calendar component is displaying.

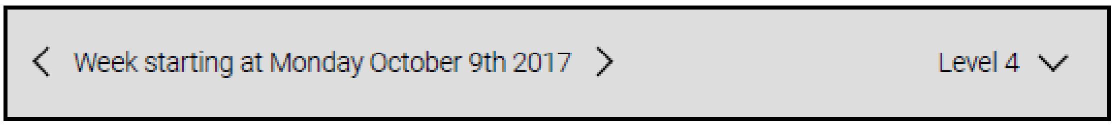


Figure 35: Header component controls.

When initialising this component both the level service and the date service are injected. This allows the component to call service methods as well as listen to changes by subscribing to the observable these services provide.

```
@Injectable()
export class DateService {
  date: Date;
  private dateChange = new Subject<any>();
  dateChange$ = this.dateChange.asObservable();

  setDate(date: Date) {
    this.date = date;
    this.dateChange.next(this.date);
  }
}
```

Figure 36: Date Service Initialization.

```
@Injectable()
export class LevelService {
  level: number;
  private levelChange = new Subject<any>();
  levelChange$ = this.levelChange.asObservable();

  setLevel(level: number) {
    this.level = level;
    this.levelChange.next(this.level);
  }
}
```

Figure 37: Level Service Initialisation.

By looking at Figure 36 and Figure 37 it can be seen that an Observable object is created. This Observable is updated by using manipulation methods from within the service, `setLevel()` for example in Figure 37.

This process can be seen in use by analysing the planner-header component in detail:

```
export class HeaderComponent implements OnInit {
  today: string;
  todayDate: Date;
  level: number;

  constructor( private dateService: DateService,
    private levelService: LevelService) {
  }
}
```

Figure 38: Planner-Header Component Initialisation.

The above Figure 35 shows the `LevelService` and `DateService` being injected in to the HeaderComponent allowing these services to be used and called upon using this.dateService and this.levelService.

Using Figure 35 as reference the left and right arrows indicate going back a week in the past and going forward a week in the future. Below the HTML template for these arrows can be seen followed by the methods that are called to manipulate the dateService.

```
<div class="row">
  <div class="col col-md-4">
    <a (click)="backWeek()"> << BACK </a>
  </div>
  <div class="col col-md-4">
    <h6 >Week starting {{today}}</h6>
  </div>
  <div class="col col-md-4">
    <a (click)="forwardWeek()"> FORWARD >> </a>
  </div>
</div>
```

Figure 39: HTML template for date changer

```
private backWeek() {
  this.todayDate.setDate(this.todayDate.getDate() - 7);
  this.today = moment(this.todayDate).startOf('isoweek')
    .format('dddd MMMM Do YYYY');
  this.dateService.setDate(this.todayDate);
}
```

Figure 40: Method for setting date back one week

```
private forwardWeek() {
  this.todayDate.setDate(this.todayDate.getDate() + 7);
  this.today = moment(this.todayDate).startOf('isoweek')
    .format('dddd MMMM Do YYYY');
  this.dateService.setDate(this.todayDate);
}
```

Figure 41: Method for setting date forward one week

By using the Angular event binding (click), component logic can be called. In the above example it can be seen that the event binding has been added on to the anchor (<a>) tags. For the first example: <a (click)="backWeek()"> << BACK </a> a click on the tag calls the backWeek() method inside the component. This in turn will set the member variable this.todayDate to be one week prior, following this the dateService.setDate method is called which manipulates the dateChange Observable sending a change to all components subscribed to it.

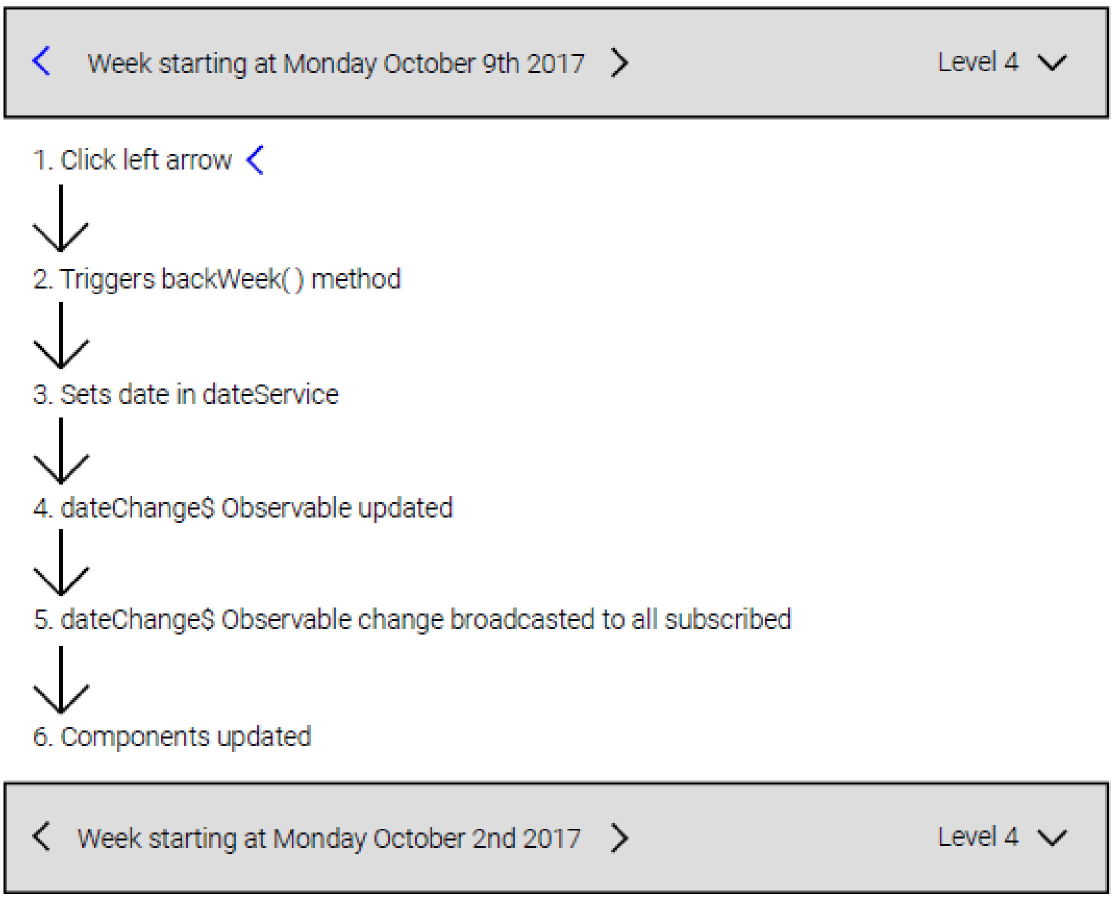

Week starting at Monday October 9th 2017
Level 4
1. Click left arrow
2. Triggers backWeek( ) method
3. Sets date in dateService
4. dateChange$ Observable updated
5. dateChange$ Observable change broadcasted to all subscribed
6. Components updated
Week starting at Monday October 2nd 2017
Level 4

Figure 42: Flow of actions following click on left arrow

The `levelService` works in a very similar way to `dateService`. By clicking on the level drop down list component logic is called which will use the `levelService` `setLevel` method to manipulate the `levelChange` Observable which will then be recognised throughout the application.

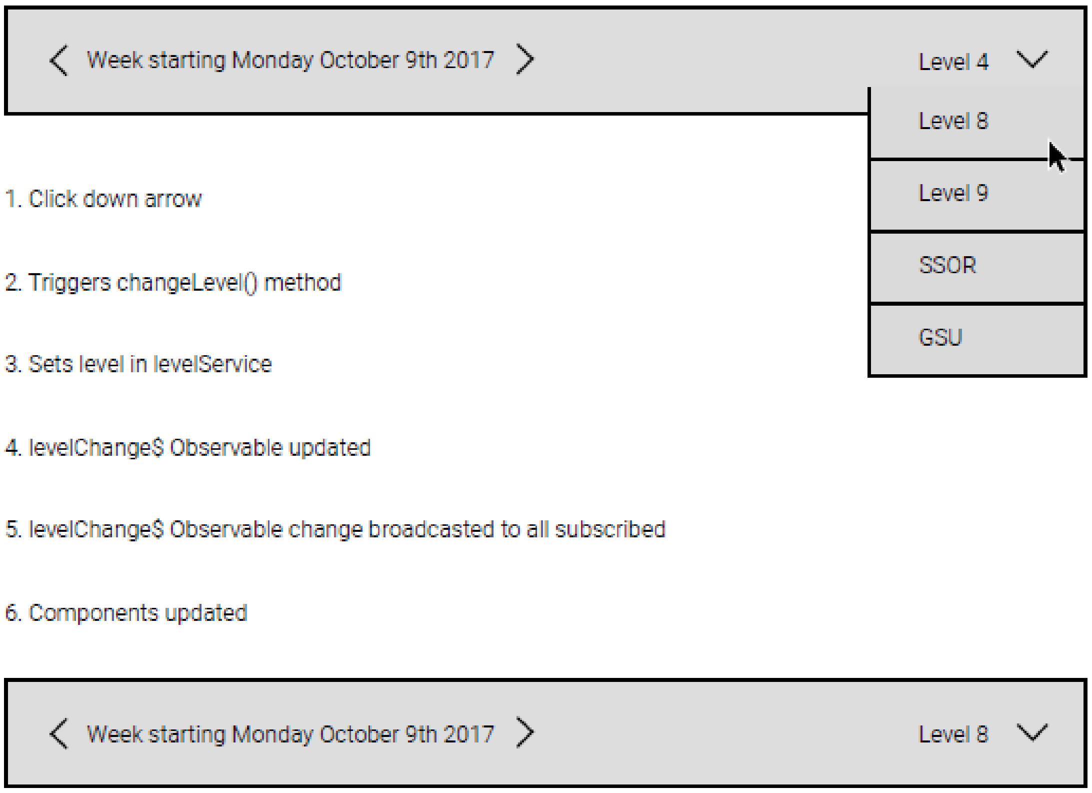

Week starting Monday October 9th 2017
Level 4
Level 8
Level 9
SSOR
GSU
1. Click down arrow
2. Triggers changeLevel() method
3. Sets level in levelService
4. levelChange$ Observable updated
5. levelChange$ Observable change broadcasted to all subscribed
6. Components updated
Week starting Monday October 9th 2017
Level 8

Figure 43: Flow of actions following click on down arrow

5.1.2 Calendar Component

The consequence of the above service updates in the header component can best be seen in the calendar component. The calendar component does not manipulate any state to do with level or date it just subscribes to the respective services and then displays the correct bookings to the page.

```
constructor(
  private auth: AuthService,
  private bookingService: BookingService,
  private dateService: DateService,
  private levelService: LevelService,
  private permissionService: PermissionService
  ) {

  if(!auth.isAuthenticated()) {
    this.auth.login();
  }
}

ngOnInit() {
  if(this.auth.isAuthenticated()) {
    this.getLevel();
    this.getDates();
    this.populate();

    if(!localStorage.getItem("permission")) {
      this.permissionService.setPermission();
    }

    let _subscription = this.dateService.dateChange$
      .subscribe((value) => {
      this.getDates();
      this.populate();
    });

    let _subscription = this.levelService.levelChange$
      .subscribe((value) => {
      this.getLevel();
      this.dateService.setDate(new Date());
      this.populate();
    });
  }
}
```

Figure 44: Construction of the Calendar Component

Using Angular life cycle hooks all service dependencies are injected through the constructor. `ngOnInit` is an Angular life cycle hook to indicate that the component is done being created, it is during this hook that there are multiple checks to determine if and what should be rendered to the page.
Firstly there is a check to `authService` to ensure there is a logged in user, there is no scenario where a non-authorised person should have access to this page. If the current

user is not logged in they are redirected to the login page opposed to seeing a rendered calendar.

If the user is logged in the populating of the component commences:

```
private getLevel() {
  this.level = this.levelService.getLevel();
}
```

Figure 45: levelService getLevel( ) method

The `levelService getLevel()` method is called to set the local `this.level` variable for future use in populating the bookings.

```
private getDates() {
  this.mondayDate = this.dateService.getMonday();
  this.tuesdayDate = this.dateService.getTuesday();
  this.wednesdayDate = this.dateService.getWednesday();
  this.thursdayDate = this.dateService.getThursday();
  this.fridayDate = this.dateService.getFriday();
}
```

Figure 46: dateService getDates( ) method

The `getDates()` method is called on the `dateService` in order to identify the dates needed to populate the current calender screen. By saving these in to a local variable they are reused during calls to the `bookingService`. The logic involved on working out which day each date relates to is left to the `dateService` to calculate.

```
public populate() {
  this.bookingService.getLevelBooking(
    this.mondayDate, this.level).subscribe(
    data => this.mondayBookings =
      (data as any).results,
    err => {
      // Log errors if any
      console.log(err);
  });
}
```

Figure 47: populate ( ) method

Using the previously set local variables the `populate()` method populates the arrays of Bookings for each day of the week. The `bookingService.getLevelBooking()` method is called passing the date and level parameters of the bookings that are to be displayed.

```
getLevelBooking(date:Date, level:number):
  Observable<Booking[]> {

  let formattedDate = moment(date).format('YYYY-MM-DD');
  this.url = "?date=" + formattedDate + "&level=" + level
  + "&valid=1";

  return this.http.get(`$this.baseUrl$this.url`)
    .map((res:Response) => res.json())
    .catch((error:any) =>
      Observable.throw(error.json().error || 'Server
      error')); //...errors if any
}
```

Figure 48: bookingService.getLevelBooking( ) method

Shown in Figure 48, by passing in a specific date and level all the applicable Booking objects are returned as an `Observable<Booking[]>` type allowing the method to be subscribed to.

A Booking object is closely coupled with the `booking_booking` table shown in Figure 31.

```
export class Booking {
  constructor(
  public url: string,
  public date: Date,
  public level: number,
  public room: number,
  public am_dept: string,
  public am_surg: string,
  public pm_dept: string,
  public pm_surg: string,
  public valid: number,
  public am_status: number,
  public pm_status: number,
  public am_confirmed: number,
  public pm_confirmed: number,
  public created: Date,
  ) { }
}
```

Figure 49: The Booking Object

The final check is against the `permissionService`. This service provides checks on what permissions the user has, whether it be admin, guest or user. In order to keep this available on page refreshes as well as application wide this is saved as a localStorage variable on the client. This variable is set using information that the `authService` returns.

```
public setPermission() {

  if(this.permission === undefined) {
    if (this.auth.isAuthenticated()) {
      if (this.auth.userProfile) {
        this.profile = this.auth.userProfile;
        this.permission = (this.profile ?
this.profile['http://test.com/roles'] :  '');
      } else {
        this.auth.getProfile((err, profile) => {
          this.profile = profile;
          this.permission = (this.profile ?
this.profile['http://test.com/roles'] :  '');
        });
      }
    }
  }

  if(this.permission == "admin") {
    localStorage.setItem("permission", "1");
  } else if(this.permission == "guest") {
    localStorage.setItem("permission", "2");
  } else {
    localStorage.setItem("permission", "0");
  }
}
```

Figure 50: permissionService setPermission method

After all the checks and variable populations have occurred subscriptions for both the `dateService` and the `levelService` are created in order to allow for changes. When these subscriptions are triggered the above processes of populating the local variables are run again to reflect the changes.

With all of the local variables set it is the Calendar components job to use these to create Booking components to fill in the screen.

5.1.3 Booking Component

The final components to create are the Booking components. These components are created in the Calendar component using the `Booking` object arrays. The Booking components hold a lot of complex logic in regards to the state of the booking and how they should be displayed, as well as an admin panel which gives control over these states.

```
<div class="row">
  <div class="col-md-1">
    {{mondayDate}}
  </div>
  <div class="col-md-11">
    <div *ngFor="let booking of mondayBookings">
      <app-booking am_confirmed="booking.am_confirmed"
        ...........  date="booking.date">
      </app-booking>
    </div>
  </div>
</div>
```

Figure 51: Calendar HTML showing creation of Booking components

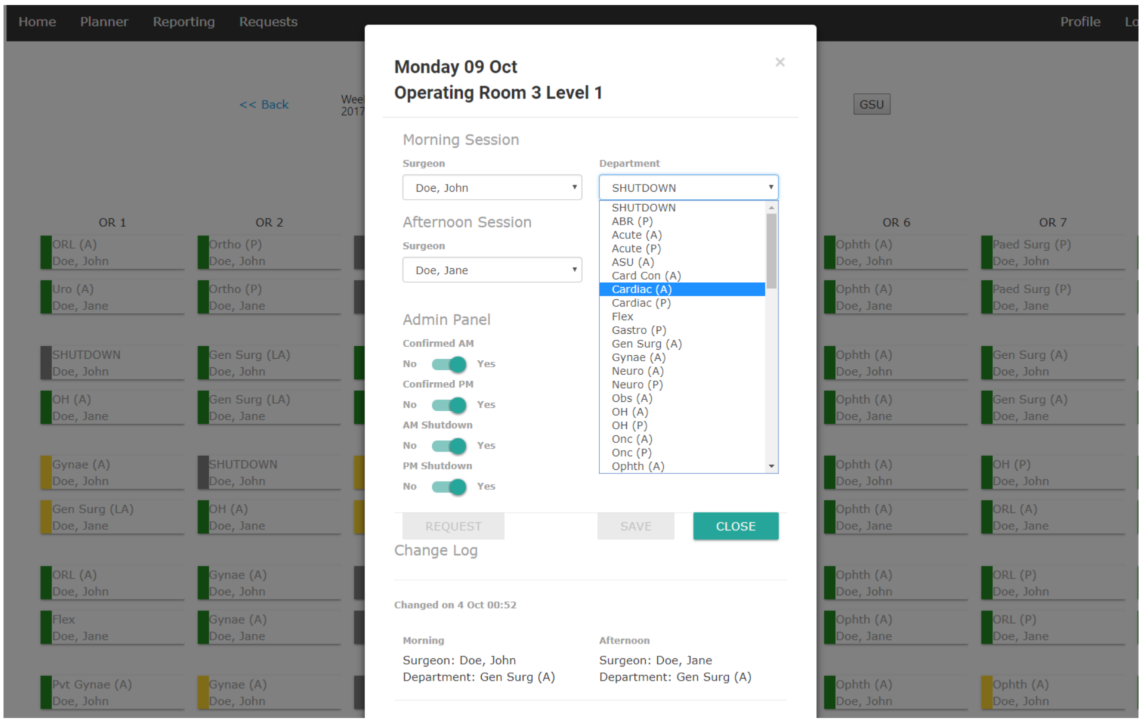
Figure 51 shows the creation of booking components. For every `Booking` object saved in the `mondayBookings` array, a row is created and all the required variables are passed to its constructor.

```
export class BookingComponent implements OnInit {
  @Input() url: string;
  @Input() surgAM: string;
  @Input() deptAM: string;
  @Input() deptPM: string;
  @Input() surgPM: string;
  @Input() level: number;
  @Input() date: Date;
  @Input() room: number;
  @Input() am_status: number;
  @Input() pm_status: number;
  @Input() am_confirmed: number;
  @Input() pm_confirmed: number;
}

  constructor(
    private dataService: DataService,
    private bookingService: BookingService,
    private calendar: CalendarComponent,
    private permissionService: PermissionService) {
  }

  ngOnInit() {
    this.is_am_confirmed = this.am_confirmed == 1;
    this.is_pm_confirmed = this.pm_confirmed == 1;
    this.is_am_shutdown = this.am_status == 3;
    this.is_pm_shutdown = this.pm_status == 3;
    this.is_am_leave = this.am_status == 4;
    this.is_pm_leave = this.pm_status == 4;

    const fb = new FormBuilder();
    this.formModel = fb.group({
      'f_deptAM': [this.deptAM],
      'f_surgAM': [this.surgAM],
      'f_deptPM': [this.deptPM],
      'f_surgPM': [this.surgPM],
      'f_am_confirmed': [this.is_am_confirmed],
      'f_pm_confirmed': [this.is_pm_confirmed],
      'f_am_shut': [this.is_am_shutdown],
      'f_pm_shut': [this.is_pm_shutdown],
      'f_am_leave': [this.is_am_leave],
      'f_pm_leave': [this.is_pm_leave],
    });

    this.getRequestLog();
}
```

Figure 52: Booking Component creation

Figure 52 shows the process involved with the creation of the Booking object. The `@Input` Angular keyword indicates that these fields are expected to be passed in when creating. This process can be seen clearly in Figure 51.

The services are injected through the constructor much like the other components analysed. `dataService` is a new service which is used uniquely in the Booking component. This service is used to populate the drop down menus for preset surgeon and service names in the modal panel relating to each individual booking. This was a requirement added late in development as a way of keeping data entry clean for reporting. `bookingService` is injected as a way to update changes made to the individual bookings, `calenderService` is injected as a way of triggering a refresh when Bookings are updated. Finally `permissionService` is injected so that the correct panels are shown dependent on the permissions the user has.



Monday 09 Oct
Operating Room 3 Level 1

Morning Session
Surgeon: Doe, John
Department: SHUTDOWN

Afternoon Session
Surgeon: Doe, Jane

Admin Panel
Confirmed AM: No / Yes
Confirmed PM: No / Yes
AM Shutdown: No / Yes
PM Shutdown: No / Yes

Change Log

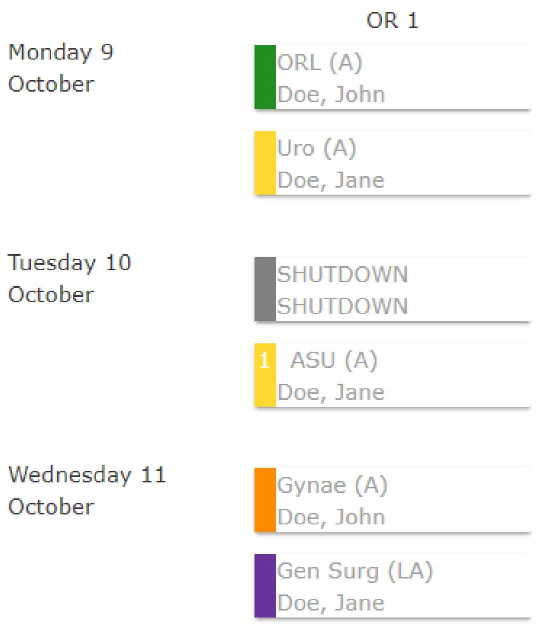
Changed on 4 Oct 00:52

Morning
Surgeon: Doe, John
Department: Gen Surg (A)

Afternoon
Surgeon: Doe, Jane
Department: Gen Surg (A)

Figure 53: The Booking modal panel showing the dataService provided data

The HTML code for Booking Components contains a lot of custom code to show the different states of sessions. The colour combinations were taken directly from the existing spreadsheet which was shown in Figure 1.



	OR 1
Monday 9 October	🟩 ORL (A) Doe, John
	🟨 Uro (A) Doe, Jane
Tuesday 10 October	⬜ SHUTDOWN SHUTDOWN
	🟨 1 ASU (A) Doe, Jane
Wednesday 11 October	🟧 Gynae (A) Doe, John
	🟪 Gen Surg (LA) Doe, Jane

Figure 54: Colour showing the various states of booking

5.2 Reporting Component

A big time and resource sink using the existing manual operating room booking system is the reporting of what has happened in regards to individual sessions. This previously involved the use of two separate systems in order match up 'what should have happened?' to 'what has happened?'. By having a system that records and displays changes in real time the need for manually reporting and matching up of separate systems becomes redundant. Three main reports are of critical importance:

1. Change History Report
2. Actual History Report
3. Six Week History Report

5.2.1 Change History Report

The change history report outputs all changes that have been made on every session. Be this a change in service, surgeon or status. This report can be set for 'all time' or alternatively a date range can be chosen to report between. This is important when reporting on usage and also when auditing that services are abiding by the set in Section 1.1.3.

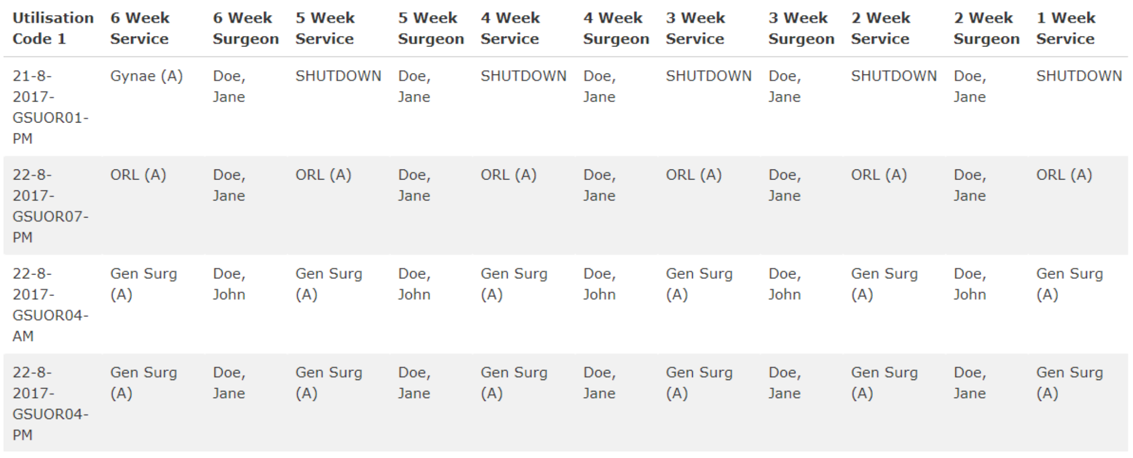
As the database is modelled in a way that it is easy to differentiate between the different statuses of sessions, simple logic can be run from within the reporting components connected to the `sessionService` and by importing a library Ala-SQL (Gershun & Wulff, 2014) this can be generated and exported in standard CSV format with a single button click taking away all the manual comparisons.

5.2.2 Actual History Report

This report is very similar to the change history report but has further filtering in order to determine what the final state of each session was. Using the logic from the change history report it is simple to further filter this by removing all sessions where Valid is set to 0 (Figure 31). Then using Ala-SQL a standard CSV formatted file can be outputted with a button press.

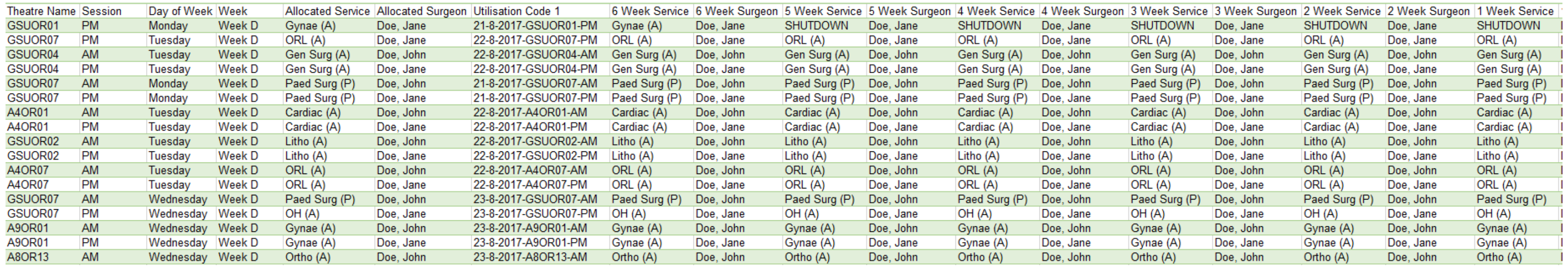
5.2.3 Six Week History Report

The six week history report is a key metric used when auditing session usage. It takes snapshots of what each session looked like in regards to service, surgeon and status at six, five, four, three, two and one week out. This report like all others can be displayed in the application but even more usefully can be exported in the identical formatting needed to analyse and for further integration in to other systems if needed.



Utilisation Code 1	6 Week Service	6 Week Surgeon	5 Week Service	5 Week Surgeon	4 Week Service	4 Week Surgeon	3 Week Service	3 Week Surgeon	2 Week Service	2 Week Surgeon	1 Week Service
21-8-2017-GSUOR01-PM	Gynae (A)	Doe, Jane	SHUTDOWN	Doe, Jane	SHUTDOWN	Doe, Jane	SHUTDOWN	Doe, Jane	SHUTDOWN	Doe, Jane	SHUTDOWN
22-8-2017-GSUOR07-PM	ORL (A)	Doe, Jane	ORL (A)	Doe, Jane	ORL (A)	Doe, Jane	ORL (A)	Doe, Jane	ORL (A)	Doe, Jane	ORL (A)
22-8-2017-GSUOR04-AM	Gen Surg (A)	Doe, John	Gen Surg (A)	Doe, John	Gen Surg (A)	Doe, John	Gen Surg (A)	Doe, John	Gen Surg (A)	Doe, John	Gen Surg (A)
22-8-2017-GSUOR04-PM	Gen Surg (A)	Doe, Jane	Gen Surg (A)	Doe, Jane	Gen Surg (A)	Doe, Jane	Gen Surg (A)	Doe, Jane	Gen Surg (A)	Doe, Jane	Gen Surg (A)

Figure 55: In application display of six week report



Theatre Name	Session	Day of Week	Week	Allocated Service	Allocated Surgeon	Utilisation Code 1	6 Week Service	6 Week Surgeon	5 Week Service	5 Week Surgeon	4 Week Service	4 Week Surgeon	3 Week Service	3 Week Surgeon	2 Week Service	2 Week Surgeon	1 Week Service
GSUOR01	PM	Monday	Week D	Gynae (A)	Doe, Jane	21-8-2017-GSUOR01-PM	Gynae (A)	Doe, Jane	SHUTDOWN	Doe, Jane	SHUTDOWN	Doe, Jane	SHUTDOWN	Doe, Jane	SHUTDOWN	Doe, Jane	SHUTDOWN
GSUOR07	PM	Tuesday	Week D	ORL (A)	Doe, Jane	22-8-2017-GSUOR07-PM	ORL (A)	Doe, Jane	ORL (A)	Doe, Jane	ORL (A)	Doe, Jane	ORL (A)	Doe, Jane	ORL (A)	Doe, Jane	ORL (A)
GSUOR04	AM	Tuesday	Week D	Gen Surg (A)	Doe, John	22-8-2017-GSUOR04-AM	Gen Surg (A)	Doe, John	Gen Surg (A)	Doe, John	Gen Surg (A)	Doe, John	Gen Surg (A)	Doe, John	Gen Surg (A)	Doe, John	Gen Surg (A)
GSUOR04	PM	Tuesday	Week D	Gen Surg (A)	Doe, Jane	22-8-2017-GSUOR04-PM	Gen Surg (A)	Doe, Jane	Gen Surg (A)	Doe, Jane	Gen Surg (A)	Doe, Jane	Gen Surg (A)	Doe, Jane	Gen Surg (A)	Doe, Jane	Gen Surg (A)
GSUOR07	AM	Monday	Week D	Paed Surg (P)	Doe, John	21-8-2017-GSUOR07-AM	Paed Surg (P)	Doe, John	Paed Surg (P)	Doe, John	Paed Surg (P)	Doe, John	Paed Surg (P)	Doe, John	Paed Surg (P)	Doe, John	Paed Surg (P)
GSUOR07	PM	Monday	Week D	Paed Surg (P)	Doe, Jane	21-8-2017-GSUOR07-PM	Paed Surg (P)	Doe, Jane	Paed Surg (P)	Doe, Jane	Paed Surg (P)	Doe, Jane	Paed Surg (P)	Doe, Jane	Paed Surg (P)	Doe, Jane	Paed Surg (P)
A4OR01	AM	Tuesday	Week D	Cardiac (A)	Doe, John	22-8-2017-A4OR01-AM	Cardiac (A)	Doe, John	Cardiac (A)	Doe, John	Cardiac (A)	Doe, John	Cardiac (A)	Doe, John	Cardiac (A)	Doe, John	Cardiac (A)
A4OR01	PM	Tuesday	Week D	Cardiac (A)	Doe, Jane	22-8-2017-A4OR01-PM	Cardiac (A)	Doe, Jane	Cardiac (A)	Doe, Jane	Cardiac (A)	Doe, Jane	Cardiac (A)	Doe, Jane	Cardiac (A)	Doe, Jane	Cardiac (A)
GSUOR02	AM	Tuesday	Week D	Litho (A)	Doe, John	22-8-2017-GSUOR02-AM	Litho (A)	Doe, John	Litho (A)	Doe, John	Litho (A)	Doe, John	Litho (A)	Doe, John	Litho (A)	Doe, John	Litho (A)
GSUOR02	PM	Tuesday	Week D	Litho (A)	Doe, Jane	22-8-2017-GSUOR02-PM	Litho (A)	Doe, Jane	Litho (A)	Doe, Jane	Litho (A)	Doe, Jane	Litho (A)	Doe, Jane	Litho (A)	Doe, Jane	Litho (A)
A4OR07	AM	Tuesday	Week D	ORL (A)	Doe, John	22-8-2017-A4OR07-AM	ORL (A)	Doe, John	ORL (A)	Doe, John	ORL (A)	Doe, John	ORL (A)	Doe, John	ORL (A)	Doe, John	ORL (A)
A4OR07	PM	Tuesday	Week D	ORL (A)	Doe, Jane	22-8-2017-A4OR07-PM	ORL (A)	Doe, Jane	ORL (A)	Doe, Jane	ORL (A)	Doe, Jane	ORL (A)	Doe, Jane	ORL (A)	Doe, Jane	ORL (A)
GSUOR07	AM	Wednesday	Week D	Paed Surg (P)	Doe, John	23-8-2017-GSUOR07-AM	Paed Surg (P)	Doe, John	Paed Surg (P)	Doe, John	Paed Surg (P)	Doe, John	Paed Surg (P)	Doe, John	Paed Surg (P)	Doe, John	Paed Surg (P)
GSUOR07	PM	Wednesday	Week D	OH (A)	Doe, Jane	23-8-2017-GSUOR07-PM	OH (A)	Doe, Jane	OH (A)	Doe, Jane	OH (A)	Doe, Jane	OH (A)	Doe, Jane	OH (A)	Doe, Jane	OH (A)
A9OR01	AM	Wednesday	Week D	Gynae (A)	Doe, John	23-8-2017-A9OR01-AM	Gynae (A)	Doe, John	Gynae (A)	Doe, John	Gynae (A)	Doe, John	Gynae (A)	Doe, John	Gynae (A)	Doe, John	Gynae (A)
A9OR01	PM	Wednesday	Week D	Gynae (A)	Doe, Jane	23-8-2017-A9OR01-PM	Gynae (A)	Doe, Jane	Gynae (A)	Doe, Jane	Gynae (A)	Doe, Jane	Gynae (A)	Doe, Jane	Gynae (A)	Doe, Jane	Gynae (A)
A8OR13	AM	Wednesday	Week D	Ortho (A)	Doe, John	23-8-2017-A8OR13-AM	Ortho (A)	Doe, John	Ortho (A)	Doe, John	Ortho (A)	Doe, John	Ortho (A)	Doe, John	Ortho (A)	Doe, John	Ortho (A)

Figure 56: Formatted CSV with generated coding

Chapter 6. System Evaluation

Originally it was planned to conduct two months of live trials in order for the client to evaluate usefulness and suitability of the application in operational use. Unfortunately because of operational and organisational changes on the client side this was not possible, despite the software being available for use at the start of the planned evaluation period.

In order to showcase the developed application the user stories introduced in Section 4.1.2 as well as the business rules from Section 1.1.3 are evaluated.

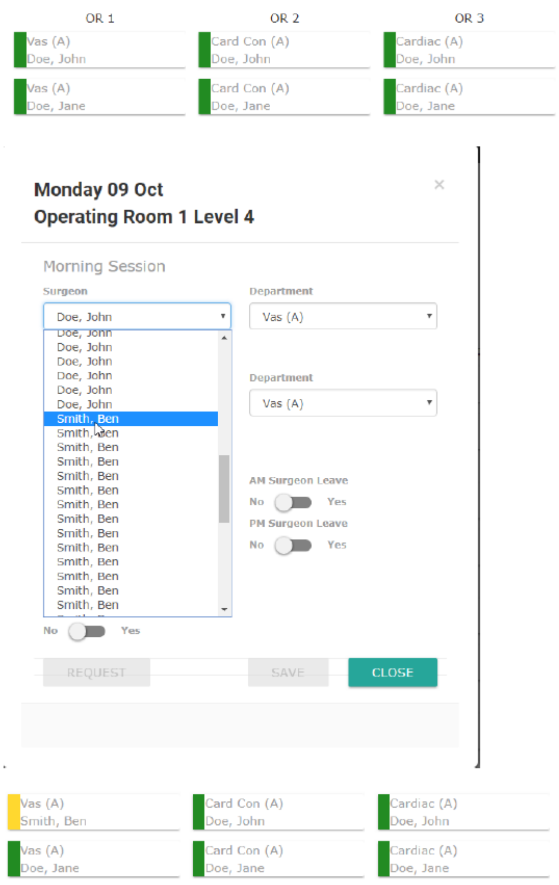
6.1. User Story Evaluation

User Story 1. Access Session Planner

User Story 1 was based around the session planner being available at any time so that the information can be available when needed. The approach taken to achieve this was to use a cloud provider to allow the application to be hosted online. The chosen cloud provider Heroku has a customer promise ”We'll do everything we can to achieve 100% up time” (Salesforce, 2017). As of the time of printing there has been no issues with up time so the acceptance test of 'session planner is available at all times' has been met.

User Story 2. Surgeon Change

User Story 2 indicates that an admin user should be able to change allocated surgeons on specific bookings. Figure 57 demonstrates an admin user changing the surgeon assigned on a specific session and the resulting change being displayed in the main planner. The associated acceptance test for this story is ”Allocated surgeon can be changed on specific room bookings.” The demonstration below fulfils this criteria.


OR 1 | OR 2 | OR 3

Vas (A) Doe, John | Card Con (A) Doe, John | Cardiac (A) Doe, John

Vas (A) Doe, Jane | Card Con (A) Doe, Jane | Cardiac (A) Doe, Jane

Monday 09 Oct
Operating Room 1 Level 4

Morning Session

Surgeon: Doe, John

Doe, John
Doe, John
Doe, John
Doe, John
Doe, John
Doe, John
Smith, Ben
Smith, Ben
Smith, Ben
Smith, Ben
Smith, Ben
Smith, Ben
Smith, Ben
Smith, Ben
Smith, Ben
Smith, Ben
Smith, Ben
Smith, Ben
Smith, Ben
Smith, Ben

Department: Vas (A)

Department: Vas (A)

AM Surgeon Leave: No / Yes

PM Surgeon Leave: No / Yes

No / Yes

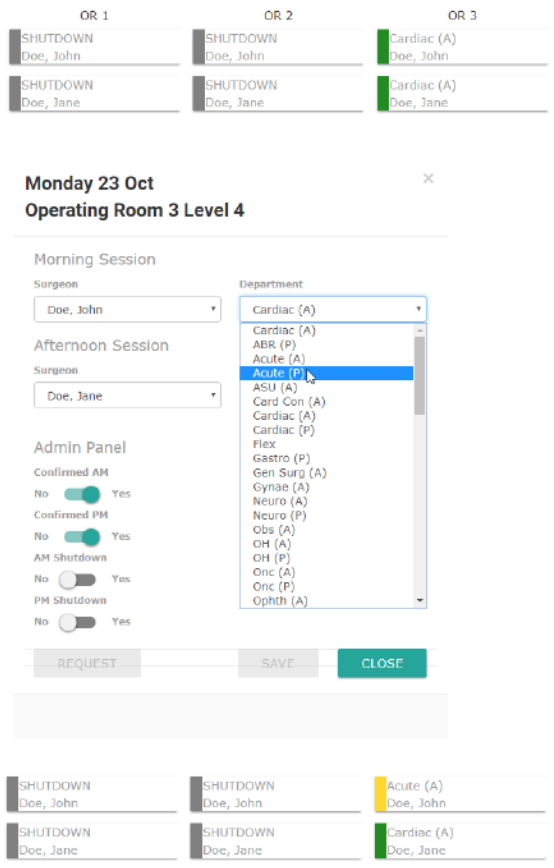
Vas (A) Smith, Ben | Card Con (A) Doe, John | Cardiac (A) Doe, John

Vas (A) Doe, Jane | Card Con (A) Doe, Jane | Cardiac (A) Doe, Jane

Figure 57: Demonstration of an admin user reallocating surgeons.

User Story 3. Service Change

Much like the above User Story 2 a session's service must be able to be reallocated by an admin user. Figure 58 shows this process and also fulfils the associated acceptance test ”Allocated service can be changed on specific room bookings”.


OR 1 | OR 2 | OR 3

SHUTDOWN Doe, John | SHUTDOWN Doe, John | Cardiac (A) Doe, John

SHUTDOWN Doe, Jane | SHUTDOWN Doe, Jane | Cardiac (A) Doe, Jane

Monday 23 Oct
Operating Room 3 Level 4

Morning Session
Surgeon: Doe, John
Department: Cardiac (A)

Afternoon Session
Surgeon: Doe, Jane

Admin Panel
Confirmed AM: No / Yes
Confirmed PM: No / Yes
AM Shutdown: No / Yes
PM Shutdown: No / Yes

SHUTDOWN Doe, John | SHUTDOWN Doe, John | Acute (A) Doe, John

SHUTDOWN Doe, Jane | SHUTDOWN Doe, Jane | Cardiac (A) Doe, Jane

Figure 58: Demonstration of an admin user reallocating services.

User Story 4. Identify Status of Session

A key feature missing from the existing manual based system was clarity of information, it was difficult to determine the status of each individual session. Borrowing from an already established set of internal standards, colors were added to each session to indicate status.

- Green: Session is as originally scheduled.
- Yellow: The Session has been recycled and confirmed.
- Orange: This session has been released and is not confirmed.
- Purple: This surgeon is on leave but session has not been released.
- Gray: This session has been shutdown (public holiday, no staff, maintenance)

The acceptance criteria for this story is that the multiple statuses can be effectively identified. Using the color based system this test is fulfilled shown in Figure 59 this has been fulfilled.

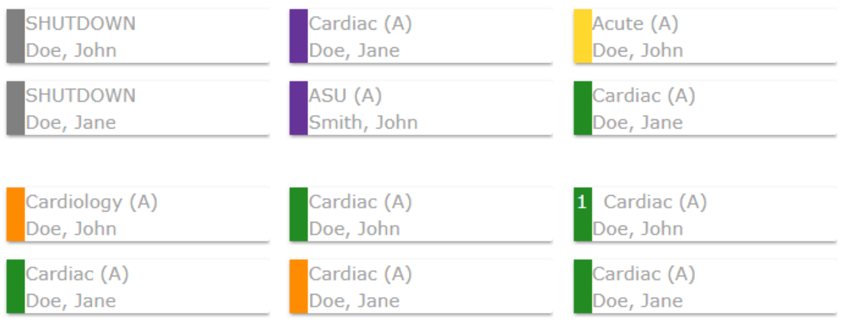


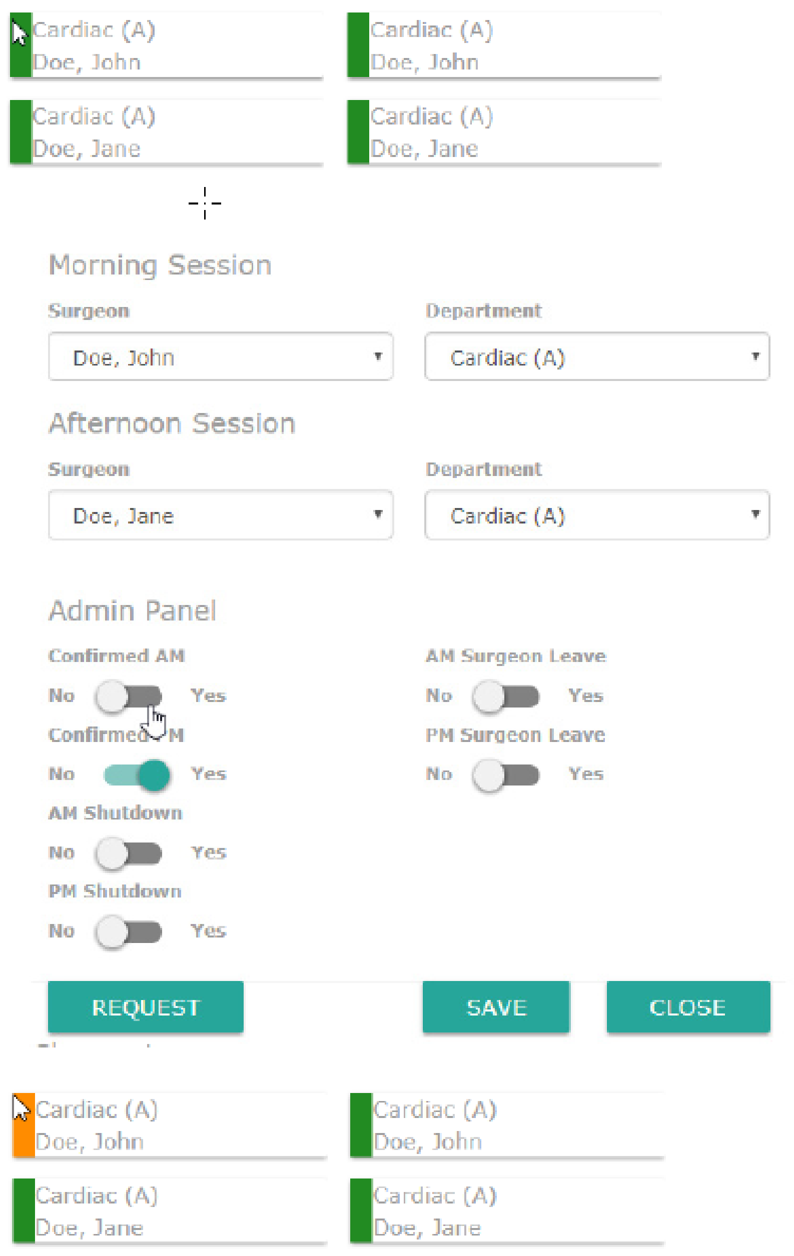
Figure 59: Demonstration of various statuses.

User Story 5: Bulk add sessions

The purpose of User Story 5 is to ease the process of importing data in to the application. The acceptance criteria states that this should be imported using CSV format as this is the existing format of the data. Although not he most elegant solution this has be fulfilled by importing via CSV option through the database IDE DataGrip (JetBrains, 2018). In the simplest sense this works as intended, yet there is no error catching or validation which could prove problematic should some dirty data be added. This is a high importance problem which needs to be further addressed.

User Story 6: Confirm/unconfirm session

Knowing whether a session is confirmed or not is a large advantage to using a real time application like the one developed in this research. Previously it was a fairly tedious job to email the relevant people to check whether a session was being used or not. By having this displayed clearly using the colour statuses users can know instantly whether sessions are available or not before requesting them. Figure **??** shows how confirming and unconfirming can be achieved.



Cardiac (A) Doe, John	Cardiac (A) Doe, John
Cardiac (A) Doe, Jane	Cardiac (A) Doe, Jane

Morning Session

Surgeon: Doe, John

Department: Cardiac (A)

Afternoon Session

Surgeon: Doe, Jane

Department: Cardiac (A)

Admin Panel

Confirmed AM: [✓ No] Yes

Confirmed PM: No [✓ Yes]

AM Shutdown: [✓ No] Yes

PM Shutdown: [✓ No] Yes

AM Surgeon Leave: [✓ No] Yes

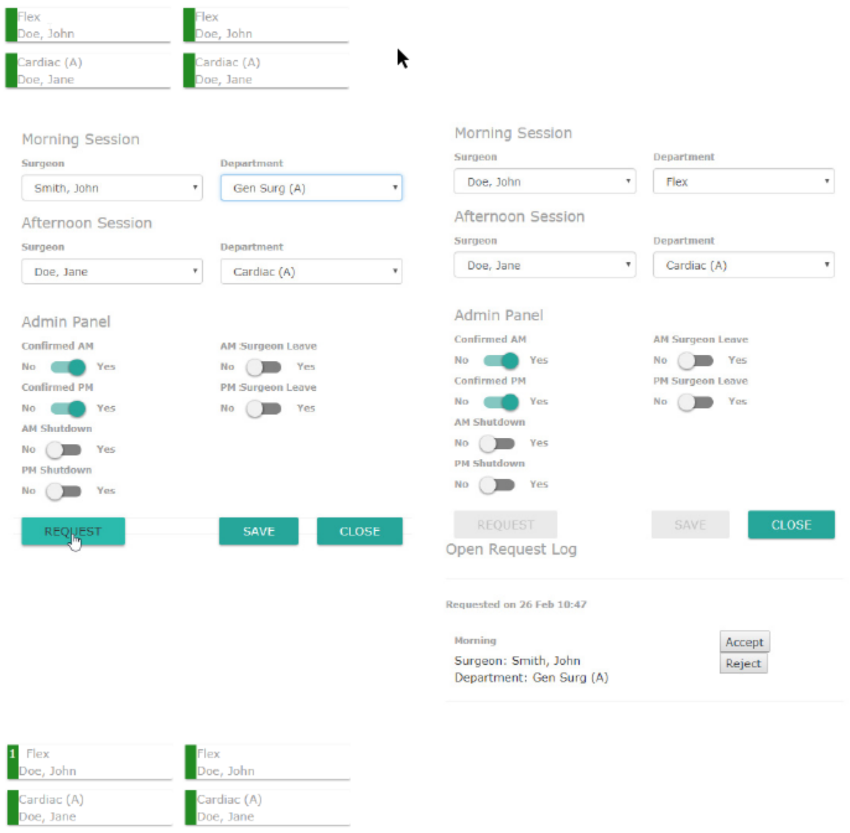
PM Surgeon Leave: [✓ No] Yes

Cardiac (A) Doe, John	Cardiac (A) Doe, John
Cardiac (A) Doe, Jane	Cardiac (A) Doe, Jane

Figure 60: Setting confirmation status

User Story 7: Request a session and User Story 8: Identify amount of requests

As User Story 7 and 8 are closely related they will be demonstrated together. In order for a user to let other users know of their request to use a certain session this must be easy to see and identify from inside the application.



Flex
Doe, John

Flex
Doe, John

Cardiac (A)
Doe, Jane

Cardiac (A)
Doe, Jane

Morning Session

Surgeon	Department
Smith, John	Gen Surg (A)

Afternoon Session

Surgeon	Department
Doe, Jane	Cardiac (A)

Admin Panel

Confirmed AM: Yes
Confirmed PM: Yes
AM Shutdown: No
PM Shutdown: No
AM Surgeon Leave: No
PM Surgeon Leave: No

Morning Session

Surgeon	Department
Doe, John	Flex

Afternoon Session

Surgeon	Department
Doe, Jane	Cardiac (A)

Admin Panel

Confirmed AM: Yes
Confirmed PM: Yes
AM Shutdown: No
PM Shutdown: No
AM Surgeon Leave: No
PM Surgeon Leave: No

Open Request Log

Requested on 26 Feb 10:47

Morning
Surgeon: Smith, John
Department: Gen Surg (A)

1 Flex
Doe, John

Flex
Doe, John

Cardiac (A)
Doe, Jane

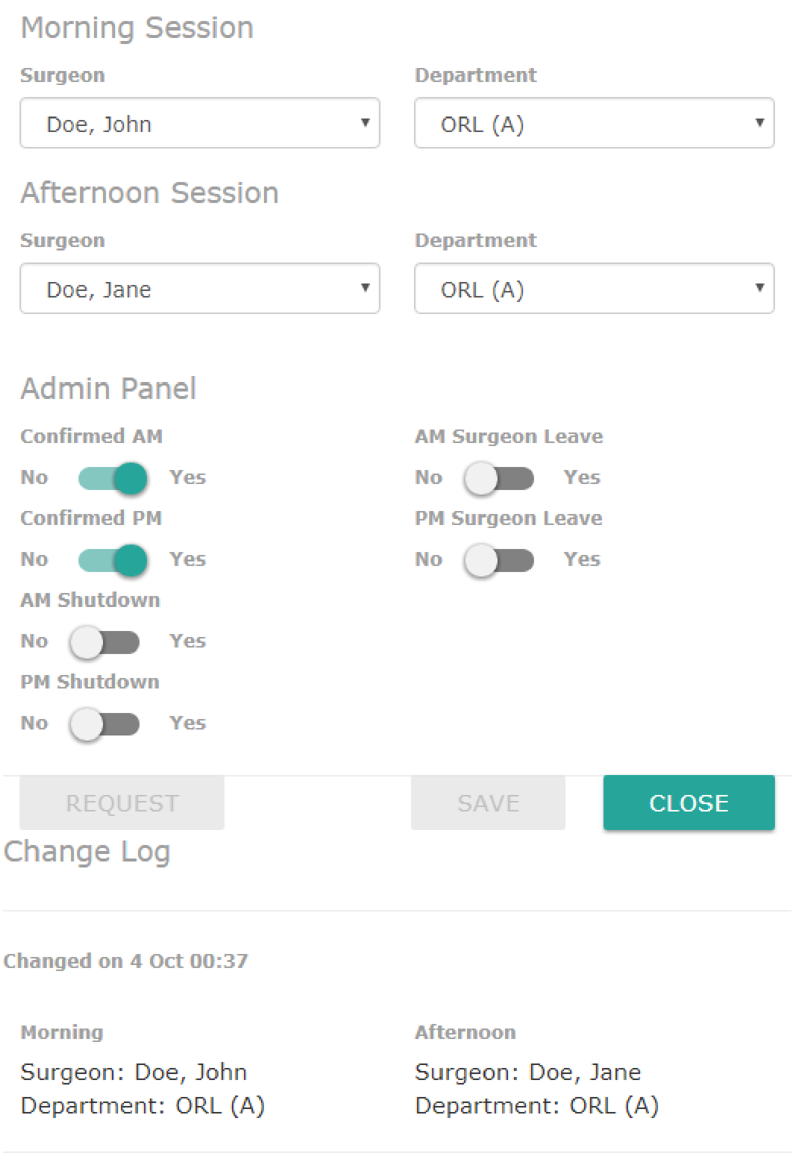
Cardiac (A)
Doe, Jane

Figure 61: Setting confirmation status and showcase of request log

Figure 61 showcases requesting a session and the resulting actions. After requesting a session a counter is added on the session which can be seen from within the planner. On top of this all requests can be viewed within the detail panel for the concerned session. If the user is an admin user they further have the option to accept or reject said session.

User Story 9: See change log of session

A new feature that was not possibly in the existing system is that of a change log for sessions. As every change is recorded it is possible to have this valuable feature on display. The information can be seen inside each sessions detailed view shown in Figure 62.


Morning Session

Surgeon	Department
Doe, John	ORL (A)

Afternoon Session

Surgeon	Department
Doe, Jane	ORL (A)

Admin Panel

Confirmed AM: Yes	AM Surgeon Leave: No
Confirmed PM: Yes	PM Surgeon Leave: No
AM Shutdown: No	
PM Shutdown: No	

Change Log

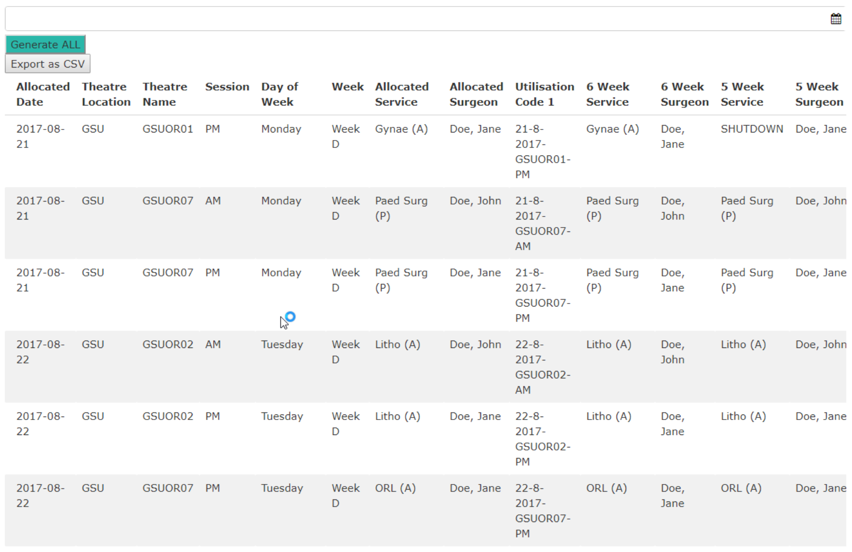
Changed on 4 Oct 00:37

Morning	Afternoon
Surgeon: Doe, John	Surgeon: Doe, Jane
Department: ORL (A)	Department: ORL (A)

Figure 62: Demonstration of timestamped change log

User Story 10 / 11 / 12: Reporting

Reporting being a huge advantage for a digital system over the existing manual system it is important that the process is simple and accurate. As described in Section 5.2 all the logic is controlled by the reporting controllers. Whereas for the user all the reports are a simple one button affair. This will display the reports to the screen for quick checks or alternatively have the option to download as an CSV file in a format suitable for integration to other systems.



Allocated Date	Theatre Location	Theatre Name	Session	Day of Week	Week	Allocated Service	Allocated Surgeon	Utilisation Code 1	6 Week Service	6 Week Surgeon	5 Week Service	5 Week Surgeon
2017-08-21	GSU	GSUOR01	PM	Monday	Week D	Gynae (A)	Doe, Jane	21-8-2017-GSUOR01-PM	Gynae (A)	Doe, Jane	SHUTDOWN	Doe, Jane
2017-08-21	GSU	GSUOR07	AM	Monday	Week D	Paed Surg (P)	Doe, John	21-8-2017-GSUOR07-AM	Paed Surg (P)	Doe, John	Paed Surg (P)	Doe, John
2017-08-21	GSU	GSUOR07	PM	Monday	Week D	Paed Surg (P)	Doe, Jane	21-8-2017-GSUOR07-PM	Paed Surg (P)	Doe, Jane	Paed Surg (P)	Doe, Jane
2017-08-22	GSU	GSUOR02	AM	Tuesday	Week D	Litho (A)	Doe, John	22-8-2017-GSUOR02-AM	Litho (A)	Doe, John	Litho (A)	Doe, John
2017-08-22	GSU	GSUOR02	PM	Tuesday	Week D	Litho (A)	Doe, Jane	22-8-2017-GSUOR02-PM	Litho (A)	Doe, Jane	Litho (A)	Doe, Jane
2017-08-22	GSU	GSUOR07	PM	Tuesday	Week D	ORL (A)	Doe, Jane	22-8-2017-GSUOR07-PM	ORL (A)	Doe, Jane	ORL (A)	Doe, Jane

Figure 63: Six week report in application

Figure 63 and Figure 56 show the reports in their two forms.

6.2. SCRUM and Business Rules Evaluation

For this application to have any chance of uptake it has to fully support the existing SCRUM and business rules. This section is going to evaluate the applications suitability in regards to this. The proposed new operating room scheduling process introduced in Section 1.1.4 has four distinct steps:

1. The surgical booker can compare their own notes against the master planner inside the application. Any leave or requests that need to be recorded can be done at this point.

2. Production coordinators have service level SCRUM meetings to review waiting lists considering surgeon leave and requests. If any sessions need to be released or requested to match demand, this is where it happens.

3. At the weekly SCRUM meeting all attendees can see the state of the master schedule in real time. Discussion around requests can be carried out and confirmed or rejected in this meeting, with all actions clearly timestamped.The change report can then be generated and sent out to all in attendance in place of the previous meeting minutes.

4. All operating managers can review the decisions made and check the master planner to ensure they are staffed and equipment is available. If they as a department decide at this stage they can notify the surgeons else the surgeons can see for themselves by logging in and checking the planner.

There is 6 main business rules around the SCRUM process which was introduced in Section 1.1.3 which also need support:

1. A surgeon on leave is shown through the planner as a purple colour. This does not indicate that the session is available for reallocation.

2. At the SCRUM meeting three weeks before a session with a surgeon on leave is due to occur, the session will be released if the service has not confirmed otherwise. This is done by unconfirming inside the application and the session in question will be orange at this stage.

3. After releasing a session the original service can claim this session back as long as it has no other requests on it. This can be done through the standard request feature inside the planner and it will be timestamped accordingly. At the next SCRUM meeting or Admin review whichever comes first this will be re-confirmed to the original service. If there was requests on this session the standard SCRUM process will occur and the resulting actions will be displayed inside the application.

4. When requesting a session, this request is added to the request log. If there are multiple requests on a session they go through the priority rules detailed in Section 1.1.3. The outcome of this process is displayed inside the application.

5. The operating room scheduler is the point of contact for confirming availability of sessions inside the application. This person has full Admin rights and can manipulate the planner as needed.

Chapter 7. Conclusion

This research investigated how a purpose-built software could increase utilisation and visibility in regards to the scheduling of operating room sessions. The research was undertaken using a combination of design science and an in depth literature review in order to grasp the problem space. As a result of this methodology an artefact in the form of an Angular application was developed based upon user stories and business rules from the existing manual system. The purpose-built software developed in this research is a new concept to the problem space far removed from the existing manual processes.

The question was asked in Section 1.2, Can the existing operating room scheduling process in Auckland Hospital be improved by implementing a purpose-built software? It was hypothesised that the usage of a purpose-built software would increase both utilisation of operating rooms and the visibility of information.

It is clear to see through the system's evaluation in Chapter 6 that the application covers all user stories and business rules that are required for this application to be regarded as usable. Further more when comparing the developed Angular application to the existing manual system it is much easier to see the status of every session, it is much easier to request and release sessions, every interested party can see the master planner in a clear and concise interface at any time through the online application and reporting has gone from a tedious and time consuming task to a one button push.

Unfortunately there was no live testing of the application during this research so it can not be confirmed whether by using the developed application operating room session utilisation would increase.

To conclude it is believed that through the use of the purpose-built Angular application the operating room scheduling system at Auckland Hospital could be improved. These improvements would be best seen through the increased clarity of information through the session planner and the time saving nature of the reporting features.

7.1 Limitations

7.1.1. Live Trial

The lack of a live trial was a large limitation in validating the research question. In an ideal situation the application could have been developed and used in a side by side manner to the existing system. This would have provided valuable insights to the development process as pain points could have been assessed in real time and fixed in further iterations of the software.

Following this development period a full scale live trial would have been conducted with utilisation metrics being compared to previous months and years giving a more quantitative answer to the research question.

7.1.2 Time

The full technology stack being new meant that a substantial amount of time was dedicated to understanding best practices and exploring capabilities in the stack. This lead to many rewrites in code as best methods were discovered.

7.1.3 Usability Study

Although tested against the business rules and user stories a usability study of the application being used by an external party would strengthen the conclusion made above. This was not possible due to the lack of a live trial and time constraints.

7.1.4 Lack of Existing Solutions

Perhaps more an opportunity than a limitation but it is worth noting that there is very little existing solutions to the problem space which meant that the application had very little to be compared too. As many applications that are developed are not necessarily new but aim to improve on existing applications, this was not a luxury available in this research.

7.2 Further Work

7.2.1 Live Trial

As discussed in the limitations a live trial would be ideal in order to quantify the improvements the application brings to the existing solution.

7.2.2 Usability Study

Also discussed in the limitations a usability study would prove valuable in confirming the findings of this research as well as a list of possible improvements which can be added in future releases of the software.

7.2.3 Improved and Additional Reporting

As every change made in the application is logged it is possible to greatly improve the reporting features of the application. This information could lead to allocation being digitally allocated based on past trends and usage without the need for manual intervention.

7.2.4 Professional Front-End Design

As well as increasingly usability by employing a front-end designer would give the application a much more polished and professional feel.

7.2.5 Mobile Optimisation

The application was solely developed for a desktop computer. By optimising it for mobile phones it would further increase its clarity of information as it could be viewed in many more situations.

7.2.6 Code Rewrites

As this application was originally a proof of concept a lot of the early code could be further optimised for readability in particular for if additional developers were to work on the project.

7.2.7 Improved Import Feature

Although potentially a feature that wouldn't be needed further down the track an in app import feature would be beneficial. This would allow an Admin user to import session data without fear of corruption or bad formatting. This feature would gate-keep all information sent to the database.

7.2.8 Advanced Alerting Feature

This feature would be implemented in order to alert users that are subscribed to specific events. Examples of this could be:

- A service wishes to be alerted every time a session is released on a certain day.
- A surgeon wishes to know when their sessions have been reallocated or released.
- An admin wishes to know when a session hasn't been confirmed within the business rules logic

With these alerts the application would allow the portrayal of information to be even greater.

References

Abedini, A., Li, W., & Ye, H. (2017). An optimization model for operating room scheduling to reduce blocking across the perioperative process. *Procedia Manufacturing*, *10*, 60–70.

Adolph, S., Cockburn, A., & Bramble, P. (2002). *Patterns for effective use cases*. Addison-Wesley Longman Publishing Co., Inc.

Appavu, S. T., Al-Shekaili, S. M., Al-Sharif, A. M., & Elawdy, M. M. (2016). The burden of surgical cancellations and no-shows: Quality management study from a large regional hospital in oman. *Sultan Qaboos University Medical Journal*, *16*(3), e298.

Atlassian. (2016). *Trello [software].* Retrieved from `https://trello.com/` (Online; accessed 1-February-2018)

auth0 inc. (2018). *Single sign on and token based authentication.* Retrieved from `https://auth0.com/` (Online; accessed 1-February-2018)

Bartlett, J. (2016). *User story vs requirement whats the difference?* Retrieved from `https://blog.testlodge.com/user-story-vs-requirements/` (Online; accessed 1-February-2018)

Belshe, M., Thomson, M., & Peon, R. (2015). Hypertext transfer protocol version 2 (http/2).

Blake, J. T., & Carter, M. W. (1997). Surgical process scheduling: a structured review. *Journal of the society for health systems*, *5*(3), 17–30.

Blake, J. T., & Carter, M. W. (2002). A goal programming approach to strategic resource allocation in acute care hospitals. *European Journal of Operational Research*, *140*(3), 541–561.

Cardoen, B., Demeulemeester, E., & Beliën, J. (2009). Optimizing a multiple objective surgical case sequencing problem. *International Journal of Production Economics*, *119*(2), 354–366.

CData Software, Inc. (2017). *Easily build angular2 database apps.* Retrieved from `https://www.cdata.com/kb/articles/apiserver-angular2.rst` (Online; accessed 1-February-2018)

CData Software, Inc. (2018). *Cdata software.* Retrieved from `https://www.cdata.com/` (Online; accessed 1-February-2018)

Christie, T. (2018). *django-rest-framework.* Retrieved from http://www.django-rest-framework.org (Online; accessed 1-February-2018)

Deering, S. (2012). *Do you know what a rest api is?* Retrieved from https://www.sitepoint.com/developers-rest-api/ (Online; accessed 1-February-2018)

Dexter, F., Macario, A., Qian, F., & Traub, R. D. (1999). Forecasting surgical groups total hours of elective cases for allocation of block time application of time series analysis to operating room management. *Anesthesiology: The Journal of the American Society of Anesthesiologists*, *91*(5), 1501–1501.

Dexter, F., Macario, A., Traub, R. D., Hopwood, M., & Lubarsky, D. A. (1999). An operating room scheduling strategy to maximize the use of operating room block time: computer simulation of patient scheduling and survey of patients' preferences for surgical waiting time. *Anesthesia & Analgesia*, *89*(1), 7–20.

Dexter, F., & Traub, R. D. (2002). How to schedule elective surgical cases into specific operating rooms to maximize the efficiency of use of operating room time. *Anesthesia & Analgesia*, *94*(4), 933–942.

Earnhart, S. (2003). Make clear rules on block scheduling. *OR manager*, *19*(9), 40–42.

Erl, T., Carlyle, B., Pautasso, C., & Balasubramanian, R. (2012). *Soa with rest: Principles, patterns &constraints for building enterprise solutions with rest.* Prentice Hall Press.

Facebook. (2017). *Sites using react.* Retrieved from https://github.com/facebook/react/wiki/Sites-Using-React (Online; accessed 1-December-2017)

Facebook. (2018). *React.* Retrieved from https://reactjs.org/ (Online; accessed 1-January-2018)

Gershun, A., & Wulff, M. R. (2014). *Alasql [software].* Retrieved from https://github.com/agershun/alasql (Online; accessed 1-February-2018)

Google. (2018). *Angular.* Retrieved from https://angular.io/ (Online; accessed 1-January-2018)

Google calendar. (n.d.). Google. Retrieved from https://calendar.google.com/

Gross, J. M., & McInnis, K. R. (2003). *Kanban made simple: demystifying and applying toyota's legendary manufacturing process.* AMACOM Div American Mgmt Assn.

Guerriero, F., & Guido, R. (2011). Operational research in the management of the operating theatre: a survey. *Health care management science*, *14*(1), 89–114.

JetBrains. (2018). *Datagrip many databases, one tool.* Retrieved from https://www.jetbrains.com/datagrip/ (Online; accessed 1-February-2018)

Kharraja, S., Albert, P., & Chaabane, S. (2006). Block scheduling: Toward a master surgical schedule. In *Service systems and service management, 2006 international* *conference on* (Vol. 1, pp. 429–435).

Krause, S. (2017). *Results for js web frameworks benchmark round 6.* Retrieved from `http://www.stefankrause.net/wp/?p=431` (Online; accessed 1-February-2018)

Li, X., Rafaliya, N., Baki, M. F., & Chaouch, B. A. (2017). Scheduling elective surgeries: the tradeoff among bed capacity, waiting patients and operating room utilization using goal programming. *Health care management science*, *20*(1), 33–54.

Macario, A., Vitez, T., Dunn, B., & McDonald, T. (1995). Where are the costs in perioperative care?: Analysis of hospital costs and charges for inpatient surgical care. *Anesthesiology: The Journal of the American Society of Anesthesiologists*, *83*(6), 1138–1144.

Masursky, D., Dexter, F., OLeary, C. E., Applegeet, C., & Nussmeier, N. A. (2008). Long-term forecasting of anesthesia workload in operating rooms from changes in a hospitals local population can be inaccurate. *Anesthesia & Analgesia*, *106*(4), 1223–1231.

Miller, R. D., Eriksson, L. I., Fleisher, L. A., Wiener-Kronish, J. P., Cohen, N. H., & Young, W. L. (2014). *Miller's anesthesia e-book.* Elsevier Health Sciences.

Minar, I. (2016). *Versioning and releasing angular.* Retrieved from `https://blog.angularjs.org/2016/10/versioning-and-releasing-angular.html` (Online; accessed 1-February-2018)

Nifty Software E.U. (2018). *Made with vue.js.* Retrieved from `https://madewithvuejs.com/` (Online; accessed 1-January-2018)

O'neill, L. (2005). Methods for understanding super-efficient data envelopment analysis results with an application to hospital inpatient surgery. *Health Care Management Science*, *8*(4), 291–298.

Oneill, L., & Dexter, F. (2004). Market capture of inpatient perioperative services using dea. *Health Care Management Science*, *7*(4), 263–273.

Operating room scheduling. (n.d.). Retrieved from `http://www.scheduleyourclinic.com/medical-operating-room-scheduler.htm`

Operating room weekly production planning (scrum) overview. (2014). Retrieved from `https://drive.google.com/open?id=0B2rflpjVyqGXVWNQSlFSM3ZEcDg`

Or scrum overview for surgical board meeting. (2014). Retrieved from `https://drive.google.com/open?id=0B2rflpjVyqGXQWJ0Mjd5V0hkM2s`

Papas, N., O'keefe, R. M., & Seltsikas, P. (2012). The action research vs design science debate: reflections from an intervention in egovernment. *European Journal of Information Systems*, *21*(2), 147–159.

Patterson, P. (1996). What makes a well-oiled scheduling system? *OR manager*, *12*(9), 19–23.

Ramsey, C. (2018). *Vue announces three new hires as part of ongoing expansion.* Retrieved from `http://www.audiomediainternational.com/business/vue-announces-three-new-hires-as-part-of-ongoing-expansion/07225` (Online; accessed 1-February-2018)

Sadler, D. (2015, Jul). Staff shortages and scheduling strategies. *OR Today Magazine*.

Salesforce. (2017). *Customer promises.* Retrieved from `https://www.heroku.com/policy/promise` (Online; accessed 1-February-2018)

Salesforce. (2018). *Heroku.* Retrieved from `https://www.heroku.com/` (Online; accessed 1-February-2018)

Schneier, B. (2009). *Schneier on security*. John Wiley & Sons.

Stack Overflow. (2016). *About stack overflow.* Retrieved from `https://stackoverflow.com/company` (Online; accessed 1-January-2018)

Stepaniak, P. S., & Pouwels, S. (2017). Balancing demand and supply in the operating room: A study for the cardiothoracic department in a large teaching hospital. *Journal of Clinical Anesthesia*, *42*, 7–8.

Strum, D. P., Vargas, L. G., & May, J. H. (1999). Surgical subspecialty block utilization and capacity planning a minimal cost analysis model. *Anesthesiology: The Journal of the American Society of Anesthesiologists*, *90*(4), 1176–1185.

Sutherland, J., & Schwaber, K. (2013). The scrum guide. the definitive guide to scrum: The rules of the game. *ScrumGuides. com*.

VanBerkel, P. T., & Blake, J. T. (2007). A comprehensive simulation for wait time reduction and capacity planning applied in general surgery. *Health care management Science*, *10*(4), 373–385.

Von Alan, R. H., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. *MIS quarterly*, *28*(1), 75–105.

Vue.js. (2018). *Single file components.* Retrieved from `https://vuejs.org/v2/guide/single-file-components.html` (Online; accessed 1-February-2018)

You, E. (2014). *The progressive javascript framework.* Retrieved from `https://vuejs.org/` (Online; accessed 1-January-2018)

You, E. (2018). *Vuejs.* Retrieved from `https://github.com/yyx990803` (Online; accessed 1-January-2018)