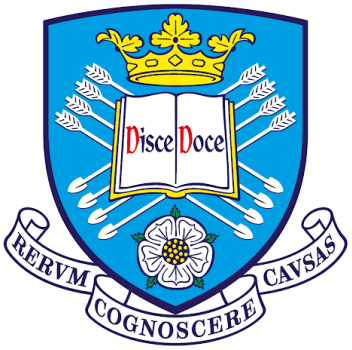


The University Of Sheffield.

A Meeting Scheduler

THOMAS PANTIORAS

Supervisor: Dr Siobhan North

COM - MSc Dissertation Project (2017-18)

This report is submitted in partial fulfilment of the requirement for the degree of
MSc Software Systems and Internet Technology
by
Thomas Pantioras.

12 September 2018

DECLARATION:

All sentences or passages quoted in this report from other people's work have been specifically acknowledged by clear cross-referencing to author, work and page(s). Any illustrations which are not the work of the author of this report have been used with the explicit permission of the originator and are specifically acknowledged. I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in this project and the degree examination as a whole.

Name: Thomas Pantioras

Signature: 

Date: 12 September 2018

ABSTRACT

Every year there are cases of University students who do not attend their courses or their progress is unsatisfactory. In order to deal with these issues there are certain procedures which are followed to assist the students with any problems they are facing and to help them to start performing well again. The appointments which take place between the student and University staff are being booked and managed by the Student Support Services (SSS) department. However, the administration task of finding available slots for all participants is always time consuming and a tedious exercise.
The project is aiming at producing a software system to facilitate the above process by providing information to the SSS admin staff about the availability of the meetings' participants. An even better solution would include functionality to automatically book these appointments for all relative parties.
The project utilised techniques from various fields of Software Engineering to elicit requirements, design, implement and thoroughly test the system. The result is a robust, modular application which exceeds expectations and covers fully its functional and non-functional requirements.

ACKNOWLEDGEMENTS

I would like to thank my supervisor Dr Siobhan North for the continuous support, guidance and time she offered throughout the Meeting Scheduler project. Equally I would like to thank Ms Cheryl A. Hewitt who was the client for this project, for her patience, the valuable feedback and availability. Finally, I would like to thank my family for their support, encouragement and for always being there when I need them.

Table of Contents

Declaration: 2
Abstract 3
Acknowledgements 4
Table of Contents 5
Table of Figures 9
1. INTRODUCTION 10
1.1 Background of Project 10
1.2 Project aim 10
1.3 Dissertation structure 11
2. LITERATURE REVIEW 12
2.1 University of Sheffield Regulations and Progress of Students procedure 12
2.2 Google Calendar Api 12
2.3 Software Project and Software Project Management 13
2.4 Project Planning 13
2.4.1 Project Methodologies 13
2.4.1.1 Waterfall 14
2.4.1.2 V- Model 14
2.4.1.3 Rational Unified Process (RUP) 15
2.4.1.4 Spiral model 15
2.4.1.5 Evolutionary Prototyping methodology 15
2.4.1.6 MSA project methodology 16
2.4.2 MSA Project Schedule 16
2.4.3 Risk Management 17
2.4.3.1 Risk Assessment 17
2.4.3.2 Risk Control 18
2.5 Desktop vs Web applications 18
2.6 Patterns 19
2.7 Model View Controller 19
2.8 Libraries and Frameworks 20
2.9 Programming languages and MVC options 20
2.9.1 Java MVCs 21
2.10 Application Architecture 22
2.10.1 Layered Architecture 22
2.10.2 Onion architecture 22
2.10.2.1 Service Locator and Dependency Injection frameworks 23
2.10.2.2 MSA chosen architecture and dependency injection method 24
2.11 Spring Boot 24
2.12 Security considerations - Login and OWASP 24
2.12.1 OWASP Top 10 Application Security Risks – 2017 25
2.13 Databases 26
2.13.1 Relational databases 26
2.13.2 NoSQL databases 26
2.13.3 MSA database solution and integration methods 27

2.13.3.1 JDBC, ORMs and Spring Data 27
2.14 Other tools and technologies used 27
2.14.1 HTML5 and CSS 27
2.14.2 CSS and Bootstrap 28
2.14.3 JavaScript, Ajax and jQuery 28
2.14.4 Maven 28
2.14.5 Git 28
2.14.6 IntelliJ IDE 28
2.14.7 Astah modelling tool 29
2.15 Summary 29
3. REQUIREMENTS AND ANALYSIS 30
3.1 Types of Users 30
3.2 Requirements gathering techniques 30
3.2.1 Interviews 30
3.2.2 Questionnaires 31
3.2.3 Task Analysis 31
3.2.4 Domain Analysis 32
3.2.5 Prototyping 32
3.3 Software Requirements 32
3.3.1 Functional Requirements 32
3.3.1.1 High Priority Requirements 33
3.3.1.2 Low Priority Requirements 34
3.3.2 Non-Functional Requirements 35
3.3.3 Use Case Diagrams 37
3.4 Summary 38
4. DESIGN 39
4.1 Modelling MSA Activities 39
4.1.1 Managing and Updating Faculty Officer data 39
4.1.2 Complete booking meeting process 40
4.4 Detailed Class Diagram 40
4.3 Modelling Interaction Sequence for the MSA 41
4.4 Logical and Physical Database Design 42
4.5 Human Computer Interaction 43
4.5.1 Consistency 43
4.5.2 Feedback and Design to Yield Closure 44
4.5.3 Error prevention 44
4.5.4 Permit easy reversal of actions/ User Control 44
4.5.5 Support internal locus of control/ Aesthetic and minimalist design 44
4.5.6 Affordance 45
4.5.7 Match between system and the real world 45
4.5.8 Constraints 45
4.6 Summary 45
5. IMPLEMENTATION AND TESTING 46
5.1 Implementation 46
5.1.1 Managing the Domain Entities data 46
5.1.2 Searching for the Faculty Officer Availability 48
5.1.2.1 Algorithm for finding Faculty Officer availability 49
5.1.3 Booking an Appointment 50
5.1.4 Login, Password Reset and Logout 51

5.2 Testing 52
5.2.1 Functional testing 52
5.2.2 Non-Functional testing 53
5.2 Summary 54
6. RESULTS AND DISCUSSION 55
6.1 Project Artefacts 55
6.2 Evaluation 56
6.2.1 Areas for Improvement 57
6.3 Future Work 59
6.4 Personal Reflections and Gains 60
7. CONCLUSION 62
REFERENCES 63
Appendix A. Acronyms 76
Appendix B. Risk Management Plan 76
B.1 Planning Phase Risk Management Plan 76
B.2 Phase-1 Risk Management Plan 80
Changes from Previous Plan 80
Risk Management Plan 80
B.3 Phase-2 Risk Management Plan 84
Changes from Previous Plan 84
Risk Management Plan 84
B.4 Phase-3 Risk Management Plan 87
Changes from Previous Plan 88
Risk Management Plan 88
B.5 Phase-4 Risk Management Plan 91
Materialised Risks 91
Changes from Previous Plan 92
Risk Management Plan 92
Appendix C. User Acceptance Testing 96
C.1 MSA User Acceptance Test Form 96
C.2 MSA User Acceptance Test Results 96
Appendix D. Non-Functional Test Results 104
D.1 MSA Non-Functional Test Form 104
D.2 MSA Non-Functional Test Results 104
Appendix E. Manage Departments screen 124
Appendix F. Algorithm for Booking Appointment 125
Appendix G. Use Case diagram for the appointment booking 126
Appendix H. Conceptual Class diagram 126
Appendix I. Conceptual Entity Relationship Diagram 127
Appendix J. Reset Password - Sequence Diagram 127
Appendix K. System Manual 128
K.1 Installation 128
K.1.1 Configuration 128
Port 128
Database 128
Primary Email Address 128
SMTP Email Address 129
K.2. User Manual 129

K.2.1 Login and Logout 129
K.2.2 Reset Password 130
K.2.3 Book Appointment 131
K.2.4 Searching for Faculty Officer Availability 133
K.2.5 Managing System Data 134
Managing Faculties 134
Managing Departments 135
Managing Faculty Officers 137
Managing Users 139
Managing Meeting Rooms 141
Appendix L. Setting up Development Environment 143
Appendix M. Project Schedule 144
Appendix N. MSA Questionnaires 147
N.1 Questionnaire 1 147
N.2 Questionnaire 2 150
N.3 Questionnaire 3 153

Table of Figures

Figure 2.1 - Waterfall software project methodology [15, p.207] 14
Figure 2.2 - V-Model software project methodology [15, p.211] 14
Figure 2.3 - Evolutionary prototyping method [19, p.147] 15
Figure 2.4 - Work Breakdown Structure template (by wbstool.wordpress.com [20]) 16
Figure 2.5 - Risk management activities [23, p.96] 17
Figure 2.6 - Risk probability ranges 18
Figure 2.7 - Risk impact ranges 18
Figure 2.8 - Client server model [28] 19
Figure 2.9 - Flow of data in an MVC application (from teamtreehouse website [33]) 19
Figure 2.10 - MVC implemented with Servlet and JSP (from tutorialpoint.com [57]) 21
Figure 2.11 - Layered architecture 22
Figure 2.12 - Onion architecture [67] 22
Figure 2.13 - Inverted dependencies after applying the DIP principle 23
Figure 2.14 - Onion architecture showing location of components 23
Figure 2.15 - Dependency injection (by Fowler [72]) 24
Figure 3.1 - HTA card for booking an Unsatisfactory Progress meeting 32
Figure 3.2 - High priority functional requirements for the MSA 34
Figure 3.3 - Low priority functional requirements for the MSA 34
Figure 3.4 - Non-functional requirements for the MSA 37
Figure 3.5 - Business use case diagram for the MSA 37
Figure 3.6 - Summary use case diagram for the MSA. 37
Figure 3.7 - User Goal use case diagram for managing the Faculty Officer in MSA 38
Figure 3.8 - Subfunction use case for updating the Faculty Officer in the MSA 38
Figure 4.1 - High level activity diagram for managing a Faculty Officer entity 39
Figure 4.2 - Detailed activity diagram showing steps for updating a Faculty Officer 40
Figure 4.3 - Activity diagram showing the sequence of steps required to book an appointment 40
Figure 4.5 - Detailed class diagram for the MSA system 41
Figure 4.6 - Sequence diagram for booking an appointment with a Faculty Officer 42
Figure 4.7 - Database model for the MSA application 43
Figure 5.1 - Menu list for managing system data 46
Figure 5.2 - Manage Faculty Officers screen 46
Figure 5.3 - Adding Faculty Officer form 47
Figure 5.4 - Confirmation dialogue box for deleting a Faculty Officer 47
Figure 5.5 - Search Faculty Officer availability screen 48
Figure 5.6 - HTML5 data type utilised to assist the user editing the form 48
Figure 5.7 - Validation for required ‘Meeting duration’ field 48
Figure 5.8 - Validation on the allowed range step values for the ‘Meeting duration’ 48
Figure 5.9 - Validation checking that values are logically correct 49
Figure 5.10 - MSA page displaying the available free slots for the Faculty Officer 50
Figure 5.11 - Book appointment form 50
Figure 5.12 - User feedback for successful booking event 51
Figure 5.13 - Google calendar event creation 51
Figure 5.14 - The MSA login page 51
Figure 5.16 - Successful password reset 52
Figure 5.15 - Reset password page 52
Figure 5.17 - Successful logout 52

1. INTRODUCTION

1.1 Background of Project

When students at the University of Sheffield make unsatisfactory progress with their courses there are a number of processes which can take place to assist the student with any issues they are facing. These originally are contained within the Department by encouraging the students to meet their personal tutor. Failing that, a meeting with the year tutor or the Head of Teaching is proposed. If even these meetings are not attended by the students, then the Department can refer them to the Faculty via an Unsatisfactory Progress report to the Student Support Services (SSS). This report has to be submitted before specific deadlines within the year. The Student Support Services, which is a part of a university structure that monitors the progress of students, invites the students to attend an interview with a Faculty Officer. A meeting with a Faculty Officer is serious because students who fail to attend those can be deemed withdrawn from the University or referred to the Progress of Students Committee with a view to exclusion.

There are certain conditions that make the administration of these appointments a particularly difficult task. The SSS admin staff deals with and arranges the meetings for all five faculties of the university, which consist of the Faculty of Arts and Humanities, the Faculty of Engineering, the Faculty of Medicine, Dentistry & Health, the Faculty of Science and the Faculty of Social Sciences. The timing of these meetings is different for each Faculty and is decided based on many factors including the exam results and release dates, student vacations and individual student circumstances. Faculties have primary Faculty Officers who consist the first choice for the meetings but also secondary ones for flexibility and availability reasons. For example, there is a policy rule where the Officer cannot belong to the student's home department or be a lecturer who has taught the student in the past, so an alternative choice is needed.

Once a date has been identified as suitable to start conducting the interviews, the SSS admin staff has to book the meeting slots within two weeks from that date. In order to do that they have to find availability for all participants and to negotiate the most convenient times for all. A lot of discussions are necessary and due to the volume of the meetings and particular circumstances the task is often daunting and always time consuming.

1.2 Project aim

The aim of this project is to create a software product which will facilitate the administration tasks of the SSS admin staff with regards to finding available slots and booking the appointments.
The system would enable the user to collect information about the availability of the Faculty Officers in specified dates. Once suitable dates have been identified, the user can then use this information to automatically book a meeting slot in the calendars of all attendees. An invitation would be sent to all relative parties. Management of the system entities would be done from within the system. As such the user would be able to add and amend Faculties, Faculty Officers, meeting rooms and SSS staff. The application would be available through the Web so that it is easily accessible from the SSS staff. Additional concerns such as usability and security would also be considered to produce a robust system. The project is not just about delivering a product which implements an initial set of requirements that are defined by the client. A big part of the project is to extract requirements which the client has not

considered and which would add real value to the product. The end result should be a solution which is reliable and contributes significantly towards lightening the administration burden of the Unsatisfactory Progress procedure.

1.3 Dissertation structure

The project report has been divided into seven chapters. Starting with the current chapter, the report provides a brief background of the existing processes. It then describes the aims of the project and the structure of the report. Chapter 2 contains the literature review which explores the various methods and technologies that are used for project management and to implement different parts of the system. It also includes details of the initial planning stage of the project, namely the schedule, the project methodology and the risk management plan. Chapter 3 focuses on the requirements gathering and analysis. This includes different ways of eliciting the requirements and categorising the requirements into functional and non-functional. Chapter 4 covers the design phases of the project. It describes the Human Computer Interaction (HCI) principles that were applied and includes design artefacts for both the domain model and the database of the application. The artefacts of the design phases were used as a basis for the implementation of the system. Chapter 5 describes in detail how each requirement was implemented and also includes the testing techniques that were used. Chapter 6 consists of an evaluation of the produced software and how effectively it meets the requirements of the project. It also includes future recommendations and ways for improvement. Finally, chapter 7 presents conclusions and summarises the results of the project.
The Meeting Scheduler application has been a very rewarding experience and allowed to put in practice a lot of the skills that were learned during the Master's first two semesters. Every phase of this project utilised some of the methods, technologies and theory which were taught during the past year. These include for example UML modelling, the Java language, database modelling, SQL skills and JavaScript to name a few. Of course, it was necessary to do further research in order to adapt to project needs and to select the best option to use and that by itself has also been very beneficial. In short, the project has provided the means to consolidate learning, but also to investigate new areas of Software Engineering. All the specific details of this experience are covered in the following report.

2. LITERATURE REVIEW

As mentioned in the Introduction, the Literature Review chapter contains the research findings to assist with the project management and the various phases of the project. There is a number of sections in the chapter covering topics around the project background, the google calendar api, project methodology and planning, risk analysis, various patterns used in the application, the system architecture, web frameworks, programming languages and database solutions among others. For most technologies, alternatives were considered weighing the advantages and disadvantages before selecting the best one to follow for the Meeting Scheduler application (MSA).

2.1 University of Sheffield Regulations and Progress of Students procedure

Admission of any student to the University of Sheffield is subject to the requirement that they will "duly observe the Charter, Statutes, Ordinances and Regulations of the University" [1]. The General University Regulations are published on the University website [2] and all students in any Faculty must comply with them.
The "General Regulations relating to the Progress of Students" [3] is part of the above regulations and is a process initiated by an academic department of the University. The process involves the review of a student's progress based on specified grounds. These include low attendance or performance levels, failure to present written work as this is required by the student's programme, failure to pass an examination, failure to perform a programme's research tasks or cooperate with a supervisor and in general any failure to demonstrate satisfactory competence in any programme of study or research. The procedure includes a meeting of the student with a Faculty Officer and an SSS (Student Support Services) staff to discuss any issues and determine if the student should be allowed to continue with their programme.
As mentioned in the Introduction chapter, there are various factors which restrict the timing of these meetings. There are also deadlines by which a department can submit a request to initialise a student progress review process. These time restrictions are different for each academic year and are also available on the University website [4]. Other relative information can be found in the published notes for the procedure of Faculty Student Review Committees [5]. These include details about who is allowed to participate in the review meetings, the process of communicating information to the student, available student responses and paths of action and finally information about appeals to any decisions taken as result of these reviews.
The forms used by the academic departments to initialise review procedures for undergraduate and postgraduate students as well as the progress appeals form are also available for download [6].

2.2 Google Calendar Api

The main feature of the Meeting Scheduler application is to find available slots and book appointments on the Google Calendar for the Faculty Officer and SSS admin staff. As such one of the first things to research was the way to integrate the MSA with the Google Calendar api [7] to realise these features. Fortunately, the Google Calendar team offers a comprehensive documentation which simplifies the integration task with other applications. This includes an overview of the concepts [8] and internal domain components [9] of the calendar api.

In order for any application to communicate with the api, an authorisation process is required which consists first of the registration of the application with Google Calendar. In addition, to access a specific

private calendar the user of this calendar must give her consent which is a process handled by Google the first time calendar access is required. A description of the steps to authorise calendar requests is offered on the site [10]. Furthermore, a guide of how to create calendar events is also available [11] as are code samples for most common languages. The java samples [12] were used as the starting point for retrieving and creating events for the MSA application. Finally, a reference of the offered api endpoints [13] is available together with an interface to test each of them online.

2.3 Software Project and Software Project Management

Cagley and Chemuturi [14] define a project as "… a temporary endeavour with the objective of manufacturing (producing or developing) a product or delivering a service, while adhering to the specifications of the customer (including functionality, quality, reliability, price, and schedule) and conforming to international / national / customer / internal standards for performance and reliability". It is temporary because it has a definitive start and end, each project is unique in its requirements and contain phases including approval, planning, design, engineering, construction, testing, deliverance and installation. Sometimes they also contain a handover to customer and maintenance phases. A project can be a standalone component or part of a larger programme.
The same authors [14] describe a *software* project to have additional unique properties. The output and artefacts of the project are not physical. The activities are similar but specified as user and software requirements extraction, software design, implementation, testing and acceptance testing, software delivery, deployment and handover. Project acquisition and post-handover activities are normally not considered part of the project. It is hard to assess progress as unfinished software cannot be easily assessed. The diagramming and modelling techniques though evolved they do not have the precision that is apparent in other types of engineering projects. Standards exist but many are not provided by standards' organisations as they do in other engineering fields.

"Software project management is the integration of management techniques into software development" [15, p.10]. The need for software project management has been identified in the 60s when failure of software projects was very common. The complexity of software systems was making it difficult to deliver projects on time, within budget and meeting the client's specifications. As a result, a number of techniques were derived to improve the software development practises.

A software project manager is involved in all the stages of the project, assessing the feasibility of the project, formalising targets, defining schedules, estimating costs, managing changes, monitoring work, assessing and applying methodologies, ensuring quality of output, managing risks and resources. The small scale of the MSA project did not demand an extensive research and effort in all the above areas. Nevertheless it was important to perform some essential project planning in order to ensure a successful outcome. The research and the outcomes of this planning exercise are discussed in the next section.

2.4 Project Planning

The project planning for the MSA consisted of selecting a suitable project methodology, deriving a project schedule, researching and preparing an initial risk management plan.

2.4.1 Project Methodologies

There is a number of project methodologies which exist and were considered for the MSA project. Villafiorita [15, Ch. 7] categorises these into three main types based on the amount of infrastructure and tolerance to requirement changes. These are the traditional, agile and extreme programming categories. The first contains the most formal procedures and highest resistance to change. These attributes are not so apparent with agile methods where there is less planning and changes are welcomed. The extreme programming methodology removes completely any formal specifications and considers changes as an integral part of the software development process.

For the MSA project, the agile and extreme programming were discarded mainly for the reason that these two require high availability of the client and this could not be guaranteed for this project. In addition, because of the small scale of the application it is not expected that any required changes would have such a high impact on the project so that a change tolerant method should be adapted.

A short discussion of the considered traditional project methodologies is offered below.

2.4.1.1 Waterfall

Traditional methodologies support a highly structured framework with a lot of formal processes and specifications. The most traditional is the waterfall methodology. The process is first described by Royce [16] (although not by using the *waterfall* name) and consists of a series of activities which are performed in sequence. Each activity produces an output which is then fed to the next activity until the last one delivers the system in its production state (figure 2.1). The waterfall model does not allow backtracking. That is once an activity has completed it should not be revisited as this would incur a high cost for the project.

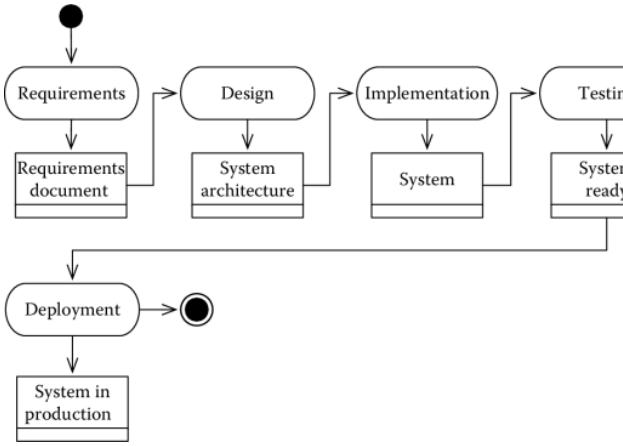


Figure 2.1 - Waterfall software project methodology [15, p.207]

2.4.1.2 V- Model

The next traditional method is called the V-Model [15, Ch. 7] and emphasizes on the validation of the produced software. It is based on the idea that the later any bugs are found in the system the bigger the impact on the project. The stages are the same as with the waterfall method up to the implementation stage. The testing phase is split in four parts, the unit testing, the integration testing, the system testing and the acceptance testing with each of them validating the component, the integration of the components, the system specification and the user requirements respectively. The testing is done sequentially and if one phase fails the project is reset from the stage that failed its validation. The figure below depicts the process.

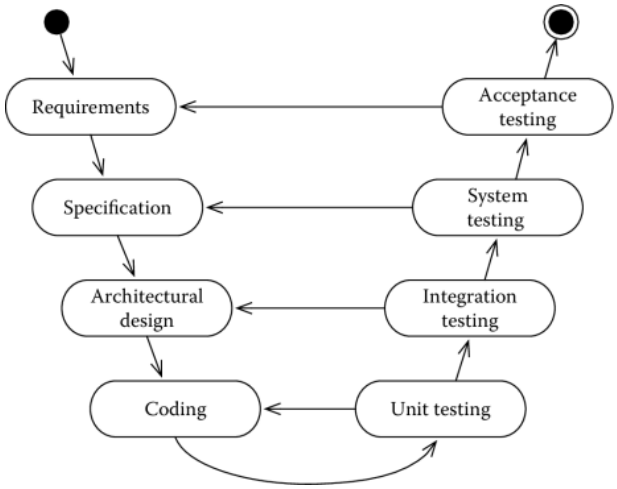


Figure 2.2 - V-Model software project methodology [15, p.211]

2.4.1.3 Rational Unified Process (RUP)

The RUP methodology was designed by Rational [17] and it introduces the concepts of project cycles, phases and workflows. Workflows are activities carried out throughout the project and listed below.

1. Business modelling
2. Requirements
3. Analysis and design
4. Implementation
5. Test
6. Deployment
7. Project Management
8. Configuration and change management
9. Environment

The project lifecycle is broken into cycles, each working towards a new generation of the product. Each cycle consists of 4 different phases, inception, elaboration, construction and transition. All activities can run in parallel in each project phase, but with different intensity. At the end of each phase, objectives are met and mark relevant project milestones. Each phase can also be further broken down in iterations with each resulting in a releasable product version.
The RUP methodology is very comprehensive and contains step by step guides and process templates to assist with the method implementation.

2.4.1.4 Spiral model

The spiral process was proposed by Boehm [18] and supports a risk driven, iterative approach. Each iteration builds on the product of the previous one. All contain the same structure of four activities.
The first activity is to *determine the objectives*. That includes deciding on the iteration objectives, considering constraints like cost and time and identify *potential alternatives* i.e. solutions which satisfy the objectives within the set constraints. The second activity is to *evaluate alternatives and risks*. This involves considering the alternatives from the previous activity, researching and evaluating the best solution which will satisfy both objectives and constraints. The third activity is to *develop*. This consists of materialising or constructing the product which was decided to be the best solution from the second activity. The first *develop* iterations would normally produce specifications and designs while later on, iterative implementations of the system would be produced. The final activity is to *plan the next iteration*. This includes a critical review of the outputs and results of previous phases as well as consideration and prioritisation of the overall project objectives.

2.4.1.5 Evolutionary Prototyping methodology

This method focuses on and facilitates communication of client and development team. The team is building a prototype which is then verified by the clients that it meets their requirements. Any observations about omissions or improvements are taken into consideration for the next evolution of the prototype which also includes enhanced system functionality. The process is formalised by McConnell [19] in four phases which are depicted in the diagram below.

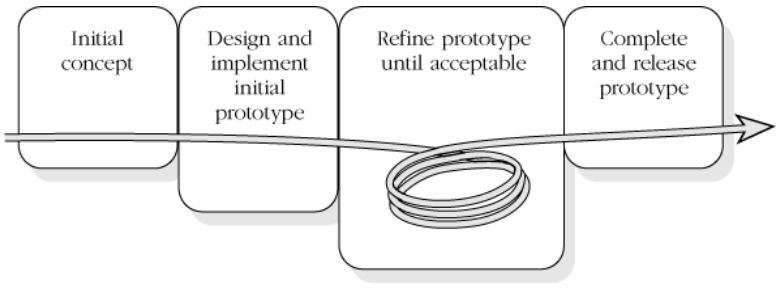


Figure 2.3 - Evolutionary prototyping method [19, p.147]

The evolutionary prototyping methodology is more suited for new projects than legacy ones. One disadvantage is that it makes it harder to plan system development as the client steers the project work after each prototype demonstration.

2.4.1.6 MSA project methodology

All of the above methods could potentially be applied for the MSA project, although some are more suitable than others.
The waterfall could work well as the system is relatively small and a thorough requirements analysis exercise could extract and specify all the client requirements at the start. The developer would then be free to proceed managing the project schedule more efficiently. However due to lack of experience in gathering requirements it was decided to keep a closer contact with the client throughout the project so that it is ensured that the software meets their expectations.
The V-Model was discarded for the same reason but also because of the developer's inexperience with certain types of testing. Retrospectively and having acquired a certain level of testing skill and ability to elicit requirements, it would have been a very good choice for the project.
The Rational Unified Process was discarded because of its comprehensiveness and formality of process. While this would assist very much in larger project it was thought that a more flexible approach would be more suitable.
The evolutionary prototyping technique was considered a good choice as the system is small and it would be possible to create a quick initial prototype with most of the application features. It would then allow the developer to get an early and continuous feedback from the user to evolve the system to the desired output state.
Nevertheless, the method favours coding and allows little time for the analysis and design activities. As the developer has small experience with software projects, it seemed important to not overlook the analysis and design processes. For that reason, it was decided to use a mix of the evolutionary prototyping method with the spiral model. The spiral model activities would be carried out to assist with each iteration of the prototype and to allow for a more analytical approach. At the end of each spiral cycle the evolved prototype would be presented to the customer and the feedback would be considered during the '*plan the next iteration'* activity to determine the next objectives.

2.4.2 MSA Project Schedule

The most simple and common method for producing a project schedule is to apply expert judgement [15, Ch. 3]. There are two main approaches to, the bottom-up and top-down. The former is done by first identifying the set of activities in a project. These are often modelled in the form of a tree structure, called the work breakdown structure (WBS). An example template is illustrated below.

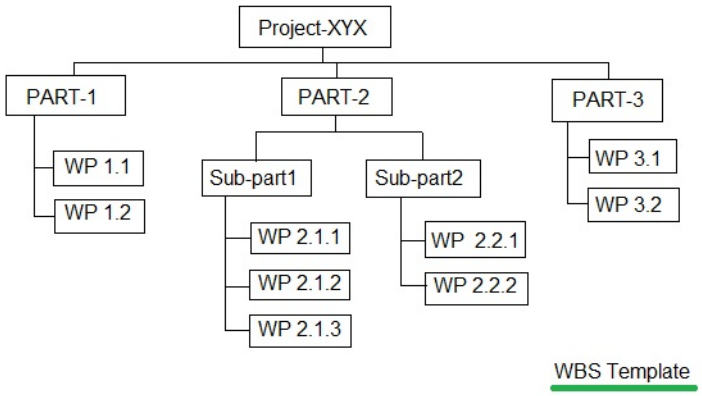


Figure 2.4 - Work Breakdown Structure template (by wbstool.wordpress.com [20])

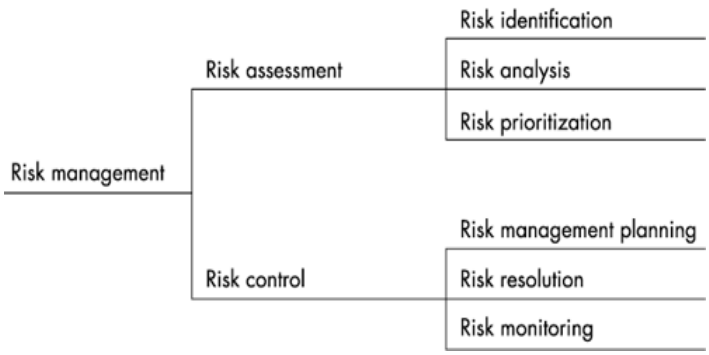
The time required for each leaf of the WBS is then estimated and added to schedule. The total effort and time are simply the addition of all the effort and time needed to perform the total of the WBS activities (allowing for some overlapping of activities).

In contrast the top-down approach is followed by estimating and allocating time for the whole project. From there individual branches are estimated and finally the leaves. This approach normally requires a more experienced project manager to make an accurate estimation. In general top-down estimates tend to underestimate the duration while the bottom-up tend to overestimate [15, Ch. 3].

From the above and considering the student's small experience, it seemed that the most accurate estimation would be achieved using the bottom up approach. Nevertheless, the top-down technique was chosen at the end as the project has strict deadlines which cannot be stretched. The produced MSA schedule spreadsheet is provided with the project artefacts. A Gantt chart [21] was used to outline the schedule of activities and the template used was offered by vertex42 [22]. Screenshots from the project schedule are attached in Appendix M.

2.4.3 Risk Management

Risks are events that may or may not happen and which can have an impact on the project. Risk management is defined by Jalote [23, p.93] as "... an attempt to minimize the chances of failure caused by unplanned events. The aim of risk management is not to avoid getting into projects that have risks but rather to minimize the impact of risks in the projects that are undertaken".



The activities that are carried out in order to mitigate the impact of these risks on the project were first described by Boehm [24] and are shown in the adjacent diagram.

Figure 2.5 - Risk management activities [23, p.96]

2.4.3.1 Risk Assessment

Starting with the risk assessment, the first task for any project manager and team is to identify risks. Any condition or event that could jeopardise the project constitutes a risk. The methods used for risk identification include surveys, interviews, process reviews and meetings with various project stakeholders and team members.
Analysis of the identified risks is concerned with grading the probability and the level of consequence for each risk. The ranges of probability and consequence are shown in figure 2.6 and 2.7 respectively and are suggested by Jalote [23, pp.98 - 99].

Probability	Range
Low	0.0 - 0.3
Medium	0.3 - 0.7
High	0.7 - 1.0

Figure 2.6 - Risk probability ranges

Level of Consequence	Range
Low	0.0 - 3.0
Medium	3.0 - 7.0
High	7.0 - 9.0
Very high	9.0 -10.0

Figure 2.7 - Risk impact ranges

The methods used for the risk analysis are similar to the ones used for risk identification such as process review meetings and personal judgment from project managers and other relevant parties.
The product of the probability Prob(R) and the level of consequence or loss Loss(R) gives the risk exposure unit RE which is then used to prioritise the risks. The formula is described by Jalote [23, p.96] and is given below.

$$RE\ (R) = Prob(R) \times Loss(R)$$

At the end of this process, the risks with the higher risk exposure are selected for mitigation and project control.

2.4.3.2 Risk Control

Risk control involves first of all, the identification and planning of steps that are required to mitigate the impact of each risk item on the project. Similar methods to the ones used in risk assessment can be employed. The team can also refer to common lists of mitigation steps used in similar projects in the past [23].

The next step in risk control exercise is the risk resolution which includes taking actions so that desired conditions are achieved where each risk item is mitigated or even eliminated. This basically involves implementing the actions from the previous risk management planning exercise.
The last element of risk control refers to the reassessment of risk perception throughout the life of a software project. The first risk assessment and control are done at the very beginning during the project planning [23]. Because risks are as mentioned, probabilistic events, the perception of them is changing over time and is based on external factors which might not be present or anticipated at this specific moment. Continuous monitoring is thus necessary and in practise this is done by revising the risk management plan in each project phase.

As advised above, the first risk management exercise and planning for the MSA was done in the project planning phase. Meetings with the client, review of the current processes, an initial analysis of project requirements and the personal judgement of the developer of the project were used to identify risks and elicit a plan for risk resolution. The initial risk management plan along with the updated versions after each spiral iteration are included in Appendix B.

2.5 Desktop vs Web applications

Two types of applications were considered for the MSA, the desktop and the web applications. The two types and the differences between them are discussed by Bychkov [25].

With the desktop type the produced software would be installed on the users' computers. Any updates to the system would also require access to the same machines. The hardware and the operating system

would affect implementation details of the application. The system would normally have quick responses as most of the resources would be local to the application.

On the other hand, a web application offers its features over the internet [26]. The application is installed on a server machine and responds to requests from client applications such as web browsers or mobile applications. The communication is done using the http protocol (Fielding *et al.* [27]).
A disadvantage of web application has sometimes been slower response times, however this has been improved greatly the recent years with high broadband speeds. In addition, a web application offers the capability of accessing the system from almost everywhere.

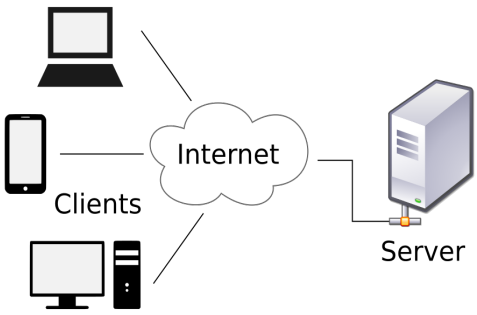


Figure 2.8 - Client server model [28]

After discussion with the client, it was confirmed that accessibility is important and for that reason the Meeting Scheduler application was decided to be a web application.

2.6 Patterns

Patterns in software engineering are reusable solutions to common, similar problems in design and implementation contexts. Using patterns increases productivity as we do not have to reinvent the wheel but instead apply a proven and previously tested solution to problems. Also it increases code readability as developers and software engineers are familiar with a pattern design and the terminology that is used. The main criticism for patterns is that it might lead to insufficient solutions as unique and well factored solutions might be a better option for unique problems. A more detailed discussion about the advantages and disadvantages of design patterns is available on the sourcemaking website [29].

Patterns can be applied in various areas including enterprise applications architecture [30], system integration and messaging [31] as well as lower level design to assist with common tasks like creation of new components, structuring them to obtain new functionality and the communication between them [32].

Various patterns were considered for the MSA application including the Model View Controller, the Front Controller, the Data Transfer Object, the Repository and the ServiceLocator (which was discarded in favour of a Dependency Injection framework) and they are discussed in further detail in subsequent sections.

2.7 Model View Controller

The Model View Controller (MVC) pattern is used to structure the code so that the business logic of the application is decoupled from how any calculations are presented to the user [30, p.330]. The diagram below shows the flow of data in an MVC application.

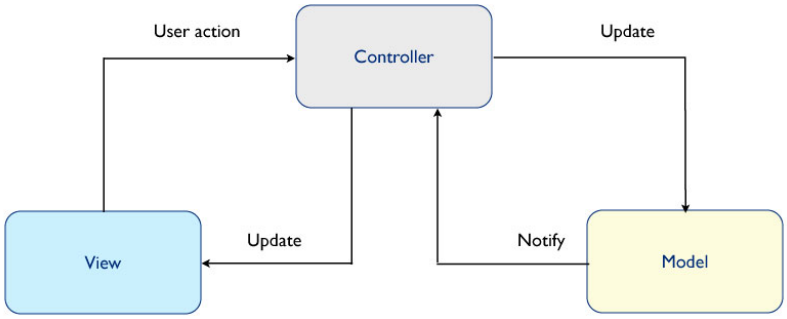


Figure 2.9 - Flow of data in an MVC application (from teamtreehouse website [33])

For the MSA application the MVC pattern was applied using the Spring MVC framework and the decision was influenced by the programming language that was chosen for the implementation of the system. More details on the considered options are provided in sections 2.9 (Programming languages section and MVC options) and 2.9.1 (Java MVCs).

2.8 Libraries and Frameworks

Two terms that are constantly mentioned in this report and which correspond to software components that were considered and used for the MSA application are libraries and frameworks.
Software libraries is code which provides functionality through an API [34]. Other applications can call this API to extend their own functionality. The main advantage is code reusability. For the MSA including a dependency to the Google Calendar API enables the code to integrate with functions such as querying calendar availability information and setting calendar events. The functionality is not implemented by the application but is offered through the calendar library.
Frameworks are similar to libraries in the sense that they offer functionality. They are often used to address non-functional concerns like security (see Spring Security in section 2.12), modularisation and application design (e.g. using an MVC framework as discussed in the following section), abstractions over the data layer (see Hibernate and Spring Data frameworks in section 2.13.3) and dependency injection (e.g. using the core Spring framework, discussed in section 2.10.2.1). They usually dictate the architecture of the application and provide callbacks and interfaces which the system can implement and built on. The framework would then call these implementations at appropriate times during the programme execution. This process is called 'Inversion of Control' [35] because the program does not call the functions as it does with the libraries, but instead the framework calls the implementations provided by the program. For that reason, it is also known as the Hollywood Principle, "Don't call us, we'll call you" [36].

2.9 Programming languages and MVC options

Another important decision to take was to choose the programming language and supporting web MVC frameworks to build the MSA application. There were many suitable choices.
The first option is the PHP [37] language, which according to w3techs.com [38] powers most of the internet websites today. There is a number of PHP web frameworks available which implement the MVC pattern [39].
JavaScript [40] is the language that is the most popular amongst developers according to a 2018 survey by the stackoverflow website [41]. Server side applications are built to run on Node.js [42] which is a runtime environment with a JavaScript engine. AngularJS [43] by Google and React [44] by Facebook are two popular frameworks which offer MVC capabilities and assist with the application structure.
Python [45] is a language with user friendly syntax, which makes it a popular choice for beginners in web programming. The Django [46] framework can be used for building web applications with Python.
Ruby [47] is another very popular language with large community support. The MVC framework used for web applications is the Ruby on Rails [48].
Java [49] is a general purpose language with a large range of libraries available. It has been established over the last twenty years and has been very popular due to its "Write once, Run everywhere" capability [50]. A more detailed discussion about Java MVC options is offered below.
ASP.NET [51] is an open source project which was developed by Microsoft and offers an MVC implementation to use for building websites. It requires a Windows server to run the application, which reduces the portability of the application. Also, at the time of this investigation it is not known what platform would be hosting the MSA system. For these reasons this option was discarded.

Although most of the above technologies (except ASP.NET) are potentially suitable for implementing the MSA application, time constraints did not allow for a more in-depth assessment of each of them. Instead, Java was chosen based on the fact that the developer was already familiar with the technology.

2.9.1 Java MVCs

Some popular MVC solutions that are available to use with a Java based application and which were considered for the MSA are described below.
The first one considered is a combination of the Servlet and the JSP technologies. JSP [52] which implements the View part of the MVC belongs to the J2EE [53] collection of libraries. Through the Expression Language [54] and JSTL [55] technologies, it offers ways to plugin dynamic content from the Model. The Controller part is implemented with the Servlets [56] technology. It works well with smaller applications but results in complex deployment descriptors which are harder to maintain as the application size increases. Each servlet is instantiated once and can serve concurrent threads. This increases the chance for errors and requires extra developer effort to ensure thread safety.

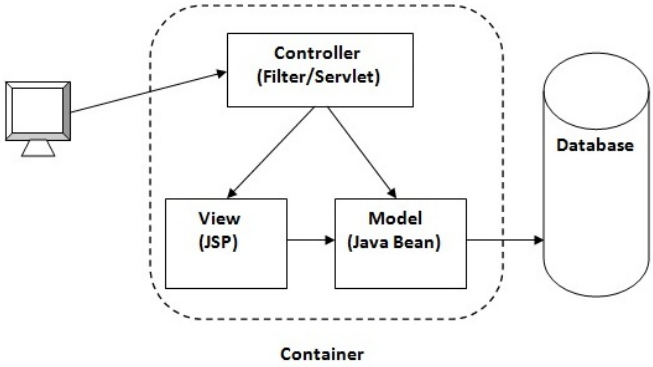


Figure 2.10 - MVC implemented with Servlet and JSP (from tutorialpoint.com [57])

Struts is another MVC technology which was considered. It is a special type of MVC which implements the Front Controller pattern, but is still applied as a way to separate the View from the Model. The advantage is that all requests are sent to a single controller, which allows to abstract common tasks [58]. Specific handling is done by setting dispatchers in a configuration file (struts.xml). It also has the advantage of initiating a new thread per request, so there is no need for synchronisation. JSP technology is most often used to implement the View.

JSF [59] is an MVC framework for building user interface components and is more suitable for larger applications. It provides functionality to easily reuse UI components, facilitate data transfer between them, manage state between http requests, assist with validation and event handling. Although JSP can still be used, Facelets [60] is the preferred option for the View implementation. As a fully fledged framework it has a steep learning curve.

Spring MVC [61] is another Front Controller implementation. It has been a leading choice for a Java MVC framework over the last 10 years in the industry [62]. XML files or annotations can be used to map requests to controller classes. The latter improves code maintainability as the mapping remains local to the controllers. Domain objects are bound to the controllers using dependency injection (discussed in the following section). This reduces coupling and makes it easier to test these components. Spring MVC offers a choice of View template technologies including JSP, FreeMarker [63], Velocity [64] and Thymeleaf [65].

All the considered MVCs above could be employed for the MSA application. However, the main options that were considered were the JSP/Servlets MVC solution and the Spring MVC because the developer had already experience with these two. The first would work really well with a small size app like the MSA, but at the end the Spring MVC was chosen because of the additional benefits of the Spring framework, especially in security and dependency injection (discussed in subsections below).
Thymeleaf was chosen for the View implementation because of its *natural templating* characteristics [65]. The Thymeleaf files look and work like html, which means they could easily be used for prototyping.

2.10 Application Architecture

Two very popular architectures were considered for the organisation and structure of the application code. These consist of the layered and the onion architectures and they are described below.

2.10.1 Layered Architecture

The layered architecture [66] is the more traditional approach. The application code is split in layers with the most common design containing the web, the business model (sometimes this layer is split to a service and model layers) and the data layer. Each layer depends on the following and the next is agnostic of the previous one. There are strict layering and flexible layering versions with the latter allowing code from one layer to access any of the layers below. In contrast the strict layering version allows access only to the layer below.
The main advantage of this architecture is that it allows for more modularised solutions. That means that reusability increases as a module can be used in other applications. It also increases maintainability and prolongs the life of the application as a module can be upgraded or replaced with a better option.
The following diagram illustrates the architecture with the dependencies between layers.

Figure 2.11 - Layered architecture

2.10.2 Onion architecture

The biggest drawback of the layered architecture is that modularisation of the application comes at the cost of higher coupling between the modules. Every high level layer depends on low level layers and these dependencies counterbalance the benefits of modularisation.

Palermo [67], [68], [69] describes an architectural design which can be employed to overcome these drawbacks. He named that design an onion architecture. The main principle behind the onion architecture is that code can depend on inwards layers but cannot depend on any outer ones. All coupling is towards the centre where the domain model resides (figure 2.12). The domain layer itself has no external dependencies. That means that the business logic can be abstracted out to be reused in other use cases with minimal effort.

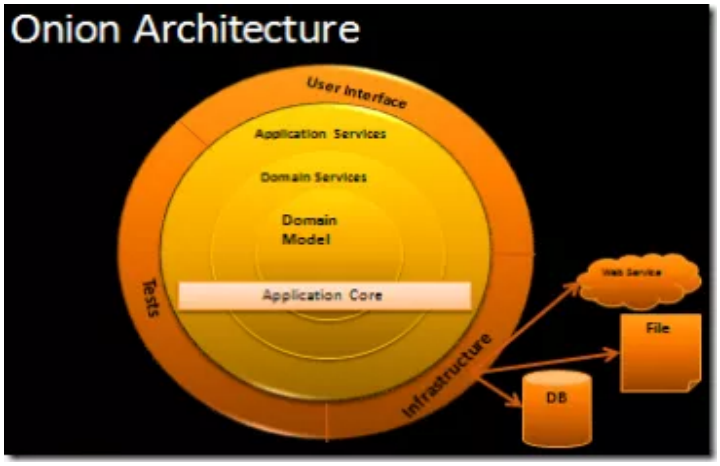


Figure 2.12 - Onion architecture [67]

The difficult part to understand about this design is how it performs operations such as storing data to the database if there is no coupling between the business and data layers. This is actually achieved by applying the Dependency Inversion Principle (DIP) [70]. The principle states:

"High-level modules should not depend on low-level modules. Both should depend on abstractions. Abstractions should not depend on details. Details should depend on abstractions."

Applying the DIP principle to the layered architecture figure diagram inverts the dependencies. This is indicated by the arrow pointing from the data to the business domain layer. Both the business domain (high level) and the data layer (low level) now depend on interfaces (abstractions).

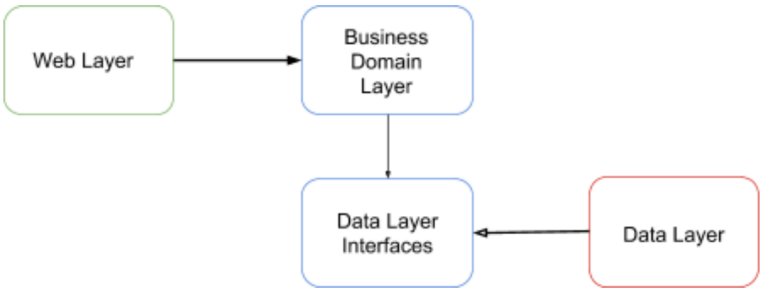
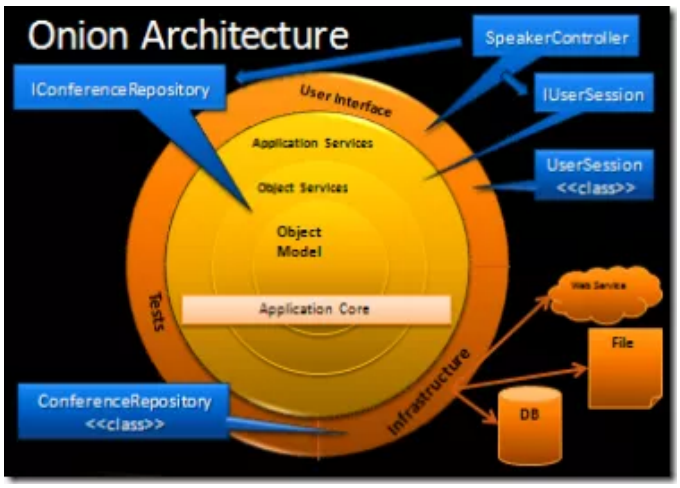


Figure 2.13 - Inverted dependencies after applying the DIP principle

The above technique is used throughout the onion architecture to achieve the decoupling of components and layers. An updated diagram from Palermo's site [69] illustrates this.



The diagram shows the object services (which is part of the core domain) having access to the IConferenceRepository class because they are in the same layer. However, the actual ConferenceRepository implementation is residing in the outer infrastructure layer and the domain has no dependency on it.

Figure 2.14 - Onion architecture showing location of components

2.10.2.1 Service Locator and Dependency Injection frameworks

One thing that needs further clarification with the onion architecture and the application of the Dependency Inversion Principle is the way that the implementation is injected into the abstraction at runtime. This is done using a Service Locator or Dependency Injection (DI) [71] which involves another component (assembler) injecting the class. Figure 2.15 below illustrates this action.

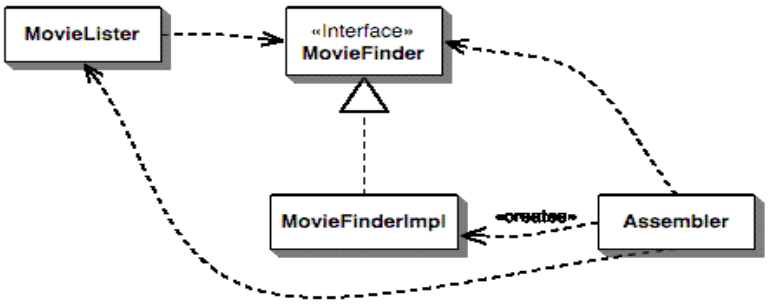


Figure 2.15 - Dependency injection (by Fowler [72])

Service locator is an object which is aware of all the services that an application might need. Other components which use the service locator, would just call its methods to plugin the right implementation service at runtime. The drawback of the service locator pattern is that it adds an extra dependency to components through object composition.

The alternative option is to use a dependency injection framework. The three forms of dependency injection are described by Fowler [72] and include the constructor injection, the setter injection and the interface injection. PicoContainer [73] and Spring [74] are popular frameworks which can be employed for dependency injection.

2.10.2.2 MSA chosen architecture and dependency injection method

For the MSA, the onion architecture was chosen as the preferred application architecture for the decoupling, modularisation and reusability advantages described above. For the dependency injection needs, the Spring framework was employed as it was a technology already familiar to the developer.

2.11 Spring Boot

Spring Boot [75] is a framework provided as part of the Spring framework architecture and it assists with the quick bootstrapping and developing of Java Spring applications. It enhances productivity by reducing boilerplate code, including default dependencies and following an "Opinionated Defaults Configuration". This enables it to provide defaults for most of the implementation and testing needs of a web application.
Spring Boot was employed for the implementation of the MSA application. Some of the default functionality that was utilised to assist with the project include the embedded tomcat container and the H2 in-memory database which was used during development (later replaced with MySQL).
Functionality like the Spring MVC, Spring Security, Spring Data were all included easily using Spring Boot starter dependencies which contain these technologies as children dependencies.
Dependencies for unit and integration testing are also included as a default.
One extra benefit is the easy creation of an executable jar artefact which contains all the required libraries for the application to run, including the embedded container. More details about how Spring Boot loads all the dependencies in the executable jar can be found in the Spring Boot documentation [76]. The jar can then be run autonomously in any platform which has an installed JVM runtime.

2.12 Security considerations - Login and OWASP

For the security needs of the MSA application, the login functionality and the possible implementation options were first considered. The first option was this feature to be implemented from scratch by the MSA developer. The advantage with this is that there would be complete control of how it is implemented and would be fine-tuned to fit the MSA unique needs. The disadvantages are more time to implement it and that there is a greater risk for the functionality to break or be compromised by an

attack as it would not be as robust as a tested off-the-shelf solution. As there was no special login functionality required for the MSA, this option was discarded.

The next option considered was to use the tomcat realms concept [77] and its implementation of contained managed security from the Servlet Specification [78]. The realm corresponds to a store of usernames and passwords (which could be a relational database) which the application is using to authenticate valid users. In addition, the implementation allows to define a list of roles against each valid user in order to enable authorisation of specific actions (the blog by doubleoctupus.com [79] describes the difference between authentication and authorisation). The decision to develop the application using Spring Boot and an embedded tomcat container made this solution less attractive as no documentation was found of how to configure the embedded server to implement the above.

A third choice is offered by the Spring Security framework [80]. After experimenting of how it can be employed for a web application (following tutorials by baeldung [81] site), it proved to be intuitive and easy to setup calling configuration methods from within the code. In addition, it offers off-the-shelf functionality to encrypt the user password during registration and authentication [82] which were requirement for the MSA system. The choice was further promoted by the fact that Spring was already selected to be used for other implementation requirements like MVC and dependency injection, and that meant an easier integration between components. As expected it also worked well with Spring Boot [83].

Spring Security was thus selected for implementing the authentication requirements of the MSA application.

2.12.1 OWASP Top 10 Application Security Risks – 2017

The OWASP list with the most critical security risks for 2017 [84] and how these could be avoided for the MSA application was also considered. These are listed below.

(i) SQL Injection

This is related to attacks which corrupt the database when running SQL queries or commands with unchecked input that is entered through the UI. The MSA would use Spring Data (see section 2.13.3.1 below) which prevents this security risk.

(ii) Broken Authentication

As mentioned earlier, Spring Security would be used to implement the login functionality. The technology is established, proven in the industry and expected to offer a robust and secure solution.

(iii) Sensitive Data Exposure

The MSA would not store any personal data except from Faculty Officers' email addresses and their roles in the University, which are already available on various University sites.
Passwords used during registration and authentication would be encrypted by the system.

(iv) XML External Entities (XXE)

This risk is concerned with the processing of hostile content within xml data documents which are usually retrieved from integration with external applications. The MSA application is only integrating with the Google Calendar API which is considered a trusted application. In addition, the JSON format would be used for retrieving data from the API.

(v) Broken Access Control

This refers to authorisation of the users to perform certain actions only. The MSA requirements do not specify different levels of authorisation, so this security risk is not applied.

(vi) Security Misconfiguration

This refers to misconfiguration of frameworks, libraries and technologies used in the system. Care would be taken to study and follow the documentation offered by these components to ensure their

correct and secure application. In the future, a follow up project or work would be required for updates and patches for the used tools and technologies.

(vii) Cross-Site Scripting (XSS)

This is dealt with using a whitelist of allowed characters and validation in all MSA forms.

(viii) Insecure Deserialization

Refers to the deserialisation of objects which might be supplied by an attacker. As mentioned earlier the MSA accepts data from the Google Calendar API which is a trusted source.

(ix) Using Components with Known Vulnerabilities

Proven, tested technologies would be considered to be used for the MSA application. Their documentation would be studied for known vulnerabilities. Any considered newer technology would be thoroughly tested.

(x) Insufficient Logging & Monitoring

Logging and monitoring have not been identified as non-functional requirements for the initial MVP product. These could be included for the next iteration of the MSA in a follow up project.

2.13 Databases

There are two database types that are more popular for web development, the relational and the NoSQL databases.

2.13.1 Relational databases

The relational database was first introduced as a concept by Codd [85] in 1970. Since then this database type has been established as the default choice for most types of applications. The relational model structures the data in tables, rows and columns and means it works well with relational data which is well structured. As a technology which has been used extensively over several decades, it is mature, offers proven value and it is well documented. Relational databases support the ACID (Atomicity, Consistency, Isolation, Durability) principles [86] which guarantee integrity of data.
The SQL (Structured Query Language) [87] language is used to retrieve and update data from a relational database.
According to DB-Engines ranking [88] the most popular relational databases in July 2018 are in order, the Oracle [89], MySQL [90], Microsoft SQL Server [91] and the PostgreSQL [92].

2.13.2 NoSQL databases

As mentioned above relational databases require the data to be well defined and structured in a rigid schema. When the data is unstructured and untyped there is a need for a different means of storage. In these cases, NoSQL database can be used as they are schema agnostic. They are faster and easier to configure and maintain. They are also more suitable to store binary data like images and for applications that their data do not require referential integrity and transaction guarantees. Because they do not guarantee these properties they are also easier to scale by adding more instances of the NoSQL databases (also known as horizontal scaling).
The NoSQL databases are further categorised in other forms. Key-value stores [93] are used for storing simple key-value pairs and are very fast. Redis [94] and Amazon DynamoDB [95] belong to this type.
Wide column stores [96] are used to store data in records with the ability to add a large number of dynamic columns. Examples of this type are Cassandra [97] and HBase [98].
Document stores [99] like MongoDB [100] and Couchbase [101] are schema agnostic and are used to store documents in JSON format.

Graph databases [102] can store and facilitate processing of data in graph form. Neo4j [103] is an example of a graph database.

2.13.3 MSA database solution and integration methods

The MySQL relational database was favoured for the MSA database as the data is well structured and the developer was already familiar with the relational model and in particular with MySQL.

2.13.3.1 JDBC, ORMs and Spring Data

The way that the application code would interact with the database was the subject of further investigation. The first option is to simply use the MySQL JDBC driver to communicate with the database. The technique is explained on the MySQL site where code examples are also provided [104] The drawback of using the MySQL JDBC driver is the boilerplate code that comes with it. In addition, it requires the developer to use database specific SQL directly in the application code.

Another solution is to use an ORM (Object Relational Mapping) [105] technology which allows to produce application code which is agnostic of the underlying database. It does that by implementing the Repository [106] pattern which isolates data access behind interface abstractions. It also provides API methods to enable CRUD operations without the need to use SQL code at all within the application. Other benefits include internal caching for faster database interactions.

JPA (Java Persistence API) [107] is the official Oracle specification for a Java specific ORM solution and the main implementation is the Hibernate [108] framework.
Another ORM alternative is MyBatis [109] (renamed from iBatis [110] after being moved away from the Apache Software Foundation [111] umbrella of software projects). It does not follow the JPA standard, however it is seemingly more lightweight and easier to use [112], [113], [114]. MyBatis decouples the database mappings from the application code by using direct SQL in separate XML configuration files. While this provides more visibility over Hibernate's way of underlying SQL generation, it also removes the benefit of a database-agnostic application.

While both Hibernate and MyBatis reduce repeated code, there is still effort required to create the mappings either in the code [115] or in configuration files [116]. Spring Data [117] is a framework that reduces further the developers' effort by providing CRUD interfaces for database integration. The developer only needs to extend these interfaces with her own entity specific interfaces and the framework creates the implementation automatically at runtime. Spring Data is built on top of a JPA implementation which means that Hibernate or another JPA compliant technology is still used underneath.

Taking into account the benefits of a database agnostic application and the savings in time and effort, Spring Data was chosen as the best solution to integrate with the database.

2.14 Other tools and technologies used

Several other technologies were considered and used for the MSA. These are outlined below.

2.14.1 HTML5 and CSS

HTML5 (HyperText Markup Language) [118] is the latest version of the declarative markup language that is used to render website pages on browsers and web applications. Its features were used in the MSA for form validation and for better rendering of special input types like dates, times, emails and number ranges.

2.14.2 CSS and Bootstrap

CSS (Cascading Style Sheets) [119] is a language used to describe the presentation of HTML and XML documents. This way, it allows the separation of the content (HTML) from its presentation (CSS), which assists with code clarity and reusability. The technology was utilised for the MSA along with the Bootstrap CSS framework [120] to design and build a responsive design which would work in various devices and sizes e.g. in mobiles, tablets, laptops and desktops.

2.14.3 JavaScript, Ajax and jQuery

JavaScript [121] consists the third core language together with HTML and CSS which is used to create web pages. It can be utilised in various ways including adding interactivity and event responses, form validation and content manipulation among others and is supported natively by most major Web browsers.
JQuery [122] is a popular JavaScript library which offers an API that simplifies the use of the above functionality. It has been employed in MSA to send asynchronous Ajax [123] requests to the server, for example when selecting a value on a dropdown field. It has also been used to send POST requests from http links, while the DataTables [124] jQuery plugin was utilised for the presentation of entity lists in table form on the MSA pages.

2.14.4 Maven

The MSA utilises functionality from a large number of libraries including the tomcat container, the Google Calendar API, the Thymeleaf template technology, various Spring libraries, the JUnit test library and others. The management of these dependencies require a dependency management software with the most popular choices for a Java application being the Maven [125] and the Gradle [126] build tools. As the student had previous experience with Maven, it was chosen for the MSA dependency management needs.

2.14.5 Git

Git [127] is a Version Control System (VCS) which is often used by teams to enable developers to work on the same applications files remotely and then merge them together and resolve any conflicts afterwards.
For the MSA, there was only one developer, however Git provided a valuable tool for backing up and recording changes to application files for later revision. The student was also able to create branches of the project to easily try new technologies and test various implementation methods before applying them on the main working branch.

2.14.6 IntelliJ IDE

The most popular development environments for Java applications are the NetBeans [128], IntelliJ IDEA [129] and Eclipse [130] IDEs. Again, due to reasons of familiarity with the tool, IntelliJ was selected for the MSA development. The tool provides an easy integration with Maven and JUnit, instant compiling error detection and correction, shortcuts for code refactorings like renaming class members and many others which assist with development and increase productivity. IntelliJ is available in both a Community Edition and a paid Ultimate Edition (the latter was employed for the MSA as is free for student projects).

2.14.7 Astah modelling tool

The Astah [131] diagramming tool was used for the UML and data modelling needs of the Meeting Scheduler application, as it has been used extensively for previous projects in the MSc course and was readily available for the student. The software is proprietary, however a previously acquired student licence allowed to be used for the project.

2.15 Summary

The chapter compared techniques and technologies in many areas of software engineering. The chosen options used for the Meeting Scheduler were described along with the reasons for their selection. As it is dictated by the risk management plan, the considered alternatives were also needed as a reference in case the original tool or technique selection has not lived up to expectations.
The bigger part of the research was completed before the core project work begun, but it was enhanced along each spiral iteration. The next chapters will offer a lot more detail of how the discussed options were applied in practise during the project.

3. REQUIREMENTS AND ANALYSIS

In this chapter the different techniques used to elicit the requirements for the Meeting Scheduler application are discussed and the extracted functional and non-functional requirements are outlined. The functional requirements are further modelled in the form of use case diagrams. These were used to confirm the requirements with the client as well as a form of specification to drive the design and implementation in later stages.
The produced artefacts for this stage were created incrementally as part of the work done in each of the spiral iterations (see MSA project methodology in section 2.4.1.6).

3.1 Types of Users

As part of the requirements process engineering, it was necessary to identify the types of users within the domain of the system. Doing that ensured fewer requirements were missed out and the most suitable technique for requirements elicitation was applied to each user category. Sharp [132] describes four different 'baseline' stakeholders which could affect decisions of how the system is implemented. These consist of the '*users'*, '*developers'*, '*legislators'* and '*decision-makers*'.
'*Users'* have been categorised and named in more detail by Eason [133]. The three main categories consist of '*primary'* which are the ones directly using the system, '*secondary'* who use the system less frequently and '*tertiary'* who are the ones affected by the system introduction in the current business process.

For the meeting scheduler application, the '*primary'* user is the main SSS admin staff who deals with the unsatisfactory progress procedures. Other SSS admin staff who would use the application less frequently are '*secondary'* users. The Faculty Officers would be '*tertiary'* at the launch of the application, but this could change if their involvement becomes more active in the future (see Chapter 6 for future system expansions).

The student undertaking this project is the only '*developer'* of the system at this point of time.

The '*legislators'* are the professional bodies, government agencies, legal representatives and policy makers. For the current system *legislators* could potentially be the University Research Ethics Committee (UREC) and the technical department of the University (CiCS). These respectively, could affect the way personal data is handled and whether the system is secure enough to be hosted by the University servers.

'*Decision makers'* are structures within the user and developer organisations, most often management which makes decisions which affect the system. For the current application, the main decision maker role is that of the project supervisor.

3.2 Requirements gathering techniques

3.2.1 Interviews

An interview is the most widely used method of gathering information. It is a process where software engineers come into contact with stakeholders, mostly face-to-face, and through a set of questions and discussions they collect project requirements.

The main advantage of the interviewing technique is the ability to "go deep" [134, p.188]. This means the interviewer can achieve a better understanding of the requirements and discover perspectives of the business domain which might be missed using other methods. The conversation is flexible, allows for the order of the questions to change and new ideas can be explored directly while conducting the interview.

On the other hand, an interview is generally a difficult process and requires experience to manage the conversation. An unskilled interviewer might not follow up on important details or as Sutcliffe [135] points out the interviewees might not be prompt in revealing their 'tacit' knowledge**.** In addition, the method is time-consuming and applicable to a small number of stakeholders.

For the meeting scheduler application, the interview technique was first used to elicit requirements from a primary stakeholder, the SSS admin staff using the system. As a domain expert [136] she provided the main source for the system requirements. It was then employed with one of the Faculty Officers who consists a tertiary stakeholder of the system (see previous section for the identified system stakeholders) in order to understand how the software would impact their way of working.
Interviewing different categories of users assisted in getting a better insight of the problem domain. The interviews were conducted in their working environment which put users at their ease [135] and contributed to better results.

3.2.2 Questionnaires

Another common technique to use is questionnaires. A questionnaire is a document with a predefined set of objective questions and corresponding answers in order to understand the project scope and requirements. It is distributed to stakeholders in the business domain and their answers are collected and drafted [137].
The questionnaires technique is a valuable tool to elicit *'shallow data'* [134, p. 106] but is difficult to utilise for retrieving detailed information. As a process, has low cost and is easy to carry out. As Garcia, Pacheco and Sumano [138] highlight, consideration should be given that a questionnaire is well written and its questions are not misinterpreted by the participants. Personal bias could also affect the answers and in contrast to interviews, this phenomenon is harder to trace.

The questionnaire technique was employed for the Meeting Scheduler application, mainly to gather requirements from staff members in the SSS office who were not able to be contacted for an interview. The answers are provided in Appendix N.

3.2.3 Task Analysis

After collecting information using the interview and questionnaire techniques, it was deemed necessary to perform a small analysis to record and group the requirements that had been identified. This can be done using 'task analysis'. Task analysis is the process of analysing the way in which people perform their jobs, things they do and they act on, as well as things they need to know.

A simple method to perform a task analysis is described by Hanington and Martin [139] where sticky notes are used to register and group together relative task actions. As already mentioned the data was fed into that process from the interviews and questionnaire and the output provided the first use cases for the Meeting Schedule Application (MSA). These use cases were further documented using use case diagrams, described in detail in section 3.3.3.

After identifying some initial use cases a more involved method of task analysis was employed, the Hierarchical Task Analysis (HTA) [140]. In HTA the tasks are broken down in subtasks and then sub-subtasks and so on. They are then grouped together as plans which specify how the tasks might be performed together in practice. The plans are documented either as hierarchical steps in cards or illustrated in diagrams. An example card for booking a meeting is shown in figure 3.1.

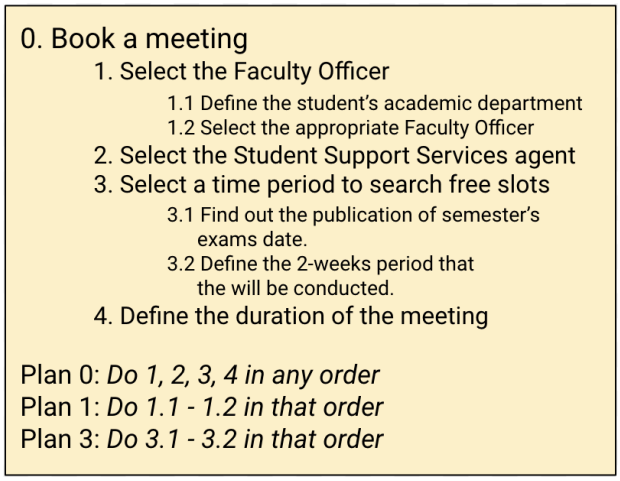


Figure 3.1 - HTA card for booking an Unsatisfactory Progress meeting

3.2.4 Domain Analysis

Domain analysis is a term first introduced by Neighbors [141] in the early 1980s as "the activity of identifying the objects and operations of a class of similar systems in a particular problem domain". Since then the technique was used successfully in many software project domains, some of which are described in Prieto-Diaz's paper [142].
However, in recent years and especially with the evolution of the Domain Driven Design [143], the term has been used in a broader sense to mean analysing the business domain and not strictly the domain of already implemented software systems.
More specifically in the requirements engineering phase, the method means the process by which domain experts are getting involved to analyse the business requirements. A developer of a system could achieve a certain level of domain expertise, but most often it is the experts within the client organisation which this term refers to.
For the MSA, the Hierarchical Task Analysis plan cards were used to discuss further with the client the business requirements. The discussions confirmed and amended these requirements.
At the end of this process, a conceptual class diagram [144] was also produced and is available in Appendix H.

3.2.5 Prototyping

As explained in the literature review, the MSA project followed a mix of the spiral and evolutionary prototyping methodologies. After employing the previous requirement elicitation techniques, an initial prototype was built to communicate and demonstrate to the user the functionality of the system. The prototype at this stage did not have any implemented functionality but allowed for quick user feedback, which was then used to amend requirements and steer the implementation.

3.3 Software Requirements

The term "*requirement*" is used in software engineering to describe the services, features and constraints of the system under development. There are two main requirement types, the functional and the non-functional requirements.

3.3.1 Functional Requirements

Robertson and Robertson [145] explain that functional requirements are the features the produced software must implement in order to support specified business needs. They should be, as far as

possible, expressed independently from any technology that would be used to implement them. The business analysis section is not attempting to craft a technological solution, but rather to specify what the technological solution must do. The functional requirements specify the product to be developed, so they must contain sufficient detail for the developer to build the correct product with only the minimum of clarification and explanation.

Based on the results of the requirement gathering techniques, the following functional requirements present the expected features of the system. They are divided in high priority requirements, to provide a minimum viable product (MVP), and in low priority which consist the extended (desirable) functionalities. The MVP concept has been popularized by Eric Ries [146], a consultant and writer on startups, and is a development strategy in which a new product is developed with sufficient characteristics to satisfy early customers.

3.3.1.1 High Priority Requirements

The following table outlines the mandatory features the MSA must have as an MVP product.

ID	*High Priority Functional Requirements*
F1-Login	Users must be able to authenticate themselves with correct credentials. Wrong credentials must fail authentication. The system must provide a login form with username and password. The username would be the user's email
F2-Registration	The users must be able to register other users with the system. Required fields are title, name, email and password
F3-Faculty	Users must be able to view the list of Faculties
F4-Department	Users must be able to view the list of academics Departments
F5-Department	Users must be able to add a new academic Department. The interface should contain a Department name and Faculty which the Department belongs
F6-Department	Users must be able to view and amend a Department's data
F7-Department	Users must be able to delete a Department
F8-FacultyOfficer	Users must be able to view the list of Faculty Officers.
F9-FacultyOfficer	Users must be able to add a Faculty Officer. Required fields are title, name, email, the Department they belong to, office phone number and comments
F10-FacultyOfficer	Users must be able to view and amend a Faculty Officer's data
F11-FacultyOfficer	Users must be able to delete a Faculty Officer
F12-SystemUser	Users must be able to view a list of system users
F13-SystemUser	Users must be able to view and amend another user's details
F14-SystemUser	Users must be able to delete a system user
F15-Room	Users must be able to view the list of meeting rooms
F16-Room	Users must be able to view and amend the details of an specified room

F17-Room	Users must be able to add a meeting room. Required fields are name, location and calendar reference
F18-Room	Users must be able to delete a room
F19-Appointment	Users must be able to book interviews with the Faculty Officer. Required fields are the student's department, Faculty Officer, admin staff, meeting room, a date and time period to search for slots and the duration of the meeting. The booking must be done automatically on the first available time slot
F20-Appointment	Users must be able to redirect to the Google Calendar application and view the created event. The system must provide the appropriate link
F21-Availability	Users must be able to view the availability of a Faculty Officer. The system must allow to specify a date period to search within, as well as the time duration that the Officer must be available.
F22-Logout	Users must be able to log out successfully

Figure 3.2 - High priority functional requirements for the MSA

3.3.1.2 Low Priority Requirements

The following table outlines any desired but not absolutely essential system features for the MVP product.

ID	***Low Priority Functional Requirements***
LF23-FacultyOfficer	Users should be able to specify a *primary* Faculty Officer
LF24-FacultyOfficer	Users should be able to identify the *primary* Faculty Officers within form dropdowns and lists
LF25-Faculty	Users should be able to add a Faculty. The interface should contain a name field
LF26-Faculty	Users should be able to amend a Faculty
LF27-Faculty	Users should be able to delete a Faculty
LF28-Appointment	During the booking appointment, if a user selects a Department, they should only be allowed to choose a Faculty Officer that belongs to that department's Faculty (but not in that Department)
LF29-Appointment	The system should not allow booking in weekend dates
LF30-Availability	The system should not return available calendar slots for the Faculty Officers within weekend dates
LF31-Password	The users should be able to reset their password

Figure 3.3 - Low priority functional requirements for the MSA

3.3.2 Non-Functional Requirements

Sommerville [147] describes non-functional requirements as constraints on the services or functions offered by the system. These do not mainly focus on specific features but they instead apply to the system as whole. Two further sources of information [137], [148] were also consulted to understand the various types of non-functional requirements and extract the ones for the MSA. The relevant types are briefly described below and the full list is provided further down in this section.

The first non-functional type of requirements is concerned with the security of the system. Security requirements describe how the system should perform against software attacks. They also include any required measures for user authentication and/or authorisation to perform specific actions. For the MSA only authentication was required. The OWASP list [84] with the top ten threats for 2017 was also consulted (as discussed in the Literature Review chapter).
Reliability was also required for the MSA. This is the extent to which the application can repeatedly perform its functionality successfully.
Usability refers to how easy it is to access and use a software product or website [149]. According to the ISO 9241-11:2018 standard, usability is defined as "the extent to which a system, product or service can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use". Usability is closely related to the HCI design which is explored in detail in Chapter 4.
Performance refers to the system capability to perform well under the expected workload. This usually focuses on the system response times and its ability to remain stable and functional under high load of requests. For the MSA the performance under load was not a necessary requirement as it is expected that a single user would be interacting with the application at a time.
The maximum response times for the different user actions and page loading are based on industry standards which were defined by Miller [150] and reiterated by Nielsen [151]. The limits are based on the user's perception of an instant response, the ability to maintain her flow of thought and the ability to keep her attention on the task she currently performs on the site.
The capability of the MSA system to be deployed in different software and hardware environments is defined in its portability requirements.
Reusability evaluates the system against how feasible it is to reuse parts of it in other applications.
Scalability refers to the degree in which an application can grow in its capacity to handle growing amounts of data and processing requirements. Considering the very small number of users for the MSA, this requirement has been de-prioritised and would not be implemented for the MVP product.
The project cost allowances include both development and operational costs for the software system and is another type of non-functional requirement.
Finally, any special configurations required to deploy and run the system are defined by its system configuration requirements.

The table below summarises the derived non-functional requirements for the Meeting Scheduler application.

Security-1	The user must be prevented from accessing pages of the MSA without authentication
Security-2	All authenticated users must be able to access any page and perform any action. There is no requirement identified for authorisation of the user
Security-3	The system must take measures against Cross Site Scripting (XSS) attacks
Reliability-1	The system must be able to repeatedly book an appointment with any of the Faculty Officers, SSS admin staff and rooms - if there is availability
Reliability-2	The system must be able repeatedly to enable a user to login and logout

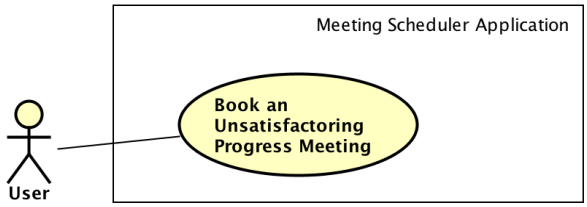
Reliability-3	The system must be able repeatedly to enable the user to update any of the system data e.g. a Faculty Officer
Usability.1	The system must be consistent with colours across the whole site
Usability-2	The system must be consistent with the location of input elements such as buttons and links. Eg buttons at the bottom middle of the html forms
Usability-3	The system must provide sufficient feedback to the user as a response to a user action eg by showing a dialogue confirming the update of Faculty Officer entity
Usability-4	The system must validate all user input that is entered in the html forms. Appropriate messages would be displayed to provide user feedback
Usability-5	The system must offer predefined values for selection where possible in html forms eg using a dropdown with populated departments to select, instead of a free textbox
Usability-6	The system must offer a confirmation dialogue before performing destructive user action eg when deleting a meeting room entity
Usability-7	The system must use terminology from the client's domain. This includes terms used in form fields, buttons, menu elements and dialogue feedback messages
Usability-8	The system must follow HCI principles for clear and aesthetic design. The pages should not be convoluted with information, they should indicate the next steps in user journeys, provide efficient navigation, allow searching and sorting of entity lists
Performance-1	Command pages such as updating a specific entity in the system's database should respond in less than 0.1 seconds
Performance-2	Query pages interacting only with the system database eg retrieving the list of Departments should respond in less than 0.3 seconds
Performance-3	Command pages which interact with external systems should take less than 0.5 seconds to respond eg booking an appointment using the Google Calendar api
Performance-4	Query pages which interact with external systems should take less than 1 seconds to respond eg when retrieving the list of free slots from the Google Calendar of a Faculty Officer
Portability-1	The system must be developed in a programming language which is portable and can be run in different software platforms. This is because deployment details will not be finalised until the end or at least the later stages of the project
Portability-2	The system must use a database which is deployable in any software platform (for the same reasons as Portability-1)
Reusability-1	The architecture of the application code will allow for a modular implementation. That means for example abstracting the business logic in one layer which could be reused in another system
Reusability-2	The application code should be database agnostic. It should allow to replace the database with minimal configuration effort and no changes in the code

Costs-1	The development costs must be zero. Any development work is required as part of the Masters degree undertaken by the developer. The tools and technologies used in any of the project phases would be open source (with a licence that allows their use for the project)
Costs-2	The operational costs must be covered by the University. The MSA should be deployed on University hosts and any technical support should be provided by the University. No separate domain purchase is required
System-Configuration-1	A SQL import script is required to set the initial data state for the MSA

Figure 3.4 - Non-functional requirements for the MSA

3.3.3 Use Case Diagrams

Another artefact of the requirements analysis phase of the project are the use case diagrams [152, pp. 639 - 652]. As already mentioned they are used to model the requirements to present to stakeholders as well as to use as a reference for later project stages. Use case diagrams can be used to model both business processes and functional requirements [153]. Business analysts write business use cases to model the business processes while developers write system use cases to model the functional requirements. The first have the business as subject while the latter have the system as subject.



The simple business use case diagram in figure 3.5 illustrates the main business requirement for this project.

Figure 3.5 - Business use case diagram for the MSA

Cockburn [154] describes three main levels of system use cases. From high to low these are the '*Summary*', '*User Goal*' and '*Subfunction*' levels.
The *Summary* use case diagrams outlines the logical contexts for the functional requirements. They offer a high-level view of what the system can do. Figure 3.6 shows the corresponding diagram for the MSA.

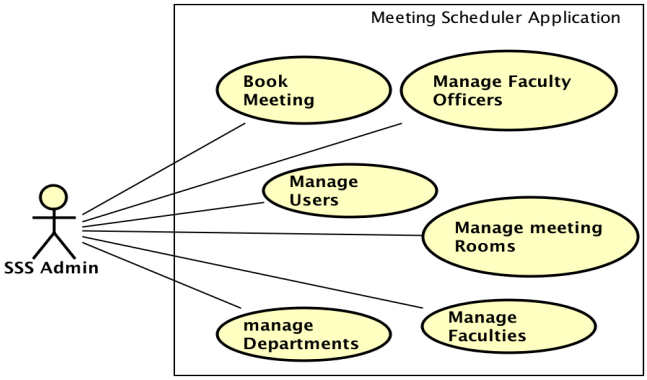


Figure 3.6 - Summary use case diagram for the MSA.

User Goal use case diagrams are modelling the actual functional requirements. The represent value for the user. Kettenis [153] describes that these types of diagrams address the question "Can the primary actor go away happily after the use case finished?".

The *Summary* use cases normally consist of a set of *User Goal* cases. This is shown below (figure 3.7) for the Manage Faculty Officers use case in the MSA.

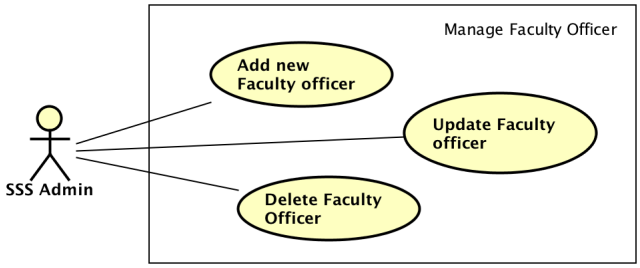


Figure 3.7 - User Goal use case diagram for managing the Faculty Officer in MSA

The *Subfunction* use case is used to model isolated parts of system features. They are moved out of the *User Goals* cases either for reusability or readability reasons. They can offer views and modelling of lower level system functions which are more useful to developers than business users.

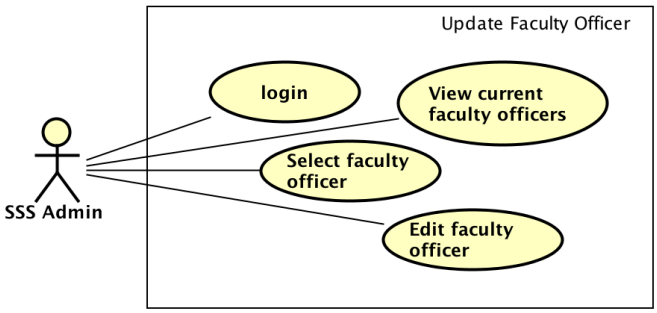


Figure 3.8 - Subfunction use case for updating the Faculty Officer in the MSA

It should be noted that the use of UML in the project was not meant to be exhaustive and cover every functionality and aspect of the system. The diagrams were produced only when the offered value outweighed the effort. For example, not all MSA use cases were modelled with use case diagrams. Once the modelling had been done for the Faculty Officer management, the same modelling was utilised for the other entities' management like Faculty, Booking Room etc (as the same functionality was required). Use case diagram for the appointment booking is attached in Appendix G.

3.4 Summary

The report above described the techniques and outcomes used for the extraction of requirements. These include interviews, questionnaires, task analysis and domain analysis techniques. The last two especially, consist the first steps towards the design phases of the project which are described in detail in the next chapter. The created prototype was also used as the base upon which the evolutionary prototyping technique was applied. The modelling of the requirements was also evolved with each spiral/ prototype iteration.

4. DESIGN

During the analysis of requirements, the various domain concepts were captured and documented using use case diagrams and a conceptual high-level class diagram. Moving into design, the class diagram would be enriched to display more detailed information about the association of the system components [155]. This will introduce and model new components across the application layers and include their associations too. UML activity diagrams [152, pp. 373-441] would be utilised to model the flow of execution and activities of each use case. Interaction sequence diagrams [152, pp. 565-638] would be used to model the component interactions and messaging that occurs in time sequence. Finally, Human Computer Interaction principles would be taken into account to design the screens for the MSA application.

4.1 Modelling MSA Activities

"A UML activity diagram shows sequential and parallel activities in a process. They are useful for modelling business processes, workflows, data flows, and complex algorithms" [156, p.477]. Where use case diagrams are mostly useful to clarify scope and to document high level functional requirements, the activity diagrams model the internal logic of single use cases or business process scenarios.
There are different levels of detail that can be captured by activity diagrams. A high-level activity diagram could be used in the analysis of the requirements to provide a view of a scenario logic that can be useful to business analysts and to assist with the understanding of requirements. A low-level activity diagram could be much more detailed and aiming to assist the developers with the understanding and implementation of the requirements.

At the end, the activity diagram should have enough detail to reflect its usefulness to the target audience but not overwhelm the reader with unnecessary information [157].

4.1.1 Managing and Updating Faculty Officer data

The activity below shows a high-level flowchart for managing the data of the Faculty Officer.

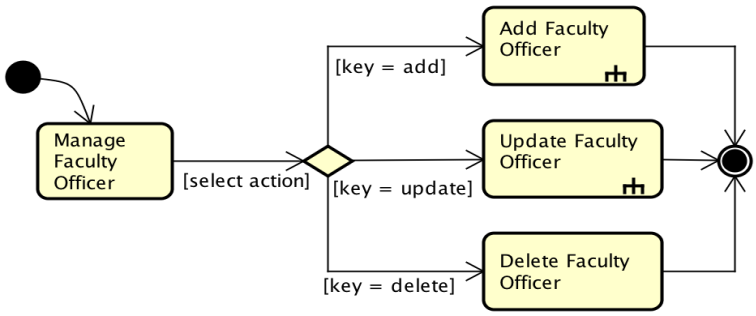


Figure 4.1 - High level activity diagram for managing a Faculty Officer entity

The individual operation of updating a Faculty Officer is abstracted out for clarity in a more detailed activity diagram below. The diagram (figure 4.2) is going to be used by the developers and is sufficient to provide the information needed for implementation.

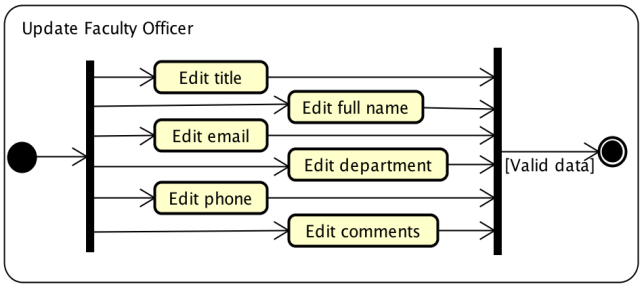


Figure 4.2 - Detailed activity diagram showing steps for updating a Faculty Officer

4.1.2 Complete booking meeting process

The diagram below (figure 4.3), illustrates the full sequence of internal steps of the process required to book an Unsatisfactory Progress interview with a Faculty Officer. It includes details for the *Login*, the *View Faculty Officer Availability* and the *Book Meeting* use cases.

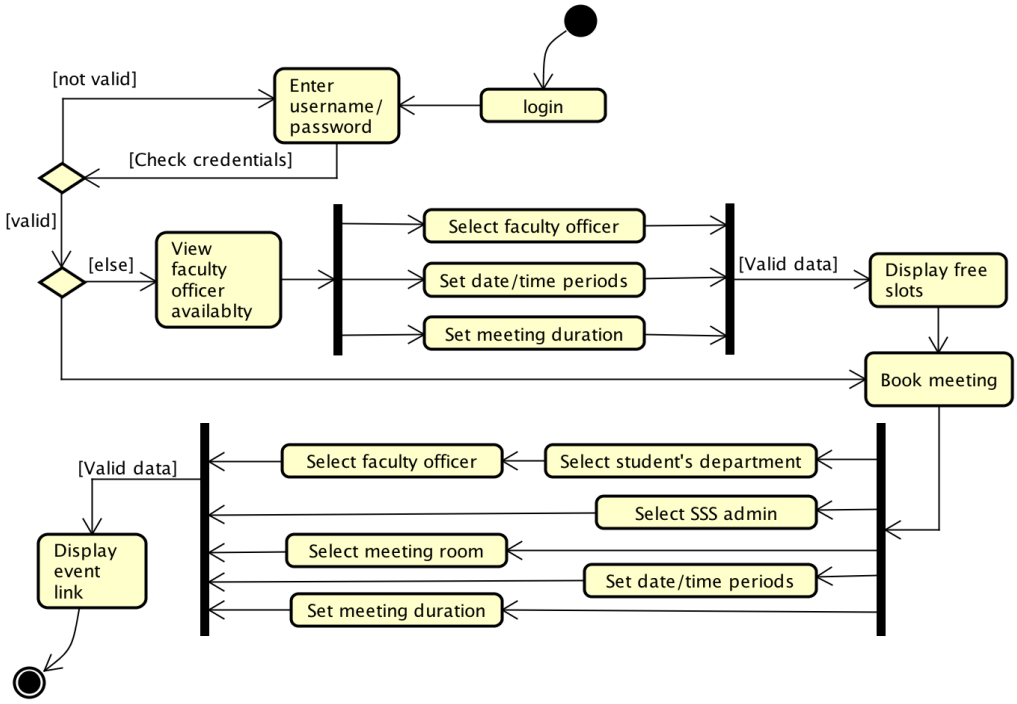


Figure 4.3 - Activity diagram showing the sequence of steps required to book an appointment

4.4 Detailed Class Diagram

As mentioned in the introduction of the chapter, the system class diagram was enhanced during the design phases to contain components across all the application layers. System behaviour was also modelled with more accuracy and detail. The detailed MSA class diagram is displayed below.

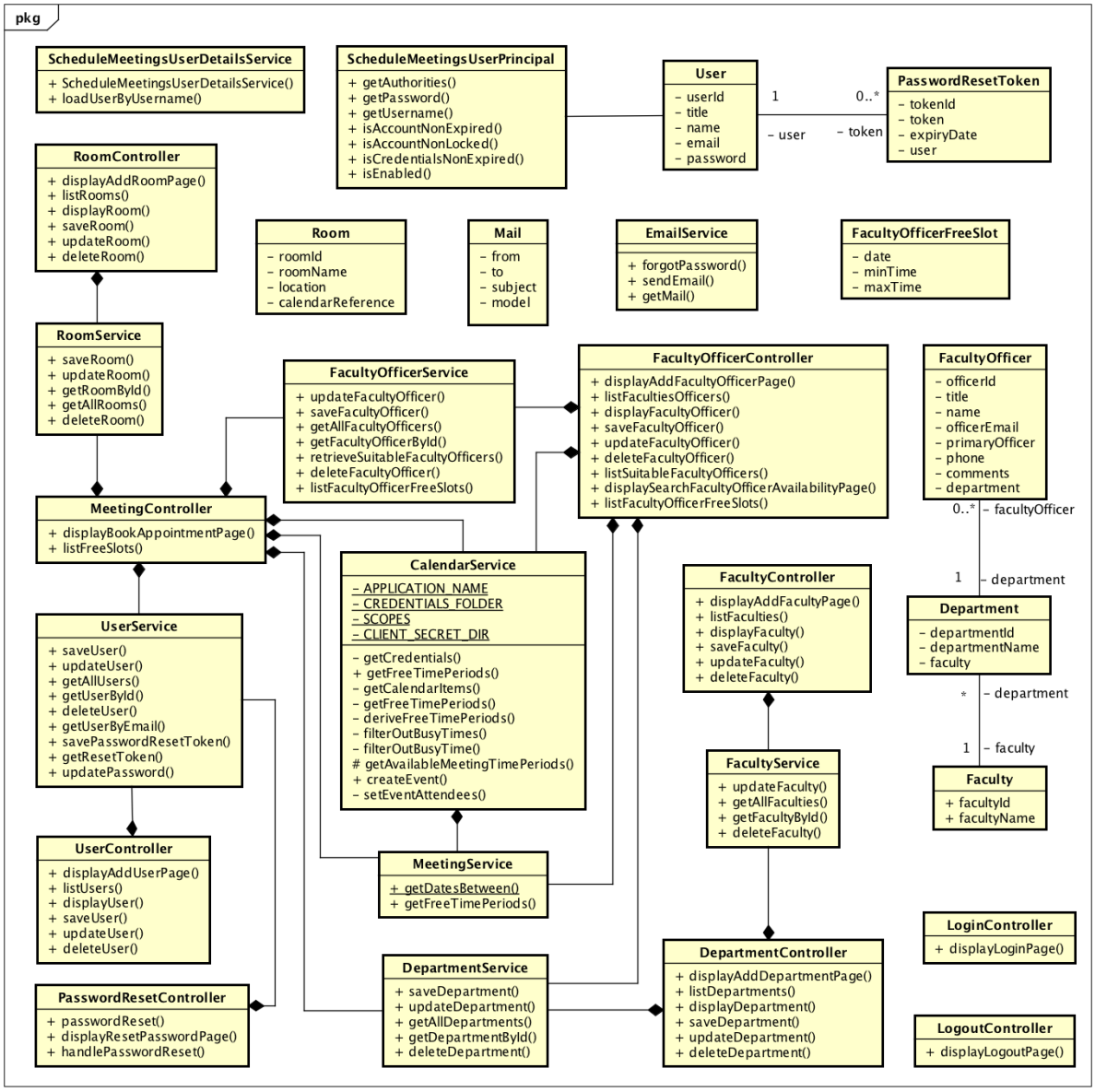


Figure 4.5 - Detailed class diagram for the MSA system

4.3 Modelling Interaction Sequence for the MSA

"The sequence diagram is used primarily to show the interactions between objects in the sequential order that those interactions occur. One of the primary uses of sequence diagrams is in the transition from requirements expressed as use cases to the next and more formal level of refinement. Use cases are often refined into one or more sequence diagrams." [158]
Thus, similar to the activity diagrams, sequence diagrams enable a higher level of refinement for use cases. However, while the former is focusing on outlining the internal activities of each use case, the sequence diagrams are used to model the interactions between components to achieve a use case target. Used in this way they probably offer a higher value for developers as they provide a very low level and clear to implement design.

The sequence diagram below (figure 4.6) depicts the object interactions for booking an appointment with a Faculty Officer and SSS admin staff in the MSA application. A sequence diagram for resetting the password by the user is also included in Appendix J.

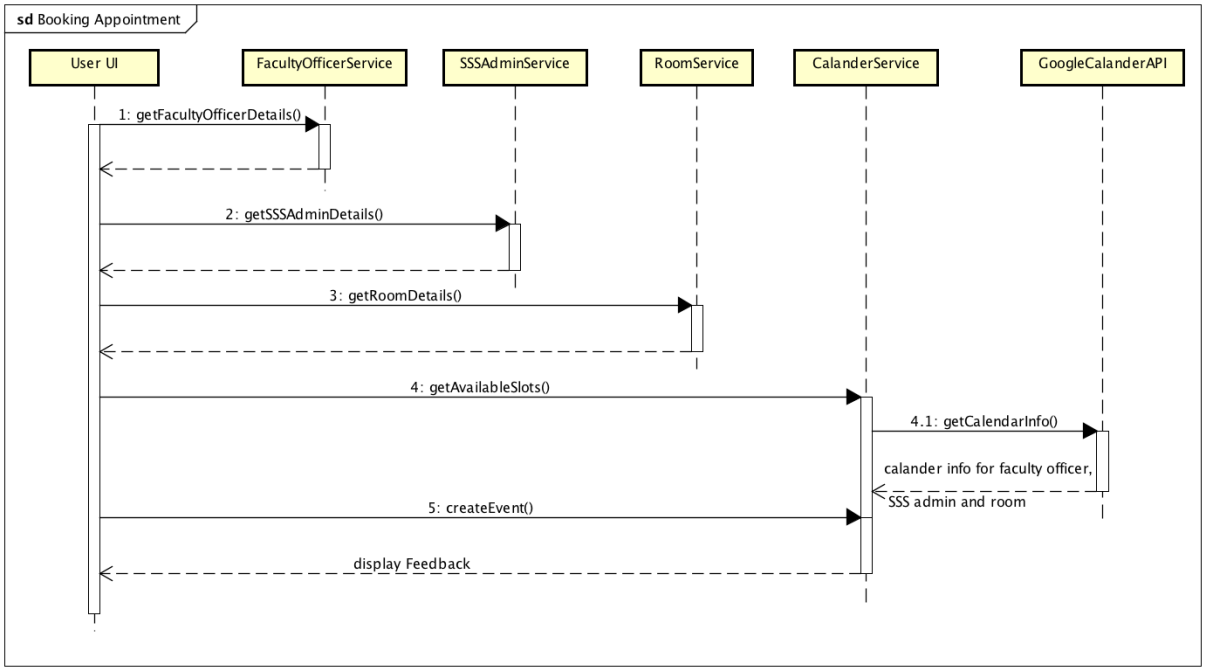


Figure 4.6 - Sequence diagram for booking an appointment with a Faculty Officer

4.4 Logical and Physical Database Design

Designing the data model is an important step in the design phase and the process artefacts are used to implement the database during implementation. The process is described by Teorey *et al.* [159, Ch. 1] and is summarised below.

The data needs of an application are initially determined during requirement analysis with the methods of interviewing and brainstorming between the project team, the client and various stakeholders. The identified data concepts are then used to implement the logical design of the database. The first step in that process is the creation of the conceptual data model which depicts the data concepts and their relationships in a simplified form. Attributes for each concept might be added but the model does not contain information about the type of each attribute at this stage. The most common form of modelling the conceptual diagram is with an Entity Relationship Diagram (ERD) [160]. The conceptual data model for the MSA is attached in Appendix I.
In a complex system, data designers would create different views of the same data needs because of different perspectives from various stakeholders. A further integration step would then be required to eliminate redundancy and inconsistency by merging the various designs in a single global view (see Teorey *et al.* [159] for example designs). This step was not required for the MSA.

The next step is to convert the conceptual design into SQL tables containing the attributes as columns and their corresponding data types. Further analysis of the requirements for each data is required but the evolution of the design can be realised in various ways. A relevant article on the guru99.com [161] site suggests evolving the entity relationship diagram (ER) to contain the additional details, [162] proposes the use forms to describe the tables and the columns/ attributes they contain, while Teorey *et al.* [159] suggest creating the data schema using Data Definition Language (DDL).

The evolved data model for the MSA is shown below. The MSA schema was also created at this stage and is provided with the system artefacts (mysql_schema.sql).

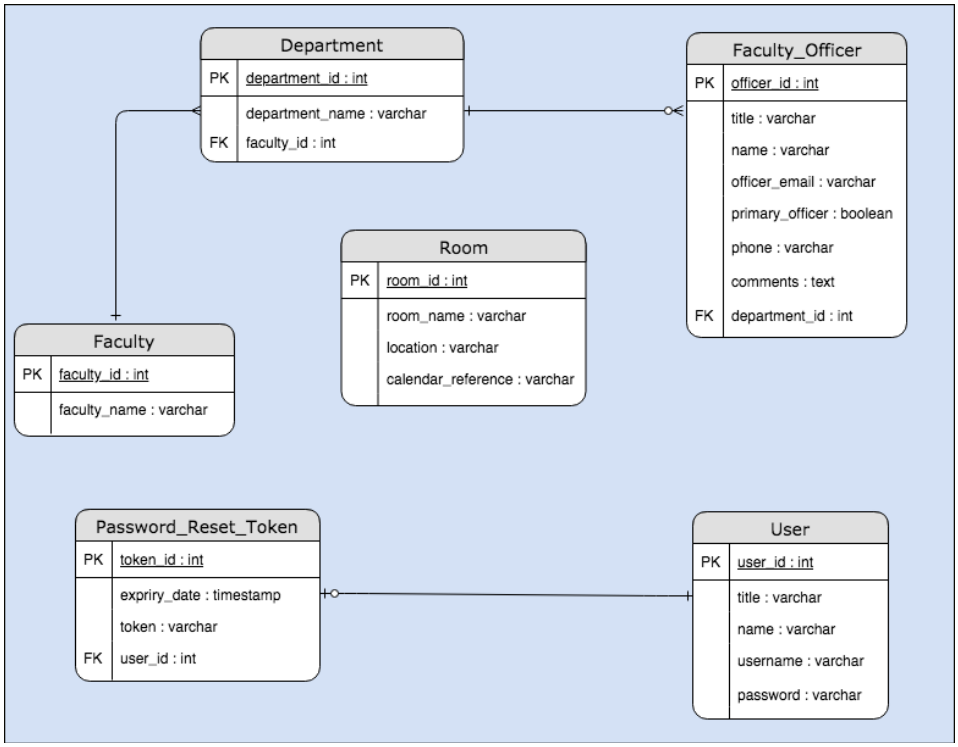


Figure 4.7 - Database model for the MSA application

The MSA data model is simple and no further refinements are required. In more complex models, normalisation [163, Ch. 6] of the data model would also normally be applied at this stage.
The final step is to create the physical design by adding constraints, indexes, clustering, partitioning and denormalisation wherever this is required. For the MSA the constraints were added to the DDL schema file to ensure referential integrity [159, Ch. 2] of the data. The rest of the mentioned techniques assist with the performance, management and scaling of the database system but they were not deemed necessary for the size and performance requirements of the MSA database. They are discussed in more detail by Teorey *et al* [159, Ch.7].

4.5 Human Computer Interaction

Human Computer Interaction (HCI) is the study of how to design user-friendly computer interfaces and interactions. The term was first used by Carlisle [164], but was popularised by Card, Moran and Newel [165]. There is a number of HCI principles which are established in the field with the most known being Norman's principles [166], Nielsen's heuristics [167] and Schneiderman rules [168]. Below there is a description of those HCI principles that were applied while designing the interface for the Meeting Scheduler Application.

4.5.1 Consistency

Consistency is one of the most important design rules and is mentioned by all three scientists above in their HCI sets of principles. It refers to using similar terminology and similar UI elements to perform similar operations. It also refers to designing similar journeys and series of steps to achieve similar targets.

For the MSA, dark grey was chosen as a consistent colour scheme across all screens. All interactive points such as buttons and links would be illustrated with blue colour. The location of buttons and naming of UI elements was designed to be identical for common operations such as when managing the various domain entities of the system e.g. adding faculties, departments, faculty officers. As another example of consistency, in order to edit any of these entity types, the series of steps to achieve this were

the same across all entities. First select 'Manage <entity>' on the menu section, then select a specific <entity> from the displayed list, then update the required details and finally press on the 'Edit <entity>' button to save the updates.

4.5.2 Feedback and Design to Yield Closure

Shneiderman [168], Nielsen [167] and Norman [166, Ch. 4] also state the importance of feedback for any actions that are taken by the user. The user should always be aware of the system status. When her action has completed a journey, this must also clearly be communicated back to her.

Any user actions in the MSA have a corresponding message which inform the user of any system changes. For Ajax CRUD (Create Retrieve Update Delete) updates of domain entities, a dialogue box is used to inform the user of the result. Dedicated 400 and 500 pages are also designed to provide consistent feedback for these types of errors. Also, upon successful booking of the Unsatisfactory Progress meeting, the information about the booking is presented in a message within the same screen. A link to access the corresponding Google Calendar page with the booked appointment is also provided in the same message. In the case of no availability another message communicates to the user the reason for failing to book an appointment.
All messages were designed with Shneiderman's rule of 'designing to yield closure' [168] in mind. The feedback at the end of a user action provides a signal of accomplishment and allows the user to continue with the next group of actions.

4.5.3 Error prevention

Both Shneiderman [168] and Nielsen [167] state the importance of error prevention. The system should be designed in such way to prevent a problem occurring in the first place and user confirmation should be required for error prone conditions.

The MSA application is designed to include validation in all the forms available to the user. Required fields are being checked for the presence of values, the type of the value such as integers or strings is being checked, the emails are being checked for the correct syntax, start and end dates need to be in chronological order and time durations are checked to be positive numbers only.
Where possible dropdown boxes are preferred over free text boxes to avoid entering an incorrect value. Dropdown examples in MSA forms include Departments, Faculty Officers, SSS admin staff and meeting rooms. Date fields are also designed as HTML5 date input elements so that the datepicker feature prevents the user from entering incorrect date formats.
In addition to the above, destructive actions such as deleting any Faculty, Department or Faculty Officer entity require extra confirmation from the user using a dialogue.

4.5.4 Permit easy reversal of actions/ User Control

Shneiderman [168] and Nielsen [167] both require in their HCI principles that the system should allow for user actions to be undone so that the user feels comfortable to explore options and navigate without anxiety through the system.

Most of the user actions offered by the MSA are designed to be simple and can easily be reverted. Where possible a single step to revert these is offered. For example, when the user logs out a link is offered to login back into the system.

4.5.5 Support internal locus of control/ Aesthetic and minimalist design

Shneiderman's rule for 'supporting internal locus of control' [168] states that the users should have the feeling that they are in control of the system and that they are the initiators of actions rather than the

responders. Nielsen's 'aesthetic and minimalist design' [167] also promotes similar UI qualities that enable the users to feel they understand the system and are in control.

The MSA screens were designed to be simple and not convoluted with information. The interactive points like buttons and links in forms and navigations menus display text information which clearly indicates the purpose of each interaction point. Navigation was designed to be clear and intuitive.

4.5.6 Affordance

Norman's affordance [166, Ch. 4] principle is concerned with the relationship between how something is perceived and how it should be actually used.

Evidence of the principle in the MSA design include using familiar symbols to indicate user actions like the 'x' for deleting a data entity. In tables where entity lists are shown, arrow symbols next to each column header indicate the sorting by the property held in this column.

4.5.7 Match between system and the real world

Nielsen's heuristic [167] refers to the quality of the system to 'speak the users' language, with words, phrases and concepts familiar to the user.

The MSA UI design uses terms and action descriptions from the user's business domain to display in form titles, buttons and navigation elements. For example, they include labels of *Faculty Officer*, *Faculty*, *Department*, *Book Appointment*, *View Officer's Availability* etc which promote system understanding and familiarity.

4.5.8 Constraints

Norman's 'constraints' [166, Ch. 4] refer to the quality of the system to constrain/ guide the user to the next appropriate action.

The interaction points in the MSA design of screens are all indicated with blue colour. Each screen contains as few interaction points as possible and the most appropriate action is more prominent that any other. For example, when adding an entity, the blue 'Add' button is the focal point of the screen. Also, because the main feature of the system is to book appointments, the user interface was designed to lead you to this journey. Upon successful login the user is immediately redirected to the appointment booking screen which basically serves as a homepage. Similarly, the navigation menu element which takes you to the same screen is more prominent by being enlarged and located leftmost on the top of the page.

4.6 Summary

During the design phases of the project, various UML diagram types were used to capture the internal behaviour of the system. The logical steps for each use case and the interactions of system components were designed as was also the data model for the application. Several examples of these models were included in this chapter. Finally, the way HCI principles affected the design decisions for the Meeting Scheduler UI were also explained. The next section will focus on the implementation of these designs and the testing procedures which were used.

5. IMPLEMENTATION AND TESTING

This chapter contains details about the implementation and testing phases of the Meeting Scheduler application. Similarly to previous project phases, the relevant artefacts which were produced were revisited and improved during each iteration of the spiral/ evolutionary prototype approach.

5.1 Implementation

The implementation details below describe how the MSA system was developed according to the design specifications from the previous chapter. Any points of interest are further explained such as the programming algorithms for searching the Faculty Officer availability and booking meeting events on the attendees' calendars.

5.1.1 Managing the Domain Entities data

The features for managing the data of the system can be accessed under the *System Management* menu (figure 5.1). Using HCI principles, appropriate names were used for the menu item elements to map system implementation terminology to real world domain [167].

The MSA offers a similar path for the management of all the entities in the system. For all CRUD (Create Read Update Delete) operations the first action is to display the list of existing entities. The figure 5.2 below shows the list of Faculty Officers once the *Manage Faculty Officers* menu item is clicked.

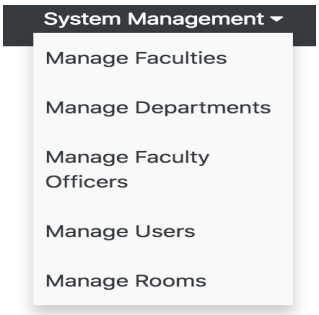


Figure 5.1 - Menu list for managing system data

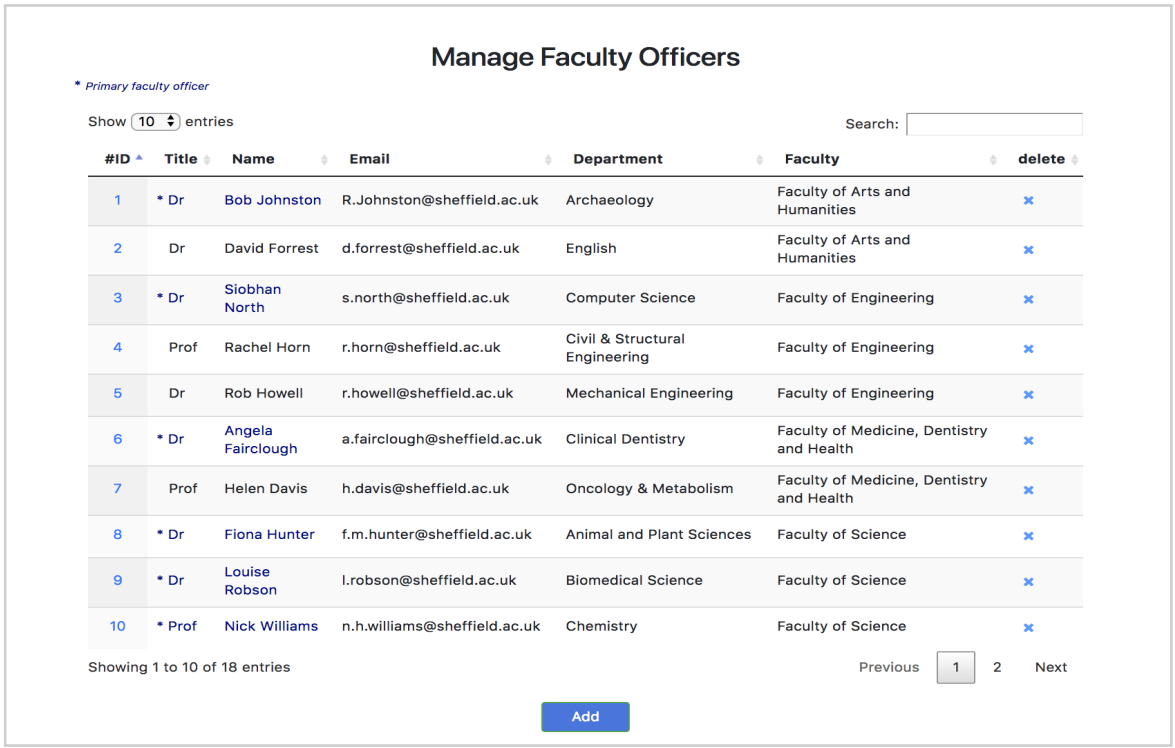


Figure 5.2 - Manage Faculty Officers screen

The visibility HCI principle has been addressed by indicating the possible next steps with blue colour. The user can select to add a new Faculty Officer with a blue 'Add' button at the bottom of the screen, edit the entity by clicking on the blue id link or click the blue 'x' symbol to remove the entity from the system. In order to ensure consistency and improve usability the same location and colour has been used for similar interactions in all entity management screens (see also the manage departments screen in Appendix E). Usability was further addressed with the inclusion of a search box at the top right for quick discovery of an entity. The number of entities that are displayed on the list is configurable by the user at the top left of the screen. Pagination is then implemented to enable the user to browse the full list of entities. Norman's affordance [166, Ch. 4] principle was also taken into account. In the above screen, the little arrows next to column headers indicate the sorting of the data by this property.
The DataTables [124] jQuery plugin was employed to assist with the features presented in the screens with entity lists.

Adding or editing an entity is done using an html form with appropriate fields for each property of the entity. Below is the corresponding screen for the Faculty Officer.

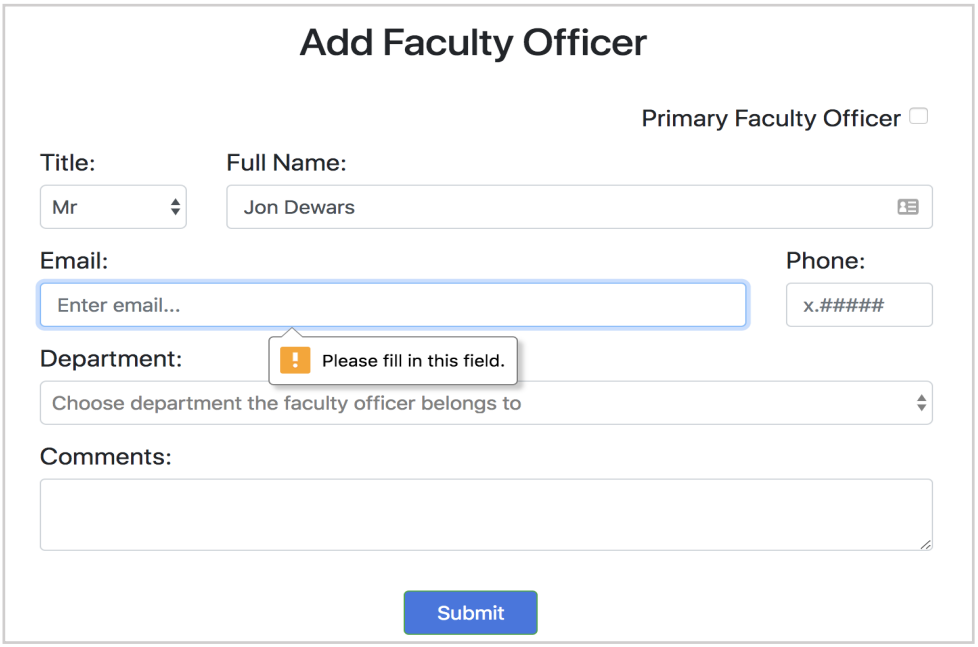
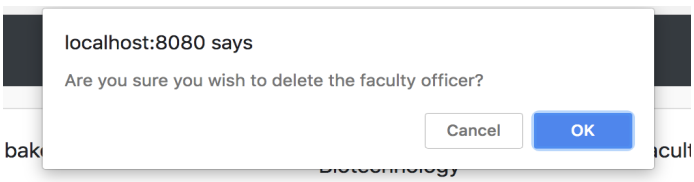


Figure 5.3 - Adding Faculty Officer form

Following HCI principles for user error prevention, predefined values are presented in the form of dropdowns such as the 'Department' and the 'Title' fields. HTML5 features have also been utilised for field validation and to add field value placeholders for an improved user experience.



Deleting an entity requires a confirmation from the user (figure 5.4). As discussed in the design section of this report, this further assists with user error prevention in the MSA.

Figure 5.4 - Confirmation dialogue box for deleting a Faculty Officer

The CRUD operations above were described for the management of the Faculty Officer entity type. As already mentioned, the same steps would be required for the management of the other MSA entities including Faculties, Departments, Users and Rooms. For brevity reasons, the description of these paths would not be repeated in this report.

5.1.2 Searching for the Faculty Officer Availability

Another important feature of the MSA is the capability to inspect a Faculty Officer calendar for available slots.

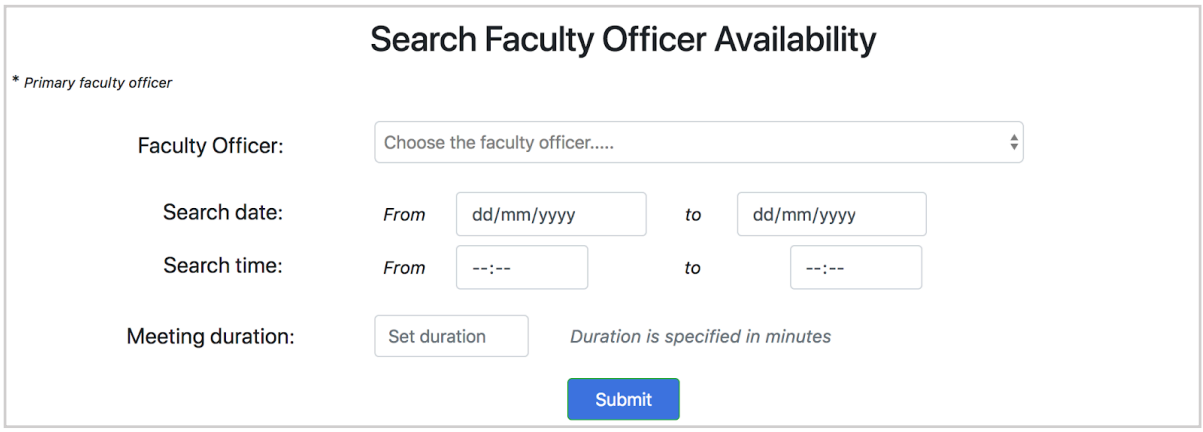


Figure 5.5 - Search Faculty Officer availability screen

The same HCI principles were applied as in the manage data entity screens. Blue colour to indicate interactive points, predefined values presented in dropdowns, HTML5 form input types such as 'date', 'time' and 'number' were used to assist the user with editing.

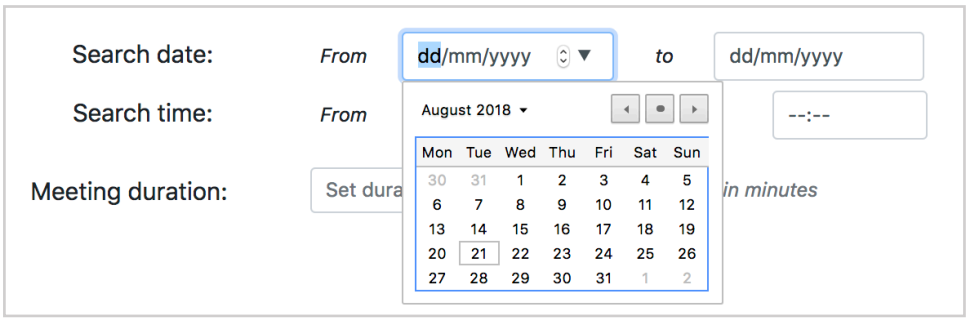


Figure 5.6 - HTML5 data type utilised to assist the user editing the form

HTML validation was applied with the 'required', 'min' and 'step' attributes to minimise user errors.

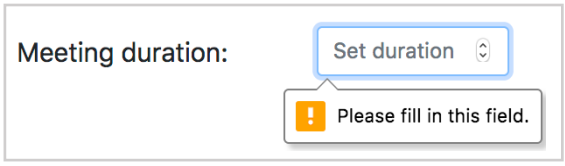


Figure 5.7 - Validation for required 'Meeting duration' field

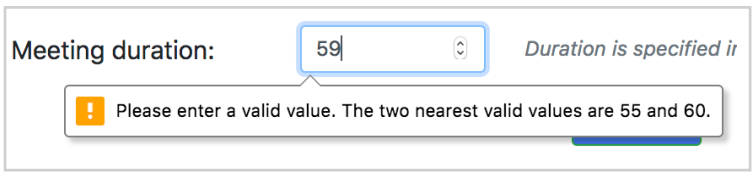


Figure 5.8 - Validation on the allowed range step values for the 'Meeting duration'

Further logical validation is also applied and the user is alerted for erroneous values (figure 5.9).

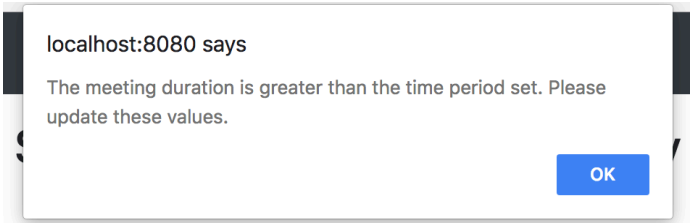


Figure 5.9 - Validation checking that values are logically correct

After passing the validation, the application will submit the query for Faculty Officer availability. The system will retrieve Faculty Officer information from the database and construct the query to send to the Google Calendar API.

5.1.2.1 Algorithm for finding Faculty Officer availability

The algorithm below describes the steps implemented by the system to derive the available slots for the Faculty officer within the user defined dates and times.

1. Retrieve data from the database for the selected Faculty Officer
2. Extract list of dates between the user defined start and end dates (LIST A)
3. Filter out Saturdays and Sundays
4. Create a list to populate with all free time slots (LIST B) in those days
5. For each date in LIST A (loop)
    a. Construct the Google Calendar API request using the Faculty Officer email and the user defined date and (start and end) times
    b. Query the Google Calendar API with this request object and get the response
    c. Get the busy time slots from the response (LIST C)
    d. Create an empty free slot list to hold the derived free slots (LIST D)
    e. Create an initial free time slot with the user defined start and end times and add it to the free times LIST D
    f. For each busy time slot in LIST C (recursion)
        i. Discard busy time slot from any free time slot it overlaps in LIST D (see sub-algorithm below)
        ii. Return updated list of derived free time slots (updated LIST D)
        iii. Use the updated LIST D of free slot times for the next iteration
    g. Add currently derived free slots in LIST D to overall free slots in LIST B
6. Filter out slots from populated LIST B which do not span the user requested duration
7. Return LIST B - contains any free slots found satisfying the user requirements for dates, times and duration

Sub-algorithm for step 5.f.i.

The sub-algorithm to filter out a busy slot from the list of free slots in step 5.f.i. is outlined below.

1. For each iteration, create a new list (LIST E) to hold the amended free time slots during the iteration
2. For each free time slot in LIST D
    a. If busy time slot overlaps completely the free time slot, then discard this free time slot
    b. If busy time start >= free time end, then add the free slot to LIST E
    c. If busy time end <= free time start, then add the free slot to LIST E
    d. If busy time start > free time start && busy time end < free time end, then filter out the busy slot (from the middle of the free slot) and add the two new free slots (from each side) to LIST E
    e. If busy time slot starts from within the free time slot and spans beyond the end of the free slot, then discard the time that is overlapped and add the remaining time slot (before the discarded time) to LIST E

f. If busy time slot starts before the free time slot and spans until a point within the free time slot, then discard the time that is overlapped and add the remaining time slot (after the discarded time) to LIST E

3. Replace the elements in LIST D with the derived free slots from LIST E (it would then be fed to the next iteration of step 5.f.i.

The screen presenting the available slots is consistent with the look and feel of the rest of the MSA site and presented below.

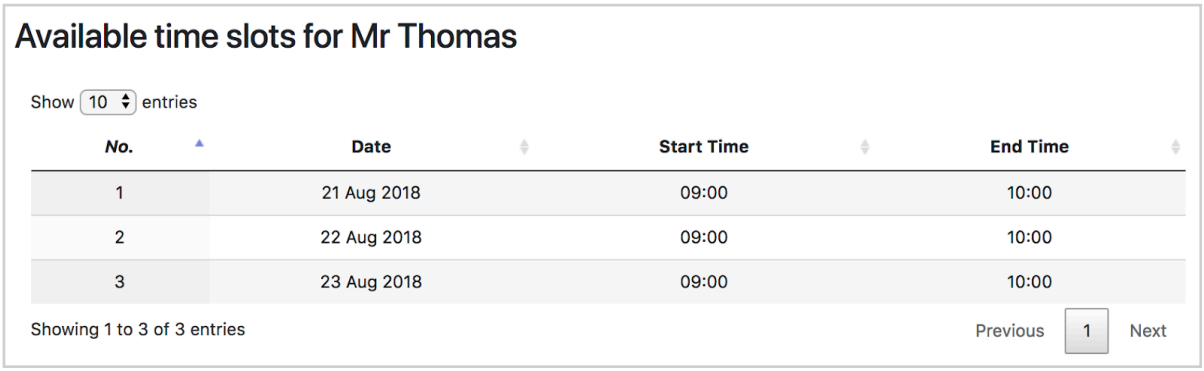


Figure 5.10 - MSA page displaying the available free slots for the Faculty Officer

5.1.3 Booking an Appointment

The page which enables an MSA user to book an appointment is shown below. Because this is the most important and most useful feature of the system, it is the screen which acts as a homepage and where the user is landed after successful login.

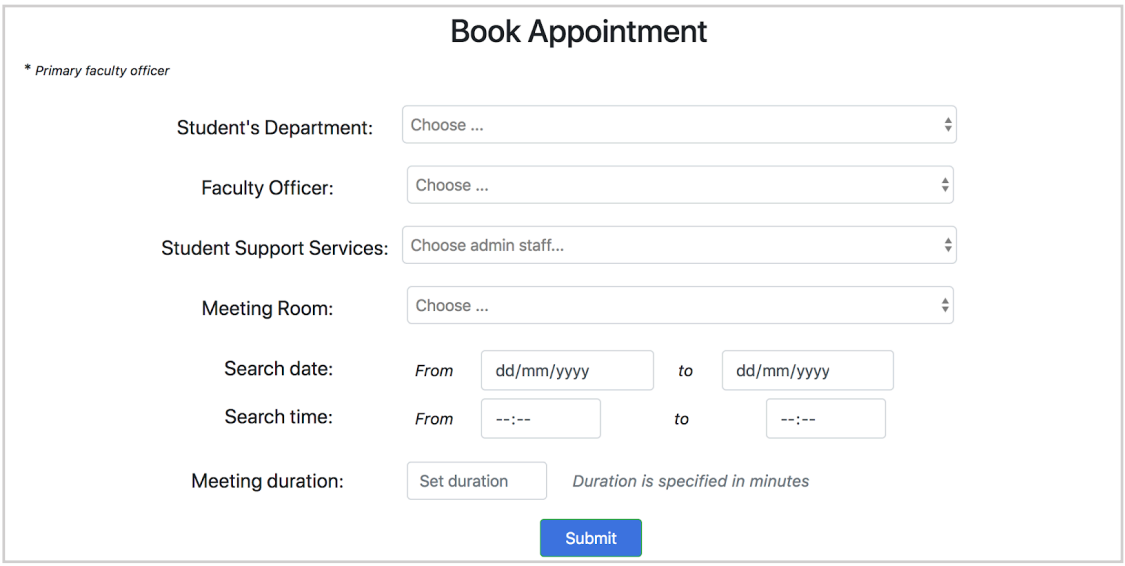


Figure 5.11 - Book appointment form

The same design and usability principles to the other MSA forms were applied on this form too. Where possible, values are presented in dropdowns and in alphabetical order to increase usability. Choosing a department value sends an Ajax request which populates the Faculty Officer dropdown field only with the Officers that belong to this department's Faculty (taking care to exclude Officers from the specified department as dictated by the requirements). HTML5 input types and validation were also employed in the same way that was described in other MSA forms (see 5.1.2 section).

On submitting the form, the application constructs a request object using information from the UI and the database. It then calls the Google Calendar API to retrieve available slots across all three selected entities, the Faculty Officer, the Admin Staff user and the room. The response is further processed to derive the first slot which spans the requested meeting duration. If found, a fresh request is sent to the calendar API to book an appointment within this slot for all relative entities. A successful response is communicated to the user.

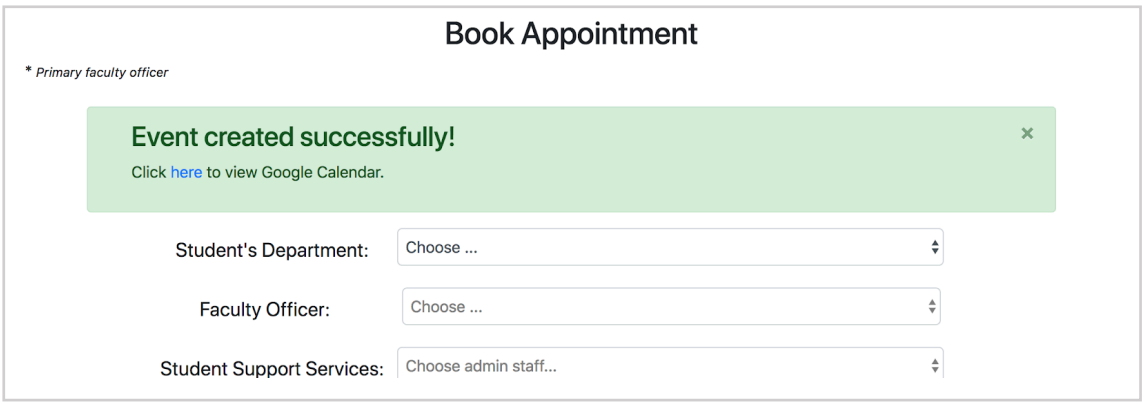


Figure 5.12 - User feedback for successful booking event

Clicking on the link within the success message redirects the user straight to the event within the Google Calendar application adding to efficiency and user satisfaction.

If no availability found for booking the meeting, an informative message is also displayed to the user. The relative screenshot is included in the user manual (Appendix K).

The algorithm for the meeting booking process is slightly more complicated than when searching for Faculty Officer availability and can be viewed in Appendix F.

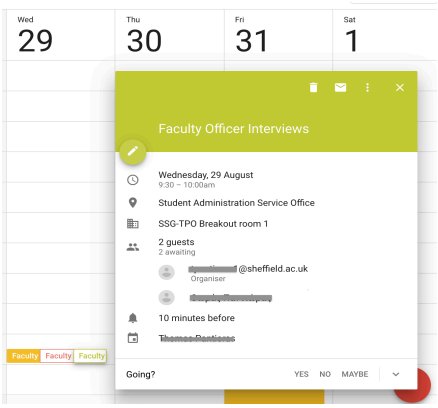


Figure 5.13 - Google calendar event creation

5.1.4 Login, Password Reset and Logout

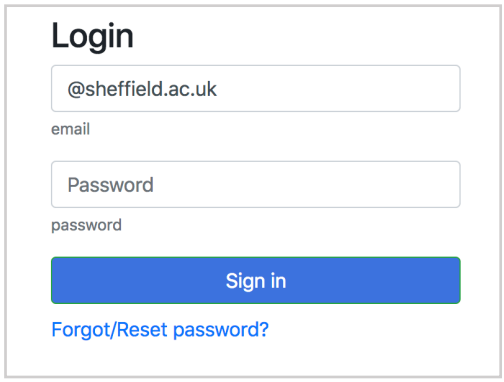


Figure 5.14 - The MSA login page

Authentication in the system was implemented using Spring Security. All pages in the MSA require an authenticated user and redirect the user to the login page if that is not the case. The Bcrypt [169] hashing algorithm was used to encrypt the password during registration. The algorithm is recommended by Spring and uses a randomly generated salt (described in more detail on the Mkyong.com [170] site) which means the same password value in different registrations would create different encoded values in the database.
The adjacent diagram shows the MSA login screen.

If the user forgets her email, a reset password functionality enables the user to recover from this situation. The process is initiated by clicking on the '*Forgot/Reset password*?' link and involves a series of small steps which are outlined below.

The user submits her email address and the system sends an email to this address containing a link to a reset password page (figure 5.15). Subsequently, the user edits values for *password* and *confirm password* fields and submits the request. Then the system encrypts and stores the updated password and redirects the user to the login page with an appropriate message (figure 5.16). The user is then able to login with the new credentials.

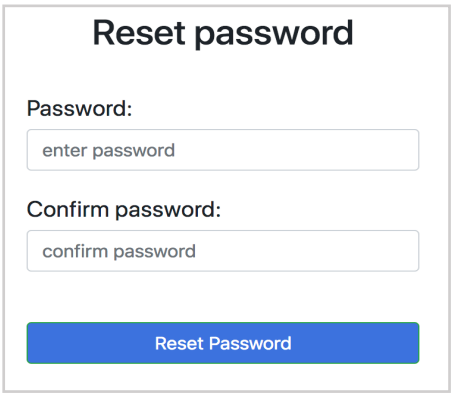


Figure 5.15 - Reset password page

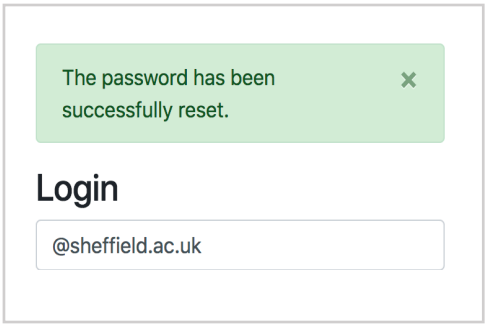


Figure 5.16 - Successful password reset

Obviously, the user should also be able to logout. Adhering to Shneiderman's rule for '*easy reversal of actions*' [168], a link is included when logged out so that the user can re-login if she wishes.

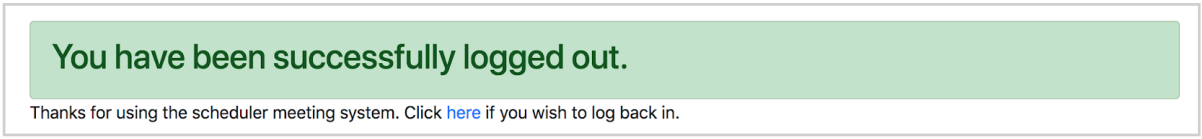


Figure 5.17 - Successful logout

5.2 Testing

Software testing is vital for the success of any software project. It is defined as "...an activity to check whether the actual results match the expected results and to ensure that the software system is defect free. It involves execution of a software component or system component to evaluate one or more properties of interest. Software testing also helps to identify errors, gaps or missing requirements in contrary to the actual requirements. It can be either done manually or using automated tools." [171] There are two main categories of software testing, the functional and non-functional which test the corresponding system requirement categories.

5.2.1 Functional testing

Functional testing in MSA was performed in three ways, unit testing [172], integration testing [173] and user acceptance testing [174].

Unit testing was performed by the developer and it was done in parallel with the implementation of the code. The Test Driven Development (TDD) [175] method was also considered, which dictates writing the tests before the implementation. This ensures that the implementation satisfies better the initial requirements. However, although the technique was briefly tested, there was not enough time to learn and apply it efficiently in this project.

Unit testing is concerned with testing a small unit of functionality. This unit constitutes part of a larger component set which contributes towards a specific functional requirement. Unit testing can be performed using either white box [176] or black box [177] testing techniques. The white box method tests the internal structure of the method/ unit while the black box tests only the output/ result. The advantage of the white box is that testing is more thorough. However, this comes with a cost as refactoring the internal logic of the method requires a change to the associated tests too.

For the MSA, the white box technique was used. Each method was tested thoroughly to ensure that the correct sequence of logical steps was followed in the code. The tool that was employed to do the application unit testing was JUnit [178]. For methods with no dependencies with other components of the system, this was the only technology required to perform the testing. For methods which contained dependencies, the Mockito [179] framework was also required in order to mock the responses of these dependencies.

Integration or component testing was also performed by the developer of the MSA. It is called so because it tests the connection of several components together and how they interact together in order to calculate the required result. This type of testing was done on the Web layer of the application and specifically to test the Spring MVC controllers. The Spring MVC Test Framework [180] was employed for this task as it is part of the same framework as the MVC implementation and was the easier approach.

The testing coverage has also been taken into consideration for the unit and integration testing of the MSA. The student attempted to achieve high coverage, but attention was also paid for the quality of the tests (as described by Fowler [181]).

The final step in the functional requirements testing process for the MSA was to perform user acceptance testing. With this, the system was tested to assess how well it met the business requirements and if it was acceptable for delivery to the client [174]. Because this type of testing was performed by the user it allowed to test what the user actually needs from the system and not what the developer understood [171]. This ensured that any gaps in the implementation of the requirements were identified and addressed.
The user acceptance testing form and the associated results are attached in Appendix C.

5.2.2 Non-Functional testing

Most of the non-functional testing was performed using a similar form with the one used for the User Acceptance testing of the functional requirements. Again, the testing was mostly done by the client although as her availability was restricted, there were cases where it was more appropriate for the student to perform some testing. These for example include the testing for reliability where the consistency of results over repeating actions was tested. Elsewhere, the testing of authentication was also done partially by the client, but checking that all MSA pages require authentication was tested by the student. Similarly with the form validation, the client was asked to test some of the forms in order to get some feedback over the validation approach. A more exhaustive testing in all the forms and for all invalid values was done by the student.

There was a large number of tests which were prepared and most by far have passed successfully. The test form and the results are both available in Appendix D. Two tests concerning the validation of the *'Comments'* field on the Faculty Officer management form have failed the validation for whitelist characters. The whitelist characters were enforced to prevent XSS (Cross Site Scripting) attacks. The validation failed due to the *textarea* HTML element not supporting the HTML5 *pattern* attribute. The easiest solution would have been to apply JavaScript to perform the validation. Due to the bug being identified very late in the project and due to the fact that the security concern was small (as there are only three users which are also required to authenticate to access the form), the fix has not been applied for the first release of the product. Nevertheless, the issue was registered with the test results and the product evaluation report (Chapter 6) and would need addressing in any follow up release or project work.

Another point worth mentioning is the performance testing for the MSA. There were several test cases prepared to measure response speeds for command and query http requests, which would require integration of the system over the network with the database and the Google Calendar API. The expected delay limits of these responses were based on accepted industry standards, as discussed in non-functional requirements section 3.3.2 of this report. The prepared tests (available in Appendix D) need to be run on the environment where the MSA system would be hosted and run, as only in this scenario the results can provide confidence for their correctness.
Other performance tests such as testing the system under load, i.e. with many simultaneous requests was not deemed necessary due to the limited number of system users.

5.2 Summary

This chapter described and showcased the implementation of the different parts of the MSA system. The algorithms for searching Faculty Officer availability and booking the appointment events on the Google calendars were outlined and discussed as were the associated UI pages. The testing procedures and outcomes of both functional and non-functional tests were also discussed. The next chapter considers the test results while offering an overall evaluation of the project outputs against the set objectives.

6. RESULTS AND DISCUSSION

In this chapter a description of the project outputs is given along with a critical evaluation of how the resulted software performs against the client expectations. Several points are also made, for ways to improve the existing functionality as well as for ideas to expand the system with new features. Some personal reflections over moments or events which impacted on the project as well as how the student has benefited from this work are also outlined below.

6.1 Project Artefacts

The most noticeable output and the one that offers the higher value for the client is obviously the produced software system, the Meeting Scheduler application (MSA).
The software is a Java web application which is contained in an executable jar format (find included scheduleMeetings-1.0.0.jar file) with an embedded tomcat container. All dependencies including any Web server dependencies are included in the jar and the only requirement is for the application to be run on a platform with an installed JVM runtime.
The system needs to integrate with a relational database to store and retrieve application data. The one employed during development was the MySQL database, however the application was designed to be completely decoupled from any database. Thus, any other relational system could be used with minimal configuration (see System Manual in Appendix K). A DDL script to create the initial schema (mysql_schema.sql) as well as a DML script (mysql_data.sql) for setting up the initial state of the database are also provided along with the other project files.
Apart from the main system a number of other artefacts were produced during the project lifetime which accompany the main product and contribute towards a robust and well documented solution.
The extracted requirements are documented clearly in functional and non-functional requirements tables which include high and low priority lists (available in Chapter 3 of this report). The answered questionnaires, used as one of the techniques for eliciting requirements, are also attached in Appendix N. Modelling of requirements has been done using UML and specifically use case, high level activity and class diagrams (also in Chapter 3). The data requirements for the system have been modelled using a conceptual Entity Relationship diagram (also in Chapter 3). These can be reused for future understanding and analysis of the client's business domain either for maintaining and expanding the MSA or to build alternative complementary solutions in the same domain.
Other artefacts produced during the design phases of the project assist further with the system understanding by future maintainers and developers. These include low level activity diagrams to depict internal activities of the business use cases, sequence diagrams for showing the interactions between system components and detailed class diagrams to express data and behaviour. Data types and other referential integrity constraints were further modelled with a physical data model and could be useful to database administrators and developers. The design artefacts are available in Chapter 4 of this report.
The code of the application is also provided in a zip file (scheduleMeetings.zip) along with the rest of project artefacts. Unzipping the file and opening the project with a Java IDE (IntelliJ was used for this project, but any other Java IDE could be used) would allow for future developers to quickly setup and start expanding the system functionality. Maven was used as a dependency management tool which enables instant download of all the required dependencies for development and testing. The system is a Spring Boot application and utilises several components of the Spring framework including Dependency Injection, Spring MVC (Model View Controller), Spring Security for authentication and Spring Data as an ORM (Object Relational Mapping) solution, so ideally any future developers should have a certain familiarity with Spring. Extensive development testing has been carried out with both unit and integration tests. These are included together with the application files. The tests provide

reassurance and confidence that the system meets all existing expectations and guarantee a robust application which can be used as a base for any future expansions. Industry standards for implementing web applications have also been followed by applying clean code practises and modularisation techniques as well as architectural and design patterns. All these should improve code readability and reusability, thus assisting with further system developments.
In addition to development testing, UAT (User Acceptance Testing) results for the functional requirements and testing outcomes for the non-functional requirements are also provided in Appendix C and D respectively. These again provide proof for the system quality and robustness but could also be used as a source for performing future business analysis and evaluating the system advantages and limitations.
Other artefacts useful to project managers and developers, include the risk management plans (attached in Appendix B) which were created in each iteration of the spiral methodology. These could be used as a reference for similar risks in new projects either to expand the MSA or to produce web applications with comparable conditions and requirements.
A big effort has been made to produce a UI which follows the HCI industry principles, is clean and intuitive to the user. Nevertheless, a system manual has also been created (available in Appendix K) with instructions and screenshots for clarifying to the end user how each of the system features can be used. The manual also contains information about how to install and run the system, including information for system configuration such as defining the listening port for the application and setting up database connection properties.
The final artefact of the project is no other than this report which contain valuable information about all the project stages and their relevant outputs. In addition to describing in detail the adopted processes in the requirements, design, implementation and testing phases, the literature review offers a discussion and critical evaluation of various alternative tools and technologies which were considered. The current chapter also provides knowledge about how elements in the software implementation as well as the employed project methods could be improved to produce better outcomes. The collection of information, descriptions and analysis of the covered techniques could provide reference material for IT students and software professionals.

6.2 Evaluation

At the end of the project it is well worth spending some time to evaluate the project results against the targets which were set at the beginning.

The main business objective of the project was to create a software to facilitate and automate the process of booking an Unsatisfactory Progress appointment with all relative participants. The main problem that needs solving is to eliminate the administration effort required to check the availability of each of the participants and negotiate the best time for them.
The MSA offers a simple UI which enables the user to edit meeting time details and select the Faculty Officer, the SSS admin staff and the room where the meeting would take place. Submitting this form, all relative parties' calendars would be checked for available slots based on the meeting time and duration requirements. Once a slot is available across all calendars, it is booked and a meeting event is created in all calendars. The process is seamless to the user and it requires a negligible amount of time.
A secondary requirement was to be able to check the availability of a Faculty Officer after applying specified date, time and duration constraints. The aim here was to allow the user to get an overview of the Faculty Officer schedule in order to decide which times should be targeted for the appointment booking process described above. Again, the system offered this functionality through a clear and intuitive UI and within seconds the user is able to retrieve the required information.

Satisfying the above requirements means that functionally the MSA system has achieved its targets. There is a number of other features and performance considerations which the system incorporates and which guarantee a satisfactory and engaging experience.
The user is able to manage all data entities stored in the system. These include Faculties, Departments, Faculty Officers, SSS admin staff and meeting rooms, and the user can easily add, edit and delete any of these using a simple interface.

A login functionality permits only authenticated users to access the application pages. The system enables existing users to register new users and a reset password feature is available for users who have forgotten their password.
Security measures have also been taken against XSS (Cross Site Scripting) and SQL Injection attacks. For the former only whitelists of characters are allowed in free-text form fields. The SQL injection attacks are prevented using Spring Data parameterised queries.
As mentioned in the previous section, exhaustive testing has been applied repeatedly by the developer as well as the client with User Acceptance tests. In this way a robust and reliable software which meets fully the client expectations was guaranteed.
Various HCI concerns were taken into account and steered the implementation to create a product with a responsive and engaging UI. Consistent colour patterns, simple UI design, easy navigation, appropriate feedback with message dialogues, validation with clear user instructions and others contributed towards a pleasant user experience. The Bootstrap framework was utilised to implement a responsive design that would work and display harmoniously across tablets, phones and desktop/ laptop machines.
Performance considerations were made to ensure response time remained within the industry recommended limits. Specifically, command responses from a local database were designed to take less than 0.1 seconds and queries less than 0.3 seconds. Command responses through the network should take less than 0.5 seconds, while query responses should take less than 1 seconds. These could be either from requests to the Google Calendar API or to a remote database.
Other considerations include the programming language and data needs of the system. The application was written in Java and can be run in any platform where a JVM machine is installed. It is also database agnostic and it can run with any relational database system. These increase the portability and maintainability of the MSA system. The costs for hosting and running the application could also potentially decrease as it can be deployed in the most preferred environment according to the currently available resources.
A considerable effort was spent to research the most suitable application architecture to increase modularisation and reusability of the system. The system is using the onion architecture which uses abstractions to decouple the different modules of the application. Together with the clean coding techniques and patterns which were applied they provide for a system which is easy to understand and maintain, can be easily expanded and its code can be reused in other applications.

Overall the resulted MSA application is a well designed and robust system which fully satisfies both its functional and non-functional requirements. Nonetheless there are areas which can be improved and these are discussed below.

6.2.1 Areas for Improvement

A proper evaluation of the MSA has revealed some points which can be addressed to further improve the quality of the system.
During the third prototype demonstration it was requested by the customer to add the feature to retrieve the Faculty Officer availability within specific dates. The analysis and design of the requirement dictated that the best implementation would have been for this feature to be incorporated within the appointment booking process. Nevertheless, the feature would have been costly to develop in this way because of the many changes that would be required in existing code. Also to guarantee a well designed and aesthetic interface more research was needed to identify the best way to present the Officer availability to the user. A possible solution would have been to use a jQuery plugin like FullCalendar [182], which allows to display events using a JSON feed and even offers google calendar integration. Clicking on calendar slots enables the user to add events with assigned callback methods integrating with the Google Calendar api. Although the apparent efficiency of the plugin more time was needed to test it as well as researching and comparing alternatives. As mentioned above, the way the plugin implementation would fit into the existing UI also required further research and prototyping for approval from the client. The problem revealed one of the drawback of the evolutionary prototyping methodology which is that “premature (and wrong) choices made on early prototypes might make their evolution into the final product cumbersome and difficult” [15, Ch. 7]. As a result and due to time

constraints it was agreed with the client to implement this as a separate feature which would display the Faculty Officer availability in a table structure on a separate screen. The solution is good enough for an MVP product, but it currently breaks the Shneiderman [168] HCI principle for reducing the short-term memory load. The user has to first visit the pages for finding the Faculty Officer availability and then use this information on a separate area of the MSA system to target specific dates for booking the Unsatisfactory Progress meeting. In a future release of the MSA the two features need to be merged together to allow for a more seamless user experience.

Currently the algorithm that returns the available slots for the Faculty Officer discards the free slots within weekends. A further improvement in the algorithm would be to discard the slots within any UK Bank Holidays. A research is necessary to discover an open source API which could provide this information.

Appropriate error handling of the Google Calendar API responses is another implementation detail which would enhance user experience. At the moment, if the calendar API responds with an error a generic error page is displayed to the user. The Google Calendar site offers a dedicated documentation page for handling error responses from its API [183], which could be used to enrich the feedback information that is presented to the user.

Feedback for all successful responses could also be further improved as currently this is done using browser alert dialogs. Although this is enough to convey information, a better solution would be to implement modals to present user feedback. There are several open source implementations including Bootstrap's Modal [184] and jQuery UI Dialog [185]. Again, some investigation is required to compare the different options.

The user experience would be further improved by adding the user manual to be available within the application web pages. Although the manual is available offline and is attached in Appendix K, the user would benefit from being able to access the manual directly while browsing the functionality of the system. A discussion with the client is needed to confirm that this would be desirable.

Another area for possible improvement is the authentication for the system. Currently a basic http authentication is implemented, which seems to provide enough security for the current requirements. A further research on alternatives like OAuth 2.0 [186] is necessary to assess if the MSA would benefit from the higher level of security they might offer.

Elsewhere, the non-functional testing of the application discovered the failing of form validation on the *'Comments'* field when adding or editing a Faculty Officer. A whitelist of characters was attempted on the field using the HTML5 *pattern* attribute in order to prevent XSS attacks. It was found that the textarea element does not support the pattern validation and that JavaScript would be needed to properly validate the field value. Due to the facts that the risk was minimal (as only three users from the same office would use the form and they would still require authentication) and also that the validation failure was discovered very close to the project deadline, it was decided to not include a fix with the MVP product. The finding was noted in the documentation and could be fixed in a following release/ student project.

In addition, adequate performance testing of the http response speeds to submitted user requests is only possible on the live environment where the system will be deployed. This is because of different server specifications and the network integration requirements which would be applied when the system is hosted and run. The prepared tests and expected values have been documented and ready to be used.

Another testing activity which could be performed is testing the UI responsiveness of the application with real mobile and tablets devices. Although this has been done using internal browser tools like Chrome's Developer Tools [187] to simulate mobile devices, a more comprehensive testing should allow to use real devices instead. It is possible that there is a University facility that offers a range of devices for the purpose of this type of testing, but the options available have not been researched further.

One other type of testing which would be beneficial to the system is automated UI testing using tools like Cucumber [188]. This type of testing checks the system functionality and specifically how it is presented on the UI using automated techniques ie not requiring the tester to manually visit the application pages. Although some initial investigation has been done, time constraints have not allowed for fully implementing this technology.

A final improvement that could be done and which would increase the maintainability and longevity of the system is adding logging capabilities to the system. Logging would allow to quickly identify the cause of errors in the system when this is deployed on the live environment. Two popular logging

libraries include logback [189] and log4j [190] and Spring Boot (which is the framework used for the MSA implementation) offers an easy integration with both of these [191].

6.3 Future Work

While the previous section offered a discussion for improvements within the scope of existing functionality, the application can also be further extended to contain additional features. One noticeable extension is to add functionality to handle meeting invitation rejections and cancellations. The functionality has not been added in the first MSA release as the three main users of the system (through questionnaires and interviews) have expressed the opinion that these features have a lower priority and could be handled offline. Nevertheless, being able to automatically book a meeting at an alternative time when the initial invitation has been rejected or cancelled would increase the effectiveness of the system and should be considered in any future system upgrades.

Another expansion to the application would be to add the Faculty Officer as a system user. According to information from a Faculty Officer interview, they would find it useful to retrieve information about the involved students prior to the meetings. That of course would require a number of new features for importing the relevant student data into the system either by the SSS admin staff or even directly from the student's Department. The latter implies that another system user could be a Department staff which could also use the MSA to initiate an Unsatisfactory Progress procedure instead of sending an offline application to the SSS team (as is currently the case). The benefits would be many for all the parties involved. For the Department staff user, the application could offer ways to store exam results, edit student attendance data, edit details for any written work failed to be submitted and other. The report for the Unsatisfactory Progress process could be started and stored in a partial state before being officially submitted. A reminder email could also be sent before the deadline. The Faculty Officer would be able to quickly access the data using searching capabilities and view it online or download it in different formats. The SSS admin staff also could access the data for any administration or management tasks as required. The MSA would act as a centralised application for processing and managing any data and activities related to the monitoring of student progress procedures (although the name would probably need to change from Meeting Scheduler application to reflect the wider functionality which would offer). Other requirements regarding personal data protection would then also have to be considered with greater attention. The law and any University procedures for data protection would have to be followed and security concerns would need to be revisited and likely upgraded. Authorisation levels would also have to be implemented so that certain types of users can perform only specific authorised actions on the system pages.

Through interviews with the client, it was found that the Unsatisfactory Progress process involves a next stage meeting which requires a larger number of attendees and is even more difficult to manage its administration. The MSA could be extended to assist with the management and administration tasks for this meeting too, e.g. searching for availability and creating appointment events.

A more challenging task, but which would be highly beneficial to the client is to find ways to define the weeks within which the appointments should take place. There is normally a window of two week to complete the meetings and the exact period is dictated by the exam periods and results announcements, public and religious holidays and individual student circumstances. If these were predefined, the user would not need to add the dates and times constraints every time a new appointment would be booked, which can increase the likelihood of mistakes. An automated way to define these events would be ideal, but this information comes from various sources which do not offer the means to programmatically consume this information (e.g. through an API). Web scraping is a technique that could be used to extract data from web pages automatically but further testing is required to test its efficiency. Also, the urls for these pages might change every year. An alternative way is to provide the interface for the user to add these events once a year, for each Faculty. A good requirement analysis and some prototyping would assist with the UI design for such a solution.

Some future work could also be done to make some of the functionality available through a mobile app. This could include searching for attendees' availability, booking the meeting events or any of the other

features discussed above. The advantages of an app over a browser application and if these justify the effort involved would need to be discussed further with the client.

Finally, an essential future activity is to upgrade the technologies used by the current system. The versions would need upgrading and patches would be required to keep the system free of security risks and to increase its maintainability and longevity.

6.4 Personal Reflections and Gains

On a personal level, the student undertaking this project has also benefited in multiple ways. First of course on various technical areas. Some of the technologies like Java, JavaScript and Spring had been already familiar, but the project gave the chance to consolidate existing knowledge and also experiment with unfamiliar language features or framework areas. Some for example include using recursion in Java, the Java Stream API, Spring Data and Spring Security among others.
In addition, the project offered the chance to enter fields of software engineering which the student had very little experience with, such as application architecture, clean coding techniques and code testing. It became more evident how applying programming principles like reducing coupling (using the onion architecture and programming with interfaces), increasing cohesion in methods and classes, choosing appropriate names for class members, keeping methods small and simple can all lead to code which is modular, reusable and maintainable. The importance of testing became apparent when started using tools like JUnit which lead to the discovery of many bugs during the implementation phases of the project.
In addition to technical skills the student has also developed softer skills like communication and listening which were essential for eliciting requirements during interviews with the client. Project management skills were necessary to prioritise the large number of tasks required and to plan and schedule the work accordingly. Researching and evaluating tools and techniques quickly and efficiently was equally important for the success of the project.
Having to perform the roles of project manager, requirements analyst, designer, developer and tester offered an insight in many aspects of software engineering and at the same time the chance to understand how everything fits together and contributes towards the achievement of the project targets. Poor performance in any project area or phase would put the project at risk.
Reflecting back there are a few cases where either because of inexperience or by trying to save time in certain project activities, wrong decisions or poor judgement had been applied. Not achieving to extract the requirement for searching the Faculty Officer availability at an early stage made it harder to merge it into the existing user interface journeys.
Using white-box testing techniques meant that the internal of each method was tested thoroughly but made code changes difficult to do. Retrospectively a black-box technique should be used where the correctness of the output is tested irrespectively of how it is implemented internally.
In another occasion a crash of the student machine used for the project, had wasted two to three days of work. According to the risk assessment plan, frequent backups of the application and report saved this work, however several diagrams were lost and these needed to be produced again. This was an unnecessary and costly oversight on behalf of the student.
An underestimation of the time needed for some tasks also added pressure on already strenuous conditions. One example was the time required to implement the algorithm for identifying available slots across all meeting attendees. Another is the time spend trying to run the application on University computers. The installed JVM version on these machines was Java 8 instead of the Java 9 used by the project. As a result, any Java 9 features had to be reimplemented and compiled with Java 8. Firewall issues also prevented the integration with the Google Calendar API and time was spent trying to resolve the issue with the help of technicians.
An initial approach of trying to model every aspect of the system with UML diagrams during the analysis and design phases was reassessed and adjusted as sometimes the value achieved was not worth the effort and time spent. For example, for the system data management only the modelling of the Faculty Officer was necessary. The management functionality for the other entities was similar and based on the same diagrams.

All of the above created challenges which put the student under a lot of pressure and required effort and discipline to overcome. The result however is greater understanding, higher competency in technical and other skills and a greater sense of satisfaction for what has been achieved.

7. CONCLUSION

The Meeting Scheduler (MSA) was a successful project undertaken as a student dissertation for the MSc Software Systems and Internet Technology.
A considerable amount of time was spent in researching software project techniques and technologies to assist with the project work. Finding the right tools for the job had put the foundations for a successful outcome.
After weighing the pros and cons from a number of project methodologies, the spiral model and the evolutionary prototyping techniques remained as favourites. The former would offer more confidence as it allows to use familiar analysis and design activities while the latter would provide quick feedback from the user. At the end a mix of the two methods was used where the activities in each spiral iteration were carried out and contributing towards building and evolving the system prototype. The client feedback was then steering the next spiral iteration.
Various techniques for eliciting requirements were applied including questionnaires, interviews, task and business analysis. The extracted requirements were modelled and analysed to identify the internal logic and steps for each use case scenario. The design techniques which were used include class, activity and sequence UML diagrams, Entity-Relationship diagrams as well as HCI principles.
For the implementation of the software a selection of the considered technologies from the earliest literature review stages, was utilised. Java was used to implement the logic and the integration with the Google Calendar API while the View part was developed with Thymeleaf, HTML5, Bootstrap and JavaScript. The application utilised the Spring framework to find solutions to many of the development challenges. These include Dependency Injection but also employing Spring MVC, Spring Data as an ORM, Spring Security for authentication and Spring Boot for quick bootstrapping and for using the embedded tomcat container. MySQL was the relational database chosen to store and retrieve system data, although the application was designed to be decoupled from the underlying data layer. That means the database is easily interchangeable if that is required.
A thorough testing has been applied to ensure a robust and efficient system. The MSA testing was done on many levels. Unit tests for the individual class methods, component tests to test the integration of components from the web layer all the way to data, user acceptance tests and non-functional tests.
The end result is a system which fully meets and, in many ways, exceeds the initial expectations. It has a clear and responsive UI for a satisfactory user experience. It is employing industry standards and programming principles to make it easier to understand and maintain in the future. It was designed and implemented using modular architecture to increase reusability and extensibility. It is bug free. It offers a comprehensive documentation about how to run and deploy the application as well as a user manual. Additionally, instructions of how to setup the development environment for future work are also provided.
On a more personal level the project was an opportunity for the student to apply knowledge and skills acquired through the first two semesters of the course. Techniques for eliciting requirements, object and data modelling, UI design, Java and JavaScript programming, HTML5 and CSS, database and SQL skills were already familiar and utilised in various stages. Others like project planning, risk analysis, application architecture and coding patterns, various plugins and frameworks, testing tools like JUnit were researched and employed for the first time but with great effect.
Softer skills were further developed to assist during the project including communication, task prioritisation, effectiveness in the research and evaluation of tools and technologies, professionalism, discipline in ways of working, flexibility in adjusting and responding to changes among others.
In summary the Meeting Scheduler project has successfully provided the client with a tool which will assist with and automate tedious and time consuming administration tasks within the Unsatisfactory Progress process. It has been a long and challenging work with a lot of obstacles to overcome and issues to resolve. It is also exactly this effort and experience which has benefited and made the student grow on both technical and personal levels.

REFERENCES

[1] University of Sheffield, "University regulations," in *sheffield.ac.uk,* January, 03, 2018, [Online]. Available: https://www.sheffield.ac.uk/history/current_students/undergraduate/regulations. Accessed on: Aug 15, 2018.

[2] University of Sheffield, "The university of Sheffield calendar 2017-18," in *sheffield.ac.uk*, [Online]. Available: https://www.sheffield.ac.uk/calendar. Accessed on: Aug 15, 2018.

[3] University of Sheffield, "General regulations relating to the progress of students," in *sheffield.ac.uk*, [Online]. Available: https://www.sheffield.ac.uk/polopoly_fs/1.743308!/file/26_Reg_XIX_General_Regulations_relating_to_the_Progress_of_Students.pdf. Accessed on: Aug 15, 2018.

[4] University of Sheffield, "Timetable for receipt of progress concerns reports," in *sheffield.ac.uk*, [Online]. Available: https://www.sheffield.ac.uk/sss/sas/progress/timetable. Accessed on: Aug 15, 2018.

[5] University of Sheffield, "Academic progress of students," in *sheffield.ac.uk*, [Online]. Available: https://www.sheffield.ac.uk/ssid/complaints-and-appeals/progress-procedures. Accessed on: Aug 15, 2018.

[6] University of Sheffield, "Progress of students: guidance, regulatory details, forms and contact information," in *sheffield.ac.uk*, [Online]. Available: https://www.sheffield.ac.uk/sss/sas/progress. Accessed on: Aug 15, 2018.

[7] Google, "Google calendar API," in *Google Developers,* [Online]. Available: https://developers.google.com/calendar/. Accessed on: Aug 27, 2018.

[8] Google, "Overview of the calendar API," in *Google Developers,* [Online]. Available: https://developers.google.com/calendar/concepts/. Accessed on: Aug 27, 2018

[9] Google, "Calendars and events," in *Google Developers,* [Online]. Available: https://developers .google.com/calendar/concepts/ events-calendars. Accessed on: Aug 27, 2018

[10] Google, "Authorizing requests to the google calendar API," in *Google Developers,* [Online]. Available: https://developers.google.com/ calendar/auth. Accessed on: Aug 27, 2018

[11] Google, "Create events," in *Google Developers,* [Online]. Available: https://developers.google .com/calendar/create-events. Accessed on: Aug 27, 2018

[12] Google, "Java quickstart," in *Google Developers,* [Online]. Available: https://developers.google .com/calendar/quickstart/java. Accessed on: Aug 27, 2018

[13] Google, "API reference," in *Google Developers,* [Online]. Available: https://developers.google .com/calendar/v3/reference/. Accessed on: Aug 27, 2018

[14] T. M. Cagley, M. Chemuturi, "Software project basics," in *Mastering Software Project Management,* Florida, US: J. Ross Publishing, 2010, pp.3 – 4

[15] A. Villafiorita, *Introduction to Software Project Managemen*t, Florida, US: CRC Press, 2014.

[16] W. Royce, "Managing the Development of large Software Systems," *Proceedings. IEEE Wescon*, pp. 328-338. August ,1970. [Online]. Available: http://www-scf.usc.edu/~csci201/lectures /Lecture11/royce1970.pdf

[17] Rational Software, "Rational unified process", Rational Software, Lexington, MA, USA, White Paper TP026B, 2011. [Online]. Available: http://www.ibm.com/developerworks/rational/library/ content/03July/1000/1251/1251_bestpractices_TP026B.pdf

[18] B. W. Boehm, "A spiral model of software development and enhancement," *Computer*, vol. 21, no. 5, pp. 61-72, May 1988. [Online]. doi: 10.1109/2.59

[19] S. McConnell, "Lifecycle planning," in *Rapid Development: Taming Wild Software Schedules,* D. J. Clark and J. Litewka, Eds. Washington, US: Microsoft Press, 1996. pp. 147 - 148

[20] WBS project management, "Creating a Work Breakdown Structure," in *WBS Project Management*, November, 4, 2017. [Online]. Available: https://wbstool.wordpress.com/. Accessed on: June 15, 2018.

[21] Gantt.com, "What is a Gantt Chart?," in *Gantt.com.* [Online]. Available: https://www.gantt.com/. Accessed on: June 10, 2018.

[22] Vertex42, "Simple Gantt Chart," in *Vertex42.com.* [Online]. Available: https://www.vertex42 .com/ExcelTemplates/simple-gantt-chart.html. Accessed on: June 10, 2018.

[23] P. Jalote, "Risk management," in *Software Project Management in Practice*, Indianapolis, IN, US: Pearson Education, 2002.

[24] B. Boehm, "Software Risk Management," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics*), vol. 387, pp.1-19, 1986. [Online]. doi: 10.1007/3-540-51635-2_29

[25] D. Bychkov, "Desktop vs. web applications: a deeper look and comparison," in *seguetech.com*, June, 7, 2013. [Online]. Available: https://www.seguetech.com/desktop-vs-web-applications/. Accessed on: Aug 19, 2018.

[26] B. M. Leiner*, et al., "*Brief History of the Internet,*"* in *www.internetsociety.org,* 1997. [Online]. Available: https://www.internetsociety.org/internet/history-internet/brief-history-internet/. Accessed on: Aug 19, 2018.

[27] R. Fielding, *et al.* "Hypertext transfer protocol -- HTTP/1.1," in *tools.ietf.org*, June,1999. [Online]. Available: https://tools.ietf.org/html/rfc2616. Accessed on: Aug 19, 2018.

[28] D. Vignoni, "File: client-server-model.svg," in *Commons.wikimedia.org,* July, 13, 2011. [Online]. Available: https://commons.wikimedia.org/wiki/File:Client-server-model.svg. Accessed on: Aug 19, 2018.

[29] Source Making, "Design patterns and refactoring," in *sourcemaking.com,* [Online]. Available: https://sourcemaking.com/design_patterns. Accessed on: Aug 19, 2018.

[30] M. Fowler, *Patterns of Enterprise Application Architecture*, Boston, MA, US: Addison-Wesley, 2002.

[31] G. Hohpe, B. Woolf, *Enterprise Integration Patterns*, Boston, MA, US: Addison-Wesley, 2004.

[32] E. Gamma, R. Helm, R. Johnson, J. Vlissides, *Design patterns: elements of reusable object-oriented software*, Boston, MA, US: Addison-Wesley, 1994.

[33] Treehouse Blog, "OS Design patterns: model view controller (part 3)," in *blog.teamtreehouse.com*, July, 14, 2015. [Online]. Available: http://blog.teamtreehouse.com/ios-Design-patterns-model-view-controller-part-3. Accessed on: Aug 18, 2018.

[34] TechTerms, "API (Application Program Interface) definition," in *Techterms.com*, June, 20, 2016. [Online]. Available: https://techterms.com/definition/api. Accessed on: July 30, 2018.

[35] M. Fowler, "Inversion of control," in *martinfowler.com*, January, 23, 2004. [Online]. Available: https://martinfowler.com/articles/injection.html#InversionOfControl. Accessed on: July 30, 2018.

[36] Matthew Mead, "Hollywood principle – Don't call us, we'll call you," in *Matthewtmead.com*, November, 8, 2008. [Online]. Available: http://matthewtmead.com/blog/hollywood-principle-dont-call-us-well-call-you-4/. Accessed on: July 30, 2018.

[37] PHP Group, "Hypertext Preprocessor," in *php.net,* 2018. [Online]. Available: https://secure.php.net/. Accessed on: Aug 20, 2018.

[38] W3Techs, "Usage statistics and market share of PHP for websites," in *w3techs.com,* 2018. [Online]. Available: https://w3techs.com/technologies/details/pl-php/all/all. Accessed on: Aug 20, 2018.

[39] Archer Software, "Top 7 Best PHP Frameworks for 2017," in *www.archer-soft.com, May*, 24, 2017. [Online]. Available: http://www.archer-soft.com/en/blog/top-7-best-php-frameworks-2017. Accessed on: Aug 20, 2018.

[40] JavaScript Community, "Free JavaScript training, resources and examples for the community," in *javascript.com,* 2018. [Online]. Available: https://www.javascript.com/. Accessed on: Aug 13, 2018.

[41] Stack Overflow, "Stack overflow developer survey 2018," in *stackoverflow.com,* 2018. [Online]. Available: https://insights.stackoverflow.com/survey/2018. Accessed on: Aug 20, 2018.

[42] Node.js Foundation, "Node.js," in *nodejs.org,* 2018. [Online]. Available: https://nodejs.org/en/. Accessed on: Aug 13, 2018.

[43] Angular, "Angular docs," in *angular.io,* 2018. [Online]. Available: https://angular.io/. Accessed on: Aug 13, 2018.

[44] React, "React – a JavaScript library for building user interfaces," in *reactjs.org,* 2018. [Online]. Available: https://reactjs.org/. Accessed on: Aug 13, 2018.

[45] Python, "Welcome to Python.org," in *python.org,* 2018. [Online]. Available: https://www.python.org/. Accessed on: Aug 20, 2018.

[46] Django, "Django overview," in *www.djangoproject.com*, 2018. [Online]. Available: https://www.djangoproject.com/. Accessed on: Aug 20, 2018.

[47] Ruby, "Ruby," in *www.ruby-lang.org*, 2018. [Online]. Available: https://www.ruby-lang.org/en/. Accessed on: Aug 20, 2018.

[48] Rails, “Ruby on Rails,” in *rubyonrails.org,* 2018. [Online]. Available: https://rubyonrails.org/. Accessed on: Aug 25, 2018.

[49] Oracle, “Oracle technology network for Java developers,” in *oracle.com*, 2018. [Online]. Available: http://www.oracle.com/technetwork/java/index.html. Accessed on: Aug 25, 2018.

[50] Zipcode, “Why Java skills matter now, and will in the future,” in *zipcodewilmington.com*, Jan, 25, 2016. [Online]. Available: http://www.zipcodewilmington.com/blog/why-java-skills-matter -now-and-in-the-future. Accessed on: Aug 26, 2018.

[51] Microsoft ASP.NET, “The official microsoft ASP.NET site,” in *www.asp.net*. [Online]. Available: https://www.asp.net/. Accessed on: Aug 21, 2018.

[52] Oracle, “JavaServer Pages Technology,” in *oracle.com*. [Online]. Available: http://www.oracle. com/technetwork/java/index-jsp-138231.html. Accessed on: Aug 21, 2018.

[53] Oracle, “Java 2 Platform, Enterprise Edition (J2EE) Overview,” in *oracle.com*. [Online]. Available: https://www.oracle.com/technetwork/java/javaee/appmodel-135059.html. Accessed on: Aug 21, 2018.

[54] Oracle, “Expression language - the Java EE 6 tutorial,” in *docs.oracle.com*. [Online]. Available: https://docs.oracle.com/javaee/6/tutorial/doc/gjddd.html. Accessed on: Aug 21, 2018.

[55] Oracle, “Using JSTL - The Java EE 5 Tutorial,” in *docs.oracle.com*. [Online]. Available: https://docs.oracle.com/javaee/5/tutorial/doc/bnake.html. Accessed on: Aug 21, 2018.

[56] Oracle, “Java Servlet Technology,” in *oracle.com*. [Online]. Available: https://www.oracle.com/ technetwork/java/index-jsp-135475.html. Accessed on: Aug 21, 2018.

[57] JavaTpoint, “MVC in JSP - javatpoint,” in *www.javatpoint.com.* [Online]. Available: https://www.javatpoint.com/MVC-in-jsp. Accessed on: Aug 21, 2018.

[58] M. Fowler, “Front Controller,” in *martinfowler.com*. [Online]. Available: https://www.martinfowler.com/eaaCatalog/frontController.html. Accessed on: Aug 21, 2018.

[59] Oracle, “JavaServer Faces Technology,” in *oracle.com*. [Online]. Available: http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html. Accessed on: Aug 21, 2018.

[60] Oracle, “Introduction to Facelets,” in *javaee.github.io*. [Online]. Available: https://javaee.github .io/tutorial/jsf-facelets.html. Accessed on: Aug 21, 2018.

[61] Pivotal, “Web on Servlet Stack,” in *docs.spring.io*, July, 26, 2108. [Online]. Available: https://docs.spring.io/spring/docs/current/spring-framework-reference/web.html. Accessed on: Aug 21, 2018.

[62] Serhiy, “7 Best Java frameworks for 2016,” in *Romexsoft,* October, 5, 2016. [Online]. Available: https://www.romexsoft.com/blog/7-best-java-frameworks-for-2016/. Accessed on: Aug 21, 2018.

[63] Freemarker, “What is Apache FreeMarker,” in *Apache FreeMarker,* Aug, 3, 2018. [Online]. Available: https://freemarker.apache.org/. Accessed on: Sept 6, 2018.

[64] Apache Software Foundation, “The Apache Velocity Project,” in *Velocity.apache.org*. [Online]. Available: http://velocity.apache.org/. Accessed on: Sept 6, 2018.

[65] Thymeleaf, “Thymeleaf,” in *Thymeleaf.org,* November, 5, 2017. [Online]. Available: https://www.thymeleaf.org/. Accessed on: Aug 21, 2018.

[66] M. Richards, “Software architecture patterns,” in *O'Reilly Media,* August, 15, 2015. [Online]. Available: https://www.oreilly.com/ideas/software-architecture-patterns/page/2/layered-architecture. Accessed on: Aug 3, 2018.

[67] J. Palermo, “The onion architecture: part 1,” in *Programming with Palermo,* July, 29, 2008. [Online]. Available: https://jeffreypalermo.com/2008/07/the-onion-architecture-part-1/. Accessed on: Aug 2, 2018.

[68] J. Palermo, “The onion architecture: part 2,” in *Programming with Palermo,* July, 29, 2008. [Online]. Available: https://jeffreypalermo.com/2008/07/the-onion-architecture-part-2/. Accessed on: Aug 2, 2018.

[69] J. Palermo, “The onion architecture: part 3,” in *Programming with Palermo,* July, 29, 2008. [Online]. Available: https://jeffreypalermo.com/2008/08/the-onion-architecture-part-3/. Accessed on: Aug 2, 2018.

[70] Object Oriented Design, “Dependency Inversion Principle,” in *Oodesign.com.* [Online]. Available: https://www.oodesign.com/dependency-inversion-principle.html. Accessed on: Aug 2, 2018.

[71] M. Fowler, “Inversion of control containers and the dependency injection pattern,” in *martinfowler.com*, January, 23, 2004. [Online]. Available: https://martinfowler.com/articles /injection.html. Accessed on: July 30, 2018.

[72] M. Fowler, “Forms of dependency injection,” in *martinfowler.com*, January, 23, 2004. [Online]. Available: https://martinfowler.com/articles/injection.html#FormsOfDependencyInjection. Accessed on: July 30, 2018.

[73] PicoContainer, “PicoContainer,” in *Picocontainer.com,* Sept, 10, 2017. [Online]. Available: http://picocontainer.com/. Accessed on: July 30, 2018.

[74] Pivotal, “Spring.io,” in *Spring.io*. [Online]. Available: https://spring.io/. Accessed on: June 02, 2018.

[75] Pivotal, “Spring boot,” in *Spring.io*. [Online]. Available: https://spring.io/projects/spring-boot. Accessed on: Aug 28, 2018.

[76] P. Webb, *et al.,* “Appendix E. The Executable Jar Format,” in *Docs.spring.io*. [Online]. Available: https://docs.spring.io/spring-boot/docs/current/reference/html/executable-jar.html. Accessed on: Sept 6, 2018.

[77] Apache, “Realm configuration How-to,” in *Tomcat.apache.org*, August, 11, 2018. [Online]. Available: https://tomcat.apache.org/tomcat-9.0-doc/realm-howto.html. Accessed on: Aug 29, 2018.

[78] Java Community Process, “Servlet 4.0 Specification,” in *Jcp.org.* [Online]. Available: https://jcp.org/aboutJava/communityprocess/final/jsr369/index.html. Accessed on: Aug 29, 2018.

[79] Secret Double Octopus Staff, "What's the difference between authentication and authorization," in *Secret Double Octopus*, September, 19, 2017. [Online]. Available: https://doubleoctopus.com /blog/authentication-vs-authorization/. Accessed on: Aug 28, 2018.

[80] Pivotal, "Spring security," in *Spring.io*. [Online]. Available: https://spring.io/projects/ spring-security. Accessed on: Aug 28, 2018.

[81] Loredana Crusoveanu, "Spring security: Authentication with a database-backed userDetailsService," in *Baeldung*, April,18,2017. [Online]. Available: https://www.baeldung.com/ spring-security-authentication-with-a-database. Accessed on: July 25, 2018.

[82] Eugen Paraschiv, "Password encoding with spring," in *Baeldung*, June, 6, 2018. [Online]. Available: https://www.baeldung.com/spring-security-registration-password-encoding-bcrypt. Accessed on: July 28, 2018.

[83] Baeldung, "Spring boot security auto-configuration," in *Baeldung*, February, 23, 2018. [Online]. Available: https://www.baeldung.com/spring-boot-security-autoconfiguration. Accessed on: July 27, 2018.

[84] OWASP, "OWASP top 10 - 2017," in *Owasp.org*. [Online]. Available: https://www.owasp.org /images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf. Accessed on: Aug 29, 2018.

[85] E. F. Codd, "A relational model of data for large shared data banks", *Communication of the ACM,* Vol. 13, No. 6, June 1970. [Online]. Available: https://www.seas.upenn.edu/~zives/03f/cis550 /codd.pdf

[86] T. Haerder, A. Reuter, "Principles of transaction-oriented database recovery", *Computing Surveys,* Vol. 15, No. 4, December 1983. [Online]. Available: https://sites.fas.harvard.edu/~cs265 /papers/haerder-1983.pdf

[87] W3schools.com, "SQL Introduction," in *W3schools.com.* [Online]. Available: https://www. w3schools.com/sql/sql_intro.asp. Accessed on: Aug 21, 2018.

[88] DB-Engines, "DB-Engines ranking," in *Db-engines.com.* [Online]. Available: https://db-engines .com/en/ranking. Accessed on: Aug 21, 2018.

[89] Oracle, "Oracle Database Technologies," in *Oracle.com.* [Online]. Available: https://www.oracle.com/database/technologies/index.html. Accessed on: Aug 21, 2018.

[90] MySQL, "Mysql," in *Mysql.com.* [Online]. Available: https://www.mysql.com/. Accessed on: Aug 21, 2018.

[91] Microsoft, "Microsoft Data Platform," in *Microsoft SQL Server.* [Online]. Available: https://www.microsoft.com/en-us/sql-server/. Accessed on: Aug 21, 2018.

[92] PostgreSQL, "PostgreSQL: The world's most advanced open source database," in *Postgresql.org.* [Online]. Available: https://www.postgresql.org/. Accessed on: Aug 21, 2018.

[93] DB-Engines, "Key-value stores," in *Db-engines.com.* [Online]. Available: https://db-engines. com/en/article/Key-value+Stores. Accessed on: Aug 19, 2018.

[94] Redis, "Redis," in *Redis.io.* [Online]. Available: https://redis.io/. Accessed on: Aug 20, 2018.

[95] Amazon, “Amazon DynamoDB - Overview,” in *Amazon Web Services, Inc.* [Online]. Available: https://aws.amazon.com/dynamodb/. Accessed on: Aug 20, 2018.

[96] DB-Engines, “Wide column stores,” in *Db-engines.com.* [Online]. Available: https://db-engines .com/en/article/Wide+Column+Stores. Accessed on: Aug 18, 2018.

[97] Apache Software Foundation, “Apache Cassandra,” in Cassandra.apache.org. [Online]. Available: http://cassandra.apache.org/. Accessed on: Aug 19, 2018.

[98] HBase, “Welcome to Apache HBase,” in *Hbase.apache.org*, June, 9, 2018. [Online]. Available: https://hbase.apache.org/. Accessed on: Aug 19, 2018.

[99] DB-Engines, “Document stores,” in *Db-engines.com.* [Online]. Available: https://db-engines. com/en/article/Document+Stores. Accessed on: Aug 18, 2018.

[100] MongoDB, “MongoDB for giant ideas,” in *MongoDB.* [Online]. Available: https://www. mongodb.com/. Accessed on: Aug 18, 2018.

[101] Couchbase, “NoSQL Engagement Database - Couchbase,” in *Couchbase.com.* [Online]. Available: https://www.couchbase.com/. Accessed on: Aug 18, 2018.

[102] DB-Engines, “Graph DBMS,” in *Db-engines.com.* [Online]. Available: https://db-engines.com/ en/article/Graph+DBMS. Accessed on: Aug 18, 2018.

[103] Neo4j, “The Neo4j graph platform,” in *neo4j.com.* [Online]. Available: https://neo4j.com/. Accessed on: Aug 18, 2018.

[104] MySQL, “Connecting to MySQL using the JDBC DriverManager interface,” in *Dev.mysql.com.* [Online]. Available: https://dev.mysql.com/doc/connector-j/5.1/en/connector-J-usagenotes-connect -drivermanager.htm. Accessed on: Aug 17, 2018.

[105] Techopedia, “What is Object-Relational Mapping (ORM)?,” in *Techopedia.com.* [Online]. Available: https://www.techopedia.com/definition/24200/object-relational-mapping--orm. Accessed on: Aug 30, 2018.

[106] E. Hieatt, R. Mee, “Repository,” in in *martinfowler.com*. [Online]. Available: https://martinfowler.com/eaaCatalog/repository.html. Accessed on: July 30, 2018.

[107] Oracle, “Java persistence api,” in *Oracle.com*. [Online]. Available: http://www.oracle.com/ technetwork/java/javaee/tech/persistence-jsp-140049.html. Accessed on: Aug 30, 2018.

[108] Hibernate, “Hibernate. Everything data,” in *Hibernate.org*. [Online]. Available: http://hibernate.org/. Accessed on: Aug 30, 2018.

[109] MyBatis, “The MyBatis Blog,” in *Blog.mybatis.org.* [Online]. Available: http://blog.mybatis .org/. Accessed on: Aug 28, 2018.

[110] iBatis, “iBatis home,” in *Ibatis.apache.org.* [Online]. Available: http://ibatis.apache.org/. Accessed on: Aug 28, 2018.

[111] Apache Software Foundation, “Welcome to the Apache software foundation,” in *Apache.org.* [Online]. Available: https://www.apache.org/. Accessed on: Aug 28, 2018.

[112] Tutorialspoint, “iBatis overview,” in *www.tutorialspoint.com.* [Online]. Available: https://www.tutorialspoint.com/ibatis/ibatis_overview.htm. Accessed on: Aug 28, 2018.

[113] M. Pickering, “What are alternatives to the Hibernate ORM library?,” in *quora.com,* Feb, 5, 2017. [Online]. Available: https://www.quora.com/What-are-alternatives-to-the-Hibernate-ORM -library. Accessed on: Aug 28, 2018.

[114] K. L. Nitin, S. Ananya, K. Mahalakshmi, S. Sangeetha, “iBatis, Hibernate, and Jpa: Which is right for you?,” in *Javaworld,* July, 15, 2008. [Online]. Available: https://www.javaworld.com/ article/2077875/open-source-tools/ibatis--hibernate--and-jpa--which-is-right-for-you-.html?page=7. Accessed on: Aug 28, 2018.

[115] Pankaj, “Spring Hibernate integration example tutorial,” in *JournalDev,* April, 2, 2018. [Online]. Available: https://www.journaldev.com/3524/spring-hibernate-integration-example-tutorial. Accessed on: July 15, 2018.

[116] A. Smith, “Introduction to iBatis,” in *Javalobby.org.* [Online]. Available: https://www.javalobby .org/articles/ibatis-introduction/. Accessed on: Aug 28, 2018.

[117] Pivotal, “Spring data,” in Projects.spring.io. [Online]. Available: http://projects.spring.io/ spring-data/. Accessed on: Aug 28, 2018.

[118] W3C, “HTML 5.2,” in *W3.org,* December, 14, 2017. [Online]. Available: https://www.w3.org/TR/html52/. Accessed on: Sept 6, 2018.

[119] ExE-Boss, *et al.,* “CSS: cascading style sheets,” in *mdn web docs,* Jun, 11, 2018. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/CSS. Accessed on: Sept 6, 2018.

[120] M. Otto, *et al.,* “Bootstrap,” in *Getbootstrap.com*. [Online]. Available: https://getbootstrap.com/. Accessed on: Sept 6, 2018.

[121] B. Aston, “A brief history of JavaScript,” in *medium*, Apr, 1, 2015. [Online]. Available: https://medium.com/@benastontweet/lesson-1a-the-history-of-javascript-8c1ce3bffb17. Accessed on: Sept 6, 2018.

[122] JS foundation, “jQuery,” in *Jquery.com.* [Online]. Available: https://jquery.com/. Accessed on: Sept 7, 2018.

[123] TechTerms, “Ajax Definition,” in *Techterms.com*, April, 13, 2011. [Online]. Available: https://techterms.com/definition/ajax. Accessed on: Sept 6, 2018.

[124] DataTables, “Table plug-in for jQuery,” in *Datatables.net*. [Online]. Available: https://datatables.net/. Accessed on: Aug 16, 2018.

[125] Apache Software Foundation, “Welcome to Apache Maven,” in *Maven.apache.org,* Sept, 09, 2018. [Online]. Available: https://maven.apache.org/. Accessed on: Sept 7, 2018.

[126] Gradle, “Gradle build tool,” in *Gradle*. [Online]. Available: https://gradle.org/. Accessed on: Sept 7, 2018.

[127] Git, “Git,” in *Git-scm.com.* [Online]. Available: https://git-scm.com/. Accessed on: Sept 8, 2018.

[128] NetBeans, "Welcome to NetBeans," in *Netbeans.org.* [Online]. Available: https://netbeans.org/. Accessed on: Sept 8, 2018.

[129] IntelliJ IDEA, "IntelliJ IDEA: the Java IDE for professional developers by JetBrains," in *JetBrains.* [Online]. Available: https://www.jetbrains.com/idea/. Accessed on: Sept 8, 2018.

[130] Eclipse Foundation, "The Platform for Open Innovation and Collaboration," in *Eclipse.org*. [Online]. Available: https://www.eclipse.org/. Accessed on: Sept 8, 2018.

[131] Astah, "Astah - software design tools for agile teams with UML, ER diagram, flowchart, mindmap and more," in *Astah.net.* [Online]. Available: http://astah.net/. Accessed on: Sept 8, 2018.

[132] H. Sharp, A. Finkelstein and G. Galal, "Stakeholder identification in the requirements engineering process," *Proceedings. Tenth International Workshop on Database and Expert Systems Applications. DEXA 99*, Florence, Italy, 1999, pp. 387-391. December 1999. [Online]. Available: http://discovery .ucl.ac.uk/744/1/1.7_stake.pdf

[133] K. Eason, *Information Technology and Organisational Change,* London, UK: Taylor and Francis, 1998

[134] H. Hochheiser, J. H. Feng, J. Lazar, *Research methods in Human-Computer Interaction,* 2nd ed., San Francisco, US: Morgan Kaufmann, 2017.

[135] A. Sutcliffe, "RE Tasks and processes," in *User-Centred Requirements Engineering***,** G. Martin, Ed. London, UK: Springer, 2002, pp. 45-46. [Online]. Available: https://books.google.co.uk/books?id=QH_lBwAAQBAJ&printsec=frontcover&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false

[136] Wiki, "Domain expert," in *wiki.c2.com,* October, 15, 2013. [Online] Available: http://wiki.c2.com/

[137] Tutorialspoint, "Software requirements," in *www.tutorialspoint.com,* July, 21, 2018. [Online] Available: https://www.tutorialspoint.com/software_engineering/software_requirements.htm. Accessed on: July 24, 2018.

[138] I. Garcia, C. Pacheco, P. Sumano, "Use of questionnaire-based appraisal to improve the software acquisition process in small and medium enterprises," in *Software Engineering Research, Management and Applications,* vol.150, L. Roger, Ed. Berlin, Germany: Springer-Verlag, 2008, p. 17 [Online]. Available: https://books.google.co.uk/books?id=SPopzSBYj_EC&printsec=frontcover&dq=editions:9vmt_IorJgQC&hl=en&sa=X&ved=0ahUKEwjfndHdwfcAhWFLsAKHe7iA0YQ6AEIKTAA#v=onepage&q&f=false

[139] B. Hanington, B. Martin, "Task analysis," in *Universal Methods of Design*, Beverly MA, US: Rockport Publishers, 2012, pp. 174 - 175

[140] B. Kirwan, L. K. Ainsworth, "Task description methods," in *A guide to Task Analysis: the Task Analysis working group,* London, UK: Taylor and Francis, 2005, pp. 102 - 110.

[141] J. Neighbors, "Software Construction Using Components," Ph.D. thesis, Department of Information and Computer Science, University of California, Irvine, 1981.

[142] R. Prieto-Diaz, "Domain analysis: an introduction", *Software Engineering Notes*, vol. 15, no. 2, pp. 47 - 53, Apr. 1990. [Online]. Available: http://www.engr.sjsu.edu/fayad/current.courses/cmpe202-

spring2015/docs/CmpE202-SE-Link-Part-Two-Fall2013/11-Domain%20Analysis/Domain%20Analysis%20-%20An%20Introduction.pdf

[143] E. Evans, *Domain-Driven Design: Tackling Complexity in the Heart of Software,* Upper Saddle River, NJ, US: Addison Wesley, 2004.

[144] M. Fowler, *Uml Distilled: A brief guide to the standard object modeling language*, 3rd ed., Boston, MA, US: Pearson Education, Sept, 2003, pp.35 – 38.

[145] J. Robertson, S. Robertson, "Functional requirements," in *Mastering the Requirements Process: Getting Requirements Right*, 3rd ed. Boston,US: Addison-Wesley, 2012, pp. 223-225.

[146] Techopedia, "What is a minimum viable product (MVP)? - definition from techopedia," in *Techopedia.com*. [Online]. Available: https://www.techopedia.com/definition/27809/minimum-viable-product-mvp Accessed on: July 15, 2018.

[147] I. Sommerville, "Software requirements," in *Software Engineering*, 8th ed, Harlow, UK: Pearson Education Limited, 2007, p.119

[148] Guru99, "What is Non-functional Testing," in *Meet Guru99 - Free Training Tutorials & Video for IT Courses,* [Online]. Available: https://www.guru99.com/non-functional-testing.html. Accessed on: Aug 6, 2018.

[149] ISO, "ISO 9241-11:2018, Ergonomics of human-system interaction - Part 11: Usability: Definitions and concepts," in *Online Browsing Platform*. 2018 [Online]. Available: https://www.iso.org/obp/ui/#iso:std:iso:9241:-11:ed-2:v1:en. Accessed on: Aug 9, 2018.

[150] R. B. Miller, "Response time in man-computer conversational transactions," in *Fall Joint Computer Conference*, Poughkeepsie, New York, 1968, pp. 267 - 277. [Online]. Available: http://yusufarslan.net/sites/yusufarslan.net/files/upload/content/Miller1968.pdf

[151] J. Nielsen, "Response times: the 3 important limits," in *Nielsen Norman group*, January, 1, 1993. [Online]. Available: https://www.nngroup.com/articles/response-times-3-important-limits/. Accessed on: Sept 7, 2018.

[152] OMG, "Unified Modeling Language," in *Omg.org,* December, 5, 2017. [Online]. Available: https://www.omg.org/spec/UML/2.5.1/PDF. Accessed on: Aug 11, 2018.

[153] J. Kettenis, "Getting started with use case modeling," Oracle, Redwood Shores, CA, USA, May 2007, p.8, [Online]. Available: http://www.oracle.com/technetwork/testcontent/gettingstartedwithusecasemodeling-133857.pdf.

[154] A. Cockburn, "Three named goal levels," in *Writing Effective Uses Cases.* S. Lilly *et al.*, Eds, New Jersey, US: Addison-Wesley, 2001 pp.69 – 86.

[155] S. Bennett, S. McRobb, R. Farmer, "Moving into design," in *Object Oriented Systems Analysis and Design using UML*, 3rd ed., London,UK: McGraw-Hill, 2005, p. 242
[xxx] Object Oriented Systems Analysis and Design using UML, Simon Bennett, Steve McRobb and Ray Farmer, pg242

[156] C. Larman, *Applying UML and Patterns: An introduction to object-oriented analysis and design and iterative development*, 3rd ed. New Jersey, US: Prentice Hall, 2004

[157] P. Wilcox, "Effective Communication of Scenarios of Usage," HWISE, Edinburgh, UK, Tech. Paper IST-2000-28402-2000-20002, Sept. 2003

[158] D. Bell, “The Sequence Diagram,” in *Ibm.com,* February, 16, 2004. [Online]. Available: https://www.ibm.com/developerworks/rational/library/3101.html. Accessed on: Aug 10, 2018

[159] T. J. Teorey*, et al., Database Design: Know It All,* San Francisco, US: Morgan Kaufmann, 2008

[160] Computer Science - University of Regina, “Data modeling and entity relationship diagram (ERD),” in *Cs.uregina.ca,* [Online]. Available: http://www.cs.uregina.ca/Links/ class-info/215/erd/. Accessed on: Aug 29, 2018

[161] Guru99, “What is data modelling? Conceptual, logical, & physical data models,” in *Meet Guru99 - Free Training Tutorials & Video for IT Courses,* [Online]. Available: https://www.guru99.com/data-modelling-conceptual-logical.html. Accessed on: Aug 22, 2018.

[162] Helen Winter, “Gathering data requirements,” in *Business Bullet,* February, 5, 2017, [Online]. Available: http://www.businessbullet.co.uk/business-analysis/gathering-data-requirements/. Accessed on: Aug 28, 2018

[163] H.V. Jagadish, T. Nadeau, S. S. Lightstone, T. J. Teorey, *Database Modeling and Design*, 5th ed., San Francisco, US: Morgan Kaufmann, 2011

[164] J. H. Carlisle, “Evaluating the impact of office automation on top management communication,” *Proceedings of the June 7–10, 1976, National Computer Conference and Exposition*. pp. 611–616, June 1976

[165] S. K. Card, T. H. Moran and A. Newell, *The Psychology of Human–Computer Interaction*. London, UK: L. Erlbaum Associates, 1983

[166] D. Norman, *The Design of Everyday Things*, revised and extended ed. New York, US: BasicBooks, 2002.

[167] J. Nielsen, “10 heuristics for user Interface design,” in *Nielsen Norman Group,* January, 1, 1995. [Online]. Available: https://www.nngroup.com/articles/ten-usability-heuristics/. Accessed on: July 28, 2018.

[168] B. Shneiderman, “Shneiderman's eight golden rules of interface design,” in *Designprinciplesftw.com,* Sept, 12, 2103. [Online]. Available: https://www.designprinciplesftw.com/ collections/shneidermans-eight-golden-rules-of-interface-design. Accessed on: July 28, 2018.

[169] N. Provos, D. Mazieres, “Bcrypt Algorithm,” in *Usenix.org*, April, 28, 1999. [Online]. Available: https://www.usenix.org/legacy/publications/library/proceedings/usenix99/full_papers/ provos/provos_html/node5.html. Accessed on: June 28, 2018.

[170] Mkyong, “Spring security password hashing example,” in *Mkyong.com*, May, 21, 2014. [Online]. Available: https://www.mkyong.com/spring-security/spring-security-password -hashing-example/ . Accessed on: June 28, 2018.

[171] Guru99, “What is Software Testing? Introduction, Basics & Importance,” in *Meet Guru99 - Free Training Tutorials & Video for IT Course.,* [Online]. Available: https://www.guru99.com /software-testing-introduction-importance.html. Accessed on: Aug 5, 2018.

[172] Software Testing Fundamentals, “Unit Testing,” in *Software Testing Fundamentals.* [Online]. Available: http://softwaretestingfundamentals.com/unit-testing/. Accessed on: Aug 5, 2018.

[173] Software Testing Fundamentals, “Integration Testing,” in *Software Testing Fundamentals.* [Online]. Available: http://softwaretestingfundamentals.com/integration-testing/. Accessed on: Aug 5, 2018.

[174] Software Testing Fundamentals, “Acceptance Testing,” in *Software Testing Fundamentals.* [Online]. Available: http://softwaretestingfundamentals.com/acceptance-testing/. Accessed on: Aug 5, 2018.

[175] Agile Data, “Introduction to Test Driven Development (TDD),” in *Agiledata.org.* [Online]. Available: http://agiledata.org/essays/tdd.html. Accessed on: Aug 5, 2018.

[176] Software Testing Fundamentals, “White Box Testing,” in *Software Testing Fundamentals.* [Online]. Available: http://softwaretestingfundamentals.com/white-box-testing/. Accessed on: Aug 5, 2018.

[177] Software Testing Fundamentals, “Black Box Testing,” in *Software Testing Fundamentals.* [Online]. Available: http://softwaretestingfundamentals.com/black-box-testing/. Accessed on: Aug 5, 2018.

[178] JUnit, “JUnit – About,” in *Junit.org,* Feb, 07, 2018. [Online]. Available: https://junit.org/junit4/. Accessed on: Aug 10, 2018.

[179] Mockito, “Mockito framework site,” in *Site.mockito.org.* [Online]. Available: https://site.mockito.org/. Accessed on: Aug 10, 2018.

[180] B. Alex, L. Taylor, R. Winch, G. Hillert, J. Grandja, J. Bryant, “Spring MVC Test Integration,” in *Docs.spring.io.* [Online]. Available: https://docs.spring.io/spring-security/site/docs/current/reference/html/test-mockmvc.html. Accessed on: Aug 10, 2018.

[181] M. Fowler, “Test Coverage,” in *martinfowler.com*, April, 17, 2012. [Online]. Available: https://martinfowler.com/bliki/TestCoverage.html. Accessed on: Aug 12, 2018.

[182] FullCalendar, “JavaScript Event Calendar,” in *Fullcalendar.io.* [Online]. Available: https://fullcalendar.io/. Accessed on: Sept 2, 2018

[183] Google, “Handle API Errors,” in *Google Developers.* [Online]. Available: https://developers.google.com/calendar/v3/errors. Accessed on: Sept 2, 2018.

[184] Bootstrap, “Modal,” in *Getbootstrap.com.* [Online]. Available: https://getbootstrap.com/docs/4.1/components/modal/. Accessed on: Sept 2, 2018.

[185] JS foundation, “Dialog,” in *Jqueryui.com.* [Online]. Available: https://jqueryui.com/dialog/. Accessed on: Sept 2, 2018.

[186] OAuth, “OAuth 2.0” in *Oauth.net.* [Online]. Available: https://oauth.net/2/. Accessed on: Sept 2, 2018.

[187] Google, “Simulate mobile devices with device mode,” in *Google Developers.* [Online]. Available: https://developers.google.com/web/tools/chrome-devtools/device-mode/. Accessed on: Sept 2, 2018.

[188] Cucumber, “Tools and techniques that elevate teams to greatness,” in *cucumber.io*. [Online]. Available: https://cucumber.io/. Accessed on: Sept 1, 2018.

[189] Logback, “Logback project,” in *Logback.qos.ch.* [Online]. Available: https://logback.qos.ch/. Accessed on: Sept 1, 2018.

[190] Log4j 2, “Apache Log4j 2,” in *Logging.apache.org.* [Online]. Available: https://logging.apache.org/log4j/2.x/. Accessed on: Sept 1, 2018.

[191] P. Webb, *et al.* “Spring boot reference guide - loggin,” in *Docs.spring.io*. [Online]. Available: https://docs.spring.io/spring-boot/docs/current/reference/html/howto-logging.html. Accessed on: June 15, 2018.

Appendix A. Acronyms

CiCS Corporate Information and Computing Services
CRUD Create Update Delete
CSS Cascading Style Sheets
ERD Entity Relationship Diagram
HCI Human Computer Interaction
HTA Hierarchical Task Analysis
HTML5 Hyper Text Markup Language
IDE Integrated Development Environment
JSON JavaScript Object Notation
JVM Java Virtual Machine
MSA Meeting Scheduler Application
MVP Minimum Viable Product
ORM Object Relational Mapping
OWASP Open Web Application Security Project
SSS Student Support Services
SQL Structured Query Language
TDD Test Driven Development
UI User Interface
UAT User Acceptance Test
UML Unified Modelling Language
UREC University Research Ethics Committee

Appendix B. Risk Management Plan

*The table structure for the risk management plan as well as the method for calculating the risk exposure was provided by Jalote [23, p.104].

B.1 Planning Phase Risk Management Plan

This plan was produced during project planning and before any core project activities like requirement analysis, design or implementation had begun.

* sorted by Risk Exposure

ID	Risks	Probability	Impact	Risk Exposure	Mitigation Plan
1	Failure to identify a risk	0.4	? (have to assume the worst	4	- Review current processes within the client's domain - Discuss with client potential risks

			value – 10)		- Discuss with supervisor potential risks - Study previous dissertations to identify cases where risk impacted on similar projects
2	Failure to extract all the requirements Incomplete project requirements (due to inexperience in the process of extracting requirements or client omitting requirements)	0.4	9	3.6	- Use a range of requirements elicitation techniques - Interview more than one user - Create an early prototype
3	Project deadline missed due to inefficient time management	0.3	10	3	- Derive a project schedule to follow and set milestones to meet - Revise schedule frequently to ensure work remains on target
4	Misunderstanding project requirements	0.3	9	2.7	- Use a range of requirements elicitation techniques - Interview more than one user - Create an early prototype
5	Failure to apply a tool or technology efficiently in the project (due to limited time available to learn and understand the tool/ technology)	0.3	7	2.1	- Where possible use tools and technologies which are familiar to the developer - if an unfamiliar technology has to be used then consider one with good documentation, good community support and not very steep learning curve
6	Technical challenges due to using unproven/ new technologies and libraries (which might have limited documentation or bugs)	0.3	7	2.1	- Where possible use established, proved technologies - Only experiment for a specified time with new technologies.

					Have alternative choices ready to use if they don't meet expectations
7	Failing to follow rules and legislation regarding the data protection	0.2	10	2	- Identify early the personal data that is required to be stored in the system - Follow all the relevant University procedures - Put in place any security measures required to protect the system from a data breach
8	Developer machine crash	0.2	10	2	- Back up application code and dissertation report at regular interval (at least on a daily basis)
9	Failure to estimate well the probability or impact of an identified risk	0.2	? (have to assume the worst value – 10)	2	- Study previous dissertations to identify cases where risk impacted on similar projects - Discuss with supervisor potential or identified risks
10	Long absence of the developer due to illness or other unexpected circumstances	0.2	9	1.8	- Been methodical with the project work and not falling behind schedule so that the impact of an unscheduled absence is minimised - Schedule to complete implementation, testing and report earlier than deadline (use the time at the end to handle the delay due to the unexpected absence – if the risk does not materialise, use the time for report revision)

11	Long absence of client due to illness or unscheduled holidays	0.2	8	1.6	- Frequent client meetings to get constant feedback so that the impact of a unexpected client absence will be mitigated - Discuss with the client to identify who could be consulted instead in their absence
12	Long absence of supervisor due to illness or unscheduled holidays	0.2	8	1.6	- Frequent supervisor meetings to get constant feedback so that the impact of an unexpected absence will be mitigated - Discuss with the supervisor to identify who would provide support/ guidance if the risk materialises
13	Change of requirements mid project	0.2	8	1.6	- Arrange frequent meetings with the client so is always aware of the work under way. That way any changes would be brought up and dealt with as early as possible
14	The system will be implemented using various technologies which are not familiar to the developer and their compatibility and integration with each other would be first tested during the project. An incompatible technology could require the application of an alternative which in turn would require more time to learn and test. The delays could affect the project completion time	0.2	7	1.4	- Each chosen technology would be researched to identify compatibility issues - For each chosen technology alternative technologies would also be considered so that time is saved from researching for one later on (if this would be required)
15	Project can only meet the specification and quality required with proprietary or expensive software or additional resources which are not available	0.1	9	0.9	- Research alternatives - Keep supervisor up to date with research results, especially if

					such a risk is identified - During the requirements phase identify requirements which are not feasible under the circumstances or require resources that are not available. Consult with client and move these out of project scope

B.2 Phase-1 Risk Management Plan

This plan was produced while evaluating risks during the first spiral iteration (see project schedule in Appendix M).

CHANGES FROM PREVIOUS PLAN

Risk ID: 6
Slower chance from an impact from using an unproven technology, as the selected technologies to be used (after some initial investigation) are mainly established and proven technologies.

Risk ID: 7
It has been assigned lower probability as there is no requirement to store personal data in the system (except Faculty Officer email and which department they belong, but this has been checked with supervisor and is fine to use for the MSA). The risk still remained in the plan in case some requirements would change and personal data would need to be stored in the system.

RISK MANAGEMENT PLAN

* sorted by Risk Exposure

ID	Risks	Probability	Impact	Risk Exposure	Mitigation Plan
1	Failure to identify a risk	0.4	? (have to assume the worst value – 10)	4	- Review current processes within the client's domain - Discuss with client potential risks - Discuss with supervisor potential risks - Study previous dissertations to identify cases where risk impacted on similar projects

2	Failure to extract all the requirements Incomplete project requirements (due to inexperience in the process of extracting requirements or client omitting requirements)	0.4	9	3.6	- Use a range of requirements elicitation techniques - Interview more than one user - Create an early prototype
3	Project deadline missed due to inefficient time management	0.3	10	3	- Derive a project schedule to follow and set milestones to meet - Revise schedule frequently to ensure work remains on target
4	Misunderstanding project requirements	0.3	9	2.7	- Use a range of requirements elicitation techniques - Interview more than one user - Create an early prototype
5	Failure to apply a tool or technology efficiently in the project (due to limited time available to learn and understand the tool/ technology)	0.3	7	2.1	- Where possible use tools and technologies which are familiar to the developer - if an unfamiliar technology has to be used then consider one with good documentation, good community support and not very steep learning curve
8	Developer machine crash	0.2	10	2	- Back up application code and dissertation report at regular interval (at least on a daily basis)
9	Failure to estimate well the probability or impact of an identified risk	0.2	? (have to assume the worst value – 10)	2	- Study previous dissertations to identify cases where risk impacted on similar projects - Discuss with supervisor potential or identified risks

10	Long absence of the developer due to illness or other unexpected circumstances	0.2	9	1.8	- Been methodical with the project work and not falling behind schedule so that the impact of an unscheduled absence is minimised - Schedule to complete implementation, testing and report earlier than deadline (use the time at the end to handle the delay due to the unexpected absence – if the risk does not materialise, use the time for report revision)
11	Long absence of client due to illness or unscheduled holidays	0.2	8	1.6	- Frequent client meetings to get constant feedback so that the impact of a unexpected client absence will be mitigated - Discuss with the client to identify who could be consulted instead in their absence
12	Long absence of supervisor due to illness or unscheduled holidays	0.2	8	1.6	- Frequent supervisor meetings to get constant feedback so that the impact of an unexpected absence will be mitigated - Discuss with the supervisor to identify who would provide support/ guidance if the risk materialises
13	Change of requirements mid project	0.2	8	1.6	- Arrange frequent meetings with the client so is always aware of the work under way. That way any changes would be brought up and dealt

					with as early as possible
14	The system will be implemented using various technologies which are not familiar to the developer and their compatibility and integration with each other would be first tested during the project. An incompatible technology could require the application of an alternative which in turn would require more time to learn and test. The delays could affect the project completion time	0.2	7	1.4	- Each chosen technology would be researched to identify compatibility issues - For each chosen technology alternative technologies would also be considered so that time is saved from researching for one later on (if this would be required)
6	Technical challenges due to using unproven/ new technologies and libraries (which might have limited documentation or bugs)	0.2	7	1.4	- Where possible use established, proved technologies - Only experiment for a specified time with new technologies. Have alternative choices ready to use if they don't meet expectations
7	Failing to follow rules and legislation regarding the data protection	0.1	10	1	- Identify early the personal data that is required to be stored in the system - Follow all the relevant University procedures - Put in place any security measures required to protect the system from a data breach
15	Project can only meet the specification and quality required with proprietary or expensive software or additional resources which are not available	0.1	9	0.9	- Research alternatives - Keep supervisor up to date with research results, especially if such a risk is identified - During the requirements phase identify requirements which are not feasible under the circumstances or

					require resources that are not available. Consult with client and move these out of project scope

B.3 Phase-2 Risk Management Plan

This plan was produced while evaluating risks during the second spiral iteration (see project schedule in Appendix M).

CHANGES FROM PREVIOUS PLAN

Risk ID: 15
The probability has been reduced as all main features of the system can be implemented with open source software. The risk remained in the plan in case requirements change.

RISK MANAGEMENT PLAN

* sorted by Risk Exposure

ID	Risks	Probability	Impact	Risk Exposure	Mitigation Plan
1	Failure to identify a risk	0.4	? (have to assume the worst value – 10)	4	- Review current processes within the client's domain - Discuss with client potential risks - Discuss with supervisor potential risks - Study previous dissertations to identify cases where risk impacted on similar projects
2	Failure to extract all the requirements Incomplete project requirements (due to inexperience in the process of extracting requirements or client omitting requirements)	0.4	9	3.6	- Use a range of requirements elicitation techniques - Interview more than one user - Create an early prototype
3	Project deadline missed due to inefficient time management	0.3	10	3	- Derive a project schedule to follow and set milestones to meet

					- Revise schedule frequently to ensure work remains on target
4	Misunderstanding project requirements	0.3	9	2.7	- Use a range of requirements elicitation techniques - Interview more than one user - Create an early prototype
5	Failure to apply a tool or technology efficiently in the project (due to limited time available to learn and understand the tool/ technology)	0.3	7	2.1	- Where possible use tools and technologies which are familiar to the developer - if an unfamiliar technology has to be used then consider one with good documentation, good community support and not very steep learning curve
8	Developer machine crash	0.2	10	2	- Back up application code and dissertation report at regular interval (at least on a daily basis)
9	Failure to estimate well the probability or impact of an identified risk	0.2	? (have to assume the worst value – 10)	2	- Study previous dissertations to identify cases where risk impacted on similar projects - Discuss with supervisor potential or identified risks
10	Long absence of the developer due to illness or other unexpected circumstances	0.2	9	1.8	- Been methodical with the project work and not falling behind schedule so that the impact of an unscheduled absence is minimised - Schedule to complete implementation, testing and report earlier than deadline

					(use the time at the end to handle the delay due to the unexpected absence – if the risk does not materialise, use the time for report revision)
11	Long absence of client due to illness or unscheduled holidays	0.2	8	1.6	- Frequent client meetings to get constant feedback so that the impact of a unexpected client absence will be mitigated - Discuss with the client to identify who could be consulted instead in their absence
12	Long absence of supervisor due to illness or unscheduled holidays	0.2	8	1.6	- Frequent supervisor meetings to get constant feedback so that the impact of an unexpected absence will be mitigated - Discuss with the supervisor to identify who would provide support/ guidance if the risk materialises
13	Change of requirements mid project	0.2	8	1.6	- Arrange frequent meetings with the client so is always aware of the work under way. That way any changes would be brought up and dealt with as early as possible
14	The system will be implemented using various technologies which are not familiar to the developer and their compatibility and integration with each other would be first tested during the project. An incompatible technology could require the application of an alternative which in turn would require	0.2	7	1.4	- Each chosen technology would be researched to identify compatibility issues - For each chosen technology alternative technologies would also be considered so that time is saved

	more time to learn and test. The delays could affect the project completion time				from researching for one later on (if this would be required)
6	Technical challenges due to using unproven/ new technologies and libraries (which might have limited documentation or bugs)	0.2	7	1.4	- Where possible use established, proved technologies - Only experiment for a specified time with new technologies. Have alternative choices ready to use if they don't meet expectations
7	Failing to follow rules and legislation regarding the data protection	0.1	10	1	- Identify early the personal data that is required to be stored in the system - Follow all the relevant University procedures - Put in place any security measures required to protect the system from a data breach
15	Project can only meet the specification and quality required with proprietary or expensive software or additional resources which are not available	0.05	9	0.45	- Research alternatives - Keep supervisor up to date with research results, especially if such a risk is identified - During the requirements phase identify requirements which are not feasible under the circumstances or require resources that are not available. Consult with client and move these out of project scope

B.4 Phase-3 Risk Management Plan

This plan was produced while evaluating risks during the third spiral iteration (see project schedule in Appendix M).

CHANGES FROM PREVIOUS PLAN

Risk ID: 5
The risk probability has been reduced as development has already been completed for booking an appointment. The work involved using most of the technologies that would be required for the system (Spring MVC, Google Calendar API integration, Spring Data and front-end technologies), so confidence has been increased.

Risk ID: 6
The risk impact has been decreased as there does not seem to be a need to use an unproven technology for any major system functionality. It remains in the plan in case some untested technologies are required for the implementation of smaller system features.

Risk ID: 14
The risk probability has been decreased as most of the required technologies have already been used for implementing the feature of booking an appointment. There were no issues with compatibility.

RISK MANAGEMENT PLAN

* sorted by Risk Exposure

ID	Risks	Probability	Impact	Risk Exposure	Mitigation Plan
1	Failure to identify a risk	0.4	? (have to assume the worst value – 10)	4	- Review current processes within the client's domain - Discuss with client potential risks - Discuss with supervisor potential risks - Study previous dissertations to identify cases where risk impacted on similar projects
2	Failure to extract all the requirements Incomplete project requirements (due to inexperience in the process of extracting requirements or client omitting requirements)	0.4	9	3.6	- Use a range of requirements elicitation techniques - Interview more than one user - Create an early prototype
3	Project deadline missed due to inefficient time management	0.3	10	3	- Derive a project schedule to follow and set milestones to meet - Revise schedule frequently to ensure work remains on target

4	Misunderstanding project requirements	0.3	9	2.7	- Use a range of requirements elicitation techniques - Interview more than one user - Create an early prototype
8	Developer machine crash	0.2	10	2	- Back up application code and dissertation report at regular interval (at least on a daily basis)
9	Failure to estimate well the probability or impact of an identified risk	0.2	? (have to assume the worst value – 10)	2	- Study previous dissertations to identify cases where risk impacted on similar projects - Discuss with supervisor potential or identified risks
10	Long absence of the developer due to illness or other unexpected circumstances	0.2	9	1.8	- Been methodical with the project work and not falling behind schedule so that the impact of an unscheduled absence is minimised - Schedule to complete implementation, testing and report earlier than deadline (use the time at the end to handle the delay due to the unexpected absence – if the risk does not materialise, use the time for report revision)
11	Long absence of client due to illness or unscheduled holidays	0.2	8	1.6	- Frequent client meetings to get constant feedback so that the impact of a unexpected client absence will be mitigated - Discuss with the client to identify who could be consulted

					instead in their absence
12	Long absence of supervisor due to illness or unscheduled holidays	0.2	8	1.6	- Frequent supervisor meetings to get constant feedback so that the impact of an unexpected absence will be mitigated - Discuss with the supervisor to identify who would provide support/ guidance if the risk materialises
13	Change of requirements mid project	0.2	8	1.6	- Arrange frequent meetings with the client so is always aware of the work under way. That way any changes would be brought up and dealt with as early as possible
5	Failure to apply a tool or technology efficiently in the project (due to limited time available to learn and understand the tool/ technology)	0.2	7	1.4	- Where possible use tools and technologies which are familiar to the developer - if an unfamiliar technology has to be used then consider one with good documentation, good community support and not very steep learning curve
7	Failing to follow rules and legislation regarding the data protection	0.1	10	1	- Identify early the personal data that is required to be stored in the system - Follow all the relevant University procedures - Put in place any security measures required to protect the system from a data breach
14	The system will be implemented using various technologies which	0.1	7	0.7	- Each chosen technology would be

	are not familiar to the developer and their compatibility and integration with each other would be first tested during the project. An incompatible technology could require the application of an alternative which in turn would require more time to learn and test. The delays could affect the project completion time				researched to identify compatibility issues - For each chosen technology alternative technologies would also be considered so that time is saved from researching for one later on (if this would be required)
6	Technical challenges due to using unproven/ new technologies and libraries (which might have limited documentation or bugs)	0.2	3	0.6	- Where possible use established, proved technologies - Only experiment for a specified time with new technologies. Have alternative choices ready to use if they don't meet expectations
15	Project can only meet the specification and quality required with proprietary or expensive software or additional resources which are not available	0.05	9	0.45	- Research alternatives - Keep supervisor up to date with research results, especially if such a risk is identified - During the requirements phase identify requirements which are not feasible under the circumstances or require resources that are not available. Consult with client and move these out of project scope

B.5 Phase-4 Risk Management Plan

This plan was produced while evaluating risks during the final spiral iteration (see project schedule in Appendix M).

MATERIALISED RISKS

Risk ID:2 has materialised as the client had requested to be able to search and review Faculty Officer availability separately from the automated booking process. The impact is estimated as about two weeks of extra time required to design and implement the extra functionality

Risk ID:8 has materialised as the laptop the student is using for the project work has crashed in the previous phase. There was a delay of two to three days as modelling diagrams were lost and had to be reproduced (the mitigation plan from previous phase only contained tasks for backing up application and report but not the diagrams).

CHANGES FROM PREVIOUS PLAN

Risk ID: 2
The mitigation plan has been updated to request more frequent feedback from the client (to avoid missing other requirements – see above).

Risk ID: 7
The risk has been eliminated as any requirement changes which would require to store personal data would not be permissible from this point onwards. This is due to the fact that proper policies and procedures regarding the handling of personal data would not be possible to be followed within the available time left.

Risk ID: 8
The mitigations plan has changed to include backing up of modelling diagrams. The impact has also been reduced as the backing up strategy is now considered more effective.

RISK MANAGEMENT PLAN

* sorted by Risk Exposure

ID	Risks	Probability	Impact	Risk Exposure	Mitigation Plan
1	Failure to identify a risk	0.4	? (have to assume the worst value – 10)	4	- Review current processes within the client's domain - Discuss with client potential risks - Discuss with supervisor potential risks - Study previous dissertations to identify cases where risk impacted on similar projects
2	Failure to extract all the requirements Incomplete project requirements (due to inexperience in the process of extracting requirements or client omitting requirements)	0.4	9	3.6	- Use a range of requirements elicitation techniques - Require more frequent feedback from the client
3	Project deadline missed due to inefficient time management	0.3	10	3	- Derive a project schedule to follow and set milestones to meet

					- Revise schedule frequently to ensure work remains on target
4	Misunderstanding project requirements	0.3	9	2.7	- Use a range of requirements elicitation techniques - Interview more than one user - Create an early prototype
9	Failure to estimate well the probability or impact of an identified risk	0.2	? (have to assume the worst value – 10)	2	- Study previous dissertations to identify cases where risk impacted on similar projects - Discuss with supervisor potential or identified risks
10	Long absence of the developer due to illness or other unexpected circumstances	0.2	9	1.8	- Been methodical with the project work and not falling behind schedule so that the impact of an unscheduled absence is minimised - Schedule to complete implementation, testing and report earlier than deadline (use the time at the end to handle the delay due to the unexpected absence – if the risk does not materialise, use the time for report revision)
11	Long absence of client due to illness or unscheduled holidays	0.2	8	1.6	- Frequent client meetings to get constant feedback so that the impact of a unexpected client absence will be mitigated - Discuss with the client to identify who could be consulted

					instead in their absence
12	Long absence of supervisor due to illness or unscheduled holidays	0.2	8	1.6	- Frequent supervisor meetings to get constant feedback so that the impact of an unexpected absence will be mitigated - Discuss with the supervisor to identify who would provide support/ guidance if the risk materialises
13	Change of requirements mid project	0.2	8	1.6	- Arrange frequent meetings with the client so is always aware of the work under way. That way any changes would be brought up and dealt with as early as possible
5	Failure to apply a tool or technology efficiently in the project (due to limited time available to learn and understand the tool/ technology)	0.2	7	1.4	- Where possible use tools and technologies which are familiar to the developer - if an unfamiliar technology has to be used then consider one with good documentation, good community support and not very steep learning curve
8	Developer machine crash (The risk has already materialised once)	0.2	6	1.2	- Back up application code, dissertation report and modelling diagrams at regular interval (at least on a daily basis)
14	The system will be implemented using various technologies which are not familiar to the developer and their compatibility and integration with each other would be first tested during the project. An incompatible technology could require the	0.1	7	0.7	- Each chosen technology would be researched to identify compatibility issues - For each chosen technology alternative technologies would

	application of an alternative which in turn would require more time to learn and test. The delays could affect the project completion time				also be considered so that time is saved from researching for one later on (if this would be required)
6	Technical challenges due to using unproven/ new technologies and libraries (which might have limited documentation or bugs)	0.2	3	0.6	- Where possible use established, proved technologies - Only experiment for a specified time with new technologies. Have alternative choices ready to use if they don't meet expectations
15	Project can only meet the specification and quality required with proprietary or expensive software or additional resources which are not available	0.05	9	0.45	- Research alternatives - Keep supervisor up to date with research results, especially if such a risk is identified - During the requirements phase identify requirements which are not feasible under the circumstances or require resources that are not available. Consult with client and move these out of project scope
7	Failing to follow rules and legislation regarding the data protection	0	10	0	No mitigation planning is required. (Any requirements for personal data storage would be out of scope from this point onwards – see notes above)

Appendix C. User Acceptance Testing

A description of the form used to perform the user acceptance testing for the Meeting Scheduler Application is offered below. The populated form with the MSA test results is provided next in Appendix C.2.

C.1 MSA User Acceptance Test Form

Test ID	Used to identify the test, mainly for referral and administration purposes
Requirement under test	The id of the functional requirement that is being tested. This is matched against the functional requirements table provided in the Requirement chapter. It might also contain a note with a small description of the functionality that is being tested
Actions	This field provides a description of the steps required to complete the test
Expected Result	Describes the expected output or result from the system when the stated actions are performed
Result	The tester's evaluation after comparing with the expected result. It can be 'Passed' or 'Failed'
Observations/Notes	Any observations or comments from the tester. If failed, it will describe the actual result. Also includes any conditions required for the testing
Tested by	Who carried out the testing. All functional requirements for the MSA were tested by the client

C.2 MSA User Acceptance Test Results

*For reasons of clarity the incremental versions of each test are not included in the table. Only the latest version is displayed below.

Date of completing UAT testing: 24/08/2018
Test Results summary: 37/37 (100%) Success

Meeting Scheduler Application: Test Plan and Results						
Test ID	**Requirement under test**	**Actions**	**Expected Result**	**Result**	**Observations/Notes**	**Tested by**
FT1	F1-Login *(successful login)*	1. Access login page 2. Enter correct email and password values 3. Press '*Sign in'*	User is successfully logged in and redirected to the '*Book Appointment'* page	Passed	-	Client

FT2	F1-Login *(unsuccessful login)*	1. Access login page 2. Enter incorrect email and password values 3. Press '*Sign in'*	User fails authentication and remains on the login screen. A message is shown to the user	Passed	-	Client
FT3	F2-Registration	1. Access the '*Manage Users'* page 2. Press the *'Add'* button 3. Edit details for title, name, email, password and confirm password fields 4. Press '*Submit'*	The new user is added and is now shown in the list in '*Manage Users'* page	Passed	-	Client
FT4	F3-Faculty *(retrieves faculty list)*	1. Click '*System Management'* on the navigation bar 2. Select *'Manage Faculties'*	A page with a list of current Faculties is shown	Passed	-	Client
FT5	F4-Department *(retrieves department list)*	1. Click '*System Management'* on the navigation bar 2. Select *'Manage Departments'*	A page with a list of current Departments in the system is shown	Passed	-	Client
FT6	F5-Department *(adds department)*	1. Access the '*Manage Departments'* page 2. Press the *Add* button 3. Edit details for new Department including name and faculty 4. Press '*Submit'*	The new Department is added to the system and is shown in the list on the '*Manage Departments'* page	Passed	-	Client
FT7	F6-Department *(updates department)*	1. Access the '*Manage Departments'* page 2. Click the id of the Department to update 3. Update all of the Department fields 4. Press *'Edit'*	The updated Department details are saved and a corresponding message is shown to the user	Passed	-	Client

FT8	F7-Department *(deletes department)*	1. Access the '*Manage Departments'* page 2. Click on the delete 'x' symbol to delete a specific department	The Department is removed from the system and the list on the page is updated	Passed	-	Client
FT9	F8-FacultyOfficer *(shows faculty officer list)*	1. Click '*System Management*' on the navigation bar 2. Select *'Manage Faculty Officers'*	A page with a list of current Faculty Officers in the system is shown	Passed	-	Client
FT10	F9-FacultyOfficer (adds faculty officer)	1. Access the '*Manage Faculty Officers'* page 2. Press the '*Add'* button 3. Edit details for new Faculty Office consisting of title, name, department, email, phone, comments 4. Press *'Submit'*	The new Faculty Officer is added to the system and is shown in the list on the *Manage Faculty Officers* page	Passed	-	Client
FT11	F10-FacultyOfficer *(updates faculty officer)*	1. Access the '*Manage Faculty Officers'* page 2. Click the id of the Faculty Officer to update 3. Update all of the Faculty Officer fields 4. Press *'Submit'*	The updated Faculty Officer details are saved and a corresponding message is shown to the user	Passed		Client
FT12	F11-FacultyOfficer *(removes faculty officer)*	1. Access the '*Manage Faculty Officers'* page 2. Click on the delete *'x'* symbol to delete a specific Faculty Officer	The Faculty Officer is removed from the system and the list on the page is updated	Passed	-	Client
FT13	F12-SystemUser *(displays user list)*	1. Click '*System Management*' on the navigation bar 2. Select *'Manage Users'*	A page with a list of current Users in the system is shown	Passed	-	

FT14	F13-SystemUser *(updates user)*	1. Access the '*Manage Users'* page 2. Click the id of the User to update 3. Update any of the User's fields (except username) 4. Press *'Edit'*	The updated User details are saved and a corresponding message is shown to the user	Passed	-	
FT15	F14-SystemUser *(deletes user)*	1. Access the '*Manage Users'* page 2. Click on the delete *'x'* symbol to delete a specific User	The User is removed from the system and the list on the page is updated	Passed	-	Client
FT16	F15-Room *(displays room list)*	1. Click '*System Management*' on the navigation bar 2. Select *'Manage Rooms'*	A page with a list of current Rooms in the system is shown	Passed	-	Client
FT17	F16-Room *(updates room)*	1. Access the '*Manage Rooms'* page 2. Click the id of the Room to update 3. Update all of the Room fields 4. Press *'Edit'*	The updated Room details are saved and a corresponding message is shown to the user	Passed	-	Client
FT18	F17-Room *(adds room)*	1. Access the '*Manage Rooms'* page 2. Press the '*Add'* button 3. Edit details for a new Room consisting of name, location, calendar reference (optional) 4. Press *'Submit'*	The new Room is added to the system and is shown in the list on the *Manage Rooms* page	Passed	-	Client
FT19	F18-Room *(deletes room)*	1. Access the '*Manage Rooms'* page 2. Click on the delete *'x'* symbol to delete a specific Room	The Room is removed from the system and the list on the page is updated	Passed	-	Client

FT20	F19-Appointment *(successful booking)*	1. Access *'Book Appointment'* page 2. Select student Department (optionally) 3. Select SSS admin staff 4. Select meeting room 5. Define date/time period to search for 6. Define meeting duration 7. Press *'Submit'*	A meeting is booked on the calendars of all participants. The screen displays a successful message	Passed	Use date and time to search which is known to be free across all participants	Client
F21	F19-Appointment *(booking not possible - no faculty officer availability)*	1. Access *'Book Appointment'* page 2. Select student Department (optionally) 3. Select SSS admin staff 4. Select meeting room 5. Define date/time period to search for 6. Define meeting duration 7. Press *'Submit'*	A booking for a meeting is not being made. The screen displays a message for no availability	Passed	Use date and time to search which is known to be busy for the Faculty Officer	Client
F22	F19-Appointment *(booking not possible - no admin staff availability)*	1. Access *'Book Appointment'* page 2. Select student Department (optionally) 3. Select SSS admin staff 4. Select meeting room 5. Define date/time period to search for 6. Define meeting duration 7. Press *'Submit'*	A booking for a meeting is not being made. The screen displays a message for no availability	Passed	Use date and time to search which is known to be busy for the SSS admin staff	Client

F23	F19-Appointment *(booking not possible - no room availability)*	1. Access *'Book Appointment'* page 2. Select student Department (optionally) 3. Select SSS admin staff 4. Select meeting room 5. Define date/time period to search for 6. Define meeting duration 7. Press *'Submit'*	A booking for a meeting is not being made. The screen displays a message for no availability	Passed	Use date and time to search which the meeting room is not available	Client
FT24	F20-Appointment	1. Complete booking process as in FT-20 2. Click on the link that is shown after successful login	The google calendar site is launched on screen with information about the booking and meeting participants	Passed	-	Client
FT25	F21-Availability *(has availability)*	1. Access the '*Faculty Officer Availability'* page 2. Select Faculty Officer 3. Select date and time period to search within 4. Define minimum duration 5. Press *'Submit'*	The screen displays a list with the available slots in the Faculty Officer calendar which satisfy the requirements	Passed	Select date, time and duration which are known to offer availability slots for the Faculty Officer	Client
FT26	F21-Availability *(no availability)*	1. Access the '*Faculty Officer Availability'* page 2. Select Faculty Officer 3. Select date and time period to search within 4. Define minimum duration 5. Press *'Submit'*	The screen shows a message describing that no availability found for this Faculty Officer	Passed	Select date, time and duration which overlap busy times in the Faculty Officer calendar	Client
FT27	F22-Logout	1. Click *'Logout'* button on the navigation bar	The user is logged out and a message is displayed with a link to re-login if wished	Passed	-	Client

FT28	LF23-FacultyOfficer *(define primary)*	1. Access a specific Faculty Officer details page 2. Check the 'Primary Faculty Officer' checkbox 3. Press *'Submit'*	The officer is now stored as primary officer in the system. The font colour changes for the title and name Faculty Officer details to indicate the she/he is now primary. An asterisk '*' also appears next to the title	Passed	In previous version of the test, client indicated that more is needed to identify the Faculty Officer than just the font. An asterisk is thus also attached next to the title.	Client
FT29	LF24-FacultyOfficer (*identifies primary on booking page)*	1. Access the '*Book Appointment'* page 2. Click on the dropdown which displays the '*Faculty Officer*' choices	An asterisk '*' is shown to identify the primary Faculty Officers Faculty Officer details	Passed	-	Client
FT30	LF24-FacultyOfficer *(identifies primary on view availability page)*	1. Access the '*Faculty Officer Availability'* page 2. Click on the dropdown which displays the '*Faculty Officer*' choices	An asterisk '*' is shown to identify the primary Faculty Officers Faculty Officer details	Passed	-	Client
FT31	LF25-Faculty *(adds Faculty)*	1. Access the '*Manage Faculties'* page 2. Press the '*Add'* button 3. Edit name for the new Faculty 4. Press *'Submit'*	The new Faculty is added to the system and is shown in the list on the '*Manage Faculties'* page	Passed	-	Client
FT32	LF26-Faculty *(updates Faculty)*	1. Access the '*Manage Faculties'* page 2. Click the id of the Faculty to update 3. Update the Faculty name 4. Press *'Edit'*	The updated Faculty details are saved and a corresponding message is shown to the user	Passed	-	Client
FT33	LF27-Faculty *(deletes Faculty)*	1. Access the '*Manage Faculties'* page 2. Click on the delete *'x'* symbol	The Faculty is removed from the system and the list	Passed	-	Client

		to delete a specific Faculty	on the page is updated			
FT34	LF28-Appointment *(department selection restricts faculty officer choice)*	1. Access the '*Booking Appointment*' page 2. View full list of Faculty Officers in '*Faculty Officer*' dropdown 3. Select a department	The '*Faculty Officer'* dropdown is showing only officers which belong to the selected department's faculty (but not in that department)	Passed	-	Client
FT35	LF29-Appointment *(does not allow bookings on weekends)*	Repeat booking process of FT20, but choose weekend dates to book	A booking for a meeting is not being made. The screen displays a message for no availability	Passed	-	Client
FT36	LF30-Availability *(does not return officer availability on weekends)*	Repeat the process of FT25 for searching for Faculty Officer availability, but select weekend dates	The screen shows a message describing that no availability found for this Faculty Officer	Passed	-	Client
FT37	LF31-Password *(password reset)*	1. Access the MSA login pages 2. Click the '*Forgot password?'* link 3. After redirected to '*Forgot Password*' page enter email address and press *'Reset Password'* 4. An email is sent to the inserted address with a link to continue the process 5. When clicked a form is shown to provide the new password. It should contain	The password is reset to the password provided by the user. The user is redirected to the login page with a message confirming the password reset. The user should now be able to login with the updated password	Passed	-	Client

		'Password' and 'Confirm password' fields 6. Edit and press 'Reset Password'				

Appendix D. Non-Functional Test Results

A description of the form used to perform the non-functional testing for the Meeting Scheduler Application is offered below. The populated form with the MSA test results is provided next in Appendix D.2.

D.1 MSA Non-Functional Test Form

Test ID	Used to identify the test, mainly for referral and administration purposes
Requirement under test	The id of the non-functional requirement that is being tested. This is matched against the non-functional requirements table available in the Requirements chapter. It might also contain a note with a small description of the functionality that is being tested
Test Actions	This field provides a description of the steps required to complete the test
Expected Result	Describes the expected output or result from the system when the stated actions are performed
Result	The tester's evaluation after comparing with the expected result. It can be 'Passed' or 'Failed'. When no value, the Observations field would contain clarifications
Observations and Actions Required	Any observations or comments from the tester. If failed, it will describe the actual result. Also includes any conditions required for the testing
Tested by	Who carried out the testing

D.2 MSA Non-Functional Test Results

*For reasons of clarity the incremental versions of each test are not included in the table. Only the latest version is displayed below.
*'*Special characters*' refer to any punctuation or mathematical symbols (not including numbers)

Date of completing non-functional testing (not on live environment): 24/08/2018
Test Results summary: 96 Success, 2 Failed (see Actions Required on table), 22 Pending (need to be run on the live environment)

Meeting Scheduler Application: Non-Functional Test Plan and Results						
Test ID	**Requirement under test**	**Test Actions**	**Expected Result**	**Result**	**Observations and Actions Required**	**Tested by**
NFT1	Security-1	Access url to book appointment	User is redirected to login page	Passed	-	Client
NFT2	Security-1	Access url to view faculty officer availability	User is redirected to login page	Passed	-	Client
NFT3	Security-1	Access url to manage faculties	User is redirected to login page	Passed	-	Client
NFT4	Security-1	Access url to manage departments	User is redirected to login page	Passed	-	Developer
NFT5	Security-1	Access url to manage faculty officers	User is redirected to login page	Passed	-	Developer
NFT6	Security-1	Access url to manage admin staff/ users	User is redirected to login page	Passed	-	Developer

NFT7	Security-1	Access url to manage meeting rooms	User is redirected to login page	Passed	-	Developer
NFT8	Security-2	1. Login 2. Access all features in manage faculties pages. Perform CRUD operations	User should be allowed to perform all the CRUD actions without restrictions	Passed	There is only one user level implemented. No need to test different authorisation levels	Client
NFT9	Security-2	1. Login 2. Access all features in manage departments pages. Perform CRUD operations	User should be allowed to perform all the CRUD actions without restrictions	Passed	There is only one user level implemented. No need to test different authorisation levels	Client
NFT10	Security-2	1. Login 2. Access all features in manage Faculty Officers pages. Perform CRUD operations	User should be allowed to perform all the CRUD actions without restrictions	Passed	There is only one user level implemented. No need to test different authorisation levels	Client
NFT11	Security-2	1. Login 2. Access all features in manage users pages. Perform CRUD operations	User should be allowed to perform all the CRUD actions without restrictions	Passed	There is only one user level implemented. No need to test different authorisation levels	Client
NFT12	Security-2	1. Login 2. Access all features in manage rooms pages. Perform CRUD operations	User should be allowed to perform all the CRUD actions without restrictions	Passed	There is only one user level implemented. No need to test different authorisation levels	Client

NFT13	Security-3	1. Login 2. Access Book Appointment form 3. Add alphabetical or special characters* on the date, time and duration fields 4. Press 'Submit'	The action should be prevented	Passed	-	Client
NFT14	Security-3	1. Login 2. Access Search Faculty Officer Availability form 3. Add alphabetical or special characters* on the date, time and duration fields 4. Press 'Submit'	The action should be prevented	Passed	-	Client
NFT15	Security-3	1. Login 2. Select Manage Faculties from menu 3. Press 'Add Faculty' 4. Enter any of the prohibited special characters in the name field (the allowed characters are displayed when hovering over the field) 5. Press 'Submit'	Action should be prevented	Passed	-	Client
NFT16	Security-3	1. Login 2. Select Manage Faculties from menu 3. Click the ID of an existing entity to edit 4. Enter any of the prohibited special characters in the name field (the allowed characters are displayed when hovering over the field) 5. Press 'Edit'	Action should be prevented	Passed	-	Developer

NFT17	Security-3	1. Login 2. Select Manage Departments from menu 3. Press ‘Add’ 4. Enter any of the prohibited special characters in the name field (the allowed characters are displayed when hovering over the field) 5. Press ‘Submit’	Action should be prevented	Passed	-	Client
NFT18	Security-3	1. Login 2. Select Manage Departments from menu 3. Click the ID of an existing entity to edit 4. Enter any of the prohibited special characters in the name field (the allowed characters are displayed when hovering over the field) 5. Press ‘Edit’	Action should be prevented	Passed	-	Developer
NFT19	Security-3	1. Login 2. Select Manage Faculty Officers from menu 3. Press ‘Add’ 4. Enter any of the prohibited special characters in the name, email, phone and comments fields (the allowed characters are displayed when hovering over the field) 5. Press ‘Submit’	Action should be prevented	Failed	The name, email and phone were validated successfully. The comments field did not pass the validation. The HTML5 pattern validation is not applicable on a text area. JavaScript is needed for this. Allowed for MVP due to low security risk (only three users which must authenticate)	Client

NFT20	Security-3	1. Login 2. Select Manage Faculty Officers from menu 3. Click on an ID to edit a Faculty Officer 4. Enter any of the prohibited special characters in the name, email, phone and comments fields (the allowed characters are displayed when hovering over the field) 5. Press 'Edit'	Action should be prevented	Failed	The name, email and phone were validated successfully. The comments field did not pass the validation. The HTML5 pattern validation is not applicable on a text area. JavaScript is needed for this. Allowed for MVP due to low security risk (only three users which must authenticate)	Developer
NFT21	Security-3	1. Login 2. Select Manage Users from menu 3. Press 'Add' 4. Enter any of the prohibited special characters in the name and email fields (the allowed characters are displayed when hovering over the field) 5. Press 'Submit'	Action should be prevented	Passed	-	Developer
NFT22	Security-3	1. Login 2. Select Manage Rooms from menu 3. Press 'Add' 4. Enter any of the prohibited special characters in the name, location and calendar reference fields (the allowed characters are displayed when hovering over the field) 5. Press 'Submit'	Action should be prevented	Passed	-	Developer

NFT23	Security-3	1. Login 2. Select Manage Rooms from menu 3. Click on the ID of the room entity to edit 4. Enter any of the prohibited special characters in the name, location and calendar reference fields (the allowed characters are displayed when hovering over the field) 5. Press 'Submit'	Action should be prevented	Passed	-	Developer
NFT24	Reliability-1	1. Login 2. Book an appointment with a combination of Faculty Officer, admin staff, room. 3. Repeat step 2 to book appointment with the same combination 4. Repeat step 2 with another combination	Three successful bookings should be made	Passed	-	Developer
NFT25	Reliability-2	1. Login 2. Logout 3. Login with the same credentials used in step 1 4. Logout	The system should successfully login and logout twice. Upon login the Book Appointment form should be displayed. Upon logout a page with an informative message and a link to login again should be displayed.	Passed	-	Developer

NFT26	Reliability-3	1. Login 2. Click on Manage Faculties on the menu 3. Add a Faculty 4. Add another Faculty	The system should add two Faculties	Passed	-	Developer
NFT27	Reliability-3	1. Login 2. Click on Manage Faculties on the menu 3. Edit a Faculty 4. Inspect result 5. Edit same Faculty again 6. Inspect result	The system should successfully edit the Faculty twice	Passed	-	Developer
NFT28	Reliability-3	1. Login 2. Click on Manage Departments on the menu 3. Add a Department 4. Add another Department	The system should add two Departments	Passed	-	Developer
NFT29	Reliability-3	1. Login 2. Click on Manage Departments on the menu 3. Edit a Department 4. Inspect result 5. Edit same Department again 6. Inspect result	The system should successfully edit the Department twice	Passed	-	Developer
NFT30	Reliability-3	1. Login 2. Click on Manage Faculty Officer on the menu 3. Add a Faculty Officer 4. Add another Faculty Officer	The system should add two Faculty Officers	Passed	-	Developer

NFT31	Reliability-3	1. Login 2. Click on Manage Faculty Officers on the menu 3. Edit a Faculty Officer 4. Inspect result 5. Edit same Faculty Officer again 6. Inspect result	The system should successfully edit the Faculty Officer twice	Passed	-	Developer
NFT32	Reliability-3	1. Login 2. Click on Manage Users on the menu 3. Add a User 4. Add another User	The system should add two User entities	Passed	-	Developer
NFT33	Reliability-3	1. Login 2. Click on Manage Rooms on the menu 3. Add a Room 4. Add another Room	The system should add two Room entities	Passed	-	Developer
NFT34	Reliability-3	1. Login 2. Click on Manage Rooms on the menu 3. Edit a Room 4. Inspect result 5. Edit same Room again 6. Inspect result	The system should successfully edit the Room twice	Passed	-	Developer
NFT35	Usability-1	Visit all pages of the application	There should be a consistent black grey colour scheme across all pages. Interactive points such as links and buttons should be indicated with blue	Passed	-	Client

NFT36	Usability-2	Visit all pages of the application	The location of buttons on forms should be centralised at the bottom. Edit and delete links should be located in the same position for all entity lists. Delete links should be represented with an ‘x’ symbol next to the entities in all lists	Passed	-	Client
NFT37	Usability-3	Add a Faculty Officer	A message is displayed confirming the save of the Faculty Officer	Passed	-	Client
NFT38	Usability-3	Edit a Faculty Officer	A message is displayed confirming the update of the Faculty Officer	Passed	-	Client
NFT39	Usability-3	Add a Faculty	A message is displayed confirming the save of the Faculty	Passed	-	Client
NFT40	Usability-3	Edit a Faculty	A message is displayed confirming the update of the Faculty	Passed	-	Client

NFT41	Usability-3	Add a Department	A message is displayed confirming the save of the Department	Passed	-	Client
NFT42	Usability-3	Edit a Department	A message is displayed confirming the update of the Department	Passed	-	Client
NFT43	Usability-3	Register a new user	A message is displayed confirming the new user registration	Passed	-	Client
NFT44	Usability-3	Add a Room	A message is displayed confirming the save of the Room	Passed	-	Client
NFT45	Usability-3	Edit a Room	A message is displayed confirming the update of the Room	Passed	-	Client
NFT46	Usability-3	Complete the booking process	A message is displayed confirming the booking of the meeting	Passed	-	Client

NFT47	Usability-4	Access '*Add Faculty'* form. Press *'Add'* without entering a value for the name field	Validation of the form prevents form submission	Passed	-	Client
NFT48	Usability-4	Access '*Edit Faculty'* form. Press *'Submit'* without entering a value for the name field	Validation of the form prevents form submission. Appropriate messages are shown	Passed	-	Client
NFT49	Usability-4	Access '*Add Department'* form. Press *'Add'* without entering values for the name and/or Faculty fields	Validation of the form prevents form submission. Appropriate messages are shown	Passed	-	Client
NFT50	Usability-4	Access '*Edit Department'* form. Press *'Submit'* without entering values for the name and/or Faculty fields	Validation of the form prevents form submission. Appropriate messages are shown	Passed	-	Client
NFT51	Usability-4	Access '*Add Faculty Officer'* form. Press *'Add'* without entering values in the fields. Also submit with email which does not contain '@'. Also with a phone which does not follow the specified format.	Validation of the form prevents form submission. Appropriate messages are shown	Passed	-	Client-partially, Developer

NFT52	Usability-4	Access '*Edit Faculty Officer'* form. Press *'Submit'* without entering values in the fields. Also submit with email which does not contain '@'. Also with a phone which does not follow the specified format.	Validation of the form prevents form submission. Appropriate messages are shown	Passed	-	Client-partially, Developer
NFT53	Usability-4	Access '*Add User'* form. Press *'Submit'* without entering values in the fields. Also submit with username/email which does not contain '@'	Validation of the form prevents form submission. Appropriate messages are shown	Passed	-	Client
NFT54	Usability-4	Access '*Book Appointment'* form. Press *'Submit'* without entering/selecting values in the fields	Validation of the form prevents form submission. Appropriate messages are shown	Passed	-	Client
NFT55	Usability-4	Access '*Book Appointment'* form. Enter duration longer that time period specified. Press *'Submit'*	Validation of the form prevents form submission. Appropriate messages are shown	Passed	-	Client
NFT56	Usability-4	Access '*Book Appointment'* form. Enter '*to Date*' which is earlier than '*from Date*'. Press *'Submit'*	Validation of the form prevents form submission. Appropriate messages are shown	Passed	-	Client

NFT57	Usability-4	Access '*Search Faculty Officer Availability'* page. Press *'Submit'* without entering/selecting values in the fields	Validation of the form prevents form submission. Appropriate messages are shown	Passed	-	Client
NFT58	Usability-4	Access '*Search Faculty Officer Availability'* form. Enter duration longer that time period specified. Press *'Submit'*	Validation of the form prevents form submission. Appropriate messages are shown	Passed	-	Developer
NFT59	Usability-4	Access '*Search Faculty Officer Availability'* form. Enter '*to Date*' which is earlier than '*from Date*'. Press *'Submit'*	Validation of the form prevents form submission. Appropriate messages are shown	Passed	-	Developer
NFT60	Usability-5	Access '*Add Department'* form	Faculty dropdown contains predefined values	Passed	-	Client
NFT61	Usability-5	Access '*Edit Department'* form	Faculty dropdown contains predefined values	Passed	-	Developer
NFT62	Usability-5	Access '*Add Faculty Officer'* form	Department dropdown contains predefined values	Passed	-	Client
NFT63	Usability-5	Access 'Edit *Faculty Officer'* form	Department dropdown contains predefined values	Passed	-	Developer

NFT64	Usability-5	Access '*Book Appointment'* form	Department, Faculty Officer, SSS staff and Room dropdowns contains predefined values	Passed	-	Client
NFT65	Usability-5	Access '*Search Faculty Officer Availability'* form	Faculty Officer dropdown contains predefined values	Passed	-	Client
NFT66	Usability-6	Submit request to delete a Faculty	Confirmation dialogue is displayed	Passed	-	Developer
NFT67	Usability-6	Submit request to delete a Faculty Officer	Confirmation dialogue is displayed	Passed	-	Client
NFT68	Usability-6	Submit request to delete a Department	Confirmation dialogue is displayed	Passed	-	Developer
NFT69	Usability-6	Submit request to delete an SSS staff	Confirmation dialogue is displayed	Passed	-	Developer

NFT70	Usability-6	Submit request to delete a Room	Confirmation dialogue is displayed	Passed	-	Developer
NFT71	Usability-7	All pages are accessed to confirm that buttons, menus and all text are using terminology familiar to the user	Client confirms that he understands all the terms used in the UI	Passed	-	Client
NFT71	Usability-8	All pages are accessed to assess clear and aesthetic design. Navigation should be intuitive and data in lists should allow for searching and sorting	Client confirms intuitive and clear design	Passed	-	Client
NFT72	Performance-1	Submit '*Add Faculty*' request	Confirm that response takes less than 0.1 secs	-	To be tested on the live environment	Developer/ Technical Team
NFT73	Performance-1	Submit '*Edit Faculty*' request	Confirm that response takes less than 0.1 secs	-	To be tested on the live environment	Developer/ Technical Team

NFT74	Performance-1	Submit '*Add Department*' request	Confirm that response takes less than 0.1 secs	-	To be tested on the live environment	Developer/ Technical Team
NFT75	Performance-1	Submit '*Edit Department*' request	Confirm that response takes less than 0.1 secs	-	To be tested on the live environment	Developer/ Technical Team
NFT76	Performance-1	Submit '*Add Faculty Officer'* request	Confirm that response takes less than 0.1 secs	-	To be tested on the live environment	Developer/ Technical Team
NFT77	Performance-1	Submit '*Edit Faculty Officer'* request	Confirm that response takes less than 0.1 secs	-	To be tested on the live environment	Developer/ Technical Team
NFT78	Performance-1	Submit '*Add Room'* request	Confirm that response takes less than 0.1 secs	-	To be tested on the live environment	Developer/ Technical Team
NFT79	Performance-1	Submit '*Edit Room'* request	Confirm that response takes less than 0.1 secs	-	To be tested on the live environment	Developer/ Technical Team

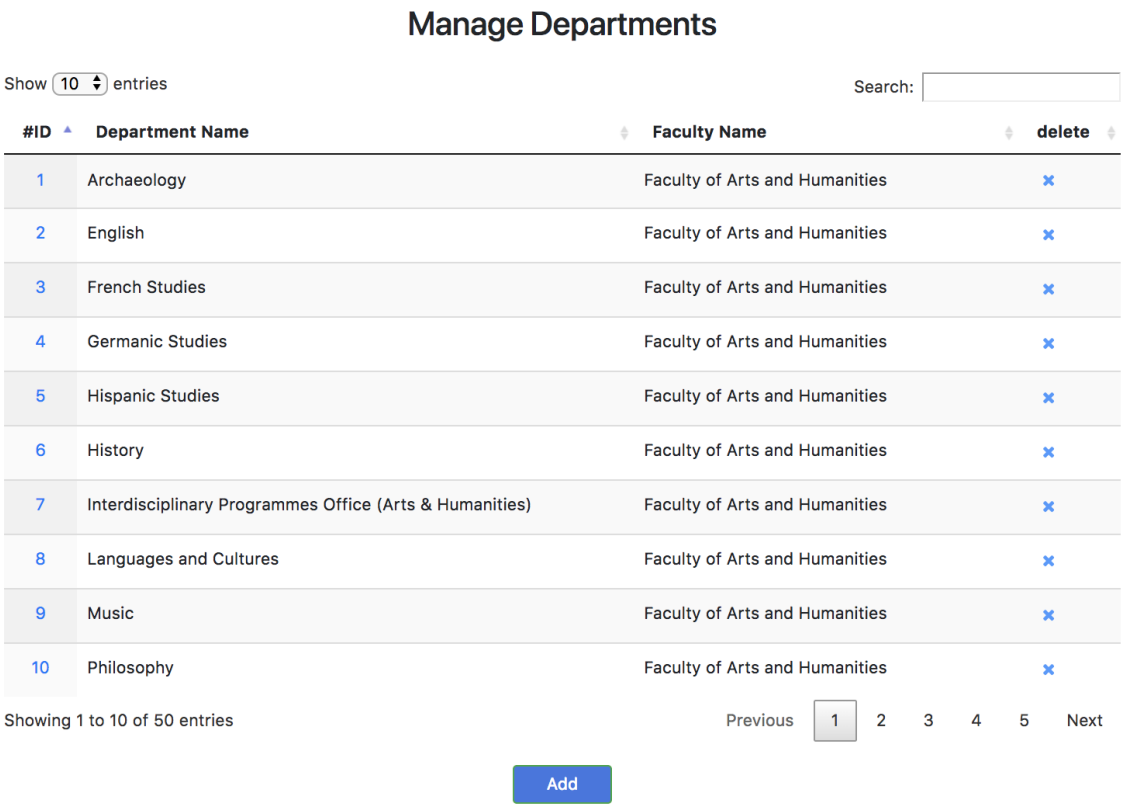
NFT80	Performance-1	Submit '*Add User'* request	Confirm that response takes less than 0.1 secs	-	To be tested on the live environment	Developer/ Technical Team
NFT81	Performance-1	Submit request to delete a User	Confirm that response takes less than 0.1 secs	-	To be tested on the live environment	Developer/ Technical Team
NFT82	Performance-1	Submit request to delete a Faculty	Confirm that response takes less than 0.1 secs	-	To be tested on the live environment	Developer/ Technical Team
NFT83	Performance-1	Submit request to delete a Department	Confirm that response takes less than 0.1 secs	-	To be tested on the live environment	Developer/ Technical Team
NFT84	Performance-1	Submit request to delete a Faculty Officer	Confirm that response takes less than 0.1 secs	-	To be tested on the live environment	Developer/ Technical Team
NFT85	Performance-1	Submit request to delete a Room	Confirm that response takes less than 0.1 secs	-	To be tested on the live environment	Developer/ Technical Team

NFT86	Performance-2	Submit request to retrieve list of current Faculties	Confirm that response takes less than 0.3 secs	-	To be tested on the live environment	Developer/ Technical Team
NFT87	Performance-2	Submit request to retrieve list of current Departments	Confirm that response takes less than 0.3 secs	-	To be tested on the live environment	Developer/ Technical Team
NFT88	Performance-2	Submit request to retrieve list of current Faculty Officers	Confirm that response takes less than 0.3 secs	-	To be tested on the live environment	Developer/ Technical Team
NFT89	Performance-2	Submit request to retrieve list of current Users	Confirm that response takes less than 0.3 secs	-	To be tested on the live environment	Developer/ Technical Team
NFT90	Performance-2	Submit request to retrieve list of current Rooms	Confirm that response takes less than 0.3 secs	-	To be tested on the live environment	Developer/ Technical Team
NFT91	Performance-3	Submit request to book appointment using google calendar api	Confirm that response takes less than 0.5 secs	-	To be tested on the live environment	Developer/ Technical Team

NFT92	Performance-4	Submit request to search Faculty Officer availability using google calendar api	Confirm that response takes less than 1 secs	-	To be tested on the live environment	Developer/ Technical Team
NFT93	Portability-1	Run the system (written in Java) in linux, mac and windows operating systems/servers	Confirm it runs successfully	Passed (tested on PCs and personal laptops)	-	Developer
NFT94	Portability-2	Install MySQL in linux, mac and windows operating systems/servers and connect to it from the MSA system	Confirm it runs and connects to MySQL successfully	Passed (tested on PCs and personal laptops)	-	Developer
NFT94	Reusability-1	Use Dependency Injection to inject alternative implementations during unit and integration testing	Confirm testing can be performed with dependency injection. This means that the modules can be reused with different dependencies/ implementations	Passed	-	Developer
NFT95	Reusability-2	Use the H2 as the MSA database during development. The code should not need to change when MySQL is used in the live environment	Confirm no changes in the code are necessary (except possibly some configuration)	Passed	-	Developer
NFT95	Costs-1	No development/ project costs should occur for the client	Confirm this is the case at the end of the project	Passed	-	Developer

NFT95	Costs-2	No hosting/running costs should occur for the client	Confirm this is the case when hosted/run on the live environment	-	No costs expected as the system is using open source software and the hosting/running should be free with the University servers. Should be confirmed with the technical team which will setup hosting	University technical team
NF96	System-Configuration-1	Run the initial script to setup the MSA database on different database instances	Confirm the script works when setting up the system on a different environment with a new database	Passed	-	Developer

Appendix E. Manage Departments screen



Appendix F. Algorithm for Booking Appointment

The algorithm below describes the steps implemented in the MSA for booking an Unsatisfactory Progress appointment.

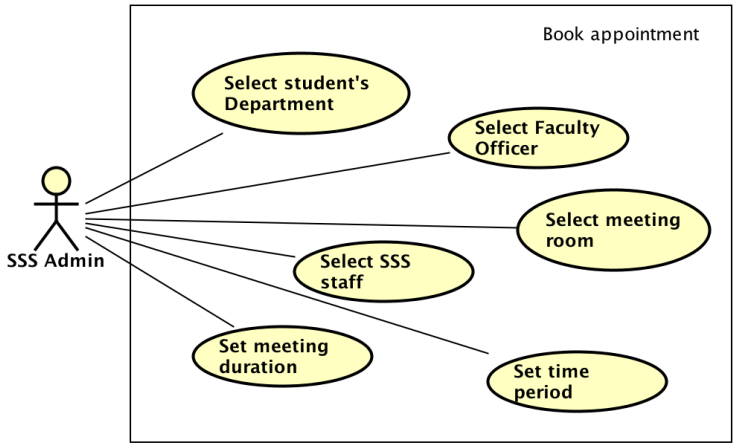
1. Retrieve data from the database for the selected Faculty Officer, admin staff and room
2. Extract list of dates between the user defined start and end dates (LIST A)
3. Filter out Saturdays and Sundays
4. Create a list to populate with all free time slots (LIST B) in those days
5. For each date in LIST A (loop)
    a. Construct the Google Calendar API request using the meeting participants emails and the user defined date and (start and end) times
    b. Query the Google Calendar API with this request object and get the response
    c. Get all the meeting participants' calendars (containing the busy time slots) from the response (LIST C)
    d. Create an empty free slot list to hold the derived free slots (LIST D)
    e. Create an initial free time slot with the user defined start and end times and add it to the free times LIST D
    f. For each calendar in LIST C (recursion)
        i. Get the busy time slots from the calendar (LIST E)
            1. For each busy time slot in LIST E (recursion)
                a. Discard busy time slot from any free time slot it overlaps in LIST D (see sub-algorithm below)
                b. Return updated list of derived free time slots (updated LIST D)
                c. Use the updated LIST D of free slot times for the next iteration
        ii. Use the updated LIST D of free slot times for the next iteration
        iii. Add currently derived free slots in LIST D to overall free slots in LIST B
6. Filter out slots from populated LIST B which do not span the user requested duration
7. Return LIST B - contains any free slots found satisfying the user requirements for dates, times and duration across all Faculty Officer, SSS admin staff and meeting room calendars
8. Book an appointment in the first available slot in LIST B - this action would create an event in all three calendars of Faculty Officer, SSS admin staff and meeting room

Sub-algorithm for step 5.f.i.1.a

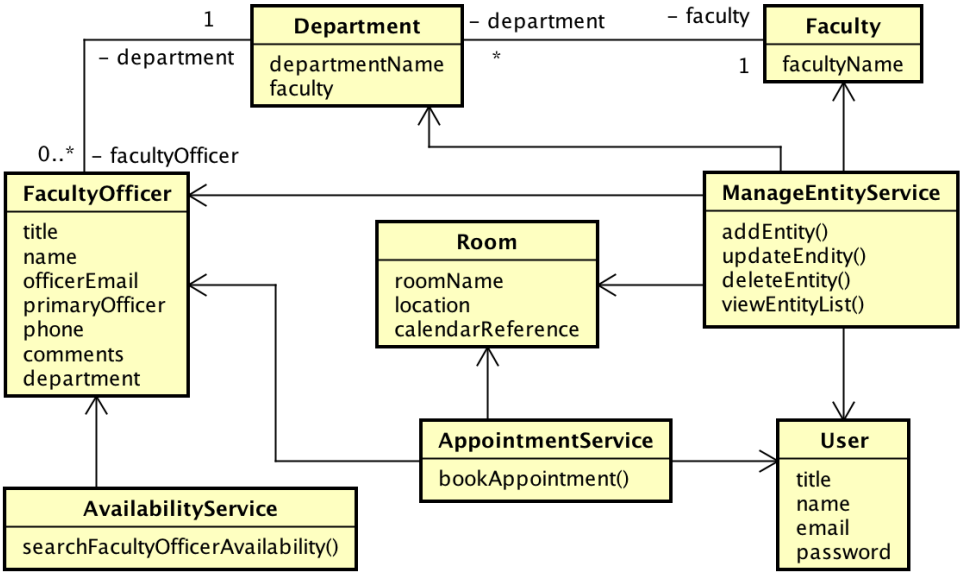
The sub-algorithm to filter out a busy slot from the list of free slots in step 5.f.i.1.a is outlined below.

1. For each iteration, create a new list (LIST F) to hold the amended free time slots during the iteration
2. For each free time slot in LIST D
    a. If busy time slot overlaps completely the free time slot, then discard this free time slot
    b. If busy time start >= free time end, then add the free slot to LIST F
    c. If busy time end <= free time start, then add the free slot to LIST F
    d. If busy time start > free time start && busy time end < free time end, then filter out the busy slot (from the middle of the free slot) and add the two new free slots (from each side) to LIST F
    e. If busy time slot starts from within the free time slot and spans beyond the end of the free slot, then discard the time that is overlapped and add the remaining time slot (before the discarded time) to LIST F
    f. If busy time slot starts before the free time slot and spans until a point within the free time slot, then discard the time that is overlapped and add the remaining time slot (after the discarded time) to LIST F
3. Replace the elements in LIST D with the derived free slots from LIST F (it would then be fed to the next iteration of step 5.f.i.1.a

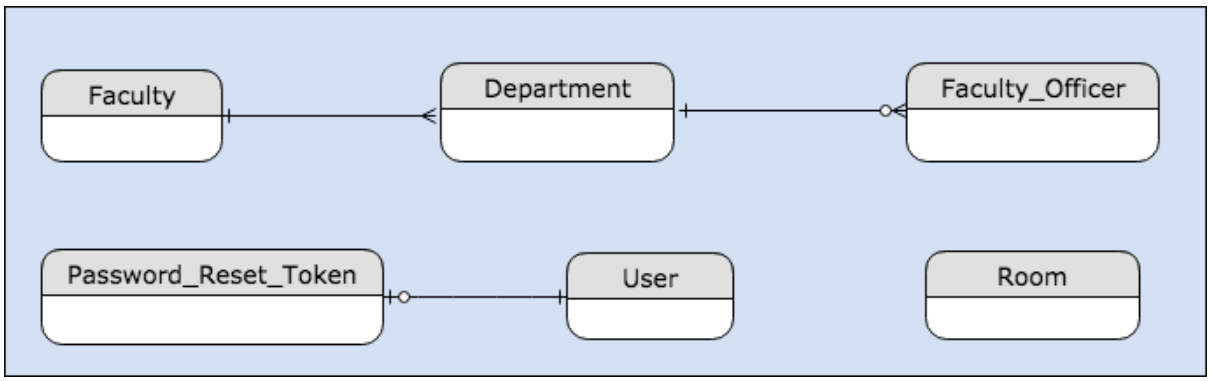
Appendix G. Use Case diagram for the appointment booking



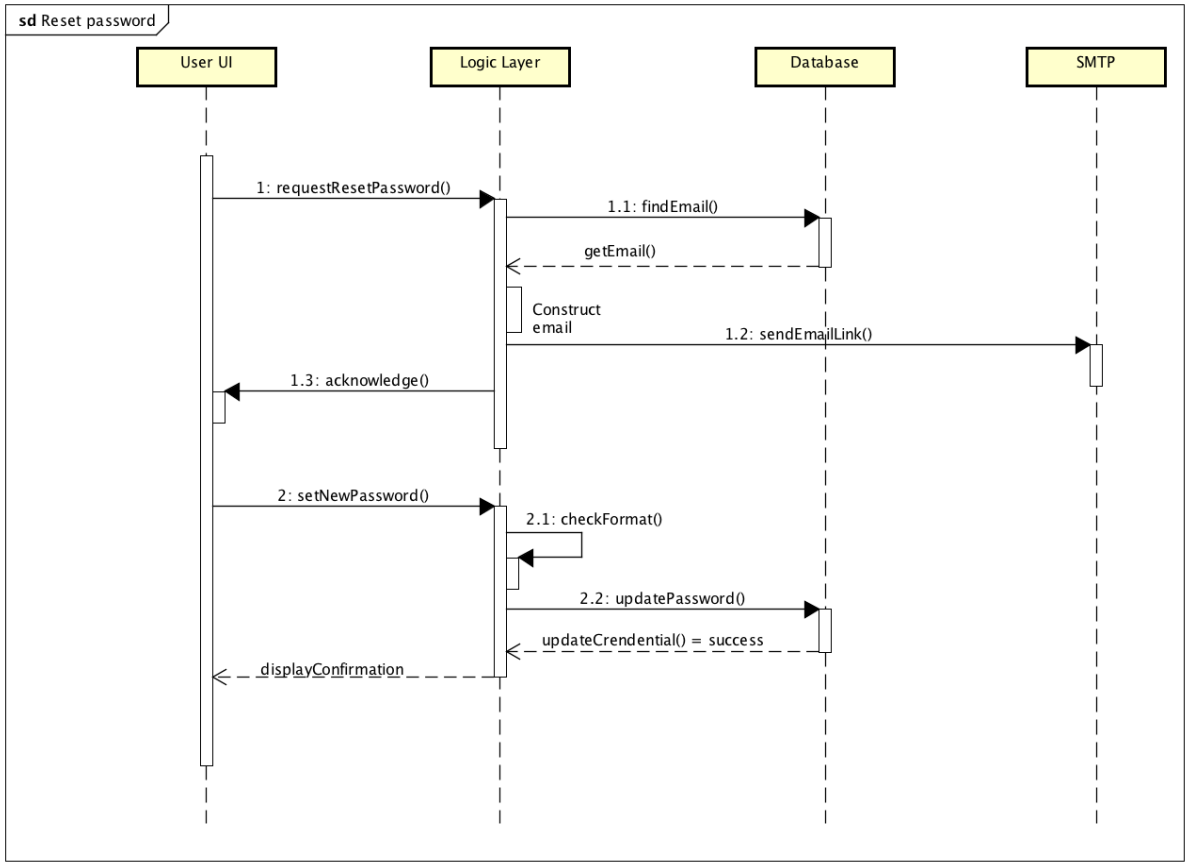
Appendix H. Conceptual Class diagram



Appendix I. Conceptual Entity Relationship Diagram



Appendix J. Reset Password - Sequence Diagram



Appendix K. System Manual

K.1 Installation

The whole application and all its dependencies are contained within a single executable jar file, the *scheduleMeetings-1.0.0.jar*. The jar file contains also the embedded tomcat container which hosts the application. There is no need for another external standalone container. The only requirements are a platform with an install JVM and a relational database (configuration instructions are provided in the next section).
Run the application using the following command:

java -jar scheduleMeetings-1.0.0.0.jar

There is a one registered system user to enable to login the first time after installation.
username: *admin@sheffield.ac.uk*
password*: password*

After the first login, you can register a new user and delete the existing one.

K.1.1 Configuration

The essential configuration for the application is described below.

Port

The default port for the application is 8080. To change it you would need to update the following in *application.properties*.

server.port=8080

Database

The configuration of the system's database is also done in the *application.properties* file within the application jar. Use a compression software to unzip, update the file and zip again the jar. Otherwise follow the instructions from Appendix L to open the application in an IDE and update the file.

The system uses a MySQL database for storing and retrieving data. The database connection properties need to be setup in the *application.properties*. The properties to update are listed below (values are added as examples):

spring.datasource.url=*jdbc:mysql://localhost:3306/MSA*
spring.datasource.username=*root*
spring.datasource.password=*password*

Two sql scripts have been prepared to create the schema and load the initial data in MySQL. These are the *mysql_schema.sql* and *mysql_data.sql* respectively.
Although the system has been tested with MySQL, the application is designed to work with any relational database. The properties above and the associated initialisation files would need to be updated accordingly for an alternative database system.

Primary Email Address

The system integrates with the Google Calendar API to retrieve calendar events and create new ones. Google Calendar requires a primary email address which would be used to display as the requester

address to other users, when their calendar information is requested. It would also be used as the organiser's address when a new booking event is sent. This needs to be added to the application the first time it is run. It will be requested with a popup during the running process and the email from the Student Support Services office (or another address recognisable by Faculty Officers and other relative parties) would likely be used.

SMTP Email Address

The system is configured with an email account in order to send email to users who have forgotten and want to reset their password. The email account can be updated with the following properties (values are added as examples).

spring.mail.username=*meeting.scheduler.app@gmail.com*
spring.mail.password=*passmeet18*

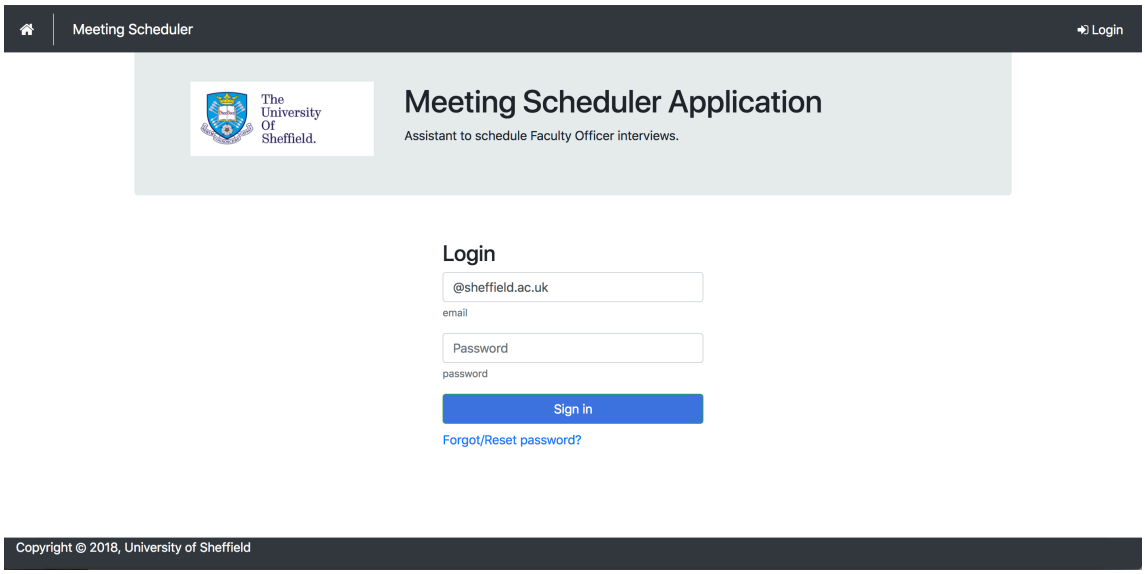
Again, the Student Support email account could be used (although any valid Google account is compatible).

K.2. User Manual

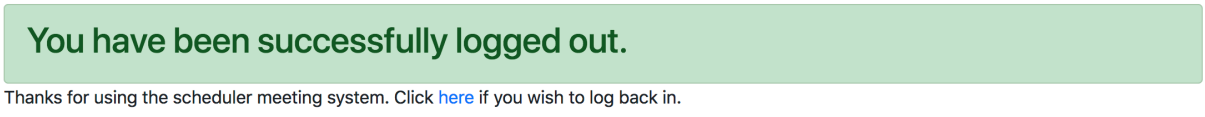
The user manual below contains instructions for accessing and using all of the Meeting Scheduler features.

K.2.1 Login and Logout

All pages in the Meeting Scheduler application require authentication for the user. Login using the page below. The page is displayed upon accessing the root url of the application. Accessing any other page without prior authentication will also redirect you to this page.

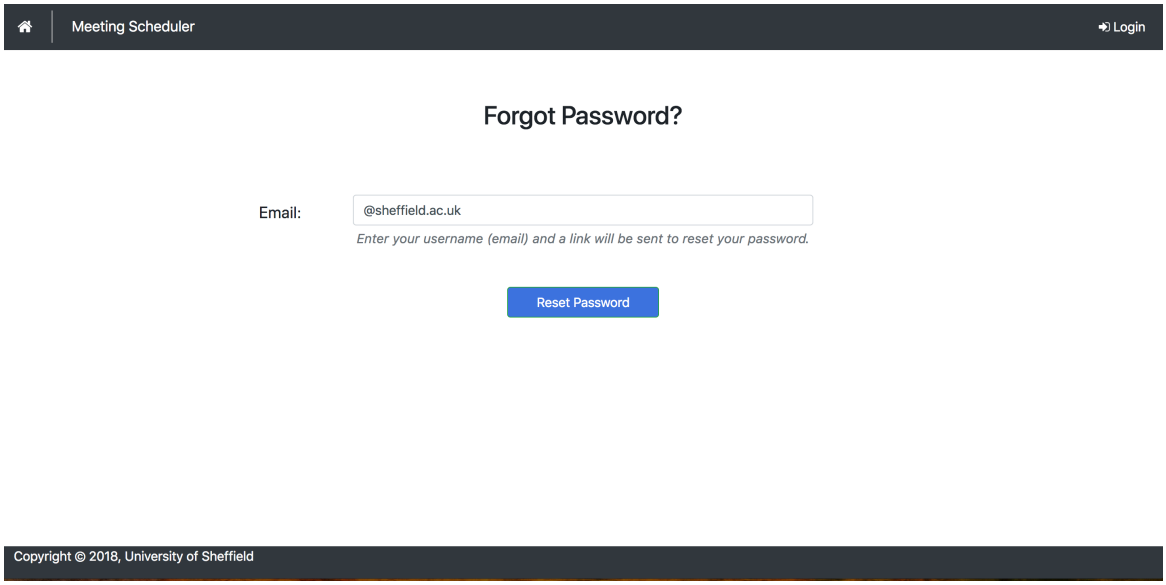


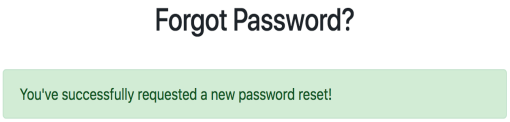
You can logout at any point using the Logout link at the top right of any MSA page. A message will confirm the logout and would contain a link to re-login if this is required.



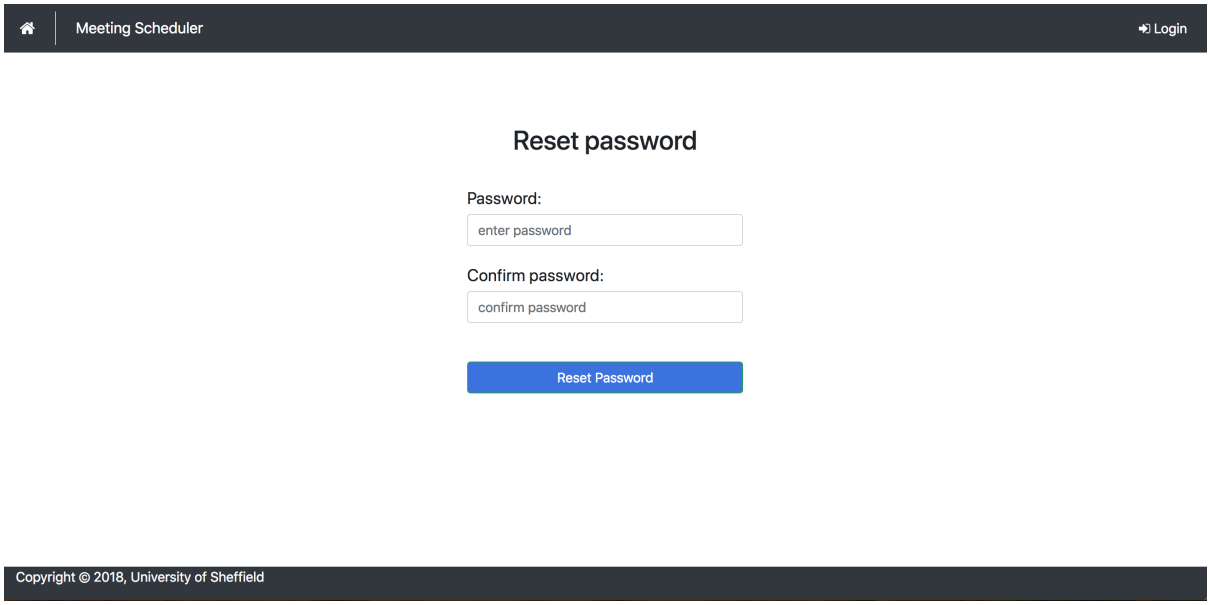
K.2.2 Reset Password

You can reset your password by clicking on the *Forgot/Reset Password* link on the login page. You will be shown a form to enter your email address.

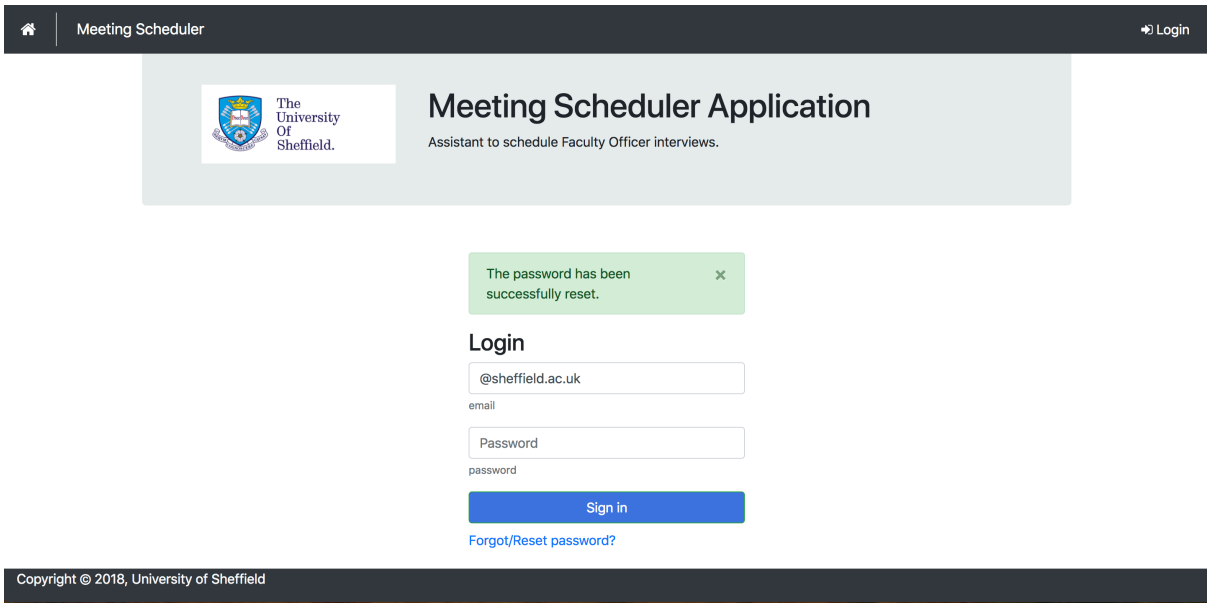




Submitting a request will display a success message.
An email will be sent to your account containing a link to reset your password.
The form is shown below.

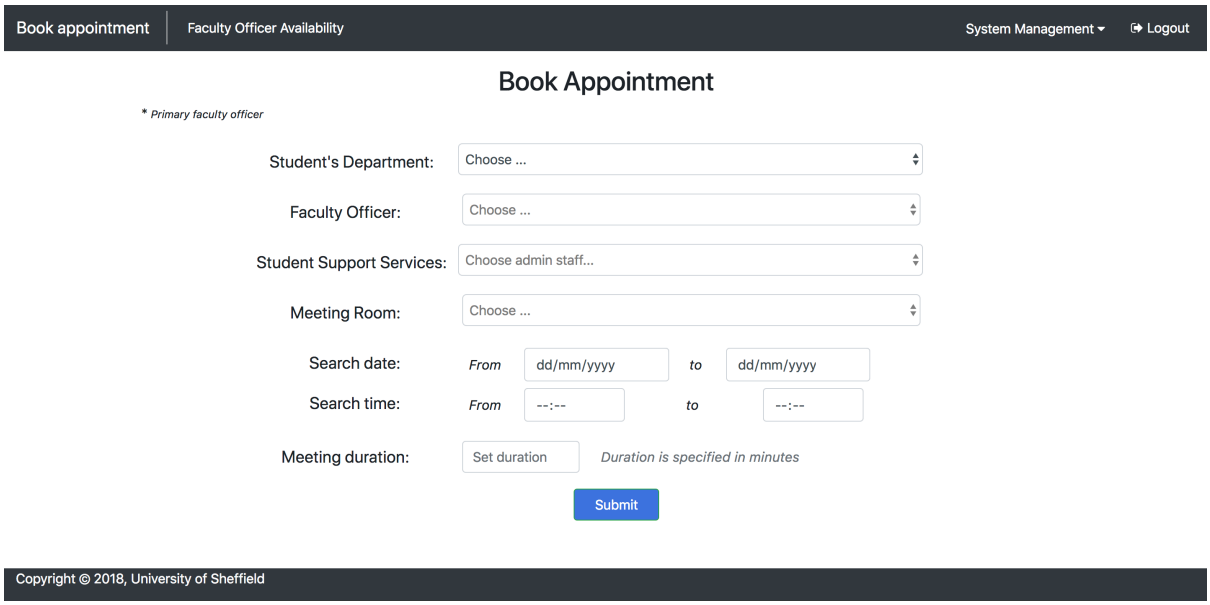


Edit the new password and submit the form. The updated password is stored in the system and you will be redirected to the login page where you can use your new credentials.

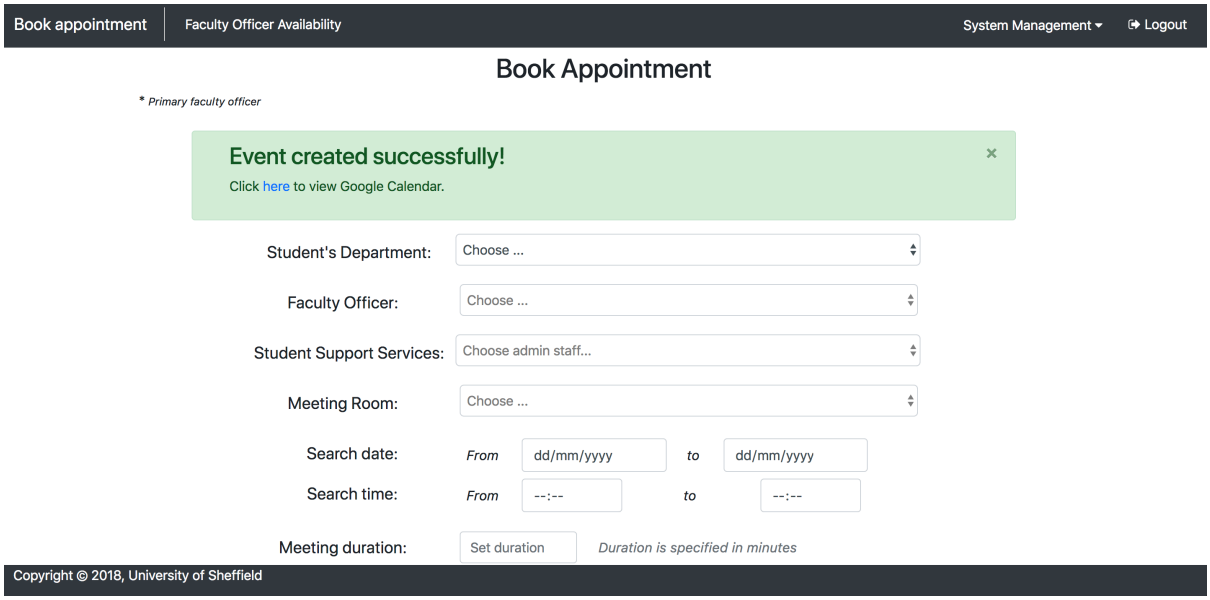


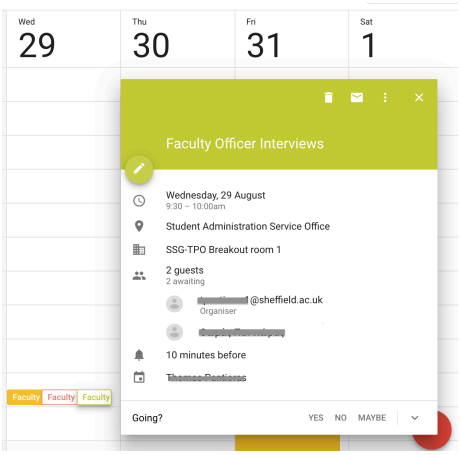
K.2.3 Book Appointment

Click on the *Book Appointment* navigation element on the top left of the screen. If a student Department value is selected, the *Faculty Officer* field will be populated with Faculty Officers which belong to that Department's Faculty (excluding Officers from the selected Department). If no value is selected in the Department field, the user would be able to select a Faculty Officer value from the full list of Officers in the University. Edit values for the rest of the fields including date, time and duration constraints for the appointment. Press *Submit*.



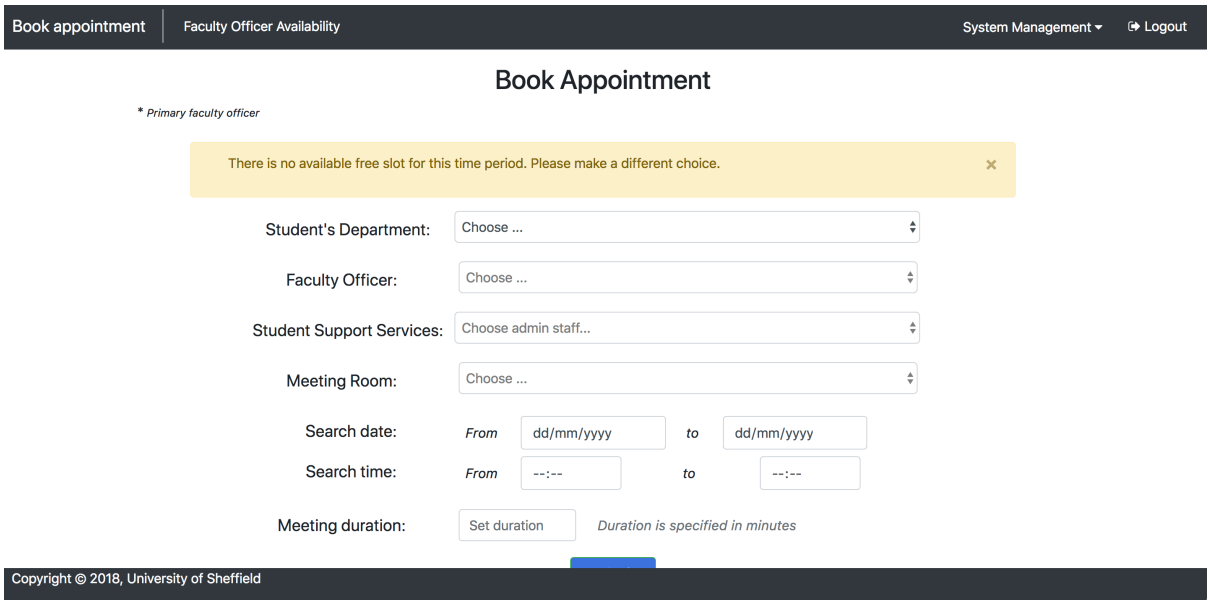
If availability is found across the calendars of the Faculty Officer, SSS admin staff and meeting room, the appointment is booked and a message is displayed.





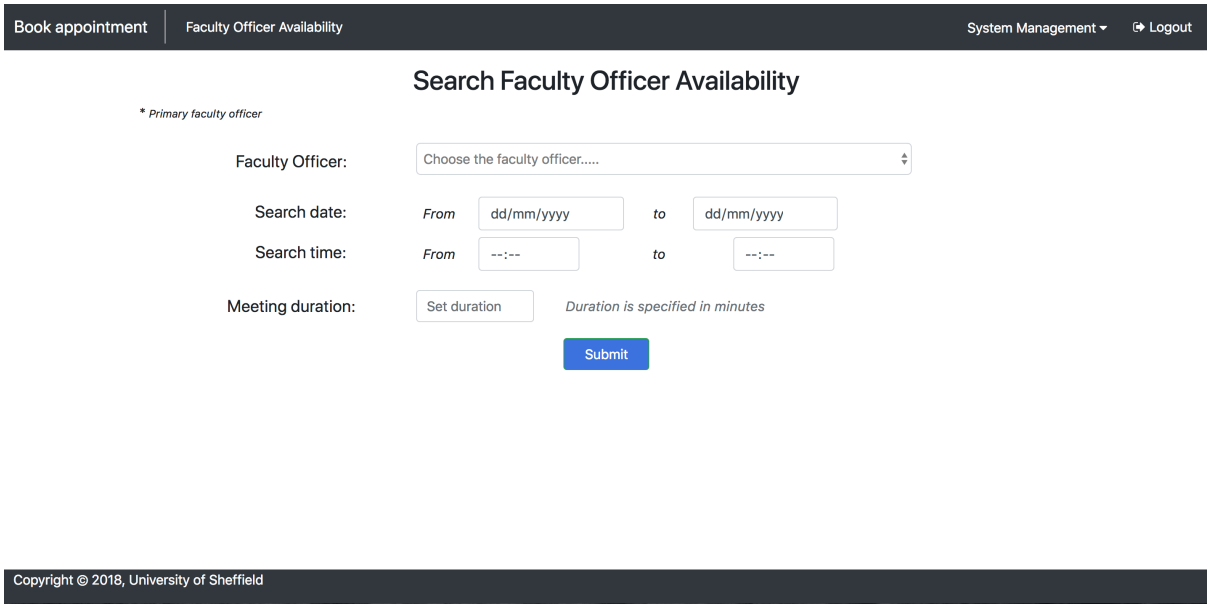
Clicking on the link within the response message, you will be redirected to your Google Calendar application where the created meeting event is displayed.

In the case of no availability an appropriate message will be displayed.

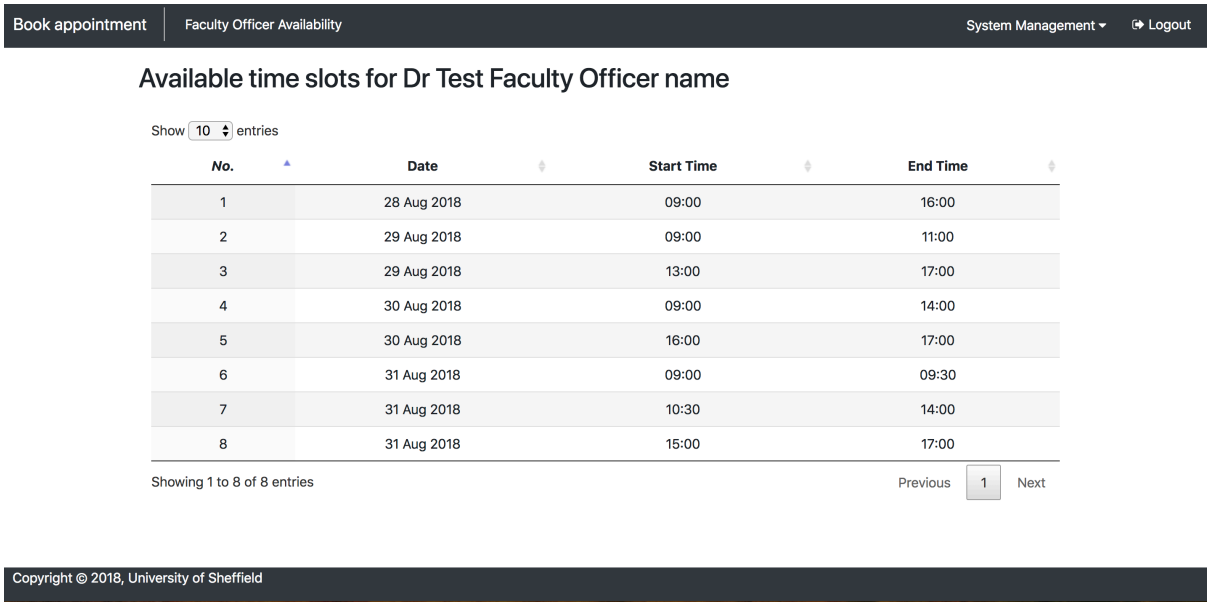


K.2.4 Searching for Faculty Officer Availability

Click on the *Faculty Officer Availability* navigation element on the top of the page. Edit the Faculty Officer to search for availability. Edit date, time and duration constraints for the search. Press *Submit*.

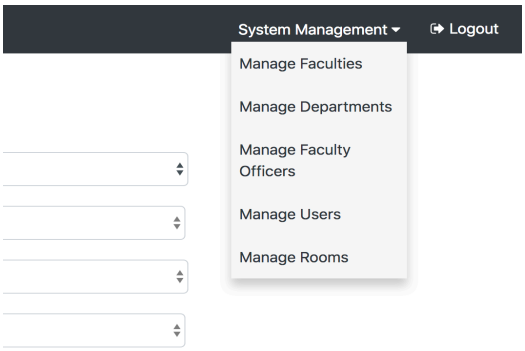


The retrieved list of available slots is displayed. Use the pagination links at the bottom to navigate through the pages of available slots. The number of slots displayed on the page is configurable at the top left of the table.



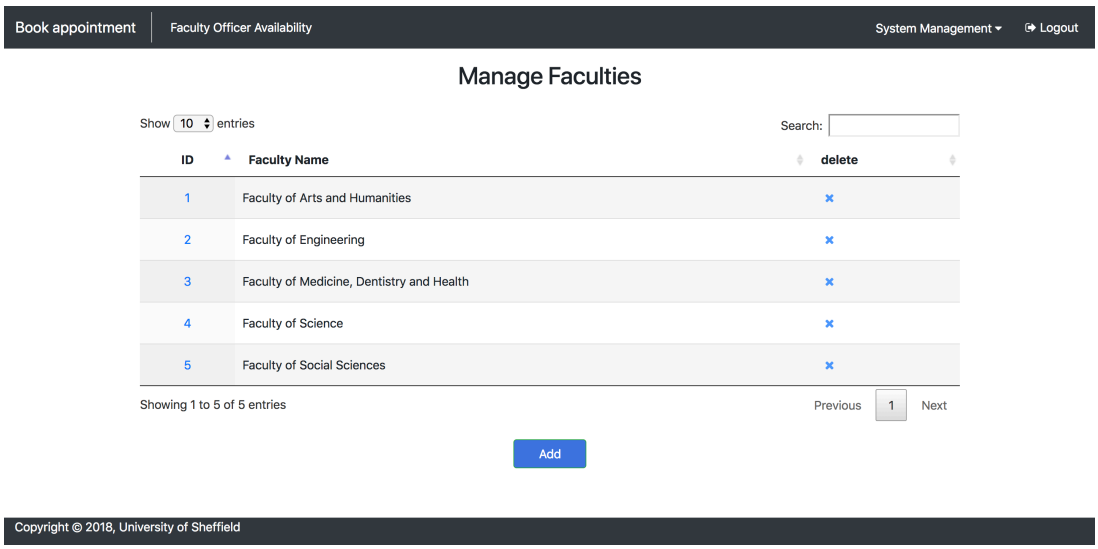
K.2.5 Managing System Data

The data management functionality is accessible through the System Management menu.



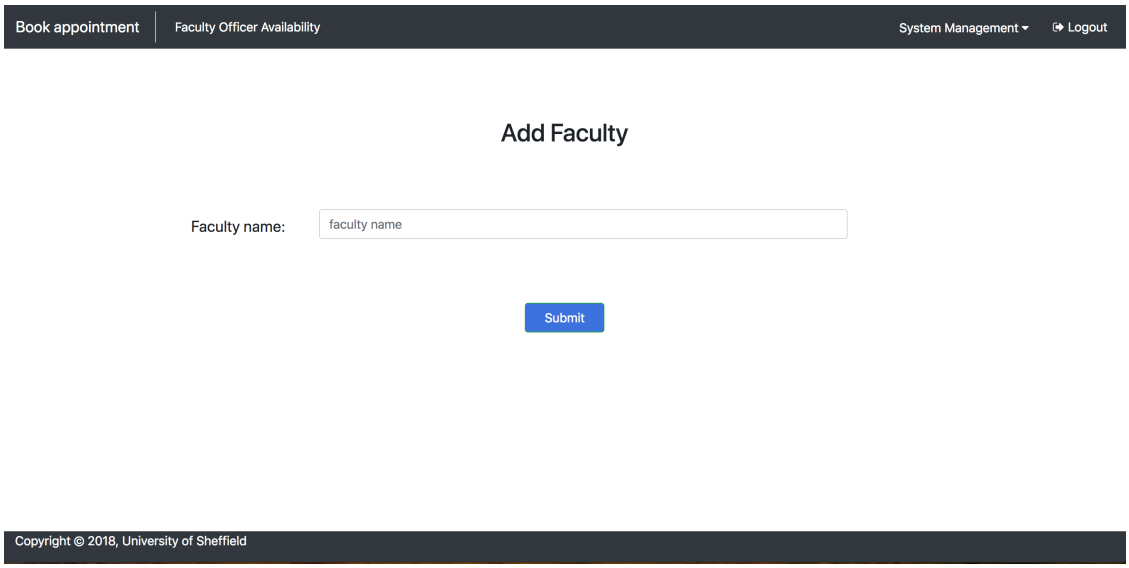
Managing Faculties

To manage a Faculty first access the Manage Faculties page to view the list of faculties. Click on the *Manage Faculties* element in the *System Management* menu. Search functionality is available at the top right of the page. The number of entities displayed on the screen is configurable on the top left. Pagination navigation links are available at the bottom right.



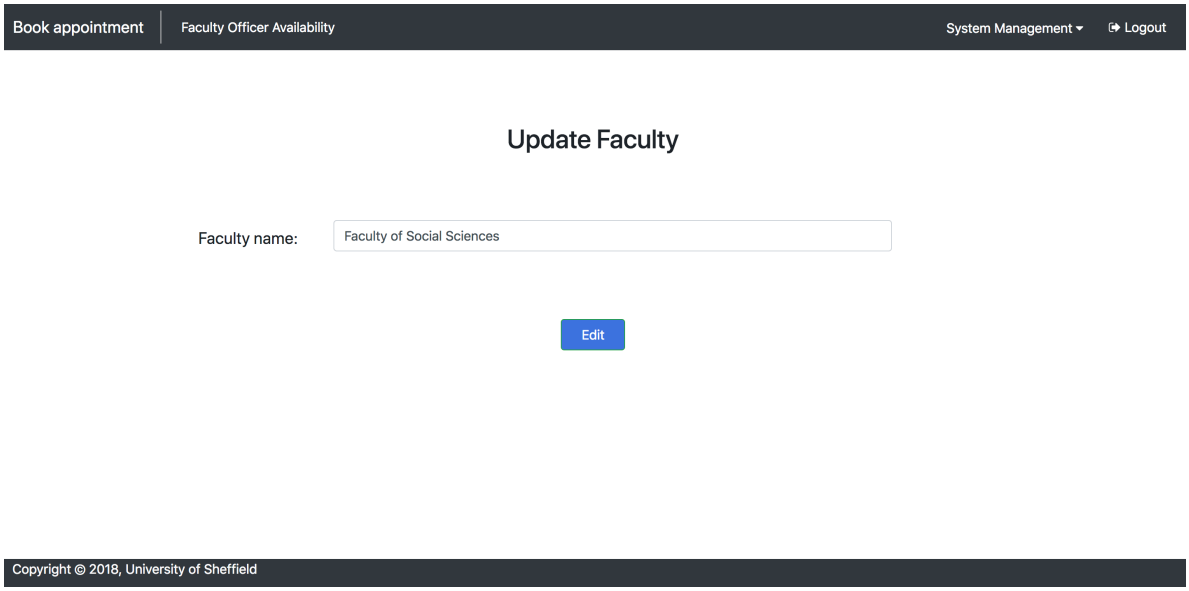
(i) Add a Faculty

To add a Faculty, press the *Add* button on the *Manage Faculties* page. Enter *Faculty name* value and press *Submit*.



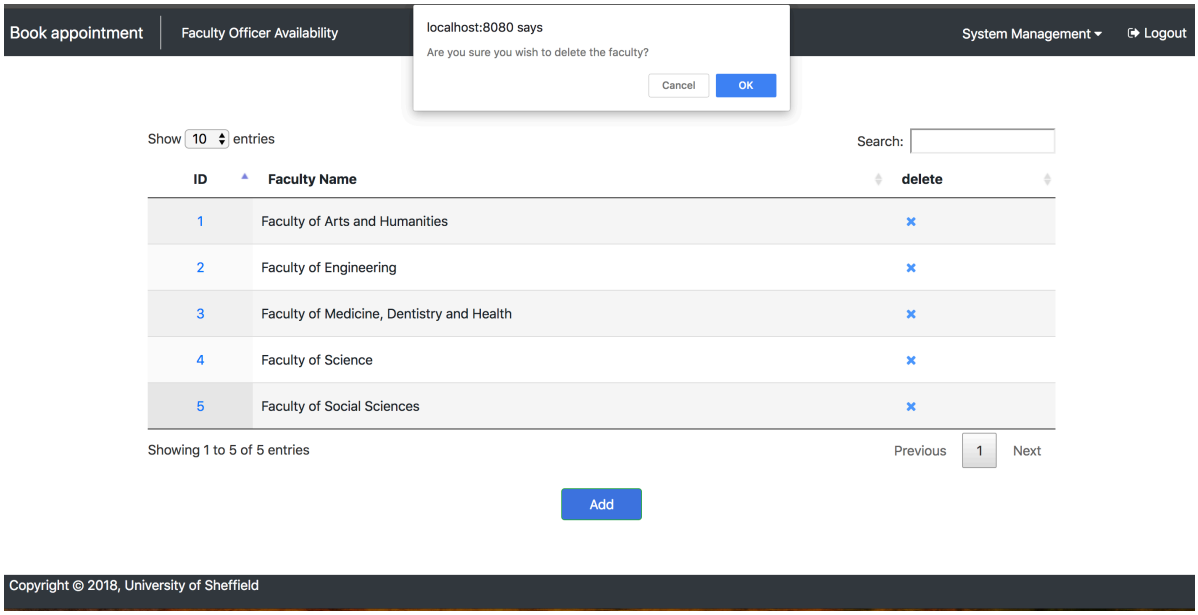
(ii) Edit a Faculty

To add a Faculty, click on the id of a specific Faculty on the *Manage Faculties* page. Edit *Faculty name* value and press *Edit*.



(iii) Delete a Faculty

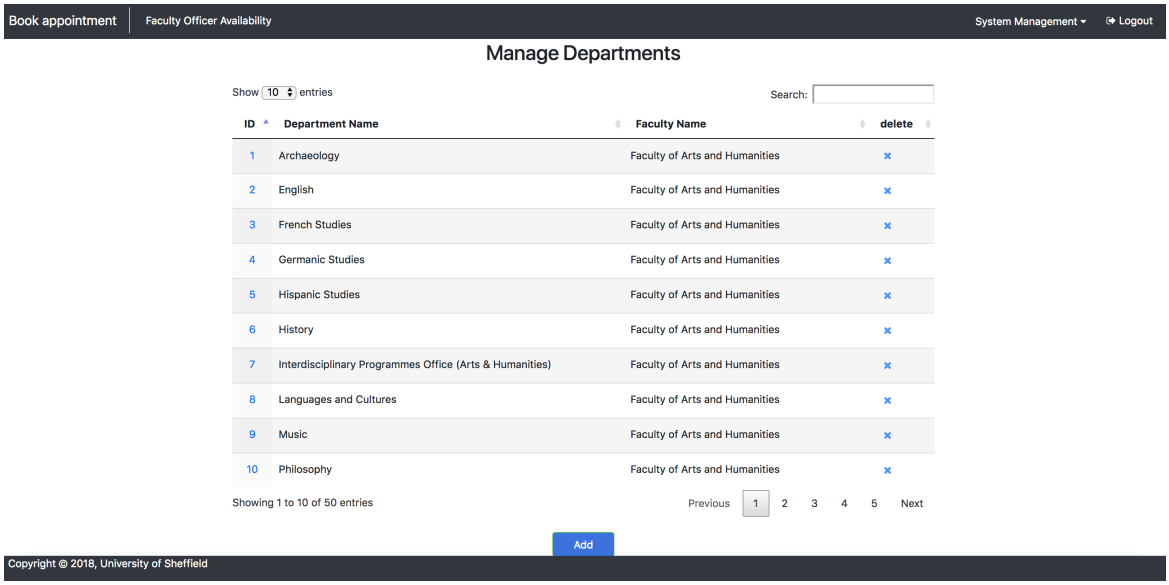
Access the *Manage Faculties* page. To delete a Faculty entity, press the delete symbol 'x' next to the Faculty to be removed.



Click *OK* to confirm the removal of the Faculty entity from the system.

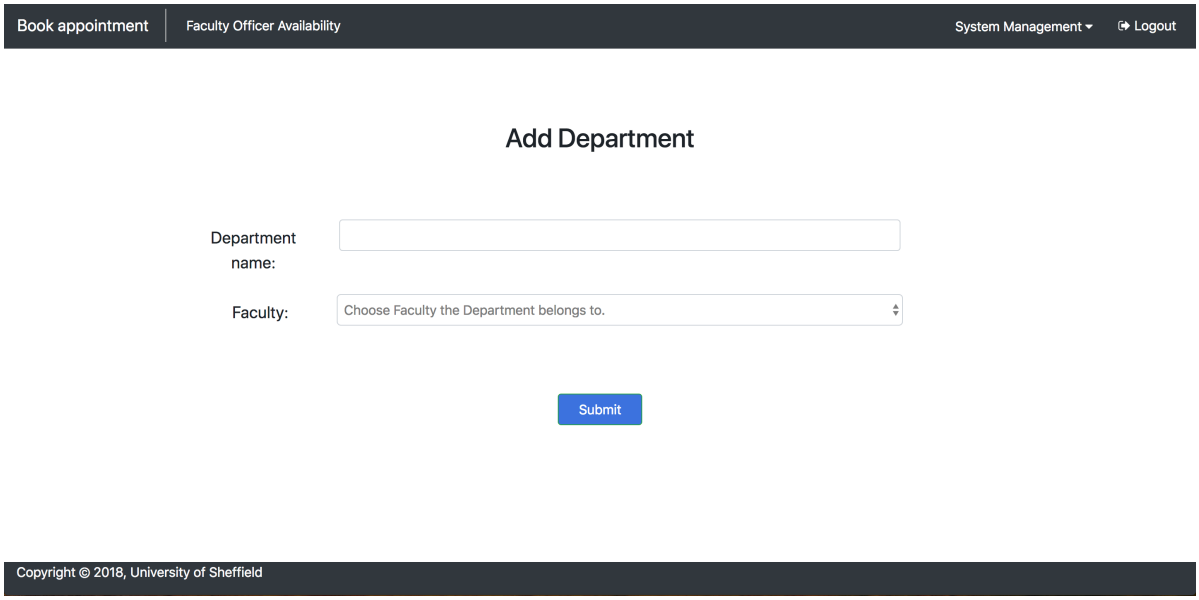
Managing Departments

To manage a Department first access the Manage Departments page to view the list of academic departments. Click on the *Manage Departments* element in the *System Management* menu. Search functionality is available at the top right of the page. The number of entities displayed on the screen is configurable on the top left. Pagination navigation links are available at the bottom right.



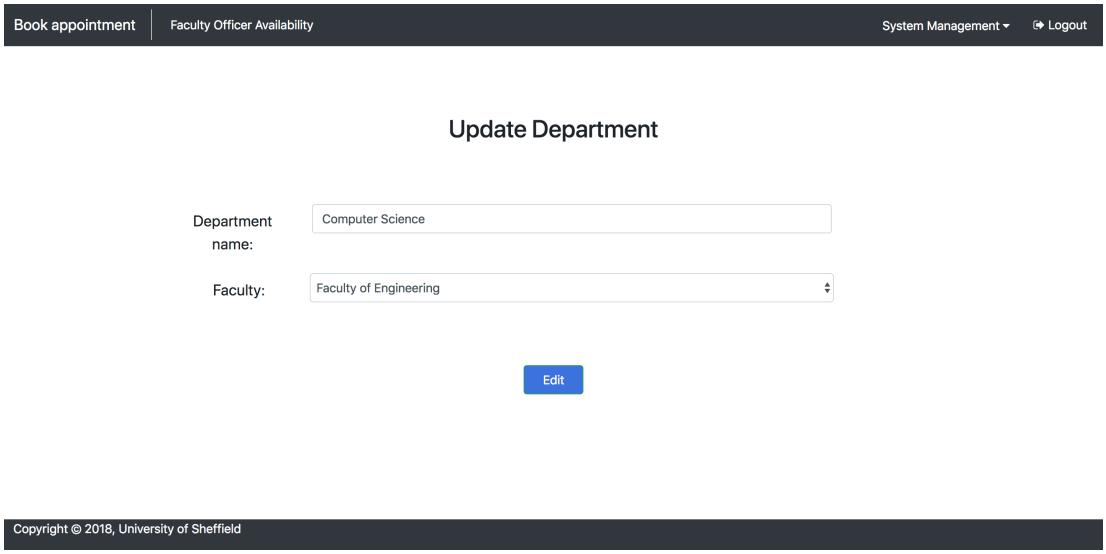
(i) Add a Department

To add a Department, press the *Add* button on the *Manage Departments* page. Enter values for Department and Faculty fields and press *Submit*.



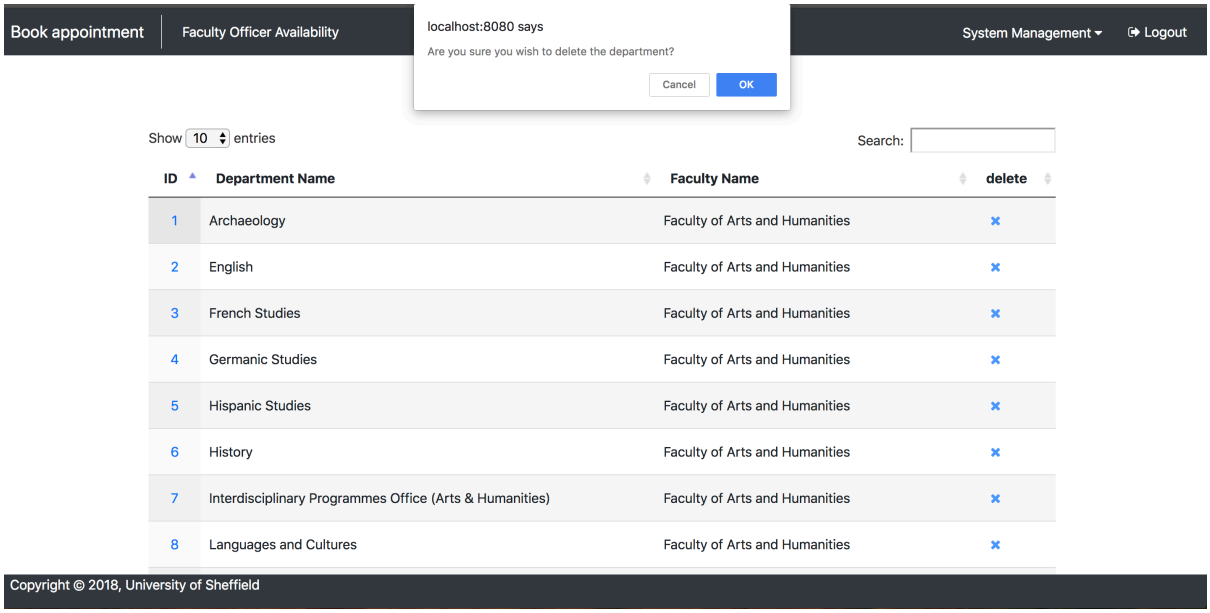
(ii) Edit a Department

To edit a Department, click on the id of a specific Department on the *Manage Departments* page. Update required values and press *Edit*.



(iii) Delete a Department

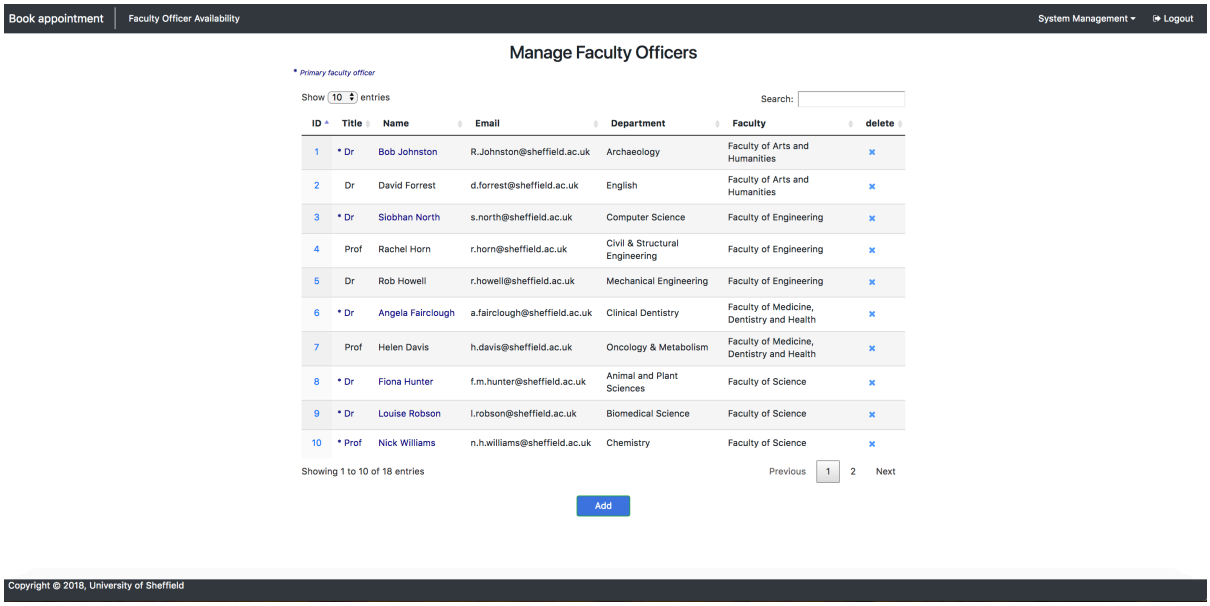
Access the *Manage Departments* page. To delete a Department entity, press the delete symbol 'x' next to the Department to be removed.



Click *OK* to confirm the removal of the Department entity from the system.

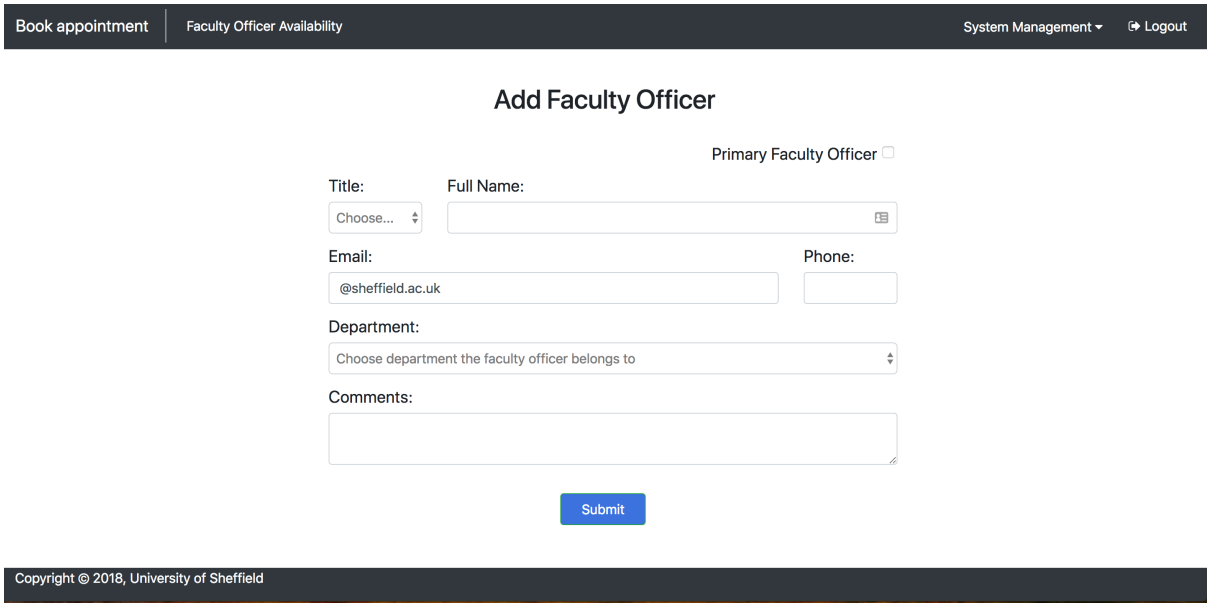
Managing Faculty Officers

To manage a Faculty Officer first access the Manage Faculty Officers page to view the list of current Officers. Click on the *Manage Faculty Officers* element in the *System Management* menu. Search functionality is available at the top right of the page. The number of entities displayed on the screen is configurable on the top left. Pagination navigation links are available at the bottom right.



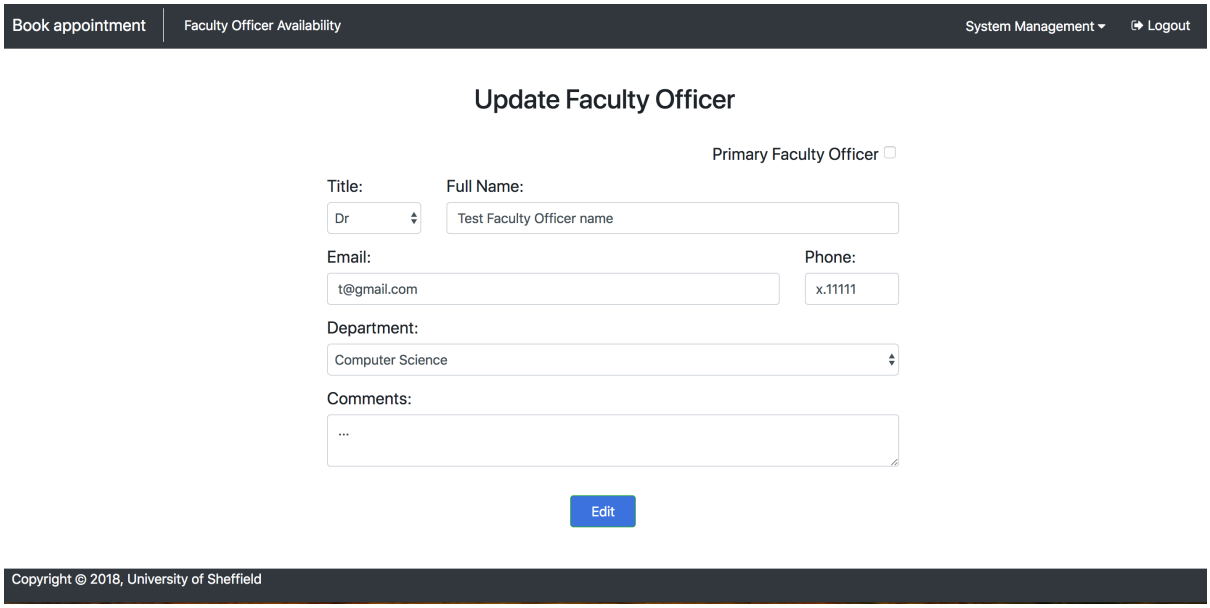
(i) Add a Faculty Officer

To add a Faculty Officer, press the *Add* button on the *Manage Faculty Officers* page. Enter values for the new Faculty Officer. Use the *Primary Faculty Officer* checkbox to define if the Officer should be preferred over others when booking an appointment automatically. Press *Submit.*



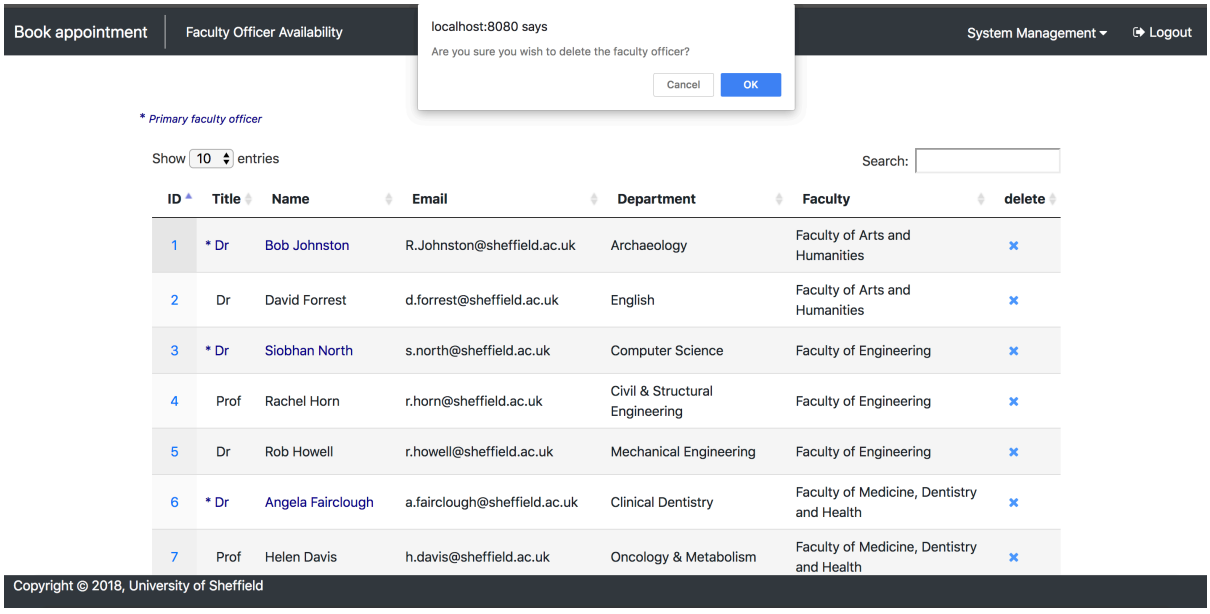
(ii) Edit a Faculty Officer

To edit a Faculty Officer, click on the id of a specific Faculty Officer on the *Manage Faculty Officers* page. Update required values and press *Edit*.



(iii) Delete a Faculty Officer

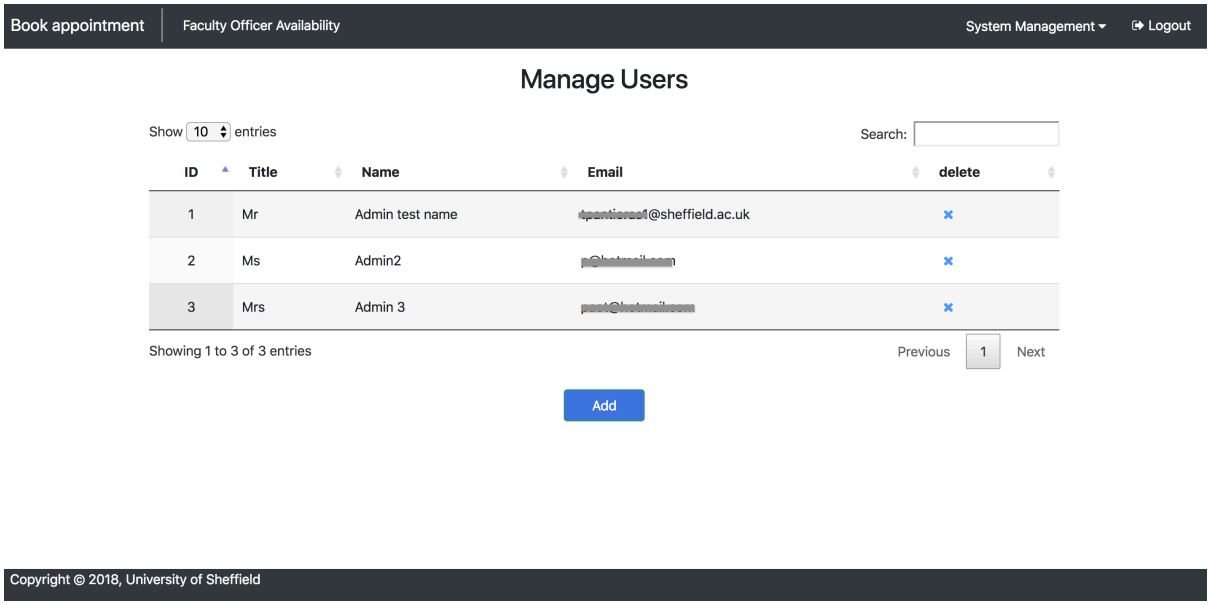
Access the *Manage Faculty Officers* page. To delete a Faculty Officer entity, press the delete symbol 'x' next to the Faculty Officer to be removed.



Click *OK* to confirm the removal of the Faculty Officer entity from the system.

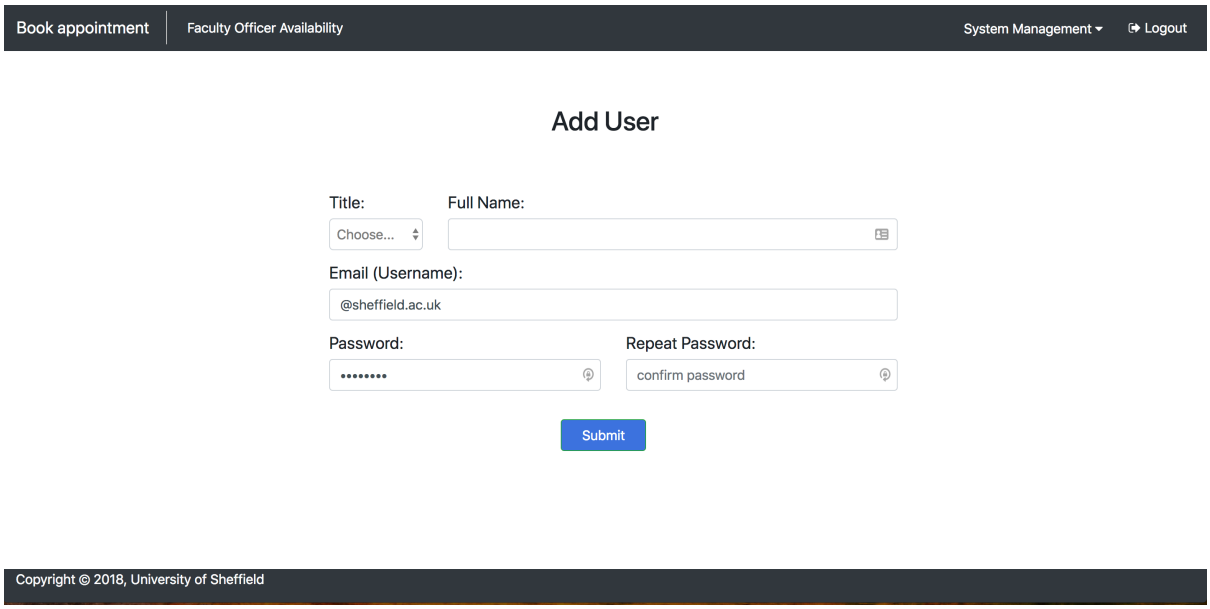
Managing Users

To manage a system user first access the Manage Users page to view the list of current users. The system users would be staff from the Student Support Services office. Click on the *Manage Users* element in the *System Management* menu. Search functionality is available at the top right of the page. The number of entities displayed on the screen is configurable on the top left. Pagination navigation links are available at the bottom right.



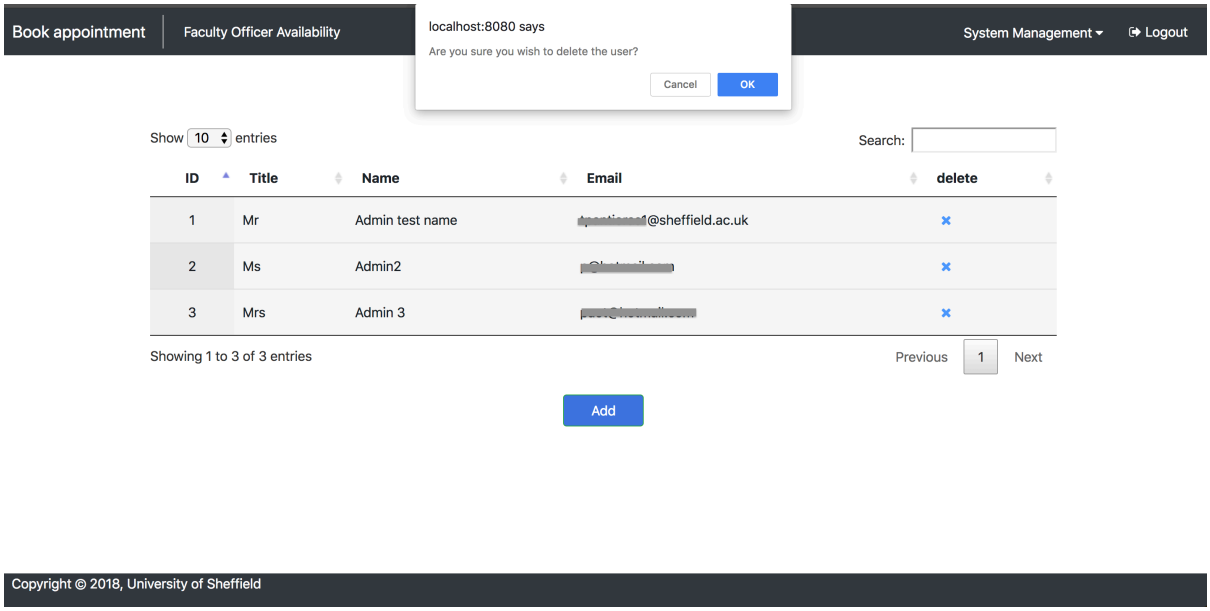
(i) Register a new User

To register a new User, press the *Add* button on the *Manage Users* page. Enter values for the new User. The password would need to be repeated to avoid mistakes. Press *Submit*.



(ii) Delete a User

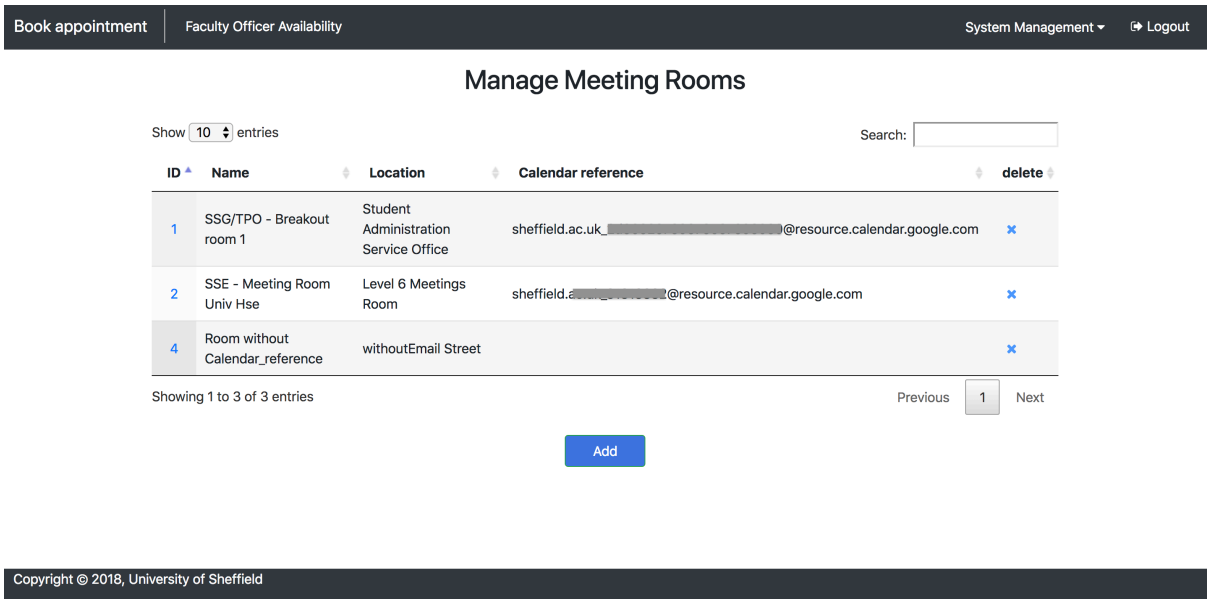
Access the *Manage Users* page. To delete a User entity, press the delete symbol ‘x’ next to the User to be removed.



Click *OK* to confirm the removal of the User entity from the system.

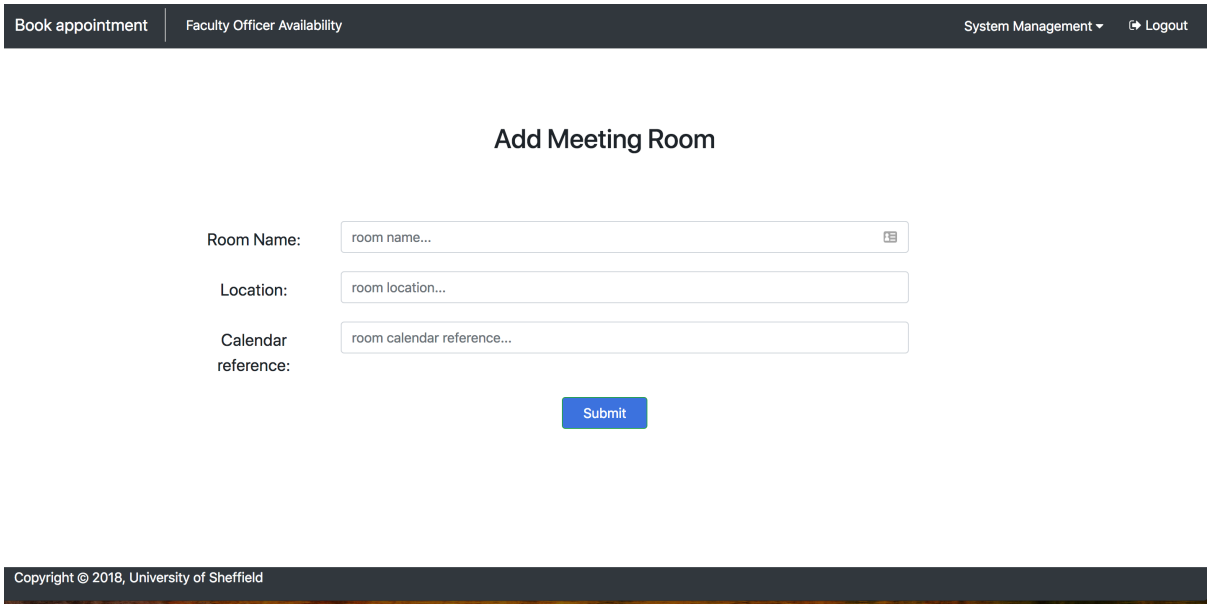
Managing Meeting Rooms

To manage a Meeting Room first access the *Manage Meeting Rooms* page to view the list of current rooms. Click on the *Manage Rooms* element in the *System Management* menu. Search functionality is available at the top right of the page. The number of entities displayed on the screen is configurable on the top left. Pagination navigation links are available at the bottom right.



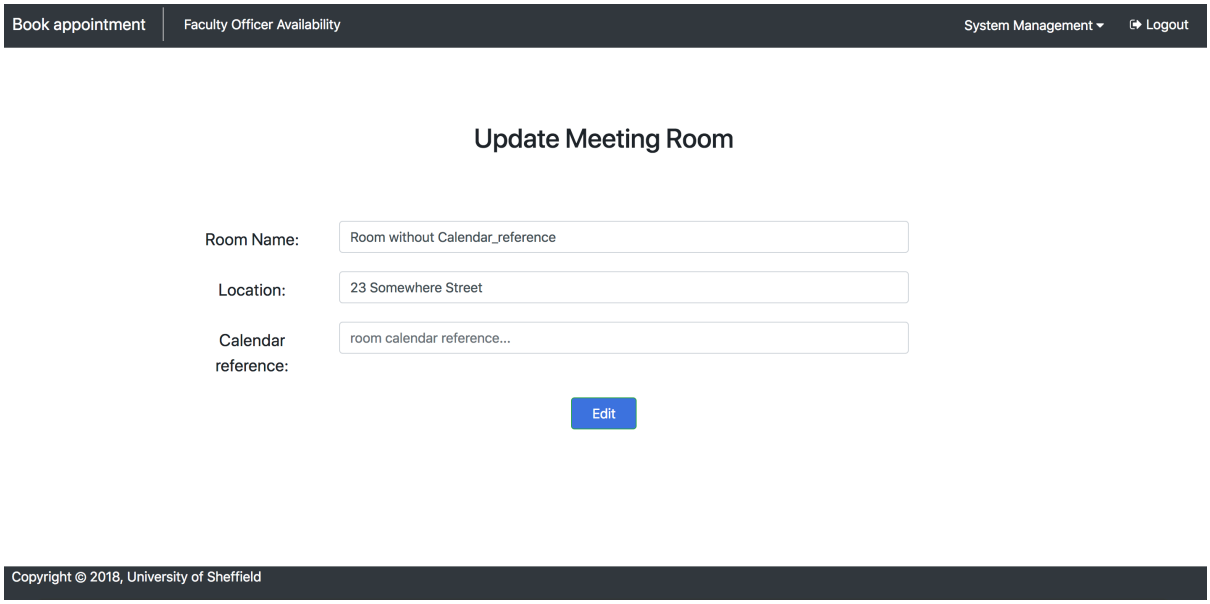
(i) Add a Meeting Room

To add a Meeting Room, press the *Add* button on the *Manage Meeting Rooms* page. Enter values for the new room. If added, the calendar reference would be used to book a meeting event on this room's calendar, otherwise the room availability would not be taken into account when searching for free appointment slots. Press *Submit.*



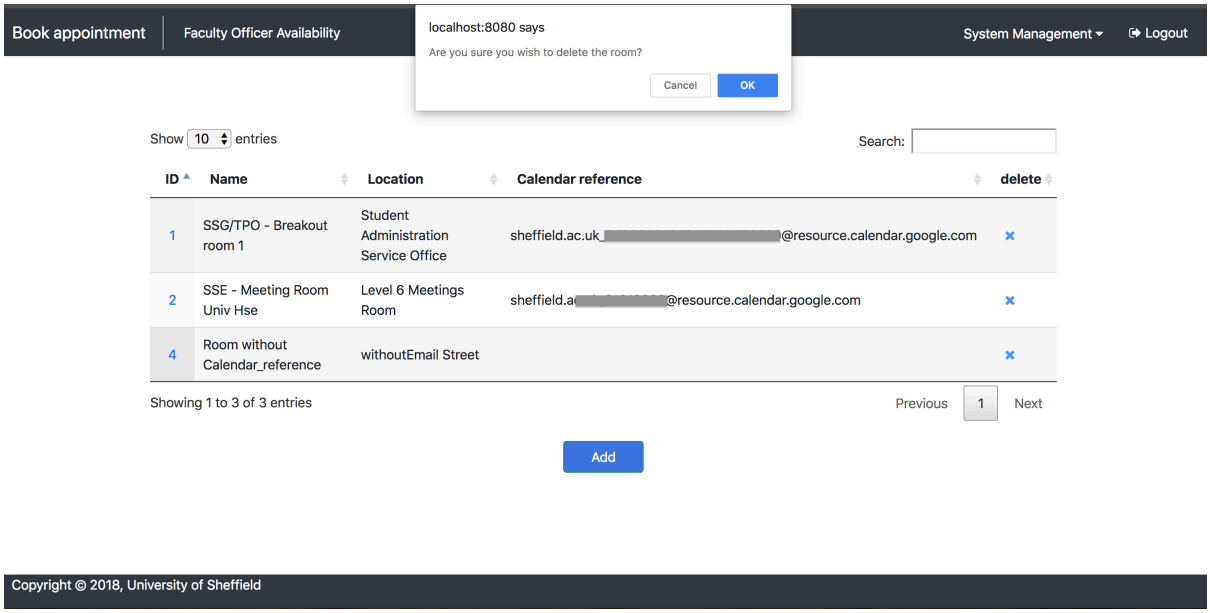
(ii) Edit a Meeting Room

To edit a Meeting Room, click on the id of a specific Meeting Room on the *Manage Meeting Rooms* page. Update required values and press *Edit*.



(iii) Delete a Meeting Room

Access the *Manage Meeting Rooms* page. To delete a Meeting Room entity press the delete symbol ‘x’ next to the Room to be removed.



Click *OK* to confirm the removal of the User entity from the system.

Appendix L. Setting up Development Environment

In order to setup the development environment, first unzip the scheduleMeetings.zip file into your workspace. You will need Java 8 and Maven 3 to be installed in the environment.
Run the following command within the unzipped folder.

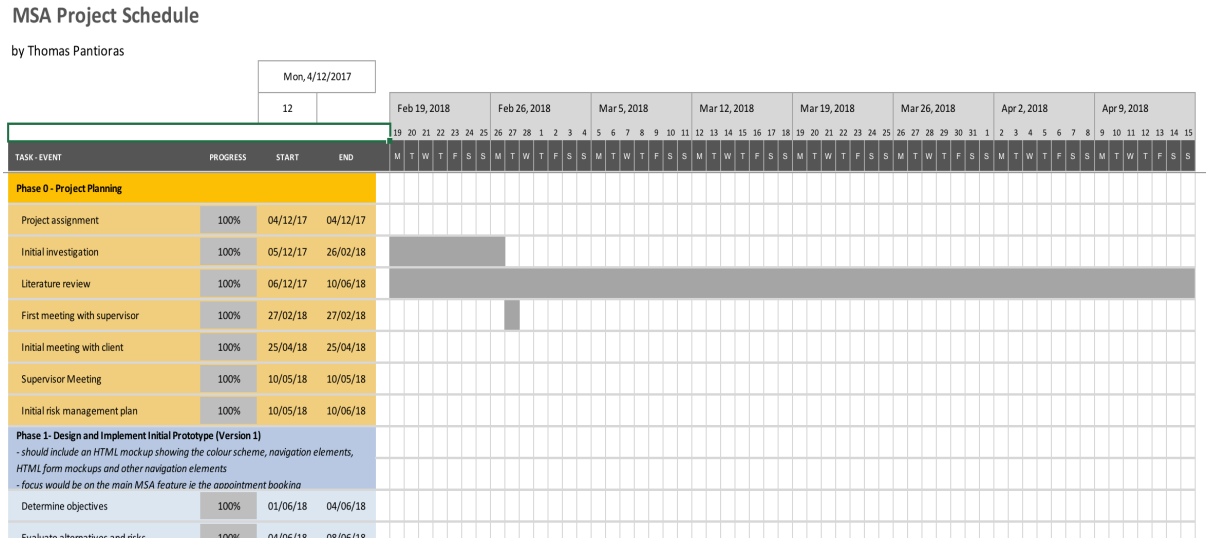
mvn clean install

The command will install all the required dependencies for the application to compile and run. The original work was done using the IntelliJ IDE but any Java IDE can be used for further development.

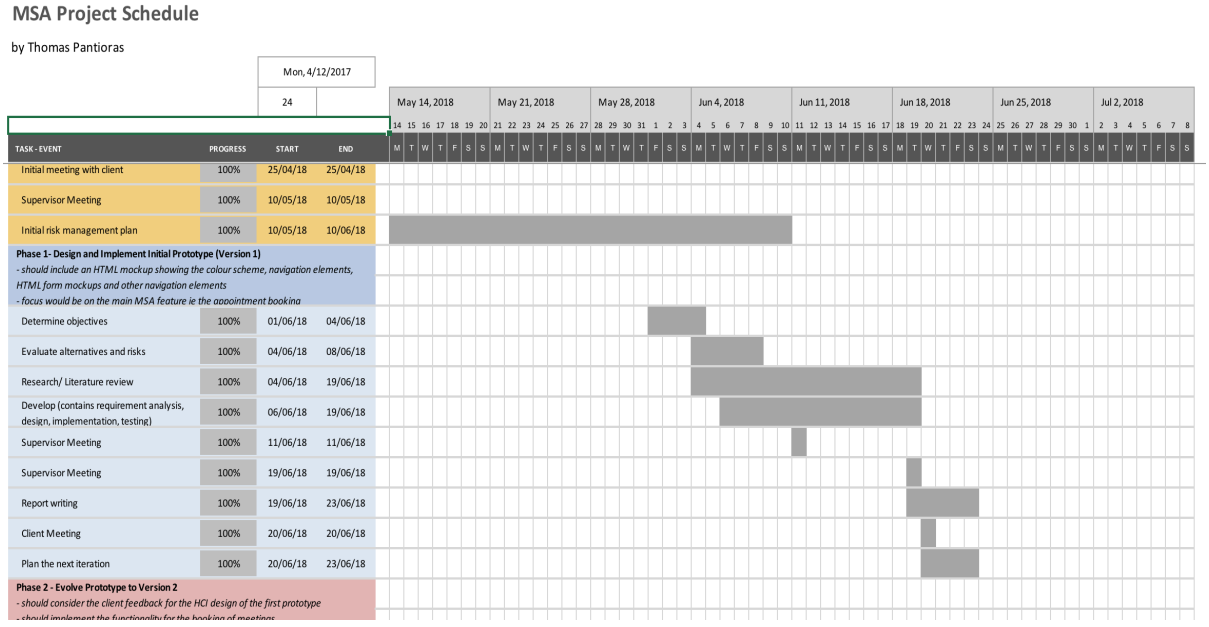
The first time the application is run, a popup form will request for the primary calendar address to be added. This will be used in requests to the Google Calendar API when retrieving or creating calendar events.

Appendix M. Project Schedule

Screenshots from the created Gantt chart showing scheduled activities in each spiral iteration of the Meeting Scheduler project. The Excel spreadsheet is provided with the project artefacts.



Screenshot of Gantt chart schedule showing phase-0 activities



Screenshot of Gantt chart schedule showing phase-1 activities

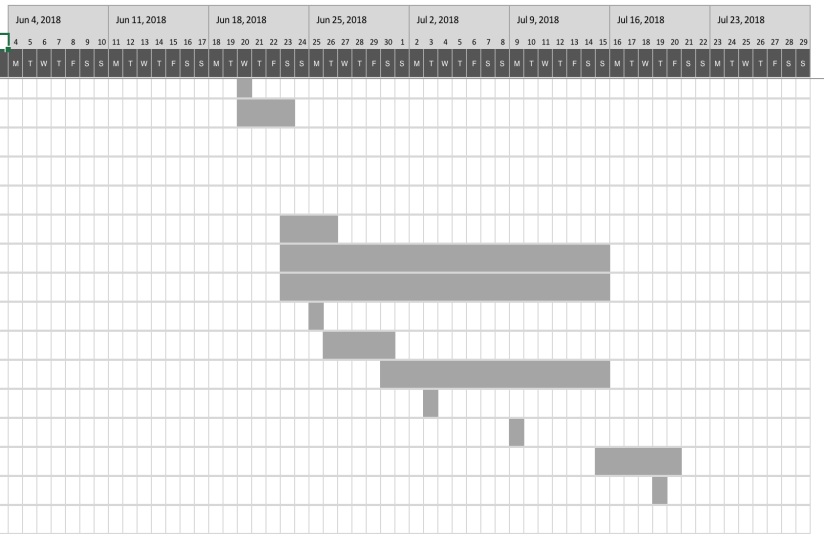
MSA Project Schedule

by Thomas Pantioras

Mon, 4/12/2017

27

TASK - EVENT	PROGRESS	START	END
Client Meeting	100%	20/06/18	20/06/18
Plan the next iteration	100%	20/06/18	23/06/18
Phase 2 - Evolve Prototype to Version 2 *- should consider the client feedback for the HCI design of the first prototype* *- should implement the functionality for the booking of meetings* *- should include mockup for the management of data entities in the system* *- should include mockup for the login functionality*			
Determine objectives	100%	23/06/18	26/06/18
Report writing	100%	23/06/18	15/07/18
Research/ Literature review	100%	23/06/18	15/07/18
Supervisor Meeting	100%	25/06/18	25/06/18
Evaluate alternatives and risks	100%	26/06/18	30/06/18
Develop (contains requirement analysis, design, implementation, testing)	100%	30/06/18	15/07/18
Supervisor Meeting	100%	03/07/18	03/07/18
Supervisor Meeting	100%	09/07/18	09/07/18
Plan the next iteration	100%	15/07/18	20/07/18
Client Meeting	100%	19/07/18	19/07/18
Phase 3 - Evolve Prototype to Version 3 *- should consider the client feedback for the HCI design and functionality*			



Screenshot of Gantt chart schedule showing phase-2 activities

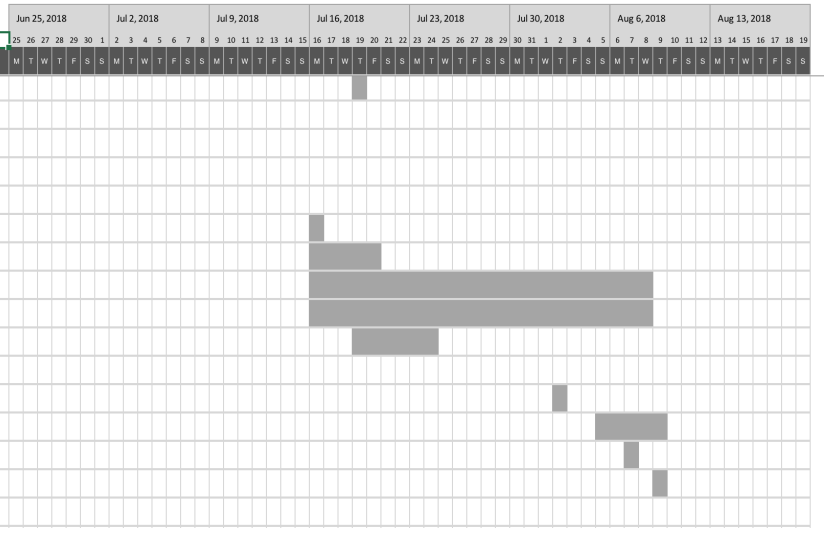
MSA Project Schedule

by Thomas Pantioras

Mon, 4/12/2017

30

TASK - EVENT	PROGRESS	START	END
Client Meeting	100%	19/07/18	19/07/18
Phase 3 - Evolve Prototype to Version 3 *- should consider the client feedback for the HCI design and functionality of the second prototype* *- should implement the functionality for the management of data entities in the system* *- should implement the login functionality* *- should include mockup for searching the availability of Faculty Officers* (the requirement was identified during the current phase)			
Supervisor Meeting	100%	16/07/18	16/07/18
Determine objectives	100%	16/07/18	20/07/18
Research/ Literature review	100%	16/07/18	08/08/18
Report writing	100%	16/07/18	08/08/18
Evaluate alternatives and risks	100%	19/07/18	24/07/18
Develop (contains requirement analysis, design, implementation, testing)	100%	22/08/18	08/08/18
Supervisor Meeting	100%	02/08/18	02/08/18
Plan the next iteration	100%	05/08/18	09/08/18
Client Meeting	100%	07/08/18	07/08/18
Supervisor Meeting	100%	09/08/18	09/08/18
Phase 4 - Evolve Prototype to Version 4 (Final Version) *- should consider and adjust functionality based on the client feedback for the version 3*			



Screenshot of Gantt chart schedule showing phase-3 activities

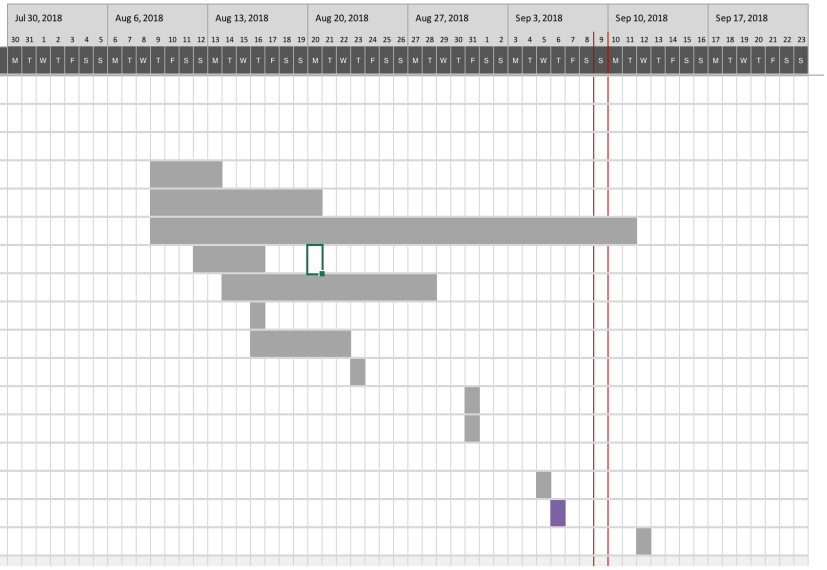
MSA Project Schedule

by Thomas Pantioras

Mon, 4/12/2017

35

TASK - EVENT	PROGRESS	START	END
Phase 4 - Evolve Prototype to Version 4 (Final Version) *- should consider and adjust functionality based on the client feedback for the version 3 of the prototype* *- should implement the functionality for searching Faculty Officer availability* *- should consider hosting issues*			
Determine objectives	100%	09/08/18	13/08/18
Research/ Literature review	100%	09/08/18	20/08/18
Report writing	100%	09/08/18	11/09/18
Evaluate alternatives and risks	100%	12/08/18	16/08/18
Develop (contains requirement analysis, design, implementation, testing)	100%	14/08/18	28/08/18
Supervisor Meeting	100%	16/08/18	16/08/18
Investigate hosting requirements	100%	16/08/18	22/08/18
Supervisor Meeting	100%	23/08/18	23/08/18
Supervisor Meeting	100%	31/08/18	31/08/18
Client Meeting - User acceptance testing - Final demonstration and system approval	100%	31/08/18	31/08/18
Project presentation	100%	05/09/18	05/09/18
Supervisor Meeting (not required)		06/09/18	06/09/18
Project deadline/ report submission	100%	12/09/18	12/09/18



Screenshot of Gantt chart schedule showing phase-4 activities

Appendix N. MSA Questionnaires

N.1 QUESTIONNAIRE 1

Meeting Scheduler Questionnaire

Currently

1. *Is there always availability of Faculty Officers?*
   - *Yes*
   - *√ No*
2. *Is there always availability of rooms for the meetings?*
   - *Yes*
   - *√ No*
3. *What happens in the case of no availability as in questions 1 and 2?*

We have to find another date and/or another venue

In the new software application

1. *How often do you require to update (add/edit/delete) the faculties?*
   - *Every year*
   - *Every few years*
   - *√ Every decade*
2. *How often do you require to update (add/edit/delete) the faculty departments?*
   - *Every year*
   - *√ Every few years*
   - *Every decade*
3. *How often do you require to update (add/edit/delete) the faculty officers?*
   - *√ Every year*
   - *Every few years*
   - *Every decade*
4. *Do the faculty officers move often between departments?*
   - *Yes, every year*
   - *Sometimes, every few years*
   - *√ Almost never*
5. *How often does the allocation priority of the Faculty Officers changing?*

- *√ Often, within the year*
- *Sometimes, every few years*
- *Very rarely*

6. *How likely is to update the list of suitable rooms where the meetings take place?*
    - *√ Very likely*
    - *Possible*
    - *Almost never*
7. *Would you require a single account credentials or one per system user?*
    - *Single*
    - *√ One per user*
8. *Would you require any student data to be stored in the system?*
    - *√ No*
    - *Yes. Please describe below*

9. *Are the Faculty Officers aware that data against them will be stored in the system?*
    - *No*
    - *Yes, they have given their consent*
10. *Would you prefer a manual selection of the appointment time or an automated allocation based on the availability of all relative parties?*
    - *Manual selection*
    - *√ Automated allocation*
11. *Would you prefer to select the Faculty officer for a meeting or automatically selected (based on predefined priority order)?*
    - *√ Manually select the Faculty Officer*
    - *Automatically selected by the system*
12. *Would you prefer to select the room for a meeting or automatically selected (based on predefined priority order)?*
    - *√ Manually select the room*
    - *Automatically select the first available*
13. *Would you need to send any message to the meeting participants?*
    - *No*
    - *√ Yes*
14. *What should the system do if a Faculty Officer rejects an appointment request?*

- *√ It should try to find an alternative allocation with the same Faculty Officer*
- *It should try to find an alternative allocation with another Faculty Officer*
- *Nothing, it will be dealt offline*

15. *What should the system do if a Faculty Officer cancels an appointment booking?*
    - *√ It should try to book an alternative appointment at another time*
    - *Nothing, it will be dealt with offline*

16. *Should the current logged in user of the system always be the Admin staff participating in meeting or should the system allow for other Admin Staff users to be selected*
    - *The current user would be participating in the meeting*
    - *√ The system should allow to select the Admin staff to participate in the meeting*

17. *What should the system do if an Admin Staff rejects an appointment (only applicable if 'any' Admin Staff users can be selected for a booking)?*
    - *Try to book an alternative appointment at a different time*
    - *√ Nothing, it will be deal with offline*

18. *What should the system do if an Admin Staff cancels an appointment?*
    - *Try to book an appointment at a different time*
    - *√ Nothing, it will be dealt with offline*

19. *What should the system do if a student rejects an appointment?*
    - *It should try to book another appointment*
    - *√ Not applicable, the student will be notified offline*

20. *What should the system do if a student cancels an appointment?*
    - *It should try to book another appointment*
    - *√ Not applicable, the student will be notified offline*

21. *Would you require to view current bookings against students/officers/admins from within the system?*
    - *Yes*
    - *√ No, this will be done directly on the google calendar*

N.2 QUESTIONNAIRE 2

Meeting Scheduler Questionnaire

Currently

1. *Is there always availability of Faculty Officers?*
   - ~~*Yes*~~
   - *No*
2. *Is there always availability of rooms for the meetings?*
   - ~~*Yes*~~
   - *No*
3. *What happens in the case of no availability as in questions 1 and 2?*

Either move the date to suit the FO, use our secondary FO. If all that fails, ask them to suggest another FO we could use. Rooms – we have an idea of some other suitable rooms and we would use Room Bookings system to see if they were free.

In the new software application

1. *How often do you require to update (add/edit/delete) the faculties?*
   - ~~*Every year*~~
   - ~~*Every few years*~~
   - *Every decade*
     Or even more than that – only a change in the way the University runs would prompt this kind of change.
2. *How often do you require to update (add/edit/delete) the faculty departments?*
   - ~~*Every year*~~
   - *Every few years*
   - ~~*Every decade*~~
3. *How often do you require to update (add/edit/delete) the faculty officers?*
   - *Every year*
   - ~~*Every few years*~~
   - ~~*Every decade*~~
4. *Do the faculty officers move often between departments?*
   - ~~*Yes, every year*~~
   - ~~*Sometimes, every few years*~~
   - ~~*Almost never*~~
     They don't change departments – though they do stop being FOs.

5. *How often does the allocation priority of the Faculty Officers changing?*
    - *Often, within the year*
    - ~~*Sometimes, every few years*~~
    - ~~*Very rarely*~~
6. *How likely is to update the list of suitable rooms where the meetings take place?*
    - ~~*Very likely*~~
    - *Possible*
    - ~~*Almost never*~~
7. *Would you require a single account credentials or one per system user?*
    - *Single*
    - ~~*One per user*~~
8. *Would you require any student data to be stored in the system?*
    - *No*
    - ~~*Yes. Please describe below*~~

9. *Are the Faculty Officers aware that data against them will be stored in the system?*
    - *No*
    - ~~*Yes, they have given their consent*~~
10. *Would you prefer a manual selection of the appointment time or an automated allocation based on the availability of all relative parties?*
    - *Manual selection*
    - ~~*Automated allocation*~~
11. *Would you prefer to select the Faculty officer for a meeting or automatically selected (based on predefined priority order)?*
    - *Manually select the Faculty Officer*
    - ~~*Automatically selected by the system*~~
12. *Would you prefer to select the room for a meeting or automatically selected (based on predefined priority order)?*
    - *Manually select the room*
    - ~~*Automatically select the first available*~~
13. *Would you need to send any message to the meeting participants?*
    - ~~*No*~~
    - *Yes*

14. *What should the system do if a Faculty Officer rejects an appointment request?*
    - *It should try to find an alternative allocation with the same Faculty Officer*
    - ~~*It should try to find an alternative allocation with another Faculty Officer*~~
    - ~~*Nothing, it will be dealt offline*~~
15. *What should the system do if a Faculty Officer cancels an appointment booking?*
    - ~~*It should try to book an alternative appointment at another time*~~
    - *Nothing, it will be dealt with offline*
16. *Should the current logged in user of the system always be the Admin staff participating in meeting or should the system allow for other Admin Staff users to be selected*
    - ~~*The current user would be participating in the meeting*~~
    - *The system should allow to select the Admin staff to participate in the meeting*
17. *What should the system do if an Admin Staff rejects an appointment (only applicable if 'any' Admin Staff users can be selected for a booking)?*
    - ~~*Try to book an alternative appointment at a different time*~~
    - *Nothing, it will be deal with offline*
18. *What should the system do if an Admin Staff cancels an appointment?*
    - ~~*Try to book an appointment at a different time*~~
    - *Nothing, it will be dealt with offline*
19. *What should the system do if a student rejects an appointment?*
    - ~~*It should try to book another appointment*~~
    - *Not applicable, the student will be notified offline*
20. *What should the system do if a student cancels an appointment?*
    - ~~*It should try to book another appointment*~~
    - *Not applicable, the student will be notified offline*
21. *Would you require to view current bookings against students/officers/admins from within the system?*
    - *Yes*
    - ~~*No, this will be done directly on the google calendar*~~

N.3 QUESTIONNAIRE 3

Meeting Scheduler Questionnaire

Currently

1. *Is there always availability of Faculty Officers?*
   - *No*
2. *Is there always availability of rooms for the meetings?*
   - *No*
3. *What happens in the case of no availability as in questions 1 and 2?*

I would have to seek a Faculty Officer that we might not usually use. / Would resort to the Room Bookings system or ask the Faculty Officer if they could reserve a suitable room in their department.

In the new software application

1. *How often do you require to update (add/edit/delete) the faculties?*
   - *Very rarely happens*
2. *How often do you require to update (add/edit/delete) the faculty departments?*
   - *Every few years*
3. *How often do you require to update (add/edit/delete) the faculty officers?*
   - *Every year*
4. *Do the faculty officers move often between departments?*
   - *Not between departments, no.*
5. *How often does the allocation priority of the Faculty Officers changing?*
   - *Academic staff can be asked to act as Faculty Officers or step down from the Faculty Officer role and this happens quite regularly. Academic staff may act as Faculty Officers for many years or may choose to do it for a short period (say one academic year at the least) only. A lot depends on their other work commitments.*
6. *How likely is to update the list of suitable rooms where the meetings take place?*
   - *Possible*
7. *Would you require a single account credentials or one per system user?*
   - *Single – we work as a team and would be happy to share details.*
8. *Would you require any student data to be stored in the system?*
   - *No*

9. *Are the Faculty Officers aware that data against them will be stored in the system?*
    - *No – but I can ask them if they have any objections.*

10. *Would you prefer a manual selection of the appointment time or an automated allocation based on the availability of all relative parties?*
    - *Manual selection – there may be outside factors that have to be considered – for example: a Faculty Officer can do a full morning but must leave by 11.30am for another meeting.*

11. *Would you prefer to select the Faculty officer for a meeting or automatically selected (based on predefined priority order)?*
    - *Manually select the Faculty Officer*

12. *Would you prefer to select the room for a meeting or automatically selected (based on predefined priority order)?*
    - *Manually select the room – if the system shows the rooms that are available to me.*

13. *Would you need to send any message to the meeting participants?*
    - *Yes – but this is usually done automatically by Google Calendar when the meeting event is added to their Calendar.*

14. *What should the system do if a Faculty Officer rejects an appointment request?*
    - *Nothing, it will be dealt offline*

15. *What should the system do if a Faculty Officer cancels an appointment booking?*
    - *Nothing, it will be dealt with offline*

16. *Should the current logged in user of the system always be the Admin staff participating in meeting or should the system allow for other Admin Staff users to be selected*
    - *The system should allow to select the Admin staff to participate in the meeting*

17. *What should the system do if an Admin Staff rejects an appointment (only applicable if 'any' Admin Staff users can be selected for a booking)?*
    - *Nothing, it will be deal with offline*

18. *What should the system do if an Admin Staff cancels an appointment?*
    - *Nothing, it will be dealt with offline*

19. *What should the system do if a student rejects an appointment?*
    - *Not applicable, the student will be notified offline*

20. *What should the system do if a student cancels an appointment?*
    - *Not applicable, the student will be notified offline*

21. *Would you require to view current bookings against students/officers/admins from within the system?*
    - *Yes – that might be helpful (though I don't need the student details in this system).*