Managing desk space allocation and usage within Modern Office Environments

A dissertation submitted in partial fulfilment of

the requirements for the degree of

BACHELOR OF ENGINEERING in Computer Science

in

The Queen's University of Belfast

by

Saoirse McCann

9TH May 2017

**SCHOOL OF ELECTRONICS, ELECTRICAL ENGINEERING and COMPUTER SCIENCE**

**CSC3002 – COMPUTER SCIENCE PROJECT**

**Dissertation Cover Sheet**

A signed and completed cover sheet must accompany the submission of the Software Engineering dissertation submitted for assessment.

Work submitted without a cover sheet will **NOT** be marked.

Student Name: Saoirse McCann Student Number: 40104396

Project Title: Managing desk space allocation and usage within Modern Office Environments

Supervisor: Dr Barry McCollum

**Declaration of Academic Integrity**

Before signing the declaration below please check that the submission:

1. Has a full bibliography attached laid out according to the guidelines specified in the Student Project Handbook
2. Contains full acknowledgement of all secondary sources used (paper-based and electronic)
3. Does not exceed the specified page limit
4. Is clearly presented and proof-read
5. Is submitted on, or before, the specified or agreed due date. Late submissions will only be accepted in exceptional circumstances or where a deferment has been granted in advance.

**I declare that I have read both the University and the School of Electronics, Electrical Engineering and Computer Science guidelines on plagiarism - http://www.qub.ac.uk/schools/eeecs/Education/StudentStudyInformation/Plagiarism/ - and that the attached submission is my own original work. No part of it has been submitted for any other assignment and I have acknowledged in my notes and bibliography all written and electronic sources used.**

*Student’s signature* *Date of submission*

Acknowledgements

Firstly, to my supervisor, Dr Barry McCollum, thank you for all of the support and guidance you provided through this project and for always answering my endless questions.

To all the lecturer’s at Queens’ who have taught me over the past four years, and to all of my teachers at Dominican College Fortwilliam, thank you for passing on your knowledge and for your enthusiasm for teaching.

Finally, to my family and friends, thank you for your patience and for always encouraging me to keep going and never give up.

Abstract

This thesis examines the methods that could be implemented by a company to maximise the spatial utilisation of resources such as desks and meeting rooms in an open planned office space. In this paper a system is developed for managing the allocation of resources in an open planned office. As part of the system, an executive Officer Dashboard will display the spatial utilisation rates for the company allowing the workspace to be analysed and managed. As part of this evaluation, research is presented on the drive behind the paradigm shift from private individual offices to open-planned workspaces.

Table of contents

[Acknowledgements iii](#_Toc481679437)

[Abstract iii](#_Toc481679438)

[Table of contents iv](#_Toc481679439)

[Table of figures vi](#_Toc481679440)

[Chapter One 1](#_Toc481679441)

[1.0 Introduction and Problem Specification 1](#_Toc481679442)

[1.1 INTRODUCTION 1](#_Toc481679443)

[1.2 BACKGROUND TO THE PROBLEM 2](#_Toc481679444)

[1.2.1 Current trends 2](#_Toc481679445)

[1.2.2 The Problem 2](#_Toc481679446)

[1.3 RESEARCH FOR THE SOLUTION 2](#_Toc481679447)

[1.3.1 Agile working 2](#_Toc481679448)

[1.3.2 \*\*\* Civica’s approach 4](#_Toc481679449)

[1.3.3 PARETO’S 80-20 RULE 4](#_Toc481679450)

[1.4 ANALYSING SPACE UTILISATION 4](#_Toc481679451)

[1.4.1 Frequency rate 5](#_Toc481679452)

[1.4.2 Occupancy rate 6](#_Toc481679453)

[1.4.2 Utilisation rate 7](#_Toc481679454)

[1.4.3 Utilisation rate 8](#_Toc481679455)

[Chapter 2 9](#_Toc481679456)

[2.0 System Requirements Specification 9](#_Toc481679457)

[2.1 SOLUTION OVERVIEW 9](#_Toc481679458)

[2.1.1 Introduction 9](#_Toc481679459)

[2.1.2 End users 9](#_Toc481679460)

[2.1.3 Executive Officer Dashboard 9](#_Toc481679461)

[2.1.4 Resource allocation 10](#_Toc481679462)

[2.2 ASSUMPTIONS 10](#_Toc481679463)

[2.2.1 Functional Requirements 11](#_Toc481679464)

[2.2.2 Non-Functional requirements 12](#_Toc481679465)

[Chapter 3 14](#_Toc481679466)

[3.0 Design 14](#_Toc481679467)

[3.1 SYSTEM ARCHITECTURE 14](#_Toc481679468)

[3.1.1 System Architecture Overview 14](#_Toc481679469)

[3.2 PRESENTATION LAYER 15](#_Toc481679470)

[3.3 DOMAIN/BUSINESS LAYER 15](#_Toc481679471)

[3.4 DATA LAYER 15](#_Toc481679472)

[3.5 USER INTERFACE 16](#_Toc481679473)

[3.5.1 Design Overview 16](#_Toc481679474)

[3.5.2 User Interface Design Guidelines 16](#_Toc481679475)

[3.5.3 Screen Mock-Ups 16](#_Toc481679476)

[Chapter 4 17](#_Toc481679477)

[4.0 Implementation and Testing 17](#_Toc481679478)

[4.1 IMPLEMENTATION 17](#_Toc481679479)

[4.1.1 Server 17](#_Toc481679480)

[4.1.2 Languages and Project Practices 18](#_Toc481679481)

[4.1.3 Libraries 18](#_Toc481679482)

[4.1.4 Coding Conventions 19](#_Toc481679483)

[4.1.5 Source Control 20](#_Toc481679484)

[4.1.6 Development 21](#_Toc481679485)

[DATABASE 21](#_Toc481679486)

[MODELS 22](#_Toc481679487)

[SERVICE 23](#_Toc481679488)

[CONTROLLERS 23](#_Toc481679489)

[VIEWS 24](#_Toc481679490)

[4.2 TESTING 25](#_Toc481679491)

[Chapter 5 26](#_Toc481679492)

[System Evaluation and Experimental Results 26](#_Toc481679493)

[Chapter 6 26](#_Toc481679494)

[Conclusion 26](#_Toc481679495)

[Appendices 30](#_Toc481679496)

[Appendix A – Office Distribution Scenario 30](#_Toc481679497)

[Appendix XX – Use Cases 31](#_Toc481679498)

[Appendix XX – User Interface Design Guidelines 32](#_Toc481679499)

[Appendix XX – Screen Mock-Ups 33](#_Toc481679500)

[Appendix XX – Acceptance Tests 38](#_Toc481679501)

[Appendix XX – User Manual 39](#_Toc481679502)

Table of figures

Chapter One

1.0 Introduction and Problem Specification

* 1. INTRODUCTION

In this day and age, companies face major overheads for office space, resources, licenses, etc. Many companies now have an open-planned workspace for their employees in order to allow the company to expand and continue to grow.

The paradigm shift from private individual offices to open-planned workspaces was driven by financial advantages, the promise of increased productivity and the potential for better interaction between employees.

With an open planned layout companies could make better use of space, allowing for more employees in one building reducing the overheads of paying for extra space. In environments were teams have to interact with one another on a continual basis, then an open-planned layout could increase productivity and allows colleagues to approach one another without the need to arrange a meeting.

The issue many companies are now facing is how best to allocate the space and resources. It is this area that this thesis will focus on.

* 1. BACKGROUND TO THE PROBLEM

1.2.1 Current trends

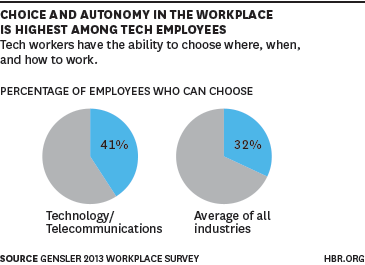
For the most part, office workspaces allocate awork area for each employee. It is easy for the company as they just have to do a simple calculation of one desk for every employee and no further planning or analysis is required. Employees tend to prefer to have their own personal desk space which they can personalise and become accustomed to. This approach, however, is inefficient for the company, as when resources are left vacant they cannot be reassigned when required.

1.2.2 The Problem

With people working from home, off sick, on annual leave or on customer sites, a number of resources can be left vacant. If each employee has a personal desk, then these resources cannot be utilised by others as they are still technical “occupied” by said employee even when they are vacant.

It’s becoming more normal for companies to move away from the normal resource allocation process of providing each employee with a desk. The more efficient agile approach is to allocate resources when requested, which ensures no resources is not utilised when it could be, and allows employees to request the type resource they require.

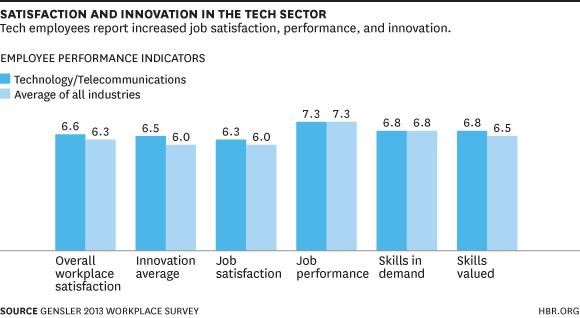
* 1. RESEARCH FOR THE SOLUTION

1.3.1 Agile working

With technological advances it is becoming more common for people to work from home. Different industries require employees to be on company sites or out of the office for meetings.

For this reason, it is becoming less common for all employees to be in the office every day. For this reason, let us consider each employee of the company as a “visitor”, who has the option to use office space if required. If space is required then the most appropriate resource for the “visit” can simply be booked, e.g. a hot desk.

An article in the Hazard Business Review by Diane Hoskins states that overall performance and attitudes to the working environment increases when employees have free reign of the space where they work. The study stated this approach worked particularly well in technology companies.



The reason why these technology companies noticed an increase in productivity and satisfaction of the workforce were due to the principles of agile working. Agile working gives employees a choice of where, when and how to work. A number of different space options are provided, such as meeting rooms, hot desks, team workspaces, etc. Agile working gives the workforce the freedom and flexible to pick the best working environment for them and their task.

An agile workspace has more focus on the actual space and what the employees do and require, rather than having an emphasis on departments and eliminates desk ownership.

* + 1. \*\*\* Civica’s approach

1.3.3 PARETO’S 80-20 RULE

The Pareto principle states that roughly 80% of outputs come from 20% of the inputs.

Applying Pareto’s rule to the issue of space allocation within an office would indicate that 80% of space is underutilised. In order to improve this rate and to understand how best to manage it, a thorough analysis of the space and how it is used is required.

* 1. ANALYSING SPACE UTILISATION

Space utilisation is a measure of well a space is being used, and how it is being utilised.

Frequency, occupancy and utilisation rates measure how efficiently a space is being use. In this thesis, we will consider resources such as desks, meeting rooms, team work areas, etc. as space.

Calculating the frequency, occupancy and the utilisation rates for a company can help in many ways. These rates are very useful when a company is seeking to expand, these indicators will allow them to see which resources are being underutilised, meaning they could be replaced with workspaces for new employees and resources which are in high demand. In turn, this will have fiscal benefits for the company allowing them to expand, while staying at the same location and not having to consider paying for more space. Overall, if a company has a thorough analysis of their space utilisation, it provides endless possibilities for the company to improve and manage the company’s workspace. For a scenario on how these rates can aid in analysing the layout of an office space refer to appendix A.

1.4.1 Frequency rate

The frequency rate represents the percentage of time space is used compared to its availability.

Example:

A company has a working day from 9am to 5pm. The table below shows when a resource was in use for one working week, Monday through to Friday

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | MONDAY | TUESDAY | WEDNESDAY | THURSDAY | FRIDAY |
| 9am - 10am | **VACANT** | **UTILISED** | **VACANT** | **UTILISED** | **VACANT** |
| 10am - 11am | **UTILISED** | **UTILISED** | **UTILISED** | **UTILISED** | **VACANT** |
| 11am – 12pm | **UTILISED** | **VACANT** | **UTILISED** | **VACANT** | **UTILISED** |
| 12pm - 1pm | **UTILISED** | **UTILISED** | **UTILISED** | **UTILISED** | **UTILISED** |
| 1pm – 2pm | **VACANT** | **UTILISED** | **UTILISED** | **UTILISED** | **VACANT** |
| 2pm – 3pm | **UTILISED** | **UTILISED** | **UTILISED** | **VACANT** | **UTILISED** |
| 3pm – 4pm | **UTILISED** | **UTILISED** | **UTILISED** | **UTILISED** | **UTILISED** |
| 4pm – 5pm | **UTILISED** | **UTILISED** | **UTILISED** | **UTILISED** | **UTILISED** |
| Hours in use | 6 | 7 | 7 | 6 | 5 |

HOURS IN USE

Then the sum of all of the hours the resource was utilised needs to be calculated.

(Monday) 6 + (Tuesday) 7 + (Wednesday) 7 + (Thursday) 6 + (Friday) 6 = (Total) **30 hours**

HOURS AVAILABLE

Total availability is the total amount of hours the resource is available to be used, 8 hours a day for 5 days.

Total availability: 5 \* 8 = **40 hours**

%

1.4.2 Occupancy rate

The occupancy rate represents how full the space is compared to its capacity.

Example:

A company has a working day from 9am to 5pm. The steps below show how to calculate the occupancy rate for a resource. The table shows how many people occupied the resource for one working week, Monday through to Friday

|  |  |
| --- | --- |
| CAPACITY | 20 |
|  | MONDAY | TUESDAY | WEDNESDAY | THURSDAY | FRIDAY |
| 9am - 10am | **0** | **0** | **0** | **20** | **0** |
| 10am - 11am | **10** | **20** | **20** | **15** | **0** |
| 11am – 12pm | **10** | **0** | **18** | **0** | **19** |
| 12pm - 1pm | **15** | **20** | **20** | **10** | **20** |
| 1pm – 2pm | **0** | **0** | **17** | **10** | **0** |
| 2pm – 3pm | **15** | **5** | **0** | **0** | **20** |
| 3pm – 4pm | **20** | **1** | **15** | **15** | **16** |
| 4pm – 5pm | **20** | **0** | **20** | **15** | **16** |
| HOURS IN USE | **26** |

HOURS IN USE

Add up all of the hours the resource was utilised (where the occupants is greater than 0).

Hours in use: **26**

NUMBER OF PEOPLE OCCUPAYING THE SPACE

Add up the number of occupants for when the resource is in use.

Total number of occupants: **412**

%

1.4.2 Utilisation rate

The occupancy rate represents how full the space is compared to its capacity.

Example:

A company has a working day from 9am to 5pm. The steps below show how to calculate the occupancy rate for a resource. The table shows how many people occupied the resource for one working week, Monday through to Friday

|  |  |
| --- | --- |
| CAPACITY | 20 |
|  | MONDAY | TUESDAY | WEDNESDAY | THURSDAY | FRIDAY |
| 9am - 10am | **0** | **0** | **0** | **20** | **0** |
| 10am - 11am | **10** | **20** | **20** | **15** | **0** |
| 11am – 12pm | **10** | **0** | **18** | **0** | **19** |
| 12pm - 1pm | **15** | **20** | **20** | **10** | **20** |
| 1pm – 2pm | **0** | **0** | **17** | **10** | **0** |
| 2pm – 3pm | **15** | **5** | **0** | **0** | **20** |
| 3pm – 4pm | **20** | **1** | **15** | **15** | **16** |
| 4pm – 5pm | **20** | **0** | **20** | **15** | **16** |
| HOURS IN USE | **26** |

HOURS IN USE

Add up all of the hours the resource was utilised (where the occupants is greater than 0).

Hours in use: **26**

NUMBER OF PEOPLE OCCUPAYING THE SPACE

Add up the number of occupants for when the resource is in use.

Total number of occupants: **412**

%

1.4.3 Utilisation rate

For a scenario on how these rates can aid in analysis the layout of an office space refer to appendix A.

The utilisation rate is a function of a frequency rate and an occupancy rate. The frequency rate measures the proportion of time that space is used compared to its availability, and the occupancy rate measures how full the space is compared to its capacity. Utilisation rates can be assessed in terms of both actual use and predicted use.

system constraints.

* A complete set of function definitions (as use cases if preferred), as far as possible written so as to be testable
* Measurable and testable non-functional requirements
* Description of interfaces required such as with other software or systems
* Any specific user interface requirement
* User characteristics

The target to aim for here is that your requirements could be the basis for a contract or handing to external developers to complete.

Chapter 2

2.0 System Requirements Specification

2.1 SOLUTION OVERVIEW

2.1.1 Introduction

The goal is to develop an end to end software solution which will aid in the planning and management of resources within a modern office environment.

Research has shown that technology companies appear to be most flexible and are more likely to adopt an agile working environment. For this reason the prototype solution will be developed with a technology company in mind. However, this is mainly to aid with examples and scenarios, the end solution should be versatile enough for all office environments.

Research shows that employees perform better when they can control and choose their working environment. This was the main motivation for the system, a focus on allowing employees to choose their space as opposed to simply automatically assigning the next available space. There will be no concept of desk ownership; if a resource is required it must be booked.

2.1.2 End users

The system will be designed with two end users in mind:

1. Administrators – all end users who are added to the system as administrators will have access to the Executive Officer Dashboard and the resource booking functionality. Administrators will also be able to add, update and remove users, teams, resources and time slots.
2. Non-Administrators – all end users who do not have administrative rights will only have access to the resource booking functionality.

2.1.3 Executive Officer Dashboard

The system must have an intuitive user experience design. Information relating to employees, teams, resources and time slots should be easy and straightforward to add to the system. Once all of the information has been entered the system will provide an Executive Officer Dashboard that will display any relevant Key Performance Indicators that the company requires to effectively manage their resource allocation. The dashboard will display the frequency, occupancy and utilisation rates in the form of charts and graphs which will aid the end user in making decisions on how best to manage and allocate their resources to make the workspace more cost efficient. The Key Performance Indicators will be displayed per week, for the overall company and for each resource

2.1.4 Resource allocation

When a user is selecting a resource the system will provide functionality for them to auto assign their desk space, selecting the desk that is located closest to their line manager.

The system shall also have an end point to allow users to log on and book a resource from their mobile device. This will involve touch screen development.

When booking a resource, consideration will have to be taken into the schedule of each resource to ensure it is not double booked or booked beyond its capacity.

2.2 ASSUMPTIONS

Due to the time constraints that have been set for developing the solution the following assumptions have been made:

* A prototype solution will be developed for one office consisting of a single one level building, for an IT company.
* When an employee is off work on annual leave or working from home they do not need to book any resources for said days
* It is the responsibility of each user to book their resources for the upcoming week
* Archiving of the “Timetable” table in the database will not be implemented for the prototype developed. Although possible approaches on how this would be carried out will be investigated.
* Users only have to book resources for Monday – Friday between 8am and 6pm.
* Users will only be able to book resources for the current week and the upcoming week.
* Resources can only be booked against existing users of the system

2.2.1 Functional Requirements

1. Administrators shall have access to functionality to add new users, teams, resources and time slots to the system
2. Administrators shall have access to functionality to add members to teams
3. Information regarding existing users, teams, resources and time slots should be available to the administrator to update
4. Accounts shall be automatically created for each employee when they are added to the system
5. Administrators shall have access to an executive officer dashboard which will provide information of measurements of utilisation
6. Analytics for the key performance indicators on the executive officer dashboard shall be displayed in the form of charts and graphs:
7. Utilisation rate: frequency rate \* occupancy rate
8. Frequency rate: percentage of time space is used compared to its availability
9. Occupancy rate: how full the space is compared to its capacity
10. A user shall be able to send a change request for an occupied resource
11. Each user should have access to an allocation timetable displaying all resources they have booked for the current week
12. A user shall have access to functionality to delete a resource booking which is displayed in their allocation timetable
13. All users shall have access functionality to book resources for themselves
14. Users should be able to book a resource for a single time slot
15. Users should be able to block book a resource spanning across time slots and days
16. Users should be able to add a group booking for multiple attendees
17. The system shall prevent users from booking multiple resources for the same time period
18. The system shall allow users to add attendees to an existing group booking
19. Group bookings must be accepted by each users before it is added to their allocation timetable
20. The system shall keep a timetable of allocation for each resource
21. The system shall prevent a resource being double booked
22. The system shall prevent a resource from being over booked

2.2.2 Non-Functional requirements

1. The system should be mobile compatible to allow a user to easily:
   1. delete a resource booking from their schedule
   2. book resources
   3. view resource allocation timetable
2. Modified data on the system should be updated in the database within 2 seconds for all other users accessing it.
3. The system should allow the user to easily switch between booking a desk, room or other resource.
4. The UI should be intuitive and user friendly, requiring minimal training for the end user.

**3. Design**

This section should describe the design of your proposed system. Normally this several parts, depending on your project:

1. Architectural Description of the system – textual and/or diagrammatic. This could be a simple diagram showing the components and how they relate or it could describe the choice of architectural style or pattern used.
2. User Interface Design (if applicable). Show sketches of the design or screenshots with explanations of choices made, if necessary.
3. Software System Design.

The role of each component and the interfaces between components should be described. There should be a clear correlation between your design and your specification.

The design should be linked to requirements and, where applicable give a critical discussion of key design decisions/styles/patterns used. There might be a data model, a UI design, details of external interfaces, and of other important issues e.g. concurrency, event handling, error and exception handling, security, data persistence. No particular notation or tool is mandated.

Chapter 3

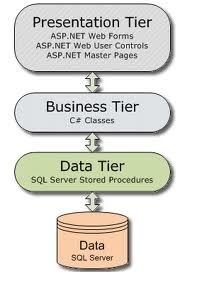
3.0 Design

In this chapter I am going to talk about the overall architecture of the end solution. An insight into how each layer was implemented will be discussed along with reasoning as to why each decision was made.

3.1 SYSTEM ARCHITECTURE

3.1.1 System Architecture Overview

A three tiered architecture approach was selected as it allows for separation of concerns. After doing research, there was a lot of help and support for implementing an n-tiered architecture, especially with ASP.NET MVC.

The end solution will consist of a presentation layer, a business/domain layer and a data layer.

The separation into layers results in a more practical end solution which is easier to update and extend, making it overall more flexible, manageable and extendable.

Having all of the business logic in a separate layer means that’s the service methods can be updated as business rules change. This means that only the project containing the domain layer needs to be updated, and no changes will need to be carried out to the data layer or the presentation layer.

Within the solution, each layer was created as a separate project to further help separate all of the areas of concern.

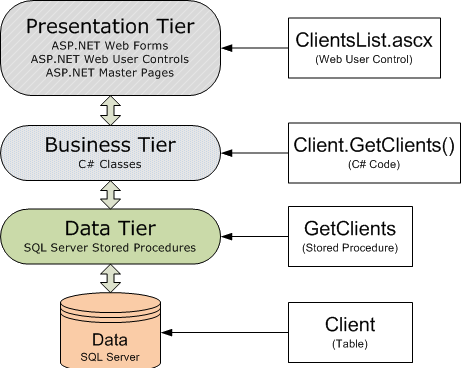
\*\*\*add photo of folder layout -> need to add testing projects first

3.2 PRESENTATION LAYER

The presentation layer will consists of the unser interface which the end user will interact with. It will be made up of an ASP.NET MVC application. Using MVC will provide a solid platform for separation of concerns, it is also easy to integrate with other frameworks and librarie such as JQuery and Highcharts. MVC provides a solid structure which makes unit testing more manageable, providing an extra layer of protection.

3.3 DOMAIN/BUSINESS LAYER

The domain layer will consist of C# classes which will contain service methods containing the business rules of the system. The business rules will be separated out into components making the solution more manageable. This means, the will be a User Service containing all of the service methods required to get, manipulate, delete and add users to the system. The same pattern will follow for teams, resources, time slots, bookings, accounts and rate calculations. The domain layer communicates with the data layer to retrieve the data to then filter and manipulate it into the required results and then send them to the presentation layer to be displayed for the end user.

3.4 DATA LAYER

The data layer follows an entity framework code-first migrations approach. This approach was selected due to its flexibility. It allows the database structure to be changed and updated as new requirements are added. Also when more thought and understanding is given into a requirement providing a clearer view of what is required, resulting in a change of the current database structure. The data layer is used for interacting with the database to access, manipulate and add data.

3.5 USER INTERFACE

There are two main endpoints in mind for this solution. Ideally, there would be two end solutions, a phone app and a web application. The phone app would simple allow users to book a resource and view their allocation. And the web app would allow full use of the system, including access to the executive officer dashboard.

Although, dud to time constraints it was decide to design one end solution that would cater for both end points. For this reason, the system will be responsive and compatible to use on both mobile phones, computer, tablets, etc.

3.5.1 Design Overview

The main goal is for the end system to have a clean and professional layout. Consistency and ease of use are vital for the solution to be user friendly and intuitive. The application UI consists of Twitter Bootstrap 3. Twitter Bootstrap provides a structure which allows the system to be responsive, and therefore compatible with different hardware devices.

3.5.2 User Interface Design Guidelines

Before any implementation started, a brief list of design guidelines was created. This list contains an overview of how components should be implemented and displayed on screen. Having design guidelines in place aided in keeping the user interface consistent across the whole solution, which makes the system easier to use and in turn increases the user experience. Refer to appendix XX to view the list of design guidelines.

3.5.3 Screen Mock-Ups

Screen mock-ups were designed before any implementation started. This helped to visualize what the final solution should look like and also provided a clear goal to work towards. Although producing the mock-ups was initially very time consuming and tedious, overall, in the long run, it saved time when it came to implementation as the design decisions had already been made. Although, at times, the implementation did stray away from the design mock-ups, but the overall concept and outcome was the same. Refer to appendix XX to for the initial wireframe screen mock-ups.

Chapter 4

4.0 Implementation and Testing

This chapter provides a detailed description of the implementation approach and the testing strategies adopted in producing the final solution. All of the technology choices, libraries and frameworks used were chosen carefully and a rationale for each choice is provided. This chapter also includes an insight into how testing was carried out to ensure the system meets the requirements and that the functionality is performing as expected. The IDE for all development for the end solution was implemented on Visual Studio 2012.

4.1 IMPLEMENTATION

4.1.1 Server

The end solution is comprised of one operating system and two servers, a database server and a web server.

When researching for a database server the main requirement was for platform that could support complex SQL queries and more importantly store data in a relational way. Only open source options were considered. In the end Microsoft SQL Server was chosen due to the amount of support, information and guidance available. The SQL Server Management Studio meant that all data in the database could easily be viewed and analysed during development, helping to enhance performance and provided a vital insight into the data while debugging. MySQL, CSQL and PostgreSQL were also considered as potential database servers.

Internet Information Services (ISS) was chosen for the web server, which is hosted on Microsoft server (the operating server). Apache and nGnix were also considered as potential web servers but were not suitable for dot net applications without using a port for the dot net framework such as Mono for Linux. For this reason, ISS was selected as the best approach due to its ease to integrate with the solution framework and it’s compatible with most browsers and operating systems, which is one of the non-functional requirements for the solution.

4.1.2 Languages and Project Practices

All backend development was completed in C#. Due to using a code-first migrations approach to create the database, no SQL code was required for the database creation. However, SQL was still used throughout the project to query the database. LINQ statements were used to write strongly type query expressions to retrieve and add data to the database. HTML, CSS and JavaScript were used in the views for the UI.

Visual Studio 2012 was used the Integrated Development Environment (IDE).

\*\*table about frameworks here? MVC and EF… Cant heading?

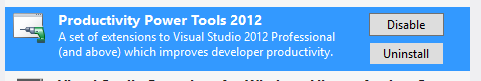
4.1.3 Libraries

A number of libraries were included in the end solution to aid in the development of the system and to enhance then end product. All of the libraries used were open-source.

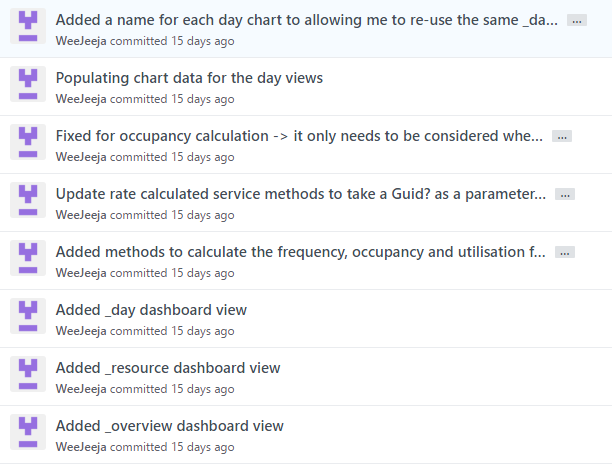
\*\*add some form of image here?

|  |  |  |
| --- | --- | --- |
| **Library** | **Version** | **How it was used** |
| Highcharts | Highcharts JS v4.0.1 | Highcharts is a JavaScript library which provides an easy framework to display interactive charts on a web page. The Highcharts library was used to display all of the charts for the dashboard views. |
| Bootstrap | Twitter Bootstrap v3.3.7 | Bootstrap is a framework which contains HTML and CSS design templates. Overall, the templates and interactive components which were readily available saved time and provided optimum results to the look and navigation of the end system. |
| JQuery | JQuery JavaScript v1.12.4 | JQuery is a javaScript library which simplifies writing javaScript code. The use of JQuery throughout the project saved time, enabling Ajax interaction and event handling to be implemented with as much ease as possible. |

4.1.4 Coding Conventions

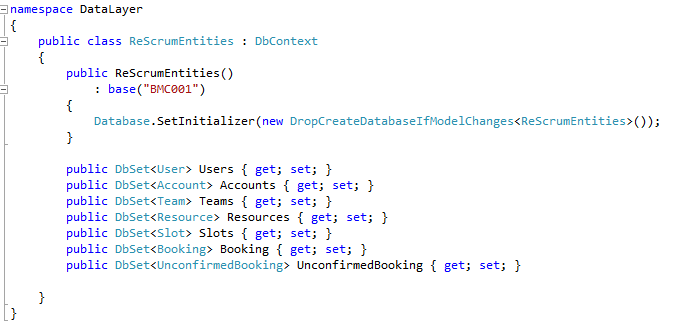
A series of coding conventions were followed throughout the project to ensure consistency and maintain coding standards. The *Productivity Power Tool 2012* was added as an extension tool to the project. This tool has a number of shortcuts to improve developer productivity and helps maintain readable and maintainable code.

|  |  |
| --- | --- |
| **Coding convention** | **Example** |
| All class names should have upper-camel case naming convention | *ResourceController.cs* |
| All variables a lower-camel case naming convention | var *name* = “Testing”; |
| All parameters should have a lower-camel case naming convention | Index(Guid? *id*) |
| Where required, longer names should be used to favour readability | *CalculateFrequencyBetweenTwoDates* instead of CalculateFrequency |
| All migrations should follow an upper-camel case naming convention | *AddedUserTableToDatabase* |
| All interfaces should begin with an “I” and follow an upper-camel case naming convention | *IResourceController.cs* |
| Partial views should begin with a “\_” symbol and follow a lower-camel case naming convention | \_*resourceInformation.cshtml* |
| Where appropriate, region breaks should be used | *#region HelperMethods*  *#endregion* |

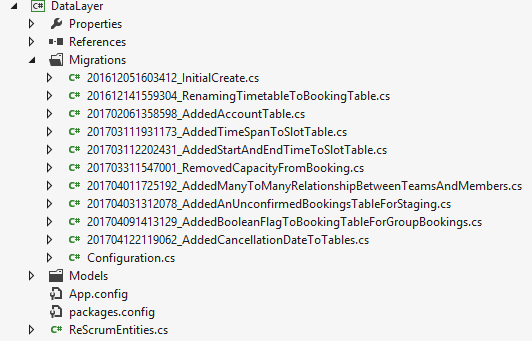
4.1.5 Source Control

GitHub was used for source control. GitHub was chosen as it integrated nicely with GitLab which was a requirement for all Queen’s students. The project was set up to facilitate both repositories meaning there were consistently two versions of the system that could be accessed at any time and were simultaneously updated anytime new code was pushed. GitHub was used on top of GitLab as GitHub allowed to be raise issues and provided a nice user-friendly interface showing the different branches were which active.

Although there is only one developer for this system, thought was still given to the practice of continuous integration. Commits were made regularly, and were as small as possible. Clear descriptions where added to each commit to allow reverting, if necessary, easier. Integrating new functionality and updates into the master code base frequently helped keep the bug count to a minimum and made regression testing easier.

4.1.6 Development

DATABASE

The first stage of development involved setting up the database. Entity Framework was used and a code-first migrations approach was implemented. A data model for each required table was created. The DBContext was then added and contains a list of DBSet classes which each present an entity. A migration was added anytime a data model class was updated or added, or when the DBContext class was changed. Whenever the database is created all the migrations are executed and a table for each entity is added to the database.

All development was carried out in an iterative process. For instance, the first stage of the development plan was to capture all of the data replacing to the company, e.g. their employees, resources, opening hours and teams. Therefore the users, resources, teams and time slots tables required to be implemented at the beginning. Using the code first migrations approach facilitated an iterative development cycle as it allowed the database to be extended when required. Later, when the resource booking functionality was to be implemented, a migration was added to create the new Booking table.

MODELS

As mentioned above, entity models are used to represent each table in the database. Each entity model defines the type for each column and how it should be stored for the table it represents. Data models also defines how data is connected together, it states the foreign keys and the relationships.

For example, the relationship between users and teams is many-to-many as each user can be a member of many teams and each team has many members. This relationship is set up in the entity models as shown in figures XX and XX. Even though the DBContext does not contain a DBSet for TeamUsers, Entity Framework knows to create this linking table in order to manage the many-to-many relationship here.

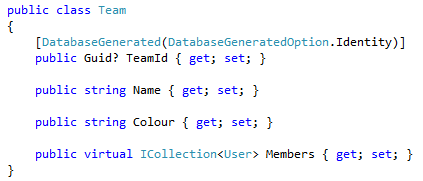


Database overview

The team entity table

The user entity table

Figure X; many-to-many relationship

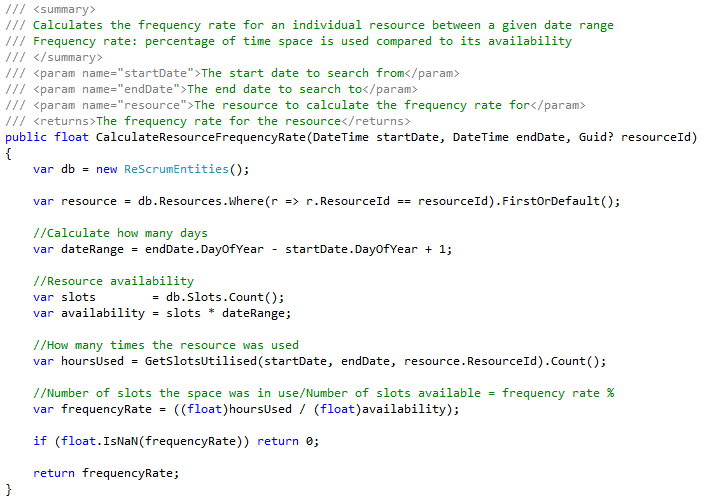
 

Wrapper models are used in the domain layer. Models contain domain specific properties and contain no knowledge of validation rules or storage procedures.

View models are used in the presentation layer. Data annotations are used for validation, and attributes such a display names were also added. View models are used to sate how the data should be presentation on screens and can add validation restrictions on data capture.

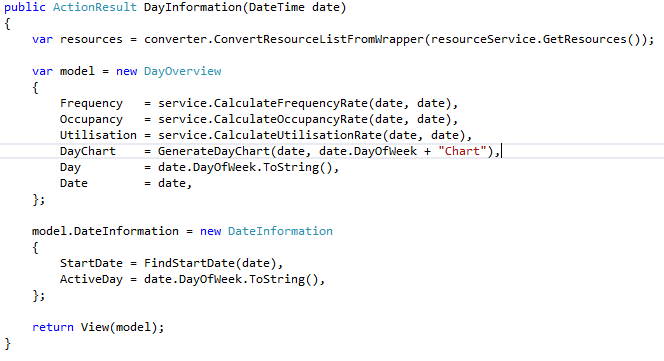
The distinction of models in each layer allows for better separation of concerns. For example, if only one model was used for every layer the display concerns, e.g. *DisplayName* attribute, would be missed with persistence concerns, e.g. *ForeignKey* attribute.

SERVICE

A service class was created for each component, e.g. ResourceService, RateCalculationService, in the domain layer. This approach is a more maintainable framework as it provides clear modules for logical functionality grouping. Each service class contains methods which retrieve, modify and delete data from the database using LINQ statements. All of the business rules and calculations are performed in the service methods.

CONTROLLERS

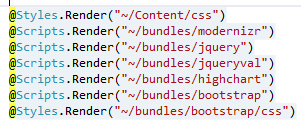
Once again, a controller for each component was created. An instance to any of the required services was added to the controller to give access to the service methods. Hence, there was no concern or implementation of business rules and logic in the controllers. GET methods were used to retrieve data to display for the user. POST methods handled validation errors and more often called service methods which modified, added or deleted data from the database.

Where possible, methods were coded with a generic framework to allow future development to take place. For example, the dashboard displays a weekly overview of the current week, last week and next week, and then has tabs for Monday through to Thursday of each week displaying an overview for that day. The method to get the weekly overview is generic, the method can be used to get the overview of any week, so long as a data is given. Using the date parameter, the method then calculates the starting date for the week (figure XX).

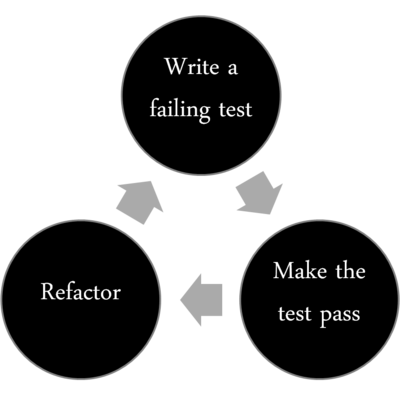
A similar approach is followed for the day, a date is passed in as a parameter and the method calculates which day it is (figure XX).



VIEWS

The razor view engine handles the display of the views on screen. Most views were strongly typed to allow access to the properties of the view model. Where possible, partial views were implemented. The use of partial views allowed the same code to be used in multiple views which in the end significantly reduced the amount of duplicated code.

Scripts were rendered on the views to allow JavaScript, bootstrap, Highcharts and JQuery libraries to be used. In order to eliminate the need to render all of the scripts for the libraries on every view, a number of bundles were added to the BundleConfig class. The bundles were then rendered on the layout page which was used as the layout for all of the other views. Overall, this approach saved time and was more efficient as less code had to be duplicated per view.

4.2 TESTING

4.2.1 Testing Strategy

It was apparent at the beginning of the project that following a Test Driven Development (TDD) approach was not feasible. TDD required the developer to write unit tests before implementing the code. After the test and code have been implemented, it is then time to refactor. TDD forces the developer to think ahead about what they are implementing and what variables, methods, services, etc. will be required to do so. This approach of making the developer think ahead produces cleaner code and overall results in fewer bugs due to high test coverage. Although, the benefits of TDD are clear, due to a lack of experience in following the TTD approach it was extremely time consuming. Writing unit tests was in itself a new technique to master and attempting the TTD approach simply overcomplicated matters.

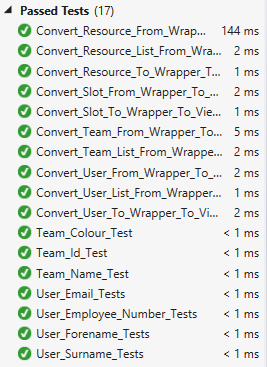
To ensure the project progressed in the time frame required a combination to manual and automated testing was used.

4.2.3 Manual Testing

Manual testing involves physically using the system to check for defects. The tester plays the role of the end users, and is normally someone other than the developer. Manual testing should not be undervalued as automated tests cannot replace user intuition and they cannot speak for user experience

A test case template was created as a framework for the acceptance tests to follow. The purpose of the acceptance tests was to test the solution against the system requirements and assess whether the solution fulfils the requirement or not. Each test is given a requirement and a scenario to assess. The steps involved in creating the scenario, the expected and the actual outcomes are recorded on the test template.

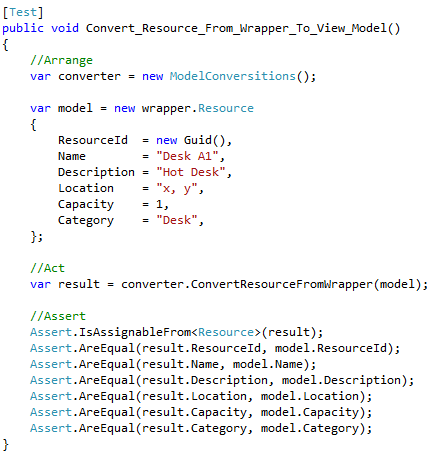
Refer to appendix XX to review the test case template that was used for all manual tests and for a sample of some of the tests that were performed.

****4.2.2 Automated Testing

Overall, automated testing is less time consuming and costly than manual testing. The main reason for this is that once created automated tests can be executed as many times and as frequently as necessary, at no additional cost.

Automated test made the process of regression testing much easier. When new functionality was added to the system it was imperative to ensure everything that was previously functioning as expected still was. Unit tests were used for automated testing.

*NUnit v.3.6.1* was installed to write the unit tests, and the *NUnit 3 Test Adapter* tool was also installed to enable the execution of the NUnit tests on Visual Studio (IDE). Although the unit tests were time consuming to implement, they provided a solid basis for regression testing. When code was added or updated, all of the unit tests could automatically be executed and provided immediate feedback on what functionality, if any, was affected by the code change.

****

Unit tests were implemented to test individual classes and methods in isolation. The *“Arrange–Act-Assert”* Pattern was followed for all unit tests. An example of the pattern can be seen in figure XX.

Chapter 5

System Evaluation and Experimental Results

**Evaluate the requirements against wat was implemented**

Chapter 6

Conclusion

References

**The first citation should be the URL to the software code repository which should contain the code and any other resource required to run the software.**

Famuyide, Stephanie, and Stephanie Famuyide. "15 Tips For Writing Better Requirements". *Business Analyst Learnings*. N.p., 2017. Web.

"What Is Executive Dashboard? - Definition From Whatis.Com". *SearchCIO*. N.p., 2017. Web.

"Writing Quality Requirements". *Processimpact.com*. N.p., 2017.

Thinking, Fresh. "Utilisation Analysis | Fresh Business Thinking". *Fresh Business Thinking*. N.p., 2017. Web.

<http://www.smg.ac.uk/documents/utilisation.pdf>

<http://reqtest.com/requirements-blog/functional-vs-non-functional-requirements/>

<http://www.requirements.com/Glossary/NonFunctionalRequirements/tabid/91/Default.aspx>

<http://softwareengineering.stackexchange.com/questions/162399/how-essential-is-it-to-make-a-service-layer>

<https://msdn.microsoft.com/en-us/library/ff648105.aspx>

<https://www.lucidchart.com/pages/er-diagrams>

<http://www.conceptdraw.com/examples/er-notation>

<http://survey.datassentialsurveys.com/help/Inquisite_ER_Diagram.htm>

<http://kc.litiumstudio.se/documentation/previous-versions/litium-studio-4-6/litium-studio-4-6-2/litium_studio_ecommerce/architecture-design/er-diagrams>

<http://www.tableau.com/sites/default/files/media/which_chart_v6_final_0.pdf>

<http://www.investopedia.com/terms/c/capacityutilizationrate.asp>

<http://www.businessballs.com/pareto-principle-80-20-rule.htm>

<https://en.wikipedia.org/wiki/Pareto_principle>

<http://ui-patterns.com/blog/The-8020-rule--the-Pareto-principle>

<http://www.archibus.com/ai/abizfiles/v23_help/archibus_help/user_en/Subsystems/webc/Content/web_user/space/planning/concepts/allocation.htm>

<https://hbr.org/2014/01/employees-perform-better-when-they-can-control-their-space>

<http://spendmatters.com/2011/01/05/six-strategies-for-it-cost-allocation/>

<http://www.flexibility.co.uk/flexwork/offices/facilities4.htm>

<http://staging.cl/wp-content/uploads/2014/07/Office-Space-How-much-is-enough.pdf>

<https://www.theregister.co.uk/2016/03/07/ibm_offshoring/>

<https://www.bloomberg.com/news/articles/2007-10-25/shaping-the-workforce-of-the-futurebusinessweek-business-news-stock-market-and-financial-advice>

<http://officeprinciples.com/agile-working/>

<https://forums.asp.net/t/1940843.aspx?Apply+3+layer+to+MVC+Application>

<http://stackoverflow.com/questions/24588838/entities-vs-domain-models-vs-view-models>

Delete functionality -> it was implement so that it would not affect the calculations Resources, Users and Slots

Add interview with workers -> reference in conclusion that perphaps and electronic tracker would be more useful. Say that employees would like to chose where they get to work

Strengths -> due to the architure fremwork, the protype can easily be updated for all kidns of companies, consisiting of diffrrent offciens and floors. Alsodoes not only have to work for an open planned floor plan.

Strenght -> due to using EF code first I was able to update the db all the time -> show migrations floder image

Strenght -> two end points, no need to design or implement second solution

Weakness -> mobile app would allow access to only whats needed

Improvement – Auto functionality should consider team memebers and sit you next to your line manager

\*\*Add chapter descitptions

\*\*tracker devices are invasion of personal privacy

\*\*Gantt chart

\*\*user manual

\*\*wireframes

\*\*methodology

\*\*folder layout in architecture

\*\* HEURISTIC HCI GUIDELINE

\*\*explain how algorithms were coded

\*\*Authentication

Appendices

Appendix A – Office Distribution Scenario

Scenario: The office is divided into two sections, one section contains desks, the second section contains two meeting rooms. The two meeting rooms each have a capacity of 100 people, while all the desk space for the office is cramped together. By looking at the frequency at which each of the rooms is used, how many occupants they have when they are utilised a company would be able to make decisions based on these calculations which will allow them to make changes to better suite their company needs.

Example 1: if the first room has an frequency rate of 90%, and an occupancy rate of 50%. And the second room has a frequency rate of 100%, and an occupany rate of 50%. This means the company could get rid of one of the rooms and spilt the toher room into two. Leaving space for more desks for the company to expand.

Example 2: Both of the rooms have a high frequency and occupancy rate. If the occupants in the room are company employess this means that their current desks are left vacant. Pareto’s rule is relevant here as it gives an insight into the fact the you do not need to provide resources for all employees. The company could still expand, keep the same location with the same rent. This is when a booking systems for resources would come in useful. And were a dashboard containing information on the frequency, occupany and the utilisation of each resource, on a certain day at a particular time may become very cost eefective.

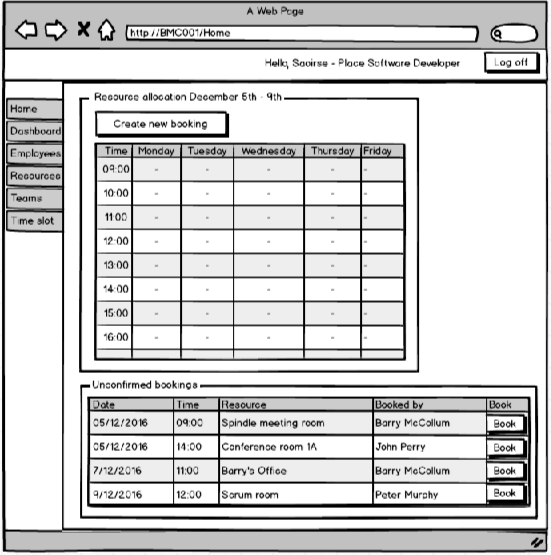
Appendix XX – Use Cases

Appendix XX – User Interface Design Guidelines

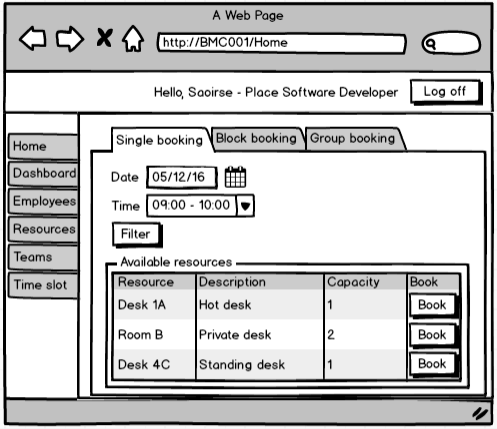
|  |  |  |
| --- | --- | --- |
| **Component** | **Design guideline** | **Example** |
| Main headlines |  |  |
| Normal text |  |  |
| Table headings |  |  |
| Tables |  |  |
| Button (to add a new entry) |  |  |
| Button (to delete an entry) |  |  |
| Button (to go to new info page) | .btn-info |  |
| Button (to go back) |  |  |

Appendix XX – Screen Mock-Ups

Home screen

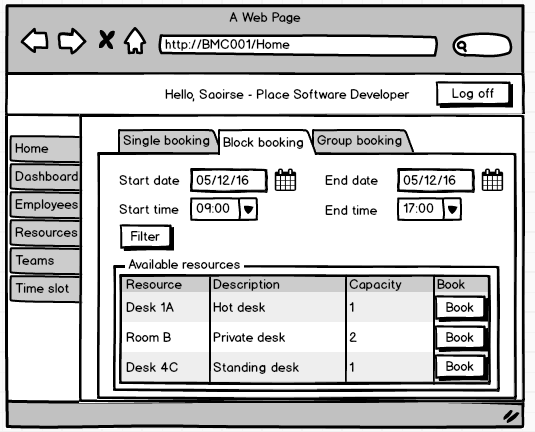


Only administrators will have access to this area

Single booking screen

This page is used to add a booking for a single time slot.

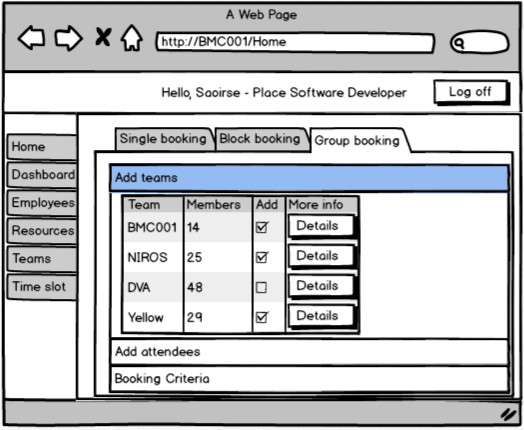
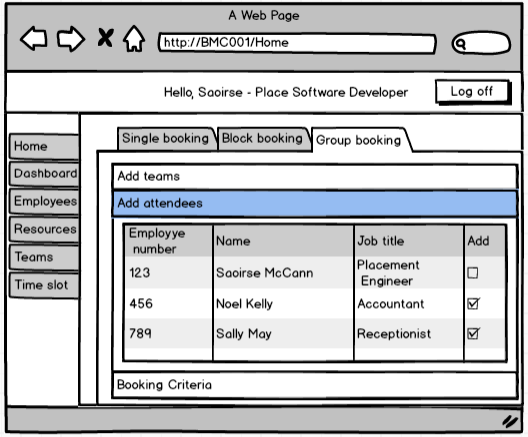
For example, a desk can be book for the 9am time slot on Monday 5th of December.

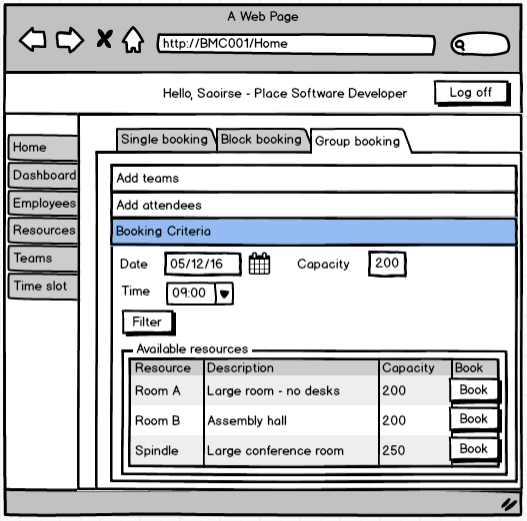


Block booking Screen

This page is used to add a booking for multiple time slots.

For example you can book at desk from 9am to 5pm for Monday 5th of December, or you can book a desk from 9am to 5pm from Monday 5th of December to Friday 9th of December.

Group booking screen



On the group booking page, the user has the option to add teams and individual attendees to the booking. Then the booking criteria is entered and only the resources that are available (i.e. not already booked) for the date and time entered, and with a capacity greater than the amount entered are returned. A resource in the “Available resources” section can then be selected and booked.

Dashboard Overview Screen

->Delete booking

->Group edit

->Add team/resource

->Add team member

Appendix XX – Acceptance Tests

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Requirement** | | 1. Administrators shall have access to functionality to add new users, teams, resources and time slots to the system | | | |
| **Scenario** | | Administrator adds new user to the system | | | |
| **Step #** | **Action** | | **Expected Outcome** | **Actual Outcome** | **Result** |
| 1 | Click on “Employees” tab | | The employee index page should be displayed with a table of all the current employees. “Add new employee” button should be displayed above the table. | The employee index page is displayed with a table of all the current employees. “Add new employee” button is displayed above the table. | **PASS** |
| 2 | Click on “”Add new employee” button | | The create employee page should be displayed with all of the fields to add an employee | The create employee page is displayed with all of the fields to add an employee | **PASS** |
| 3 | Enter employee details and click “Create” | | Should be redirected back to the index page and the new user should be displayed in the table of employees | Redirected back to the index page and the new user is displayed in the table of employees | **PASS** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Requirement** | | 1. Administrators shall have access to functionality to add new users, teams, resources and time slots to the system | | | |
| **Scenario** | | Administrator adds new resource to the system | | | |
| **Step #** | **Action** | | **Expected Outcome** | **Actual Outcome** | **Result** |
| 1 | Click on “Resources” tab | | The resource index page should be displayed with a table of all the current resources. “Add new resource” button should be displayed above the table. | The resource index page is displayed with a table of all the current resources. “Add new resource” button is displayed above the table. | **PASS** |
| 2 | Click on “”Add new resource” button | | The create resource page should be displayed with all of the fields to add a resource | The create resource page is displayed with all of the fields to add a resource | **PASS** |
| 3 | Enter resource details and click “Create” | | Should be redirected back to the index page and the new user should be displayed in the table of resources | Redirected back to the index page and the new user is displayed in the table of resources | **PASS** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Requirement** | | 1. Administrators shall have access to functionality to add members to teams | | | |
| **Scenario** | | Administrator adds an existing user to an existing team  Pre: there are users already in the system that are not members of the test team | | | |
| **Step #** | **Action** | | **Expected Outcome** | **Actual Outcome** | **Result** |
| 1 | Click on “Teams” tab | | The team index page should be displayed with a table of all the current teams. “Add members” button should be displayed beside each team in the table. | The team index page is displayed with a table of all the current teams. “Add members” button is displayed beside each team in the table. | **PASS** |
| 2 | Click on “Add members” button | | The add members page should be displayed with a list of all users who are not currently in the team, and an “Add member” button should be displayed beside each user | The add members page should is with a list of all users who are not currently in the team, and an “Add member” button is displayed beside each user | **PASS** |
| 3 | Click on the first “Add member” button | | Should be redirected back to the index page and a success message should be displayed at the top of the page stating that the user was added as a member for the team | Redirected back to the index page and a success message is displayed at the top of the page stating that the user was added as a member for the team | **PASS** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Requirement** | | 1. Information regarding existing users, teams, resources and time slots should be available to the administrator to update | | | |
| **Scenario** | | Administrator updates team name  Pre: there is a team already is the system | | | |
| **Step #** | **Action** | | **Expected Outcome** | **Actual Outcome** | **Result** |
| 1 | Click on “Teams” tab | | The team index page should be displayed with a table of all the current teams. “Edit” button should be displayed beside each team in the table. | The team index page is displayed with a table of all the current teams. “Edit” button is displayed beside each team in the table. | **PASS** |
| 2 | Click on “Edit” button | | The edit page should displayed with all of the team’s details in an editable format. | The edit page is displayed with all of the team’s details in an editable format. | **PASS** |
| 3 | Edit the team name and click on the “Save” button | | Should be redirected back to the index page and team name should be updated in the index table. | Redirected back to the index page and team name is updated in the index table. | **PASS** |

Appendix XX – User Manual

**Appendices**

These should include as appropriate:

(a) A User manual giving details on how to use the software, including details of input data, output formats and error messages.