Computing Practical Project

Rota, Scheduling & Booking System

Thien Nguyen

Candidate No. 7461

Centre No. 12290

Woodhouse College

Table of Contents

[Analysis 5](#_Toc400908138)

[1.1 Introduction 5](#_Toc400908139)

[1.1.1 Background 5](#_Toc400908140)

[1.1.2 Problem Definition 5](#_Toc400908141)

[1.1.3 The Users 6](#_Toc400908142)

[1.2.1 The Current System Analysis 6](#_Toc400908143)

[1.2.1.1 Interview 6](#_Toc400908144)

[1.2.1.2 Questionnaire 8](#_Toc400908145)

[1.2.1.3 Observation of Existing System 9](#_Toc400908146)

[1.2.2 DFD of current system (to Level 1) 9](#_Toc400908147)

[1.2.3 Data Sources and Destinations 9](#_Toc400908148)

[1.2.4 Entity-relationship diagram 10](#_Toc400908149)

[1.2.5 Discussion of problems with Current System 10](#_Toc400908150)

[1.2.6 The Proposed New System Analysis 11](#_Toc400908151)

[1.2.7 DFD of Proposed New System 11](#_Toc400908152)

[1.2.8 Data Sources and Destinations 12](#_Toc400908153)

[1.2.9 Entity-relationship diagram and entity descriptions of current system 12](#_Toc400908154)

[1.2.10 Analysis Data Dictionary 13](#_Toc400908155)

[1.2.11 Data Volumes 17](#_Toc400908156)

[1.3 Constraints 17](#_Toc400908157)

[1.3.1 Hardware constraints 17](#_Toc400908158)

[1.3.2 Software constraints 18](#_Toc400908159)

[1.3.3 Who will be allowed to use the various parts of the system 18](#_Toc400908160)

[1.4 Limitations 18](#_Toc400908161)

[1.4.1 Areas which will not be included in computerization 18](#_Toc400908162)

[1.4.2 Areas considered for future computerization 18](#_Toc400908163)

[1.5 Objectives 19](#_Toc400908164)

[1.5.1 General Objectives 19](#_Toc400908165)

[1.5.2 Specific Objectives 19](#_Toc400908166)

[1.6 Consideration of alternative solutions 20](#_Toc400908167)

[1.7 Justification of chosen solution 21](#_Toc400908168)

[Design 23](#_Toc400908169)

[2.1 Overall System Design 23](#_Toc400908170)

[2.1.1 System Flowchart 23](#_Toc400908171)

[2.1.2 Hierarchy Chart 24](#_Toc400908172)

[2.2 Description of Modular Structure of System 26](#_Toc400908173)

[2.3 Definition of Data Requirements 27](#_Toc400908174)

[2.4 Database Design (Normalised ER) 27](#_Toc400908175)

[2.5 File Organization and Processing 30](#_Toc400908176)

[2.6 Identification of Storage Media 30](#_Toc400908177)

[2.7 Identification of Suitable Algorithms (Pseudo code) 30](#_Toc400908178)

[2.7.1 Bubble sort 30](#_Toc400908179)

[2.7.2 Establishing a connection to the database. 31](#_Toc400908180)

[2.7.3 Salt and Hashed Encryption 31](#_Toc400908181)

[2.7.4 Breadcrumbs 31](#_Toc400908182)

[2.7.5 Email Automation 32](#_Toc400908183)

[2.7.6 Regex Expressions 32](#_Toc400908184)

[2.7.7 Captcha 32](#_Toc400908185)

[2.8 Sampled SQL Queries 33](#_Toc400908186)

[2.9 Class Definitions 33](#_Toc400908187)

[2.10 User Interface Rationale 34](#_Toc400908188)

[2.11 UI Sample of Planned Data 35](#_Toc400908189)

[2.12 UI Sample of Planned Output Designs 43](#_Toc400908190)

[2.13 Descriptions of Redundancy Measures (Data) 47](#_Toc400908191)

[2.14 Description of Measures Planned for System Sec. 47](#_Toc400908192)

[2.15 Overall Test Strategy 49](#_Toc400908193)

[Technical Solution 50](#_Toc400908194)

[Testing 50](#_Toc400908195)

[4.1 A minimal set of Test Data 50](#_Toc400908196)

[4.2 Details of Individual Tests 50](#_Toc400908197)

[4.3 Expected results for typical data 50](#_Toc400908198)

[4.4 Expected results for erroneous data 50](#_Toc400908199)

[4.5 Expected results for extreme data 50](#_Toc400908200)

[4.6 Samples of Actual Test Runs 50](#_Toc400908201)

[4.7 Samples of Annotated hard copy showing system 50](#_Toc400908202)

[4.8 All Samples cross-referenced to Test Plan 50](#_Toc400908203)

[Maintenance 50](#_Toc400908204)

[5.1 System Overview 50](#_Toc400908205)

[5.2 A sample of Detailed Algorithm Design 50](#_Toc400908206)

[5.3 Procedure and Variable Lists 50](#_Toc400908207)

[5.4 Annotated List of Program Code 50](#_Toc400908208)

[References 50](#_Toc400908209)

[Email Automation 50](#_Toc400908210)

[Encryption 50](#_Toc400908211)

[Regex 50](#_Toc400908212)

[Captcha 50](#_Toc400908213)

# Analysis

## 1.1 Introduction

### 1.1.1 Background

The Organization, Hanh's Nail & Beauty, is a family operated Nail Salon specializing in Waxing and Nail Care services such as Pedicures and Manicures. The store opened up in 2008, residing in Canning Town in East London. The area is surrounded by regeneration; New buildings are being constructed alongside a Morrison's supermarket sizing over 7,000 square meters. As a result, business growth is growing at a consistent rate, with new faces popping inside the store every day alongside the regular customers.

The store, whilst petite, has two floors, consisting of six nail benches and two Massage Chairs. Upstairs houses a waxing bench. The amount of workers varies depending on the situation, with its median number being four workers.

The owners and my client, Hoang and Hanh, deal with the booking system. This currently consists of a Appointment book with appointments noted down into its allocated tables. This book is then used to freely allocate customers to available staff who will then service them. The appointment book is not strictly enforced, leading to occasional freehand allocation. This sometimes leads into unpredictable timing issues where future customers have to wait at the store in order to be serviced.

### 1.1.2 Problem Definition

The current booking system is completely manual, and there is only one copy which is only dealt with by Hanh. It consists of an appointment book where each page consist of timetables, where customers can be allocated a slot into. Customers would come into the shop and ask for an appointment whenever they can if the shop is busy. There is no other way to contact them.

Not only is this manual system very lax in purpose, but occasionally it creates legibility issues due to the variation of handwriting that each worker has. Also, as it is a completely physical entity, There is no easy way to have a copy of the system, which consequently led to complications when the appointment book becomes missing. This becomes difficult for the customer also. When the current booking isn't enforced, they are forced to wait for a space, which results in the customer sitting around waiting until it is their turn. This leads to become a hassle as the customer wastes time.

This also leads to the issue of managing the money that comes into the business. The accounts are registered in another book, and weekly calculations are done on a calculator. {{expand}}

All of these problems result in hindering the progression of the business. From one perspective it shows that the business is not making as much as it possibly could due to ineffiencies such as the handling appointments in the one book, which also accounts for the invoices and weekly revenue.

### 1.1.3 The Users

The primary users will be the staff, who have a below-average IT experience. They plan to use the software to manage the booking system, where they would be able to add appointments alongside the ability to checkout customers. They will be also be able to see the expenditures and invoices for the business. Since the primary users experience with IT can be incredibly varied, it would be just to include a reference guide or similar in order to include every one of different skills.

The secondary users apply to the business owners, Hoang and Hanh. Not only do they also deal with appointments, but they also manage the finances of the company. This means that they manage company expenses and invoices from customers. They would be given elevated privileges in this context. This category of users similarly has a below-average IT experience.

The tertiary users would apply to the customers, who will be able to book appointments on the system. They would be able to book appointments whenever they can, and however they can. Their understanding of IT varies wildly, so it would be logical to assume the worst-case scenario in order to cater to all potential customers.

The user(s) should have a sufficient knowledge of how this system is operated. Otherwise the software ceases to function. Therefore it would be very important to make the software as user-friendly and intuitive as it could. A guide can be embedded onto the system and can be disabled per user using the accounts metadata.

## 1.2.1 The Current System Analysis

### 1.2.1.1 Interview

The interview happened on the 5th of September, to Hanh and Hoang, the primary users. The interview was conducted to obtain information about the current system and its problems. At the end, the users were allowed to voice their own ideas on what the website should have. Below contains a summarised report of the interview.

Interview Report

5th September 2014

What is the current system and how is the current system deployed?

The current system is composed of a appointment book that is restocked monthly. Customers would have to travel to the store in order to book an appointment. Hanh deals with the bookings. Appointment times, as well as the name of the customer, their phone number, and the service they require are noted down in a box representing the time period. Some customers can book more than one appointment, so the noted information is then duplicated into the other box(es). There is only space for one appointment per time for the book, so if there were to be two bookings in one space then it is noted seperately at the bottom of the page. When the customer comes for the appointment, they have to refer to Hanh.

Problems with the current system

The current system is managed by one person only, and if any other staff have to validate appointments they have to refer to hanh.

If they were to deal with the booking in the case Hanh is away, then they occasionally find difficulty in understanding the notation Hanh uses.

The booking process is not efficent as customers have to travel to the store to book an appointment. The shop does have a phone number, but most of the time everyone is either occupied and is unable to pick up the phone, or Hanh has to pick up the phone whilst dealing with customers.

How many people do you expect to be using the system at any given time?

As established beforehand, there is only one person dealing with the booking system. However, there are instances where the other staff take over the role temporarily as the main user, Hanh, is not available.

From the customers perspective, It is expected that all customers are to use the booking system as the store is busy. There is instances however where the staff can pop in and wait for a time that the staff are available.

How much data is there? What kind of data is there?

Data consists of customers names, their phone numbers, and the type of service they require. It also contains the time of the booking and such. This is considered sensitive data as it can be used to identify living individuals.

While there is no convertible information for information in pages as the entire system is analogue, they purchase new books monthly. It is estimated that if the entire system is to be digitalised, one copy of the system would be single digit gigabyte values.

How is the data managed?

The data is handled on books and no redundancy is taken to protect information. This has disadvantaged them in occasions where books become missing. This is crucial as the Data Protection Act (1998) essentially makes it the law to protect sensitive information such as this.

Does the client have any ideas to consider?

The client mentions that there are a lot of services that can make the booking system easier, and that they have tried sampling programs such as BookingBug. They found out however that it was very difficult to maintain due to its complexity. They preferred that a booking system did not require a computer, but rather a simple application that can be run on smartphones and tablets as that is currently used in the store.

Possible Solutions

A booking system is proposed. Lots of aspects in this system can definitely be simpler through the use of an automated system such as the duplication of customer details. The system is definitely computable.

### 1.2.1.2 Questionnaire

In order to get more accurate and valued information about the system, I opened a focus group to the rest of the staff to get their opinion on the current system. As a result I found out that the staff also found it difficult to communicate with the customers as they are forced to bend the system. The staff found it difficult to comprehend the appointment book due to legibility issues, but also it became physically impossible to know what was going on when the appointment book went missing.

{{statistics}}

### 1.2.1.3 Observation of Existing System

I have chosen to shadow the current users of the current system I could have a better understanding of how the users interact with it on a daily basis, and identify the problems they face whilst doing this.

The current process is as followed:

1. The customer comes and asks for an appointment.

1.1. The customer asks the staff what services they require.

2. The staff checks the availability of the appointment.

2.1. The staff contacts Hanh.

2.2. Either the staff waits for hanh to finish what she is currently working on or she normally jots it down.

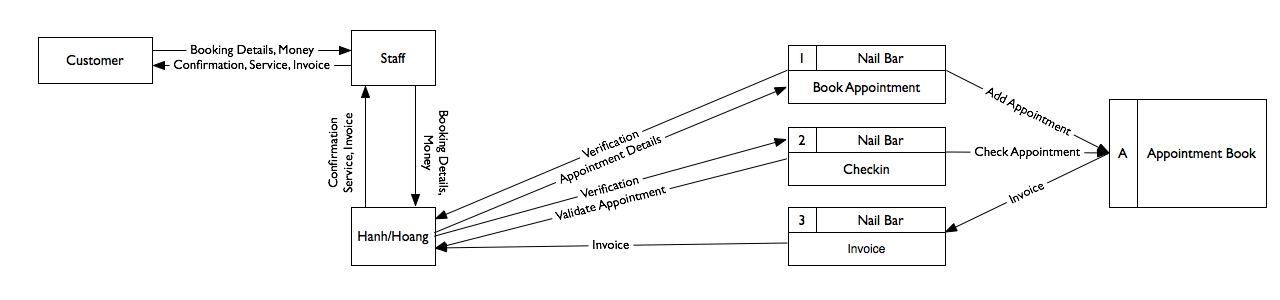
2.3. Hanh checks wether there is space available

2.3.1. Hanh calculates for the availability of staff on the day.

2.3.2. Hanh calculates the availability of staff on that time period and makes a decision based on so.

3. The customer is confirmed of their appointment, or they schedule another time. If the latter was the case, The customer repeats the process from 2.3.

### 1.2.2 DFD of current system (to Level 1)



### 1.2.3 Data Sources and Destinations

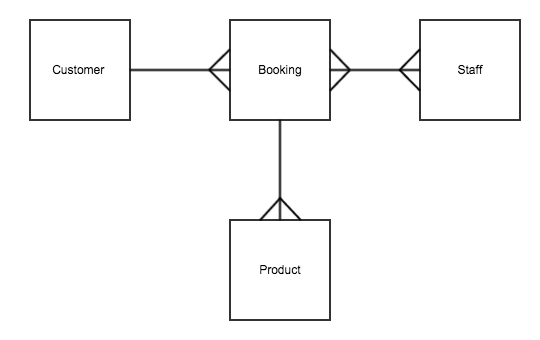
While the current system does not possess any systematic process, A brief summary of incoming information and destinations are listed in a table format below.

|  |  |  |  |
| --- | --- | --- | --- |
| What Is It? | Source | Destination | Information on Data |
| Name | Customer | Appointment Book |  |
| Date | Customer | Appointment Book |  |
| Phone Number | Customer | Appointment Book |  |
| Booking Type | Customer | Appointment Book |  |

### 1.2.4 Entity-relationship diagram

The current entity relationship diagram is not as such a complicated process. It is expected that the entity relationship diagram for the solution would be similar.

A customer has many bookings. A booking can have one customer. A booking can have many products. A product can have many bookings. A booking can have many staffs. The staff can have many bookings.



Booking(CustomerID, ProductID DateTime, Total, StaffID)

### 1.2.5 Discussion of problems with Current System

The following list is the currently complete list of issues regarding the system.

The current system is managed by one person only, and if any other staff have to validate appointments they have to refer to hanh.

If they were to deal with the booking in the case Hanh is away, then they occasionally find difficulty in understanding the notation Hanh uses.

The booking process is not efficent as customers have to travel to the store to book an appointment. The shop does have a phone number, but most of the time everyone is either occupied and is unable to pick up the phone, or Hanh has to pick up the phone whilst dealing with customers.

The staff does not enforce appointments. This becomes archaic when the store becomes busier and customers are forced to wait in the store inorder to be served.

The booking system is completely manual, leading to physical constraints such as only have one copy of the booking system.

The booking system is not secure and puts lots of customers information at risk.

## 1.2.6 The Proposed New System Analysis

#### 1.2.6.1 User Needs

Most importantly, the new system should allow the user to do do everything that can be done with the current system. If the new system doesn’t meet this requirements, then the project would cease be implemented by the staff. The staff would need to feel familiarity with the new system as they don't have a strong understanding of computer based user interfaces.

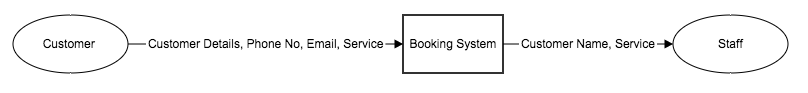
The primary user wishes to use an automated booking system to efficiently deal with customers incoming requests. For this to work, The following would be a requirement:

Information about the customer should be stored systematically. The customer details should be set in a way that the staff can pinpoint the customer details without going through many hoops to do so.

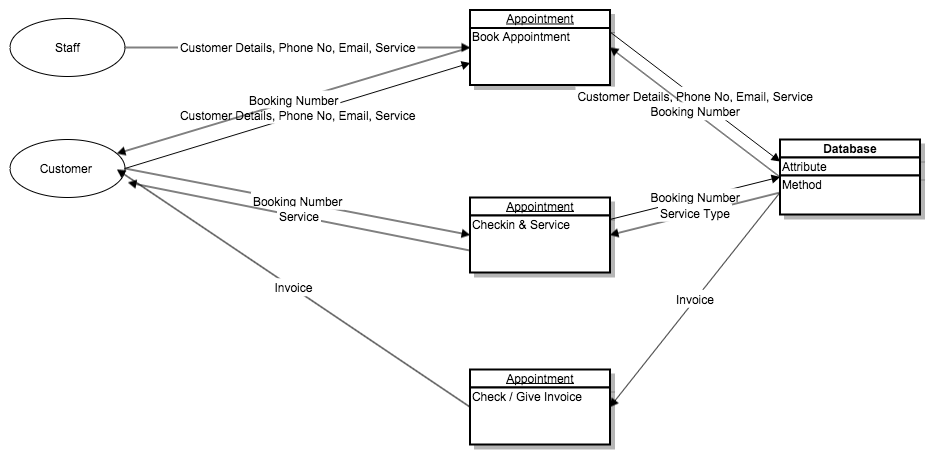
Information about the services can be shown as a list and can be showed with extra detail such as the price and what it provides. This would be shown to the customer as the staff would already have an understanding of what they offer.

### 1.2.7 DFD of Proposed New System

Level 0:



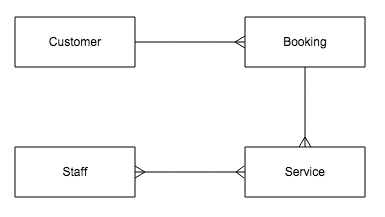
Level 1:



### 1.2.8 Data Sources and Destinations

|  |  |  |
| --- | --- | --- |
| What is it? | Source | Destination |
| Name | Customer | Database |
| Email | Customer | Database |
| Date/Time | Customer / Time of Booking | Database |
| Service | Customer | Database |
| Phone Number | Customer | Database |

### 1.2.9 Entity-relationship diagram and entity descriptions of current system



Customer: The customer entity stores data that is related to the customer such as their credentials and contact information. If we were to go into detail this would consist of usernames, passwords, names and phone numbers.

Booking: The booking entity consists of customer identification and their required service.

Service: The service entity lists the available services that the store can offer.

Staff: The staff entity contains information of all staff and what they can provide for the shop.

The customer can have many bookings. The booking can refer to many services, but only one service can be paired to a booking. Many staff can provide a single service and a service may consist of many staff.

### 1.2.10 Analysis Data Dictionary

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Client |  |  |  |  |  |  |
| Attribute | Field Purpose | Length | Type | Rules | Example Data | Validation |
| ID | Makes each entry unique and identifiable | 11 | Integer | Automatic Increment | 1 | Handled by SQL Server. |
| Username | Defines user. | 25 | Variable Character | letters, numbers and underscore only | skadir | Must not have duplicates. |
| Password | Validates authentication of user | 256 | Variable Character | Must contain a capital, a lowercase and a number | E$oD%hAcs=Urm} | Not blank |
| Forename | To identify the user by its Firstname. | 50 | Variable Character | First character is capitalised | John | Not Blank |
| Surname | To identify the user by its surname. | 50 | Variable Character | First character is capitalised | Smith | Not blank |
| Email | Serves as a directory to email to send forgotten passwords to. | 100 | Variable Character | Must contain “@“ and “.” | [jsmith@woodhouse.ac.uk](mailto:jsmith@woodhouse.ac.uk) | Must contain "@" and "." |
| Phone Number | Serves as a direct way for the staff to contact the customer if the case requires so. | 11 | Integer | Must start with 0 | 071234567889 | Must contain 11 characters |
| Service | Allow the customer to specify the service they require | 100 | Variable Character |  | 1 |  |
| DateTime | Time of appointment they wish to book at. |  | Datetime | Must be in date form. | 01/01/2014 |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Staff |  |  |  |  |  |  |
| Attribute | Field Purpose | Length | Type | Rules | Example Data | Validation |
| ID | Makes each entry unique and identifiable | 11 | Integer | Automatic Increment | 1 | Handled by SQL Server. |
| Username | Defines user. | 25 | Variable Character | letters, numbers and underscore only | skadir | Must not have duplicates. |
| Password | Validates authentication of user | 256 | Variable Character | Must contain a capital, a lowercase and a number | E$oD%hAcs=Urm} | Not blank |
| Forename | To identify the user by its Firstname. | 50 | Variable Character | First character is capitalised | John | Not Blank |
| Surname | To identify the user by its surname. | 50 | Variable Character | First character is capitalised | Smith | Not blank |
| Email | Serves as a directory to email to send forgotten passwords to. | 100 | Variable Character | Must contain “@“ and “.” | [jsmith@woodhouse.ac.uk](mailto:jsmith@woodhouse.ac.uk) | Must contain "@" and "." |
| Phone Number | Serves as a direct way for the staff to contact the customer if the case requires so. | 11 | Integer | Must start with 0 | 071234567889 | Must contain 11 characters |
| SkillSet | Specifies the skill(s) that the individual person can offer. | 100 | Variable Character |  | 1 |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Service |  |  |  |  |  |  |
| Attribute | Field Purpose | Length | Type | Rules | Example Data | Validation |
| ID | Makes each entry unique and identifiable | 11 | Integer | Automatic Increment | 1 | Handled by SQL Server. |
| Name | Type of service | 25 | Variable Character | letters, numbers and underscore only | skadir | Must not have duplicates. |
| Staff | Lists staff who can offer this service | 256 | Variable Character | Is an Array | E$oD%hAcs=Urm} | Not blank |
| TimeTaken | Describes how long this service will take |  | DateTime | numbers | 00:01:00 | must be to be set in timeform |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Booking |  |  |  |  |  |  |
| Attribute | Field Purpose | Length | Type | Rules | Example Data | Validation |
| ID | Makes each entry unique and identifiable | 11 | Integer | Automatic Increment | 1 | Handled by SQL Server. |
| CustomerID | Type of service | 25 | Variable Character | letters, numbers and underscore only | skadir | Must not have duplicates. |
| ServiceKey | Lists staff who can offer this service | 256 | Variable Character | Is an Array | E$oD%hAcs=Urm} | Not blank |
| StaffID |  |  |  |  |  |  |
| DateTime | Specifys the date and time of the appointment |  | Date |  |  | Unix Time? |

### 1.2.11 Data Volumes

Recalling from the first interview, I discovered that the majority of data would consist mainly of contact details. This shouldn’t exceed 2GB, therefore data shouldn’t be a concern. If we were to consider corruption on hard drives, the storage mediums that hold the server data should be installed in a RAID configuration, which would be explained in further detail in *1.3.1 Hardware Constraints.* Nightly backups are important, so it is recommended that the created software should be used in conjunction with another program that deals with backups and redundancy.

## 1.3 Constraints

### 1.3.1 Hardware constraints

Its fair to say that the server, the computer that would host the system and its database, should be sufficiently fast to possibly handle the maximum theoretical load. It would be preferred that the Hardware should be equipped with a X86 based processor, and should come with a sufficient amount of ram to make use of the standard 64-bit architecture that is set recently.

If we were to pinpoint the specifics of hardware, It may need to have an equivalent of a quad core processor with error correcting RAM modules installed on it. It won’t need to have a monitor connected to it, and Hard drives should be connected in a RAID array, preferably RAID5 or RAID1, dependent on how much is invested into the server. Hardware constraints should seldom be a hindrance. Many modern computers have sufficient processing horsepower to manage a small business such as this.

The store houses a computer which is capable of holding the database and the system backend. There is also an internal network where the staff normally uses to connect to the internet. This can be utilized as the terminals can be simply smartphones or tablets which connect to the backend through the internal network.

It is also reasonable to offshore the system database through a remote server. This may prove beneficial as not only that there would be a dedicated team of IT Technicians who will be to monitor the server, but the server would have its own dedicated power supply and a leased line. This proves vital in situations where the power is out, or that the server is physically removed. However that requires the shop to have a reliable source of internet, which in itself, provides more unnecessary expenditures for the company.

In order to connect to the system, tablets would server as the terminals. The tablets have to be sufficently fast in order to streamline the booking process. Modern tablets are sufficently fast and should have no problem in acting as a terminal.

### 1.3.2 Software constraints

Fortunately, with the use of web-development language, the support for such technologies is universal, meaning that if we were to consider the support from operating systems, Any operating system that includes native web-browsing capabilities is supported.

With the software being operating-system agnostic, the main requirement is the ability to run a web-server program such as the AMP packages such as MAMP, XAMP, WAMP, so on and so-forth. With these programs, the hardware would serve as the database and website host. Alternatively, we could install Apache and MySQL Server on the hardware as separate entities, as they both serve the same purpose.

The system is accessed using a web browser. The web browser used must support HTML5 to render the website properly. Legacy browsers are a concern, but with the convergence of modern technology, this should seldom be a concern. The majority of the browsers on smartphones and tablets are HTML5 Optimized, which would be the primary method in connecting to the server.

### 1.3.3 Who will be allowed to use the various parts of the system

Of course, as laid out in *1.2.7 - DFD of proposed new system,* The secondary user should have general access to the front end of the system. They should be able to have access to the end-user side of the system, such as querying the database for exam-files, videos and they should be able to comment. Most importantly, they have access to visit the website in the first place.

The primary user, the teachers, has access to the *dashboard*, which allows them to manage the content on the website, and is able to upload files and information to the database. They should not have administrative credentials, yet the option should be available to click.

## 1.4 Limitations

### 1.4.1 Areas which will not be included in computerization

Areas that will not be included is the

### 1.4.2 Areas considered for future computerization

Areas that may be included is the ability to invoice the customer automatically, and to finalize appointments by paying beforehand a deposit or the whole invoice. This way, booking process can be completely seamless and the customer doesn't have to worry about bringing money into the store. This also benefits the staff as not only are they are paid upfront, but they also don't have to deal with transaction issues such as dealing with change and handling physical money. For the time being though, it is in itself, another very complex system so it would be only considered if the original project is completed in the first place.

## 1.5 Objectives

### 1.5.1 General Objectives

The purpose of this project, and the objective of the system is to create an easier way for the primary user to manage appointments by having most of the necessary tools to do so. The system should be able to input customer details and booking types from either the secondary user or the primary user. It must be fast, efficient and easy to navigate around.

### 1.5.2 Specific Objectives

|  |  |  |
| --- | --- | --- |
| Input Objectives | Inputting a new staff | The administrator should be able to create a new staff and add the details into different records by adding text to relevant fields about the new staff. |
|  | Set new appointment | The tertiary user must be able to book an appointment by clicking on the screen and setting the required service, along with the date and time. |
|  | Updating customer details | The customer should be able to update their credentials, such as their name and password. |
|  | Inputting customer details | The customer should be able to register an account using their name, phone, email and password. |
|  | Checking-in customer | The staff must be able to check in the customer in order to validate the booking process. |
|  | Login | The staff and the customer must be able to log into their respective control panels. |
|  | Checkin Customer | The staff must be able to check in the customer in order to complete the booking process. |
| Output Objectives | Load available timings for appointments | The system should be able to get information from the database, calculate availabilty, and display it for the secondary user to see. |
|  | Confirmation Message | The customer must be notified that the booking has processed successfully. |
|  | Password Validation | When registering, a password strength meter shows how strong the password is. |
|  | Display User on Login (Dashboard) | The user credentials should be displayed when the primary user logs into the dashboard. |
|  | Error Messages | The system should output error messages relating to the problem that has occurred, such as Regex validation, Logging into the database. |
|  | Updated Information | Updated information should be displayed and the user should be notified of a change. |
|  | Showing User Information | The system should be able to display user information on a page that allows them to change credentials. |
| Processing | Identify customers uniquely | The system must be able to uniquely identify each customer. |
|  | Identify bookings uniquely | The system must be able to uniquely identify each booking. |
|  | Identify staff uniquely | The system must be able to uniquely identify each staff. |
|  | Check Availability | The system must be able to check if a period is free for an appointment. |
| Storage | Store Customer Details | The database must be capable of storing customer credentials |
|  | Store Staff Details | The database must be capable of storing customer credentials |
|  | Store Bookings | The database must be capable of storing customer credentials |
|  | Store Services | The database must be capable of storing customer credentials |
| Performance |  |  |

## 1.6 Consideration of alternative solutions

One solution may be to create a piece of software that requires installing. From a technical standpoint, the software would perform the fastest, and it would provide the most native experience for the customer, as it could follow standard design protocols depending on the operating system.

If we were to make this an actual piece of software, it would be great considering that all of the processing is dependent on the user’s hardware, but with that comes a hefty price. Not only do you have to consider the compatibility of every modern device, but you would need to port the program into different kinds of renditions, fragmenting the purpose in the process. Not only is this infeasible and prone to errors, the time constraints show that this is a completely infeasible solution to imagine, let alone to consider.

Another solution may be to improve the current process by simply designing a better appointment book that the staff can use to put appointments inside, and setting up a phone that customers can call. This would be ideal as the other staff would still be familiar with the process, and customers would be able to call remotely.

However, the staff is occupied most of the time servicing customers on site. This means that they would require a dedicated receptionist to deal with all bookings. As much as the process is more efficient, it would be the most expensive option. Also, they simply do not possess the physical space for a receptionist, as the store is small.

Alternatively, We can use premade booking solutions such as Bookingbug. However with using special purpose software, you are limited to what that software can provide. You can have a more customized piece of software by creating a bespoke solution to the problem.

While the current solution works for now, It doesn't work effectively and will hinder the performance of the business as it expands in growth.

### 1.7 Justification of chosen solution

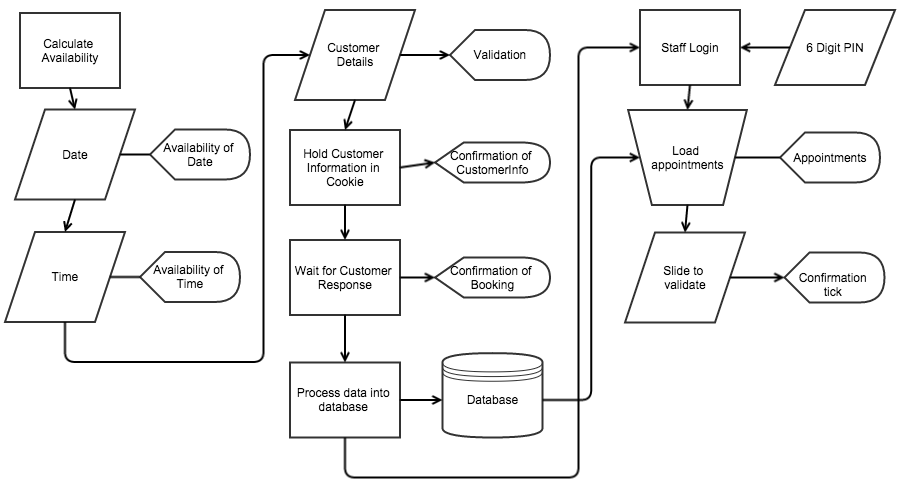
I have chosen to develop the system using a web-application using HTML, CSS, PHP and JavaScript. This way, all of the users can freely access the system online using a browser, which many people have access to. With this method, I will be able to use SQL expressions to perform searches, and data can be centralized into a database.

* This benefits the business owners as they can manage who has access to the system intuitively.
* The business owners can see the amount of customers on the day whenever due to the nature of this system.
* It utilizes computers and tablets that are in the store already, making it cheap to deploy in terms of dealing with hardware.
* The customer can book in their own time, and check without calling the store. This is especially useful as the staff can pay attention to providing the customers at the store what they need.
* There are many advantages of using PHP and MySQL. For instance, they are free to use for commercial purposes, They are fast to deploy and manage, and they are supported on all modern browsers.
* This solution is the most efficient way as I have recently developed a fair understanding of the languages that are necessary to implement into the system.

# Design

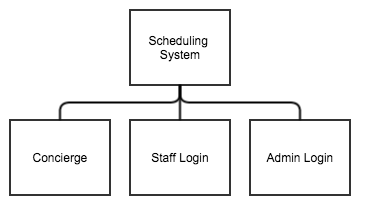
## 2.1 Overall System Design

### 2.1.1 System Flowchart

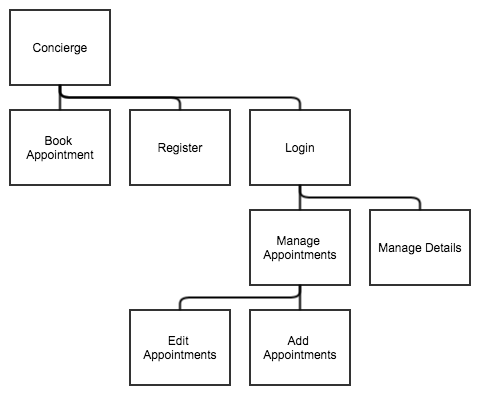


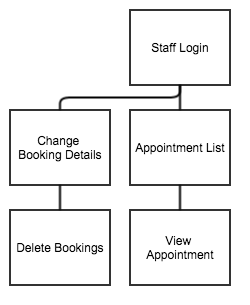
This flowchart represents how the system will operate. The majority of this process is automated, however it relies on input from the staff and the customers in order for this system to work. The flow starts with the customer choosing a date and time for the booking, followed with the rest of the credentials necessary to book. This information is then stored into the database, which the staff can process and validate once the customer arrives at the store.

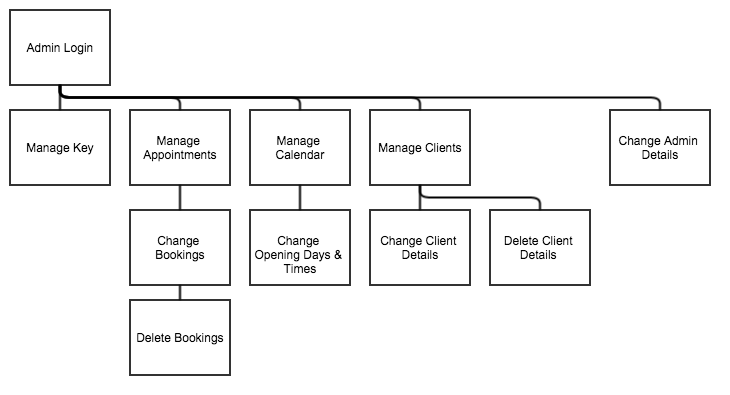
### 2.1.2 Hierarchy Chart



The scheduling system is split into three partitions, the concierge, the staff login and the admin login. This is split for the three different categories of users. The customer can only access the Concierge, the staff can access the Concierge and the Staff Login, and the administrators have full access into the system.







The administrator’s dashboard allows for flexibility with the system. It allows the administrators to change different settings such as the opening days, the daily staff key, as well as modify the details of the clients and appointments.

## 2.2 Description of Modular Structure of System

|  |  |  |  |
| --- | --- | --- | --- |
| Input | Output | Process | Storage |
| New Booking | Confirmation Post | Validate customer details | Customer Details |
| Customer Details | Registration Validation | Save customer details into database | Admin Details |
| Staff Details | Email Post | confirm booking | Staff PIN |
| Booking Options | Completed Booking | filter availability | Bookings |
| Username | Confirm Change | Encrypt password details | Opening Days |
| Password | List of Check-ins | Encrypt username |  |
| Service Option |  | Confirm Check-in |  |
| Email Address |  | Generate Daily Staff PIN |  |
|  |  | Send Automated Emails |  |
|  |  | Regex Validate |  |
|  |  | Change Password |  |

## 2.3 Definition of Data Requirements

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | UNF |  | 1st NF |  | 2nd NF |
| Booking | ClientName | Client | **ClientUsername** | Client | **ClientUsername** |
|  | ClientEmail |  | ClientPassword |  | ClientPassword |
|  | ClientPassword |  | ClientForename |  | ClientForename |
|  | ClientUsername |  | ClientSurname |  | ClientSurname |
|  | ClientPhoneNo |  | ClientEmail |  | ClientEmail |
|  | DateTime |  | ClientPhoneNo |  | ClientPhoneNo |
|  | ServiceID | Booking | BookingID | Booking | **BookingID** |
|  | ServiceType |  | *ClientUsername* |  | *ClientUsername* |
|  | ConfirmedByStaff |  | *ServiceID* |  | *ServiceID* |
|  | Comments |  | ServiceType |  | BookingDate |
| Staff | StaffPinCode |  | BookingDate |  | BookingTime |
|  | StaffPinDate |  | BookingTime |  | ConfirmedByStaff |
|  | AdminUsername |  | ConfirmedByStaff |  | Comments |
|  | AdminPassword |  | Comments | Service | ServiceID |
|  |  | Staff | StaffPinDate |  | ServiceType |
|  |  |  | StaffPinCode | Admin | **AdminUsername** |
|  |  |  | AdminUsername |  | AdminPassword |
|  |  |  | AdminPassword | StaffPIN | **StaffPINDate** |
|  |  |  |  |  | StaffPinCode |

Primary Keys are in bold, composite keys are in italics and foreign keys are underlined.

From the un-normalised form to the first normal form, DateTime is split into Date and Time. Booking has been allocated a primary key, BookingID, and a composite key (clientusername, ServiceID)

With the transition from first normal form to second normal form, StaffPIN is taken out to make its own table. ServiceID and ServiceType is also taken out and turned into its own table. ClientUsername and ServiceID now acts as foreign composite keys in the second normal form.

## 2.4 Database Design (Normalised ER)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Client |  |  |  |  |  |  |
| Attribute | Field Purpose | Length | Type | Rules | Example Data | Validation |
| Username | Defines user. | 25 | Variable Character | letters, numbers and underscore only | skadir | Must not have duplicates. |
| Password | Validates authentication of user | 256 | Variable Character | Must contain a capital, a lowercase and a number | E$oD%hAcs=Urm} | Not blank |
| Forename | To identify the user by its Firstname. | 50 | Variable Character | First character is capitalised | John | Not Blank |
| Surname | To identify the user by its surname. | 50 | Variable Character | First character is capitalised | Smith | Not blank |
| Email | Serves as a directory to email to send forgotten passwords to. | 100 | Variable Character | Must contain “@“ and “.” | [jsmith@woodhouse.ac.uk](mailto:jsmith@woodhouse.ac.uk) | Must contain "@" and "." |
| Phone Number | Serves as a direct way for the staff to contact the customer if the case requires so. | 11 | Integer | Must start with 0 | 071234567889 | Must contain 11 characters |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Service |  |  |  |  |  |  |
| Attribute | Field Purpose | Length | Type | Rules | Example Data | Validation |
| ID | Makes each entry unique and identifiable | 11 | Integer | Automatic Increment | 1 | Handled by SQL Server. |
| Type | Type of service | 25 | Variable Character | letters, numbers and underscore only | skadir | Must not have duplicates. |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Admin |  |  |  |  |  |  |
| Attribute | Field Purpose | Length | Type | Rules | Example Data | Validation |
| Username | Makes each entry unique and identifiable | 11 | Integer | Automatic Increment | 1 | Handled by SQL Server. |
| Password | Type of service | 25 | Variable Character | letters, numbers and underscore only | skadir | Must not have duplicates. |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Booking |  |  |  |  |  |  |
| Attribute | Field Purpose | Length | Type | Rules | Example Data | Validation |
| ID | Makes each entry unique and identifiable | 11 | Integer | Automatic Increment | 1 | Handled by SQL Server. |
| ClientUsername | Key to refer to the customer that is booking | 25 | Variable Character | letters, numbers and underscore only | skadir | Must not have duplicates. |
| ServiceID | Key to link to the service that the customer has booked for | 256 | Variable Character | Is an Array | E$oD%hAcs=Urm} | Not blank |
| ConfirmedByStaff | Comfirms and completes the booking | 1 | Boolean | Must be either true or false | true | handled by SQL Server. |
| Comments | Comments that the customer wishes to mention | 1000 | String |  | Lorpem Ipsum sit amet dor |  |
| Date | Specifies the date and of appointment | 8 | Date | Y/M/D | 2012/01/01 | Must be a real date |
| Time | Speficifes the time of the appointment. | 5 | Time | Hours and minutes only | 13:00 | Must be numbers. Value has to be more than the opening time and less than the closing time, |

{{Normalised ER Diagram}}

## 2.5 File Organization and Processing

The data dictionary, as shown beforehand, shows that the size of the database shouldn't be large. The contents of the database would be completely text, and no other binary content such as media files. The database must be able to hold information, as well as being able to support more content as the salon expands.

The program has been designed to work around text data and not large media files, so the size of the program should never exceed ten megabytes. The database that will hold the data that is inputted would start in the kilobyte range, expanding in correlation with the growth of the business. It is not expected for the database to contain over a million accounts; and if somehow it does, the size of the database should not exceed 100mb. These small file sizes allow the recalling of data simple and fast.

The fact that the program is a web application containing a centralised database means that the customer doesn’t need to install anything other than a web browser.

## 2.6 Identification of Storage Media

The physical storage media primarily consist of hard drives. The hard drives that would be used are two 2TB SAS Drives in a RAID 1 Array. These drives run independent of the operating system of the server, which is stored on a separate hard drive. While it seems completely arbitrary and overkill, it poses the idea that the system should seldom worry about the event of the disk storage being completely full or if one of the drive fails.

## 2.7 Identification of Suitable Algorithms (Pseudo code)

### 2.7.1 Bubble sort

Bubble sort is one of the sorting algorithms that was taught in AS Computing, which I feel can be used in this system where the secondary user has to sort through appointments for different criteria.

procedure bubbleSort( A : list of sortable items )

n = length(A)

repeat

newn = 0

for i = 1 to n-1 s do

if A[i-1] > A[i] then

swap(A[i-1], A[i])

newn = i

end if

end for

n = newn

until n = 0

end procedure

### 2.7.2 Establishing a connection to the database.

In order to connect to the database, we have to write a script that would establish the connection between the database and the website. This is written using PHP’s data objects extension, which is a prebuilt function in PHP. An attribute is set to the variable which disables prepared statements that are emulated, preventing SQL Injections.

try {

pdo = new PDO('mysql:host=host;dbname=database', 'username', 'password')

pdo -> set attribute(set prepared statements to false)

} catch (PDOException error)

{

error = ‘Unable to connect to the database server.’

include output(error)

}

### 2.7.3 Salt and Hashed Encryption

In order to make the login process secure, It is necessary to implement encryption to the user’s credentials, especially the password. I have chosen to encrypt the password using SHA-256, the industry standard.

A salt is necessary. Modern cryptographic hashes are hard to crack, but recently, hackers have compiled a reverse lookup table containing the pre done hashes of the most popular password options. If we were to not include a salt into the program, then a hacker can simply go through the lookup table and match the hash that is in the database. Hackers can also implement brute force attacks. As computers grow to become faster at an exponential rate, it becomes easier to use brute force as a means of unauthorised access.

$salt = ‘custom\_variable';

$username = $\_POST['username'];

$password = hash('sha256', 'password' . $salt);

### 2.7.4 Breadcrumbs

The breadcrumbs function is created in order to easily navigate through pages in a systematic procedure. This way, A page directory structure doesn’t need to be manually programmed for every page, since this exemplar code would do so.

Function Breadcrumbs(import separator to “/“, home to “URL”)

breadcrumbs = array(homepage)

last = end\_of\_URL

path = explode ‘/‘ and parse URL

base = homepage\_directory

foreach path = x to crumb

title = URL.end + crumb

if x is not last in array

breadcrumbs[] = URL\_title

else

breadcrumbs[] = title

endfor

print implode(separator + breadcrumbs)

<http://stackoverflow.com/questions/2594211/php-simple-dynamic-breadcrumb>

### 2.7.5 Email Automation

Mail Automation is necessary for the system as it is used to send out emails to customer about confirmations, reminders and follow-ups. Utilizes the Mail() function that is embedded in PHP. This is used to send emails to the customer when they confirm their booking. It is also used when the customer wishes to change their password, or even if they forget their password. This algorithm uses the SMTP Protocol.

msg = Get Message Form Input

email = Get Email Form Input

'Comment: ' ."\n" .$\_POST['comment'];

mail('example@example.com',

'Sample Comments', $msg);

header('location: contact-us-thank-you.html');

exit(0);

### 2.7.6 Regex Expressions

Regex is a sequence of characters that are used to form a search pattern. It will be used to validate the customers credentials such as email addresses and passwords, making sure that the password is secure enough, and the email contains an “@” symbol. We can use PHP’s pre-built function filter\_var() to verify the syntactical validity of the email address, but it is preferred if it is self-built.

function validate\_email($e){

return (bool)preg\_match("`^[a-z0-9!#$%&'\*+\/=?^\_\`{|}~-]+(?:\.[a-z0-9!#$%&'\*+\/=?^\_\`{|}~-]+)\*@(?:[a-z0-9](?:[a-z0-9-]\*[a-z0-9])?\.)+[a-z0-9](?:[a-z0-9-]\*[a-z0-9])?$`i", trim($e));

}

### 2.7.7 Captcha

Captcha is used to deter bots from flooding the process of booking and to prevent hackers from creating automated scripts that could harm the load of the server or flood the database with data, causing the database to crash.

User -> IP

Captcha-token = Scramble <- (Captcha-code + IP + Secret-code)

User <- Captcha-token

User solves the captcha, submits answer:

User -> IP, Captcha-code answer, Captcha-token

New token = Scramble <- (Captcha-code answer + IP + Secret-code)

IF Captcha-token = New token

User <- "Correct! You are a human."

ELSE

User <- "Incorrect, please try again."

## 2.8 Sampled SQL Queries

Below are examples of queries that will be used in the new system.

|  |
| --- |
| Adding a new customer into the database |
| INSERT INTO Client(ClientForename, ClientSurname, ClientEmail, ClientPassword, ClientUsername, ClientPhoneNo) VALUES(?, ?, ?, ?, ?, ?) |

|  |
| --- |
| Adding a new appointment |
| INSERT INTO Booking(ClientID, ServiceKey, DateTime, ConfirmedByStaff) VALUES(?, ?, ?, ?) |

|  |
| --- |
| Updating an appointment |
| UPDATE Booking SET ClientID, ServiceKey, DateTime, ConfirmedByStaff |

|  |
| --- |
| Removing an appointment |
| DELETE \* FROM Booking WHERE DateTime = ? |

|  |
| --- |
| Confirming an appointment |
| UPDATE Booking SET (ClientPassword) WHERE QID=? |

|  |
| --- |
| Logging into the Administrators Dashboard |
| SELECT \* FROM Admin WHERE AdminUsername = ? AND AdminPassword = ? |

|  |
| --- |
| Updating a customer’s password |
| UPDATE Client SET (ClientPassword) WHERE ClientPassword IS NOT ClientPassword AND ClientPassword=? |

|  |
| --- |
| Displaying All Appointments on database |
| SELECT DateTime, ClientID, ServiceKey, ConfirmedByStaff, ClientForename, ClientSurname FROM Client, Booking WHERE DateTime LIKE ? |

## 2.9 Class Definitions

|  |
| --- |
| Class: Calendar |
| Year as Integer  Month as Integer  Day as Integer  OpeningTime as Integer  ClosingTime as Integer  BookingFrequency as Integer  Function calculate weeks a month (month as integer, year as integer)  Year = date(y, time)  Month = date(m, time)  DaysinMonth = daysinmonth(month,year)  NumofWeeeks = DaysinMonth + DaysinMonth/7  MonthEndingDay = stringtotime(year,month,daysinmonth)  MonthStartingDay = stringtotime(year,month,1)  If(MonthEndingDay<MonthStartingDay)  {NumOfWeeks = NumOfWeeks + 1}  Return NumOfWeeks  Function calculate days in a month (month as integer, year as integer)  Year = date(y, time)  Month = date(m, time)  Return date(t, stringtotime(year,month) |

|  |
| --- |
| Class: Customer |
| Forename as String  Surname as String  Email as String  PhoneNo as String  Username as String  Password as String |

|  |
| --- |
| Class: Booking |
| ClientID as Integer  ServiceKey as String  DateTime as DateTime  ConfirmedbyStaff as Boolean |

## 2.10 User Interface Rationale

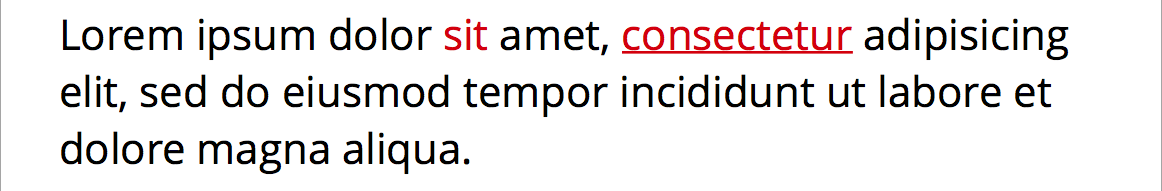
The system is designed to have a modular width, in order to cater to as many kinds of browsers possible. This is made feasible with the use of CSS3, where you can select @media queries to assign different styles depending on the size of the browser. This is convenient as you don’t have to design a completely different layout for different browsers, nor does the user have to zoom in if their browser is too small.

Since this is a web program, it would be pragmatic to create a user interface that people are already adjusted to. Conventions of website interface include a logo that is placed on the upper left, definitive link styling, button functionalities, and visual hierarchy.

The primary colour scheme will be red and white, as this is what the projects client requested. The content font will be black, with headings in a maroon red. Colour can play a big role in user interface design, so I have chosen to use colour keys which can help guide the users in the process of booking and confirming. Colour keys are universally understood and will be used on the calendar so the customer can visually distinguish between the days that are available, closed, fully booked and partially booked. Colour keys are also used elsewhere in the system, such as the registration process and the validation of booking, which will be described in more detail in *2.11 UI Sample of Planned Data.*

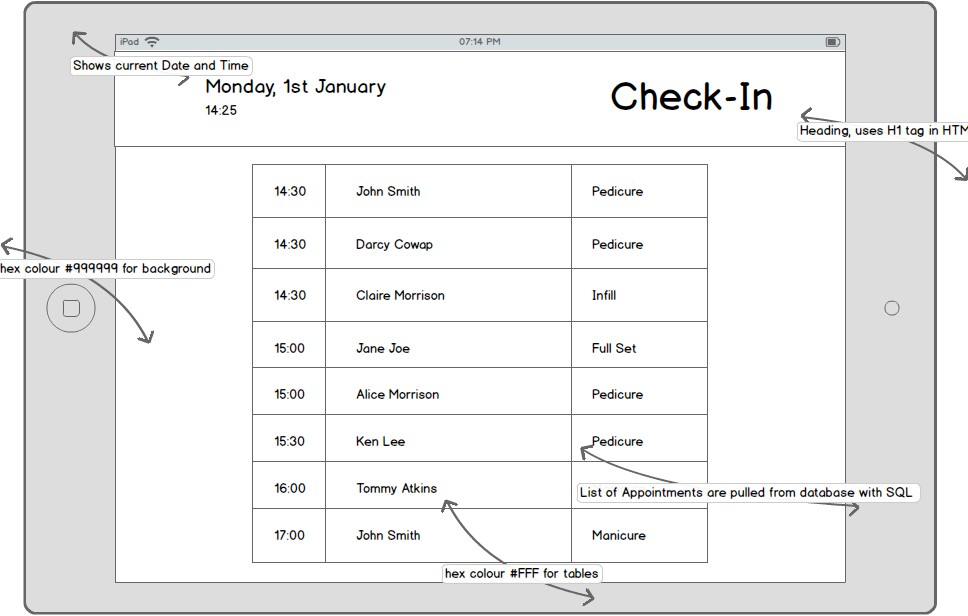
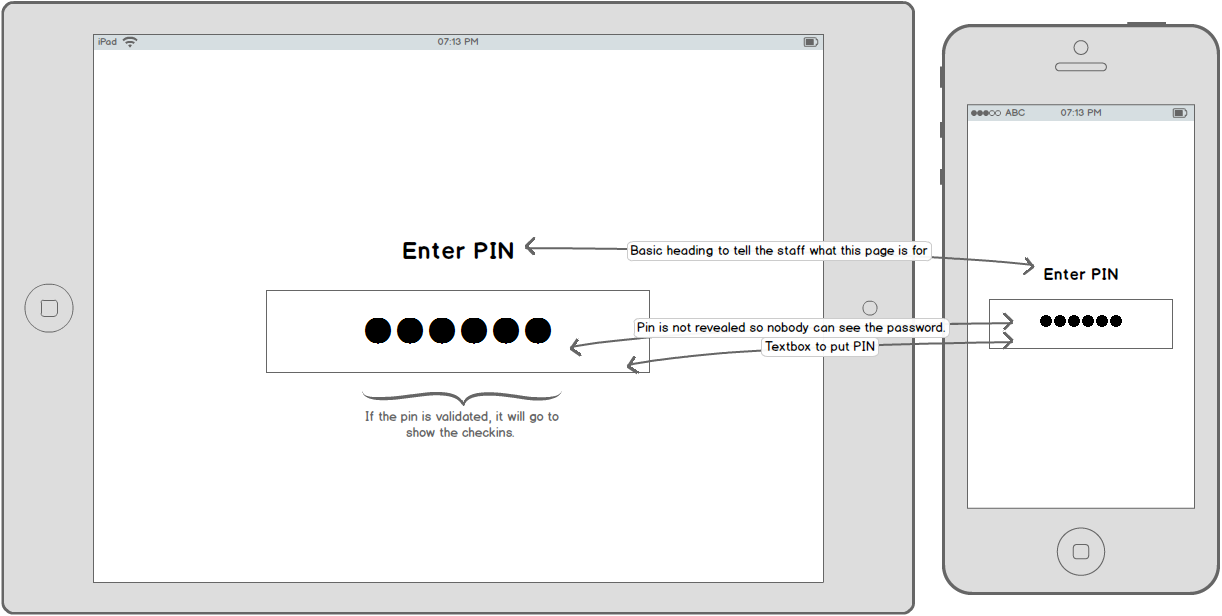
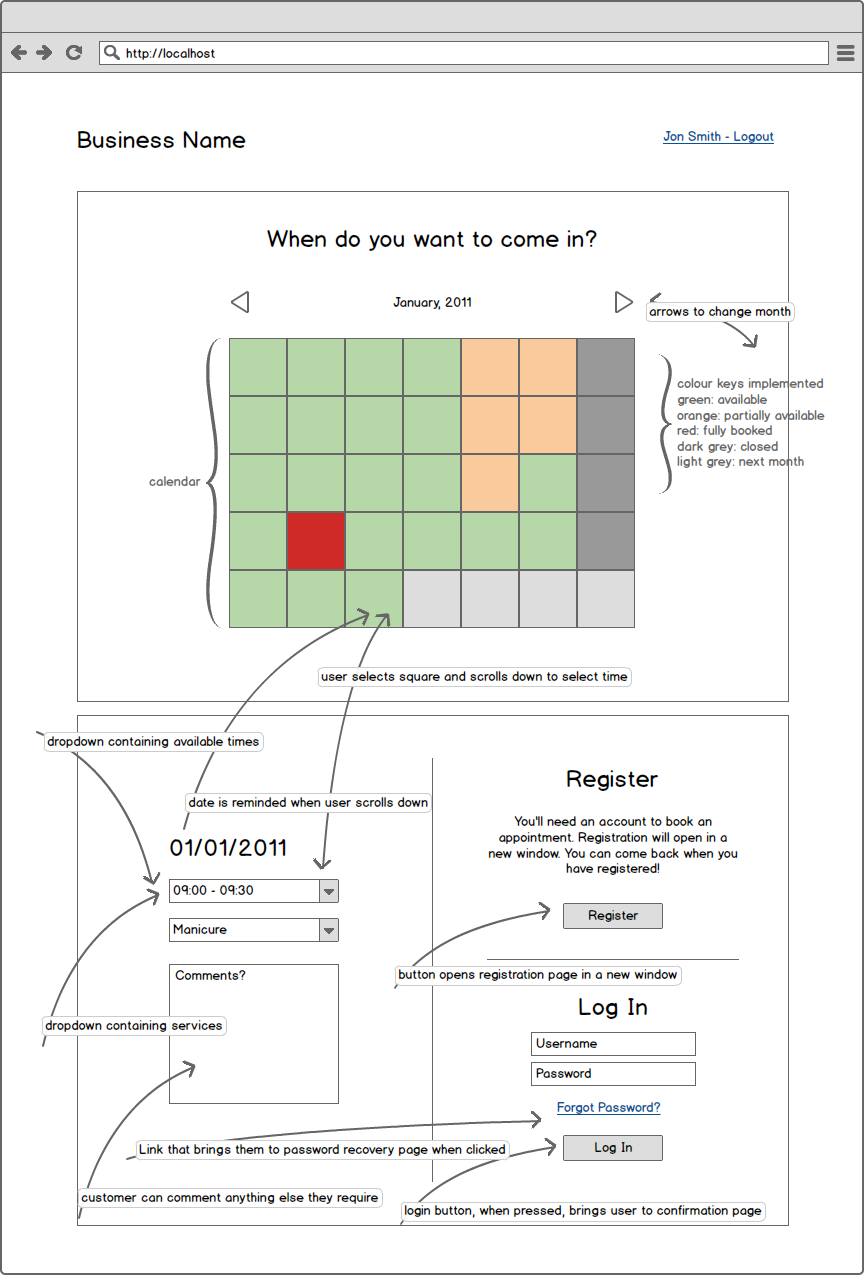
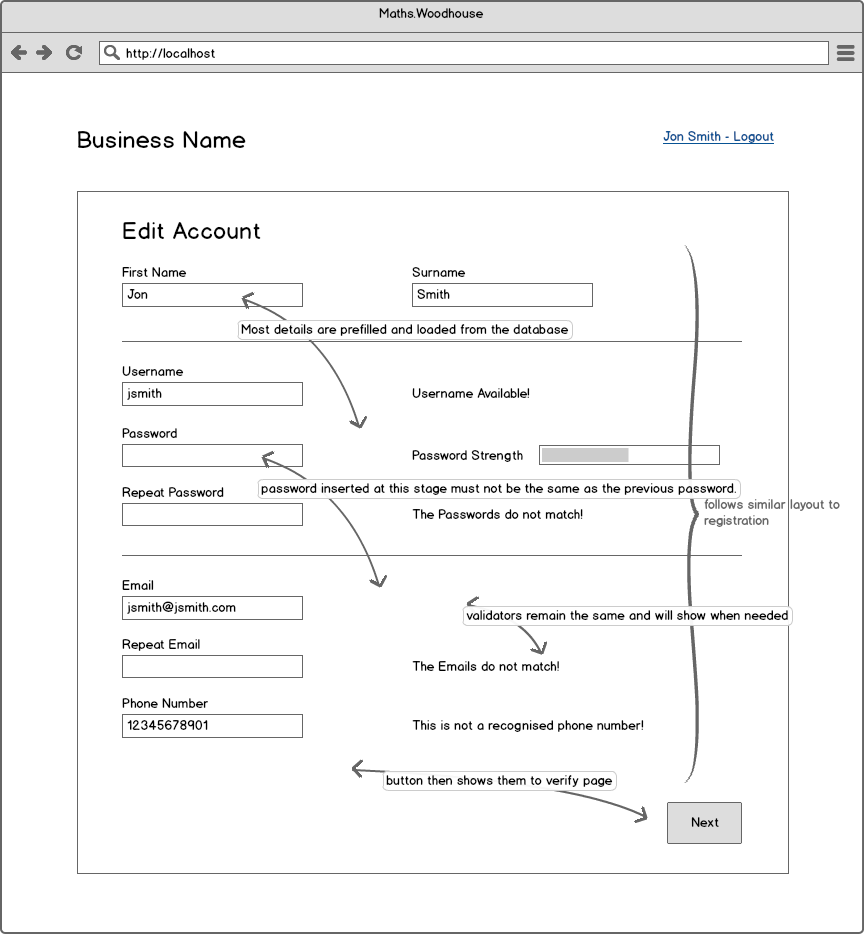
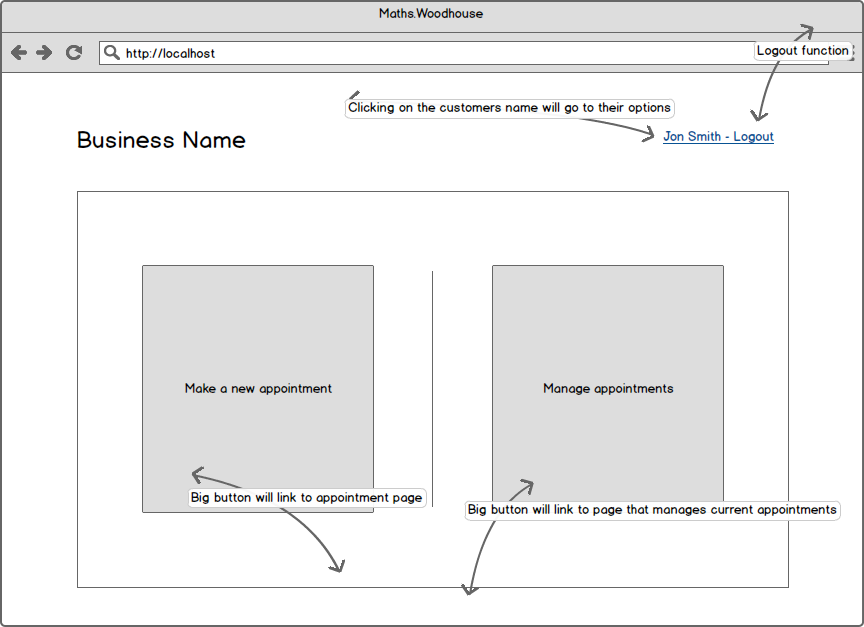
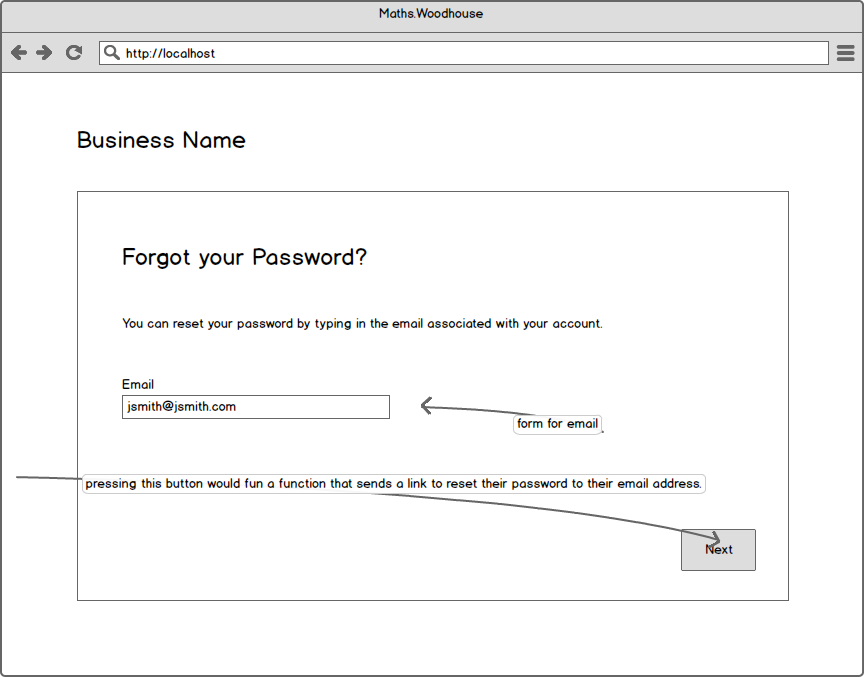
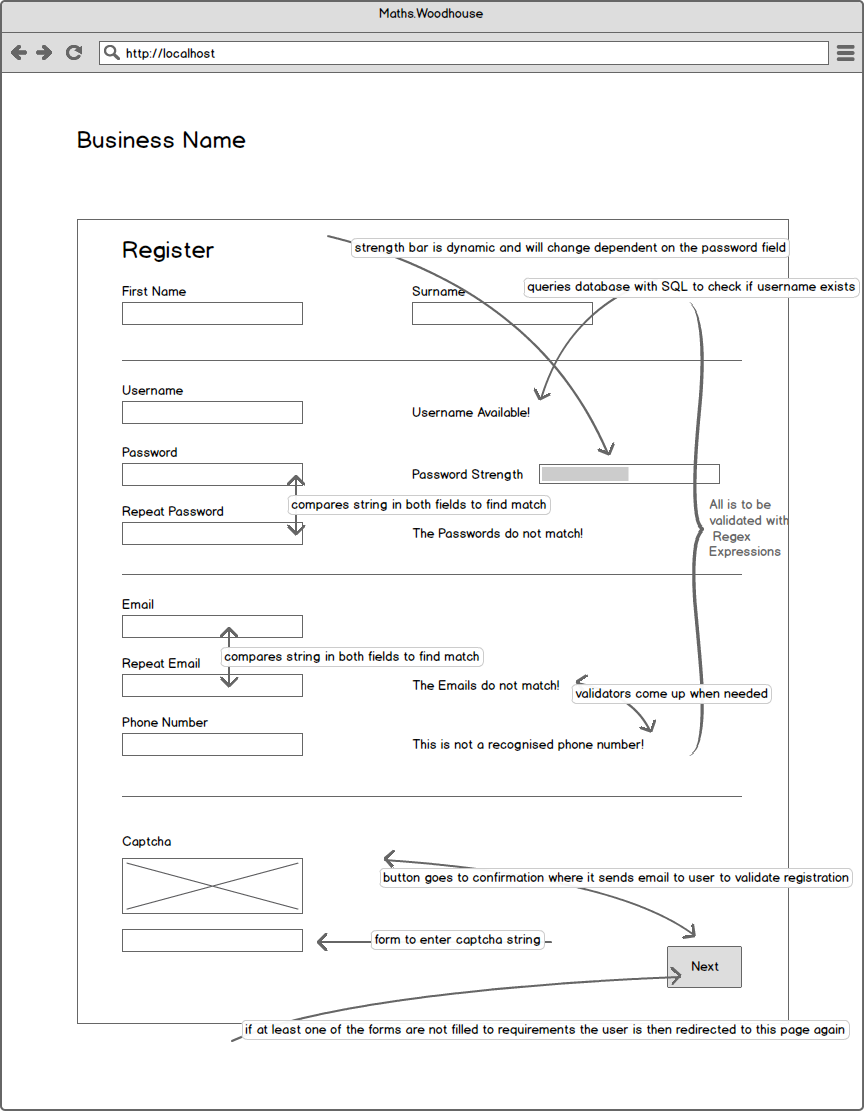
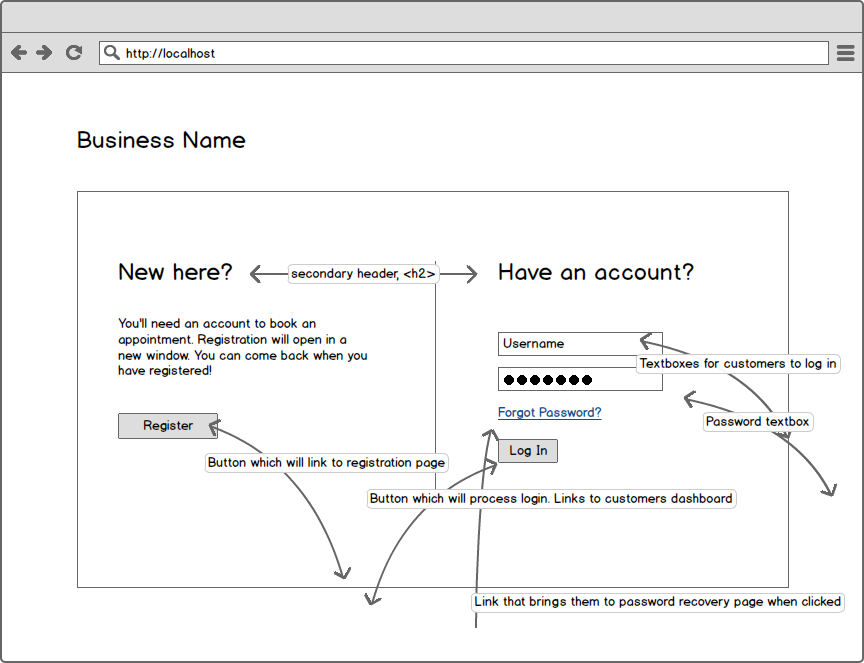
The primary font will be Open Sans, and the fall-back font will be Arial. In the special cases where neither fonts are supported, It will fall-back even further into the default sans-serif. Open Sans will be sourced by Google Web Fonts to decrease the strain of the outgoing bandwidth on the server that hosts the system. The font size will be set in 14px, with a vertical padding between words of 1em. This is chosen for its legibility; the fonts will be readable on a computer monitor as well as portable devices such as smartphones and tablets.

The primary heading font is Montserrat, with its fall-back font being Arial and sans-serif. It will also be sourced by Google Web Fonts. Font size is dependent on the size of the screen it is displayed on. The heading fonts are in red also.

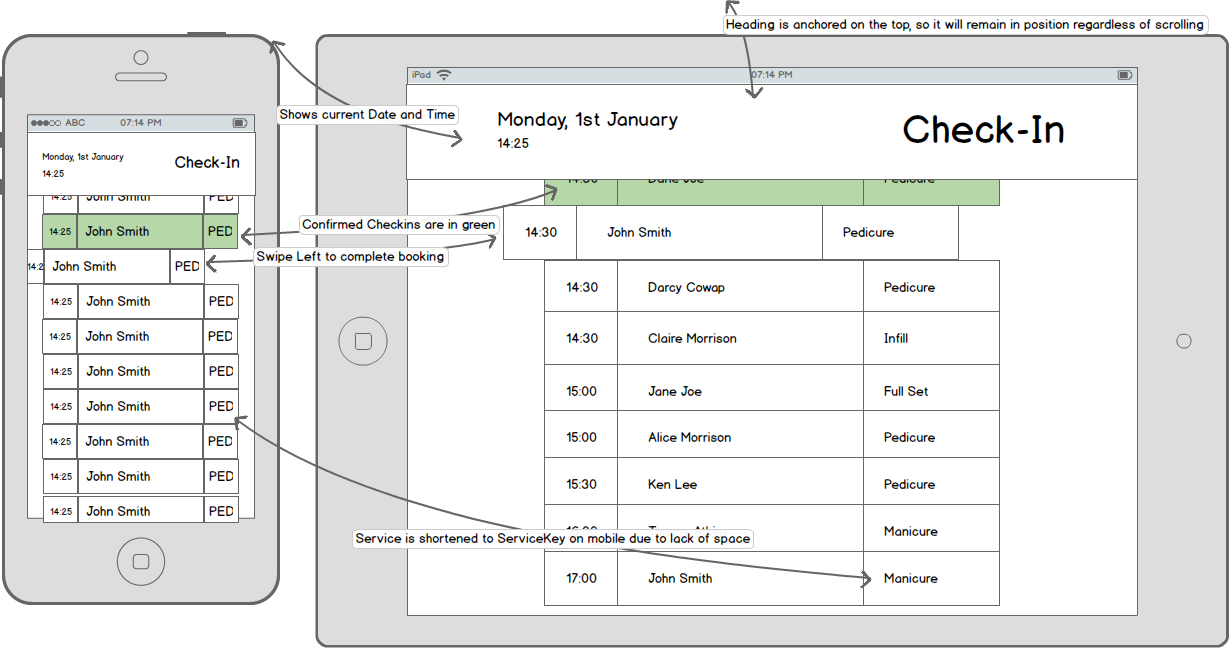
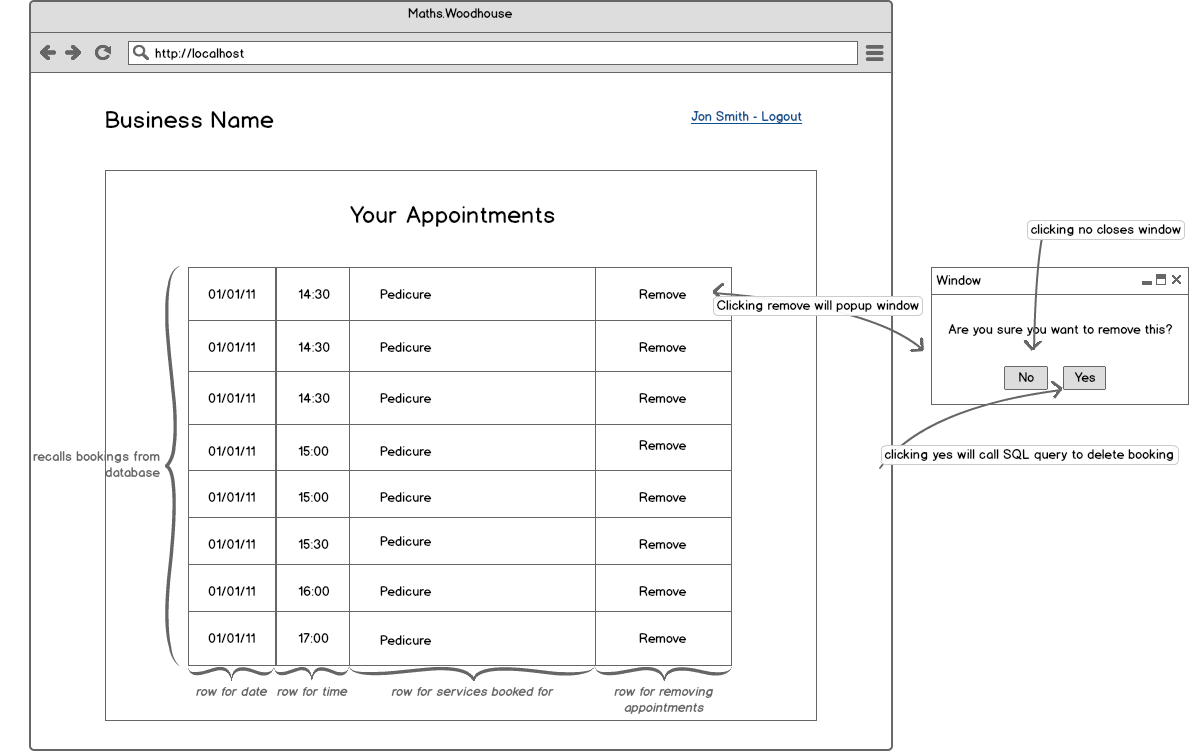
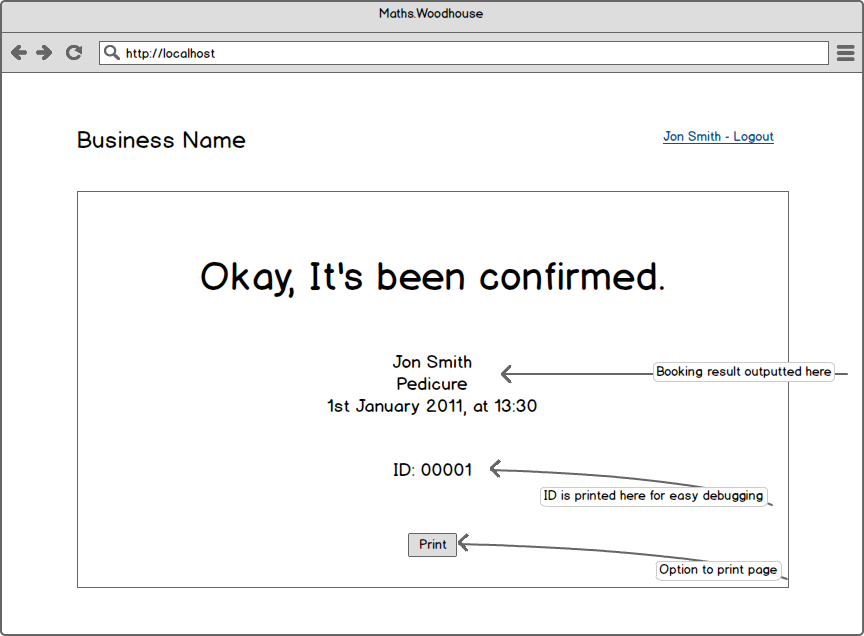
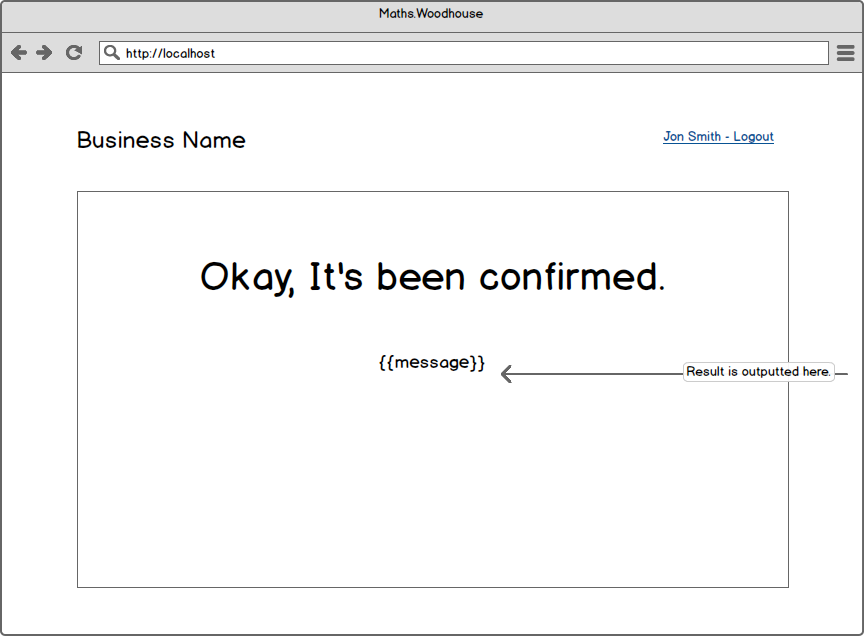
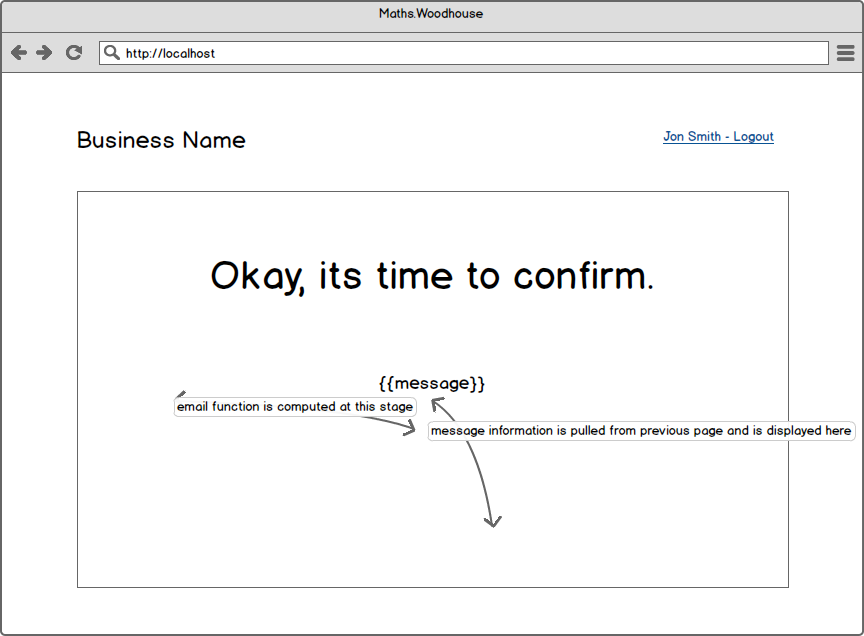


Link styling is made clear through the use of a differentiating colour from the content, which is black. The hyperlinks will be red and, when hovered over, will be underlined.

## 2.11 UI Sample of Planned Data



## 2.12 UI Sample of Planned Output Designs



## 2.13 Descriptions of Redundancy Measures (Data)

Regex is used to ensure that the inputs are correct by looking at the characters in the input string, and checking it contains certain characters. For example, to validate an email address, it must contain a “@” sign and a period “.”. It can also be used to ensure that a password is strong by checking if it has an upper-case character, a lower-case character and a number.

Confirmation popups are to be used where the customer plans to terminate a booking. The popup acts as the medium where the customer can confirm this process, making sure that the process is genuine and not accidental. Since this is not a mission critical idea, and for conveniences sake, it is appropriate to use a popup box instead of other choices such as email verification. If the popup box is not there, it increases the chance of the user accidentally deleting a booking.

Email validation used to confirm accounts and bookings. The use of emails provide a different medium of communication that can help verify if the person is certain they want to carry on with making the booking or making the account.

Data is stored server side, meaning that users don’t have to worry about data loss as there is only one point of failure which is the server. Users only have to worry about input losses such as putting in credentials and bookings.

The use of an uninterrupted power supply is necessary to prevent physical damage to the Hard Drives, which may corrupt data in the case of an electrical power cut. Also, the server has to be always on in order to serve everyone at any given time.

Password can be recovered in the case that the customer forgets it. It requires the user to input their email address that is associated with the account they wish to recover. A link is then sent to that email address which, when clicked, will bring them to a page to reset their password.

Physical backups would be enacted. The owners are prompt to back up the database and the system into a flash drive every night. This way, if the database is ever corrupt, or if there is a disaster within the program, the owners will have a physical backup they can restore from.

## 2.14 Description of Measures Planned for System Sec.

The PIN wouldn’t be digitalised and would be spoken to. The staff are notified in person of the new pin at the beginning of the work day, and if forgotten, they would ask the other staff or Hanh and Hoang the PIN in their own language. This is rarely the case however, as the staff would have to type the PIN multiple times in the day, and through repetition, it is hard to forget the 6 PIN key.

Logins are used to not only distinguish different users, but to secure sensitive information such as email addresses and phone numbers. It is important for data such as this to be protected. As a result, all accounts are required to have a password.

Captcha forms are used to make sure that the customer is an actual person. The purpose of Captcha forms is to prevent bots from creating automated accounts which could bog down the database with redundant information.

In terms of design, SHA-256, the cryptographic hash function, has been used to encrypt password details. This is used in conjunction with a randomly determined salt which fully deters packet interception from protruding hackers. The resulting hash is stored on the database, So if hackers have access into user credentials, They wouldn’t be able to reverse-engineer the hash. Also possibly implemented is the use of SSL, which essentially stops packet snooping in its entirety.

When the customer is requesting to change their password, they have to enter the email that is associated with the account they wish to recover. It is absolutely imperative that a message that identifies that the account exists is not displayed. If this was not enacted, then hackers can identify active usernames. A confirmation message that explains that an email is sent should be displayed to all requests. Then, when the user receives the email, they can explain the next process.

Preventing SQL Injections is possible by preparing statements beforehand and by using parameterised queries. As we are using PDO (PHP Data Objects) to connect to the database, we can force the system to only use real prepared statements and not injected statements that could be parsed by PHP.

One benefit of PHP is that the source code is invisible to everyone who accesses the website through a browser as the code is compiled live, with the results being displayed on the browser. To make it even more secure in terms of protecting the source code, setting file permissions is another way of doing so. I have chosen to set the permissions to 755, where the owner of the files can read, write and execute but everyone else can only read and execute. Hanh and Hoang are the only people who can access the physical files if they need to.

The check-in and the administration side of the system can only be accessed on a local network which is at the shop. The servers firewall is configured to block incoming connections from outside its network to the check-in and administration section of the system.

The physical server that will hold everything is locked away in the store, and it is physically connected to the router by Ethernet. The physical size and the design of the server is discrete and will consequently require a second glance for anyone to recognise what it is. It is stored in a place in the salon that is not disclosed.

## 2.15 Overall Test Strategy

This covers how you are going to apply testing strategies to your system. You must give examples relating to this, not just standard theory. See pages 259 – 261 in the textbook.

BLACK BOX FUNCTIONAL, WHITE BOX STRUCTURAL, DRY RUN. DO ALL THREE TSTING. SAY THAT YOU’RE CONDUCTING. DESCRIBE HOW YOU GONNA CONDUCT IT. USE PAPER, NOT COMPUTER.

TEST DATA, TALK ABOUT THE TEST DATA THAT YOU ARE GOING TO USE. NORMAL, BOUNDARY (EXTREME), ERRONEOUS.

PUT A COPY OF THEIR TEST PLAN.

|  |  |  |  |
| --- | --- | --- | --- |
| TEST NO | TEST DESC | TEST DATA | PREDICTION |
| 1 | Customer is trying to log into the database with the correct details | customer password | System will take them to their dashboard |
| 2 | Customer is trying to log into the database with incorrect details | Customer passw0rd | System will not take then into their dashboard, and will tell them that the credentials are incorrect. |
| 3 | Customer is trying to recover a password by using an email associated with the account they are trying to recover. | customer@email.com |  |
| 4 | Customer is trying to recover a password by using an email that is not recognised by the database. | Cust0m3r@em43l.com |  |
| 5 | Customer is attempting to reset their password with a different password | Password2 |  |
| 6 | Customer is attempting to reset their password with the same password that was forgotten beforehand. | password |  |
| 7 | Staff is attempting to log in with the correct PIN | 123456 |  |
| 8 | Staff is attempting to log in with the incorrect PIN | 123123 |  |
| 9 | Administrators is attempting to log in with the correct username and password. | Admin password |  |
| 10 | Administrators is attempting to log in with the correct username and an incorrect password. | Admin p4dsa23f |  |
| 11 |  |  |  |
| 12 |  |  |  |

# Technical Solution

Whilst this is a key part of your project, it is not documented as a separate section in the report. Evidence will be found by assessors in the System Maintenance, Testing and User Manual.

Annotated Program Listings (add to Appendix) – fully commented, use comments in program and annotations as well & Procedure and Variable Description Lists

Screen Designs (all data capture & output screens) – must have descriptions covering use/purpose, colour schemes, validations, changes from design, button functions

Database Structure – show relationship diagram: explain if there are any single tables of separate linked tables (password/username)

How the technical solution was achieved – bottom-up/top-down design approach? Desrcibed what you did (stages), mention testing was done at each part of development/coding stage. Testing is documented in next section.

See: <http://sjhoward.co.uk/pdfs/computing.pdf>

# Testing

## 4.1 A minimal set of Test Data

Use the template shown in lesson to create a separate Test Data Set. This is more efficient than including test data separately in each test

## 4.2 Details of Individual Tests

using the minimal set of test data above, To be corss-referenced to test plan (Actual Results column)

## 

## 4.3 Expected results for typical data

## 4.4 Expected results for erroneous data

## 4.5 Expected results for extreme data

## 4.6 Samples of Actual Test Runs

Note, just carefully selected samples are required.

‘Annotation’ should explain what is happening in the screen shot.

**Do NOT crop screenshots**, the assessors like to see the screens in the same way as the user does.

## 4.7 Samples of Annotated hard copy showing system

That is, system testing. For example, enter customer and product data, sell a product to a customer and then produce and Invoice for that customer.

Samples only again.

## 4.8 All Samples cross-referenced to Test Plan

So, the hard-copy of Test 1 in the plan should be labelled Test 1 as well.Correction actions from failed test:

Test No., Remedial Action Taken, Retest (refer to screenshots)

# Maintenance

## 5.1 System Overview

From a technical viewpoint, a description of what the system does. Include technical points, e.g. program used, files.

Include a list of the forms and reports in your system, with a description of the purpose.

## 5.2 A sample of Detailed Algorithm Design

Remember, this is what you have done, not what you designed, so **take your code and turn it into pseudocode**.

## 5.3 Procedure and Variable Lists

You might find it helpful to do this by form/report. State the purpose of each.

## 5.4 Annotated List of Program Code

Well-annotated code is good for the following reasons:

It is good practice for producing good technical documentation anyway

It will get you better marks in the system maintenance section

It will probably get you better marks in the Technical Solution since the assessor can more easily identify complex coding.

When you created the code you should have followed programming good practice so the code should already be self-documenting. If you haven’t, do it now!!! See page 266 in the textbook.

Add some clear comments to identify complex routines – this will help the assessor to mark your work.

To present the code, copy it into your report. Add screenshots of forms/reports before each program code section – this again will help the assessor.

User Manual

Contents

You must have a Contents page and page numbers

6.1 Introduction

From the user’s point of view. Think functionality – what does the system actually do?

Purpose of user manual + what does the system do.

dgillooly

5.2 System requirements

6.2 Installation Instructions

Explain how to install, load or start the application. Add screenshots to help the user.

6.3 Using the System

Change the heading above to include your system name, e.g. Using the Overtime Booking System

This section should read like a tutorial.

Start at the beginning!

How does the user get into the system?

Explain the different menu options and then go through each one.

Finish by helping the user logout.

You will be marked in ‘Using the System’ for:

Detailed description of the use of the full system

Samples of actual screen displays, in situ

Error messages and recovery procedures

At an appropriate level for the user

Enables easy use of the system

5.5 Troubleshooting

5.6 Glossary

5.7 Index

User Manual

Read **Section 7.6 Evaluation** in the textbook for more help.

In preparation for the Evaluation you should:

Arrange a time and date for the user to test the system

Complete the user manual

Complete a user acceptance test plan

Let the user test the system

Get feedback from the user – an email, a letter, or if you take notes yourself during the testing then get the user to sign and date.

In this section you evaluate the solution that you have created to solve the problem specified in the Analysis. This is done by looking at the specific objectives that you produced.

The evaluation should be critical and honest. You get no marks for just saying everything is brilliant! You will find your **diary** useful when completing this section.

Contents

7.1 Comparison of Project Performance

Take your numbered list and copy it into this section.

For each objective, state if you met it, and if not, state what has been achieved and why. Then, how effective is your solution in meeting the objective.

Refer to analysis

dgillooly

Use screenshots to show evidence of achieving it/alternative solution.

7.2 User Feedback by Assessor

The Evaluation MUST include feedback from the user. You teacher will contact your user to confirm this so don’t make it up!

Part of any completed project involves you showing the work to your target user. You might choose to sit them down and demonstrate it, then record their feedback directly, or you might choose to send them a copy and ask them to email you any responses. The important thing here is that you get feedback from your user that is definitely theirs. Try and ask them for things that would like you to improve upon. The letter / email that you receive from them should be about half a page to a page long, listing at least the following:

General feedback about how easy the system is to use, talking about all users

How does the system meet their objectives

How easy was the system to set up

Criticisms of what they don't like

Suggestions about how they would like the system to be improved or extended

Signature and date

7.3 Analysis of user feedback

Don’t just restate the feedback from the user. What do their comments mean?

Now you have the feedback, was it good was it bad, does it make you think you met all your objectives? What would you do differently next time?

dgillooly

Briefly state what you did to rectify any issues.

7.3 Possible Extensions/Improvements

This should be based on the evaluation that you and your user have made.

# References

### Email Automation

http://css-tricks.com/sending-nice-html-email-with-php/

<https://www.youtube.com/watch?v=lh1UNGA518s>

<http://stackoverflow.com/questions/17418751/reply-to-sender-php-email>

<http://access2learn.com/tutorial/php/creating-a-contact-page-with-php/>

### Encryption

https://crackstation.net/hashing-security.htm

### Regex

http://www.coderexception.com/0H1m6b1m3PUQSXXU/php-email-validation-function

### Captcha

<http://code.tutsplus.com/tutorials/build-your-own-captcha-and-contact-form--net-5362>

SQL Injection

<https://stackoverflow.com/questions/60174/how-can-i-prevent-sql-injection-in-php>

Calendar

<http://www.startutorial.com/articles/view/how-to-build-a-web-calendar-in-php>