

Computing Practical Project

Scheduling & Booking System

Thien Nguyen

Candidate No. 7461

Centre No. 12290

Woodhouse College

Table of Contents

Analysis	6
1.1 Introduction	6
1.1.1 Background.....	6
1.1.2 Problem Definition	6
1.1.3 The Users.....	7
1.2.1 The Current System Analysis	8
1.2.1.1 Interview.....	8
1.2.1.2 Observation of Existing System	8
1.2.2 Data Flow Diagram of the Current System.....	9
1.2.3 Data Sources and Destinations	10
1.2.4 Entity-relationship diagram of Current System.....	10
1.2.5 Discussion of problems with Current System	11
1.2.6 The Proposed New System Analysis	12
1.2.6.1 User Needs	12
1.2.7 DFD of Proposed New System.....	13
1.2.8 Data Sources and Destinations	14
1.2.9 Entity-relationship diagram and entity descriptions of New System	14
1.2.10 Analysis Data Dictionary.....	15
1.2.11 Data Volumes	17
1.3 Constraints.....	17
1.3.1 Hardware constraints	17
1.3.2 Software constraints	17
1.3.3 Who will be allowed to use the various parts of the system	18
1.4 Limitations	18
1.4.1 Areas which will not be included in computerisation.....	18
1.4.2 Areas considered for future computerisation.....	18
1.5 Objectives.....	19
1.5.1 General Objectives	19
1.5.2 Specific Objectives	20
1.6 Consideration of alternative solutions	22
1.7 Chosen Solution & Justification.....	23
Design	25
2.1 Overall System Design	25
2.1.1 System Flowchart	25
2.2 Description of Modular Structure of System	26
2.2.1 Hierarchy Chart	26
2.3 Definition of Data Requirements.....	29
2.3.1 Database Design (Normalised ER).....	29
2.3.2 Data Dictionary	30
2.4 File Organisation and Processing	32

2.5 Identification of Storage Media.....	32
2.6 Identification of Suitable Algorithms	33
2.6.1 Establishing a connection to the database.	33
2.6.2 Salt and Hashed Encryption.....	33
2.6.3 Email Automation	34
2.6.4 Regular Expressions	34
2.6.5 Captcha	35
2.6.6 PIN Generator	35
2.6.7 Logging Into the System	35
2.6.8 Forgot Password	36
2.6.9 Adding Tables into the System Database.....	36
2.6.10 Paginator	37
2.7 Sample of Planned SQL Queries	38
2.8 Class Definitions	40
2.9 User Interface Design Rationale	41
2.10 UI Sample of Planned Data	42
2.11 UI Sample of Planned Output Designs	50
2.12 Descriptions of Measures Planned for Security & Integrity of Data.....	53
2.13 Description of Measures Planned for System Sec.....	54
2.14 Overall Test Strategy.....	55
Technical Solution	60
3.1 Screen Designs	60
3.1.1 Setup	60
3.1.2 Login	61
3.1.3 Forgot Password.....	62
3.1.4 Customer Registration	63
3.1.5 Dashboard	64
3.1.6 Calendar	65
3.1.7 Confirmation.....	67
3.1.8 Your Appointments	68
3.1.9 Users Information	69
3.1.10 Edit Details	70
3.1.11 Staff Login.....	72
3.1.12 Staff Checkin	72
3.1.13 Administrators Login.....	73
3.1.14 Administrators Dashboard	74
3.1.15 Appointments	75
3.1.16 Calendar	76
3.1.17 Services	77
3.1.18 Users	78
3.1.19 Displaying User Details.....	79
3.1.20 Staff List.....	80

3.1.21 Settings	81
3.2 Annotated Program Listings.....	82
3.3 Procedure & Variable Description	82
3.4 Methods of establishing Database Connections	82
3.5 Database Tables (Final)	82
3.6 How was the technical Solution Achieved?	83
Testing	83
4.1 Test Table	84
4.2 Test Evidence.....	89
Maintenance	143
5.1 System Overview	143
5.2 Annotated List of Program Code.....	144
5.3 Procedure and Variable Lists	144
5.4 Samples of Algorithm Design	163
Appraisal	171
6.1 Objectives.....	171
6.1.1 Specific Objectives.....	171
6.2.1 General Objectives.....	205
6.2 User Feedback by Assessor.....	207
6.3 Analysis of User Feedback	208
6.4 Possible Extensions/Improvements.....	209
Appendix	210
7.1 Interviews & Responses.....	210
7.1.1 First Interview Report	210
7.1.2 Second Interview	212
7.1.3 Client's Feedback	213
7.2 Annotated Program Listings	214
7.2.1 Queries	214
7.2.2 User Defined Classes/Objects	218
7.2.3 Communication Protocols.....	225
7.2.4 Extensive Range of I/O & Processing.....	227
7.2.5 Graph	230
7.2.7 Regular Expressions.....	232
7.2.8 Preventing SQL Injections.....	232
7.2.9 Encryption	232
7.2.10 Paginator.....	233
7.2.11 Alphabet Sort.....	234
7.2.12 PIN Generator.....	235
7.2.13 Data Structures	236
7.2.14 Creation of Objects at Runtime.....	239
7.3 Source Code	241

References	313
Encryption.....	313
Regular Expressions	313

Analysis

1.1 Introduction

1.1.1 Background

The Organisation, Hanh's Nail & Beauty, is a family operated Nail Salon specialising in Waxing and Nail Care services such as Pedicures and Manicures. The salon 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 salon garners around five to seven thousand customers. It opens from 10AM until 8PM and staff take breaks at different periods to ensure that the store is still open during lunch time. Prices range from the simple hand and polish costing £15, to the Shellac Pedicure costing £35. Occasionally customers would come in for multiple services. The most commonly asked for service is a manicure costing £24.

The store, whilst small, has two floors, consisting of six nail benches and two Massage Chairs. Upstairs houses a waxing bench. The number of workers varies depending on the situation, with its median number being four workers. The salon receives more customers on weekends, bank holidays and also during the appropriate weather. The store is especially busy when it is opened near a seasonal event such as Christmas or the New Year.

The owners and my client, Hoang and Hanh Nguyen, deal with the booking system. This currently consists of an 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.

Bookings can be made any time in advance so long as there is an available day. Customers who are checking-in have a 10 minute timeframe, otherwise it will be presumed that the customer is not available. At this case, any customers who are waiting on the day will be served instead. Customers who wish to cancel their booking have to do so by either phone or walking to the store to notify, and it has to be done at least a day before the date of appointment.

1.1.2 Problem Definition

The current booking system is completely manual. It consists of an appointment book where each page consists of timetables, where customers can be allocated into a slot. There is only one copy of the appointment book which is only dealt with by Hanh. Customers would come into the shop and ask for an appointment whenever they can if the shop is busy, and all customers would be referred to Hanh. Only

she will note down the appointment. There is no other way to contact the salon. With this, customer flow becomes frustrating. Customers have to wait awkwardly to wait for Hanh whenever she is available, making the customer frustrated as a result of wasted time.

The client proposes an automated system that can help the salon in the process of handling appointments. It is important that the system is online based, where the customer can interact with the system to make a booking, allowing the staff to focus on the customers inside the salon.

1.1.3 The Users

There are three different users.

The Staff

The staff serve the purpose of servicing the customer with their requested service. The staff themselves have proficient IT experience, leading with their resentment to using a manual system.

The Business Owners (Administrators)

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. This category of users similarly has a below-average IT experience. They also deal with servicing the customers.

The Customer

The customers, are the people who would submit their information into the system, as it is used to make appointments. They would be able to book appointments whenever they can, and however they can. 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. Their proficiency in interacting with IT will vary significantly, so it is safe to assume that everyone would have at least a fair comprehension on using computers.

1.2.1 The Current System Analysis

1.2.1.1 Interview

An interview was conducted on the 5th of September 2014, with Hanh and Hoang. 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. A summary of the interview can be found in the appendix.

1.2.1.2 Observation of Existing System

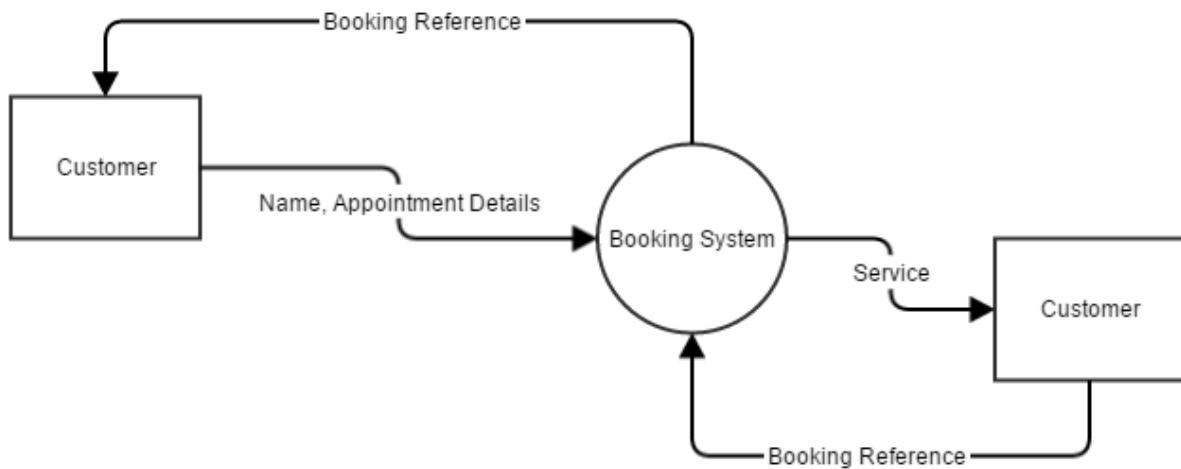
I have chosen to shadow the current users of the current system I could have a better understanding of how the users interact with it on a daily basis, and identify the problems they face whilst doing this.

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 whether 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.
4. The customer returns to the store at the designated time for their appointment.
 - 4.1. The customer notifies the staff of their appointment.
 - 4.2. The customer checks the details the customer gives such as their name and their booking date/time.
5. The customer is then serviced.

1.2.2 Data Flow Diagram of the Current System

Level 0:



The current data flow diagram is simple. The diagram is read left to right, where the customer would give their name and their appointment details. The appointment details include the following:

- The Service they require
- The Time of the Cooking
- The Customers Phone Number

The customer is then given a receipt that describes their booking. This is referenced in the diagram as “Booking Reference”.

When the customer returns, they would give the staff the booking reference to confirm their booking, and in return they are serviced per requirements set on the receipt.

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
Customers Name	Customer	Appointment Book	This is stored so the staff can refer to the customer personally.
Date & Time	Customer	Appointment Book	This is used so the staff can prepare accordingly in relation to the customers service.
Phone Number	Customer	Appointment Book	This is stored in the event that the customer is late. The staff would use this in order to contact the customer, asking that they can make the appointment or not. If the customer can't make it, then the booking is nulled and the slot would be then used to service a customer who is simply waiting for an available space.
Service	Customer	Appointment Book	Similarly with the date and time, the service is stored so the staff can prepare accordingly to what the customer requires.

1.2.4 Entity-relationship diagram of Current System

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 can request for many services, but each service is going to be different, or in this case, unique to that specific customer.

1.2.5 Discussion of problems with 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.

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.

There isn't an easy way for each staff to have their own copy of the appointment book, and if it was theoretically possible there isn't a way to prevent double bookings.

The booking process is not efficient 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 in order 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.

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 inefficiencies such as the handling appointments in the one book, which also accounts for the invoices and weekly revenue.

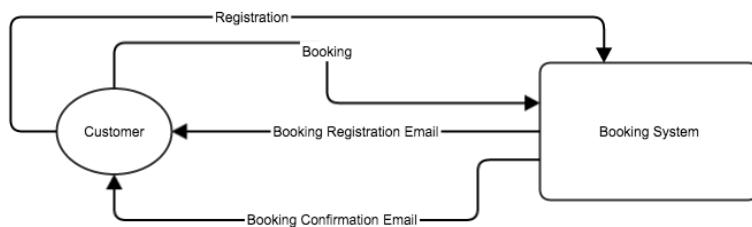
1.2.6 The Proposed New System Analysis

1.2.6.1 User Needs

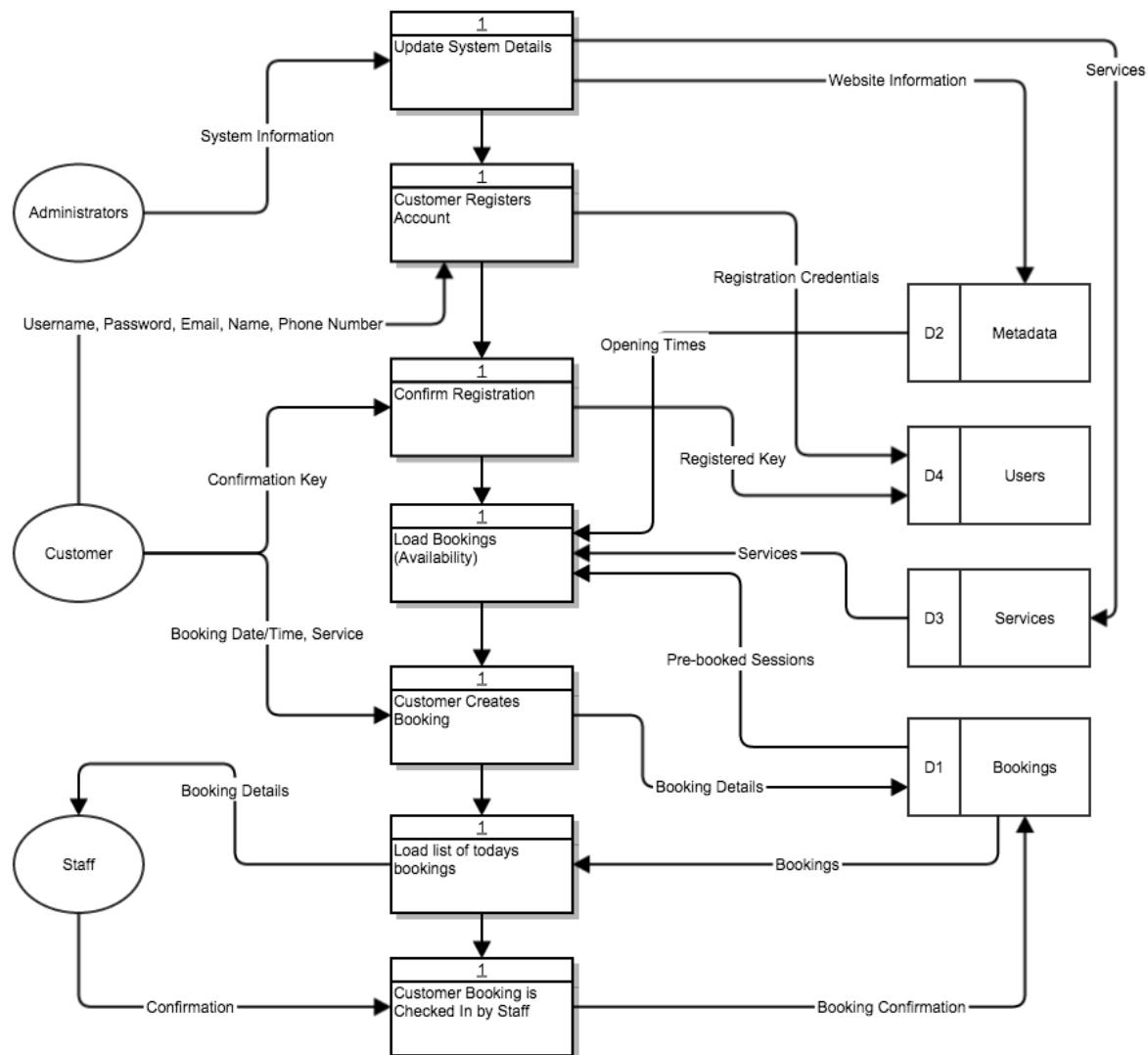
- The Second interview heavily emphasises that the new system must be able to be accessed on a phone.
- The Second interview specifies that the design must be clean.
- The Second interview specifies that the system must be able to display analytics.
- The Second interview specifies that the booking process must be intuitive enough to deter time wasters and bots, but it should be simple enough that it wouldn't annoy potential customers.
- 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.
 - 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.
 - Most importantly, the new system should allow the user to 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.

1.2.7 DFD of Proposed New System

Level 0:



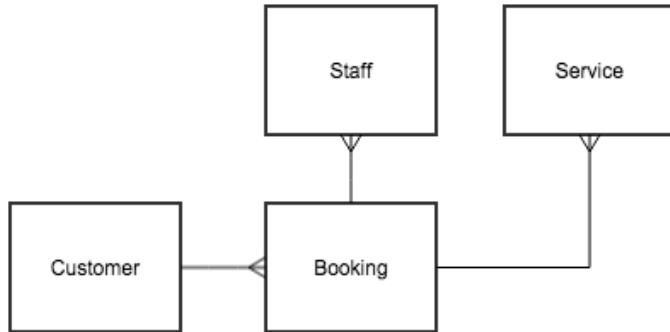
Level 1:



1.2.8 Data Sources and Destinations

Source	Destination	Description
Customer	Users Table in the Database	Customer information which is collected from the users input.
Customer	Bookings table in the database.	Customer Order details when they are booking an appointment.
Staff	Database	Staff used to verify validate booking information.
Bookings Table in the Database	User's Email Address (located in Users table)	Used to send email to customer to confirm bookings, validate accounts and such.

1.2.9 Entity-relationship diagram and entity descriptions of New 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, and each booking can only have one customer linked to it. 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

User						
Attribute	Field Purpose	Length	Type	Rules	Example Data	Validation
ID	Makes each entry unique and identifiable	11	Integer	Automatic Increment	1	Not Null
Username	Defines user.	25	String	letters, numbers and underscore only	skadir	Regular Expressions, Check if username is available
Password	Validates authentication of user, A hashed representation of the password.	256	String	Must contain a capital, a lowercase and a number	E\$oD%hAcs=Urm}	Not blank
Forename	To identify the user by its Firstname.	50	String	First character is capitalised	John	Not Blank, Must not include Numbers
Surname	To identify the user by its surname.	50	String	First character is capitalised	Smith	Not blank, Must not include numbers
Email	Serves as a directory to email to send forgotten passwords to.	100	String	Must contain "@" and ":"	<u>jsmith@woodhouse.ac.uk</u>	Must contain "@" and ":" (Regex)
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	String		1	Must not include numbers
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	Not Blank

Scheduling & Booking System

PIN	Unique PIN that the staff will use to log in.	5	Integer	Numbers Only	12345	Must not have duplicates.
Forename	To identify the user by its Firstname.	50	String	First character is capitalised	John	Not Blank
Surname	To identify the user by its surname.	50	String	First character is capitalised	Smith	Not blank
Email	Serves as a directory to email to send forgotten passwords to.	100	String	Must contain "@" and ":"	jsmith@woodhouse.ac.uk	Must contain "@" and ":"

Service						
Attribute	Field Purpose	Length	Type	Rules	Example Data	Validation
ID	Makes each entry unique and identifiable	11	Integer	Automatic Increment	1	Must be unique
Name	Type of service	25	String	letters, numbers and underscore only	skadir	Must not have duplicates.
Description	Describes the service offered	NA	Text	Must not contain illegal characters.	This is a sentence. This is another sentence.	

Booking						
Attribute	Field Purpose	Length	Type	Rules	Example Data	Validation
ID	Makes each entry unique and identifiable	11	Integer	Automatic Increment	1	
CustomerID	Type of service	25	String	letters, numbers and underscore only	skadir	Must not have duplicates.
ServiceKey	Lists staff who can offer this service	256	String	Is an Array	E\$oD%hAcs=Urm}	Not blank
StaffID	Identifies the Staff that will service the customer	11	Integer		2	Not blank
DateTime	Specifies the date and time of the appointment		Date	Is set in date format	10/10/2010 10:10	Must be set in date then time. DD/MM/YYYY HH:MM

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. In order to future-proof the system, data allocation for the system would be set at 10GB. If necessary, more data can be allocated.

The new system will not contain data from the current system, so importing current data wouldn't be an issue here in terms of space.

1.3 Constraints

1.3.1 Hardware constraints

It's 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 capable processor, and should come with a sufficient amount of RAM to make use of the standard 64-bit architecture.

If we were to pinpoint the specifics of hardware, it may need to have an equivalent of a quad core processor. 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 shouldn't be a problem. Many modern computers have sufficient processing horsepower to manage a small business such as this.

It is 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.

In order to connect to the system, smartphones or tablets could serve as the terminals. Modern tablets and smartphones have to be sufficiently fast in order to streamline the booking process.

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, which includes the clients computers.

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. This is only necessary however in the event that the client wishes to move the system into their own server. Otherwise online web hosting systems will have server programs preconfigured.

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 not be a concern. The majority of the browsers on smartphones and tablets are HTML5

Optimised, 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

Hanh and Hoang, the clients, will have access to the administrators dashboard, which allows them to manage the settings on the website, and is able to modify details on the website metadata and information to the database.

The staff will be able to have access to the check in side of the system. This is necessary as they would be the people who would verify the customers as they come in for their booking.

The customers will be able to access the front end of the system. This side would allow them to book their own appointments as well as manage their appointments. The customers will be able to access this side of the system using the credentials that they have created.

The user(s) show 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 use-friendly and intuitive as it could. A guide can be embedded onto the system and can be disabled per user using the accounts metadata.

1.4 Limitations

In the event that a off-shore server is needed, the client will be able to pay for the equipment needed to run the system. Although the development of the system is free, An off-shore server would be needed to finalise the system.

One of the limitations of the system is time, of which I intend to try and complete the implementation by 1st February. This leaves a sufficient amount of time to create other necessary items for the system such as the User Manual, before handing the system over to the client.

1.4.1 Areas which will not be included in computerisation

Areas that will not be included is the staff allocation to bookings. This feature would make a lot of complexity issues that would prolong the creation of the new system. This will be allocated manually by the staff themselves when they check in the customer, as the Staff are trained to work all the apparatus in order to service everything.

1.4.2 Areas considered for future computerisation

Areas that may be included is the ability to invoice the customer automatically, and to finalise 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 they are paid upfront, but they also don't have to deal with transaction issues that could slow down the process such as dealing with change. 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.

Another idea is a option to buy gift experiences. This may be implemented in the use of a digital code that allows them to be serviced for free or at a discount. This may be included if there is available time in the implementation of the project.

1.5 Objectives

1.5.1 General Objectives

1. The new system should be an online booking system, which is accessible anywhere by anyone, as long as there is an Internet connection.
2. The system must be able to be accessed by not only the client, but their customers and staff.
3. The new system must be better to use than the current system.
4. The system must be able to be accessed on a phone.
5. The system must be able to allow the customer to book an appointment.
6. The system must be able to allow the staff to check in the customers when they arrive.
7. The system must be user friendly.
8. The customer must be able to see what periods are available for them to book an appointment.

1.5.2 Specific Objectives

#	Type	Objective	Description
1	Input	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.
2	Input	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.
3	Input	Updating customer details	The customer should be able to update their credentials, such as their name and password.
4	Input	Inputting customer details	The customer should be able to register an account using their name, phone, email and password.
5	Input	Checking-in customer	The staff must be able to check in the customer in order to validate the booking process.
6	Input	Login	The staff and the customer must be able to log into their respective control panels.
7	Output	Load available timings for appointments	The system should be able to get information from the database, calculate availability, and display it for the secondary user to see.
8	Output	Confirmation Message	The customer must be notified that the booking has processed successfully.
9	Output	Password Validation	When registering, a password strength meter shows how strong the password is.
10	Output	Display User on Login (Dashboard)	The user credentials should be displayed when the primary user logs into the dashboard.
11	Output	Error Messages	The system should output error messages relating to the problem that has occurred, such as being unable to log in or notifying the customer for invalid input values.
12	Output	Updated Information	Updated information should be displayed and the user should be notified of a change.
13	Output	Showing User Information	The system should be able to display user information on a page that allows them to change credentials.
14	Output	Show Query Results	The system must be able to display the sorted results of queries. The queries must have the option to access multiple tables in unison. The list of queries include: Showing booking analytics. Displaying all bookings. Displaying all users.
15	Processing	Identify customers uniquely	The system must be able to uniquely identify each customer.
16	Processing	Identify bookings uniquely	The system must be able to uniquely identify each booking.

Scheduling & Booking System

17	Processing	Identify staff uniquely	The system must be able to uniquely identify each staff.
18	Processing	Check Availability	The system must be able to check if a period is free for an appointment.
19	Processing	Cookies & User Sessions	The system must be able to separate the cookies depending on the user using data structures (such as 2D arrays) and use them for user sessions.
20	Processing	Password Validation	When registering, the system must be able to calculate the strength of the password.
21	Processing	Sort Bookings	The system must be able to sort bookings depending on what each part of the system requires.
22	Processing	Hash Password	In order to protect the passwords of customers and administrators, the system must be able to hash passwords.
23	Processing	Ensure that the customer is a real person	The system must be able to ensure that the customer is indeed a customer by using a CAPTCHA.
24	Processing	String Matching	The system must be able to validate data using pattern matching sequences.
25	Processing	Queue Error Messages	The system must be able to queue up all related error messages before displaying them in the order they were sent.
26	Processing	Error Labels	In the event that the user misses out on something, they are notified with a friendly reminder to finish the inputs before continuing.
27	Processing	Email	The system will automatically notify users for the following actions: <ul style="list-style-type: none"> - Booking an Appointment - Finalising their Registration - Resetting their Password - Informing them of a new Password - Informing them of a new PIN (Staff)
28	Processing	Mathematical Calculations	The system should be able to use mathematical calculations to enhance the usability of the system. This will include the following: <ul style="list-style-type: none"> - Check the booking thresholds - Calculate the Booking Slots
29	Processing	Recursive PIN Generator	The system must be able to generate a pin that recurs in the event that the pin generated is already used.
30	Processing	Staff Search	The staff must be able to search for customers by their surname or forename.
31	Storage	Store Customer Details	The database must be capable of storing customer credentials.
32	Storage	Load Customer Details	The system must be able to load customer details from the system.
33	Storage	Store Staff Details	The database must be capable of storing staff details, such as their forename and surname.
34	Storage	Store Bookings	The database must be capable of storing bookings and their associated variables.
35	Storage	Store Services	The database must be capable of storing services, and their related information.

36	Security	Encrypting Important Details	All passwords and Staff PIN's will be securely encrypted using hashes alongside modern cryptographic methods such as MD5 or SHA.
37	Security	Password Strength	The system will attempt to insist users to use a strong password in order to improve the security of the system.
38	Security	Timeouts	In the event that the user is inactive for a long period of time the system must be able to time out the user, making them log in again. For the staff, it is in 60 seconds, the administrator 10 minutes, and the customer 10 minutes.
39	Security	Preventing Code Injection	The system must be able to sanitise inputs in order to prevent unauthorised or malicious methods to be used in effect against the system, such as SQL injections. This is done in order to maintain the integrity of the database.
40	Security	Access Rights	<p>The administrator must be able access:</p> <ul style="list-style-type: none"> • Viewing the Booking Statistics • Managing the Appointments • Managing the Calendar Settings • Managing the Customers • Managing the Staff <p>The staff must be able to access:</p> <ul style="list-style-type: none"> • Customer's Checkins <p>The customer must be able to access:</p> <ul style="list-style-type: none"> • Booking an Appointment • Viewing their current appointments • Managing their details

1.6 Consideration of alternative solutions

System Specific Software

One solution may be to create a piece of system software that requires installing. This would require the user to download the software using the internet, and install the program from there. From a technical standpoint, the software would perform the fastest, and it would provide the most native software 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.

Improved Appointment Book

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.

Prebuilt Special Purpose Software

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

Keeping the Current System

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. To Recap, the current system is based on paper books that store appointment systems. If we were to keep the current system, then at least the staff will be used to the workflow of this system.

1.7 Chosen Solution & Justification

I have chosen to develop the system using a web program 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 centralised into a database.

Client

This benefits the client as they can intuitively glance at the information of booking details. Also, as a result of this program they are able to have pre-calculated statistics on the booking.

This also benefits the business owners as they can manage who has access to the system, and they don't have to worry about data being physically lost.

User

The customer can book in their own time, and check without contacting the store. This is especially useful as the staff can pay attention to providing the customers at the store what they need.

Skills

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.

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.

Using this solution means that the customer and the staff won't have to install anything to access the system, ultimately making it easier to access.

Data Volumes

This solution proves useful as details of customers are not repeated in this system, and you can recall the same information multiple times.

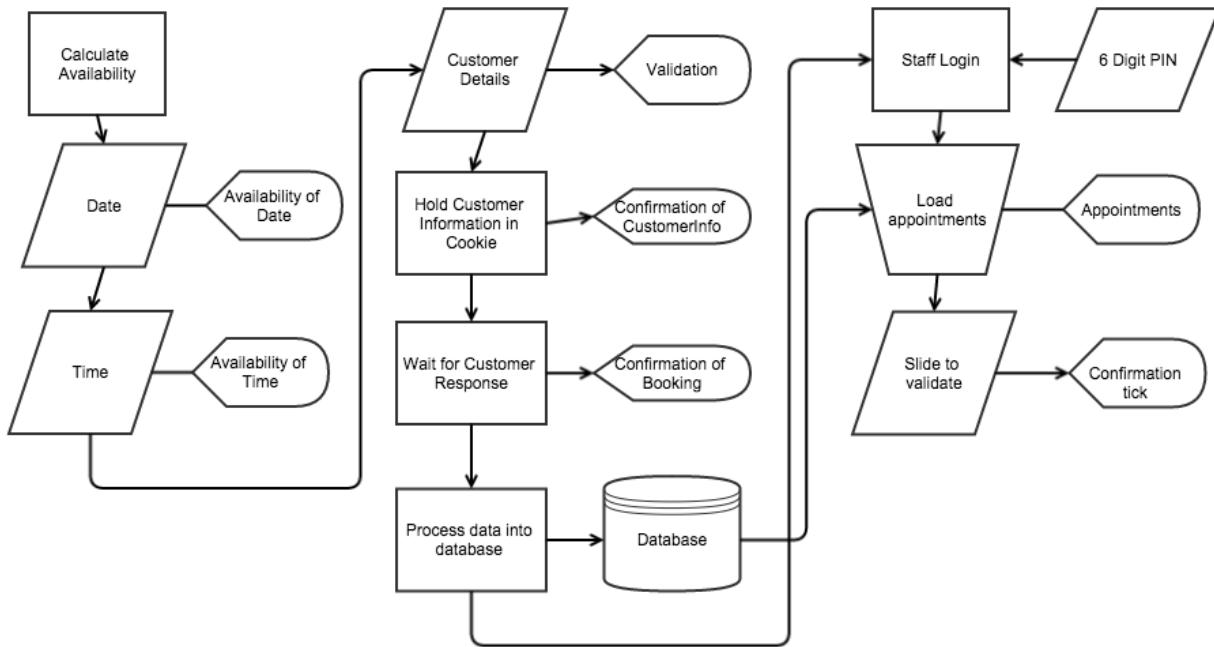
Using this solution is cost efficient compared to the current system. If they were to continue with their current system, they would have to purchase new books which have a fixed price. With digital storage, the price of a gigabyte becomes ever smaller, not to mention the density of the storage medium will be larger.

In terms of physical storage, it would benefit the Client greatly as they don't have to worry about dealing with storage concerns.

Design

2.1 Overall System Design

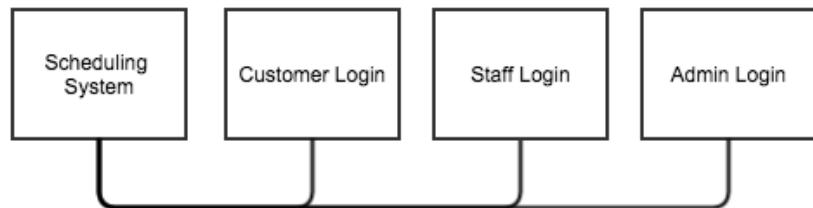
2.1.1 System Flowchart



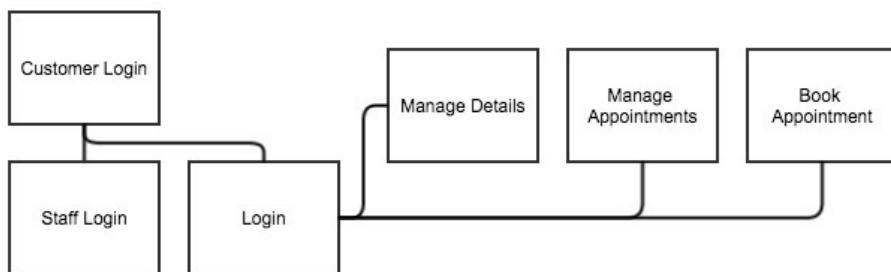
This flowchart represents how the system will operate. The majority of this process is automated, however it relies on input from the staff and the customers in order for this system to work. The flow starts with the customer choosing a date and time for the booking, followed with the rest of the credentials 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.

2.2 Description of Modular Structure of System

2.2.1 Hierarchy Chart



The scheduling system is split into three partitions, the scheduler, the staff login and the clients (Hanh & Hoang's) login. This is split for the three different categories of users. The customer can only access the scheduler, the staff can access the scheduler and the Staff Login, and the Hanh and Hoang have full access into the system.

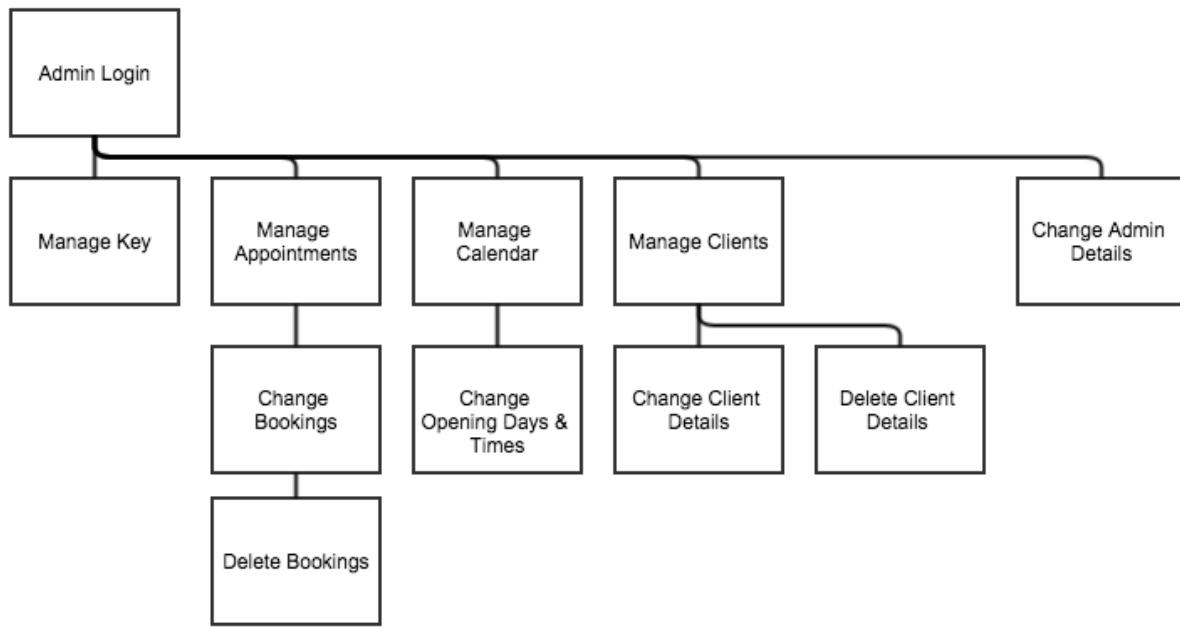


The customer will be able to book an appointment, register an account, and log in. When they have logged in, they will be able to manage their appointments and modify their details that are stored in the system.



For the staff, they only need access to an appointment list in order to check in the customers, ready to service them. This is only available when they have successfully logged in.

Scheduling & Booking 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.

IOPS Table

Input	Output	Process	Storage
Username	Confirmation Post	Validate customer details	Customer Table
Password	Registration Validation	Save customer details into database	Admin Table
Email Address	Email Post	confirm booking	Staff PIN
Phone Number	Completed Booking	filter availability	Bookings Table
Captcha	Confirm Change	Encrypt password details	Metadata
Forename	List of Check-ins	Encrypt username	
Surname		Confirm Check-in	
Activation Code		Generate Daily Staff PIN	
Service Type		Send Automated Emails	
Service Price		Regex Validate	
Service Description		Change Password	
Booking Date			
Booking Time			
Booking Comments			
Booking Confirm			
Registration Confirm			
List All Attributes			

2.3 Definition of Data Requirements

2.3.1 Database Design (Normalised ER)

UNF

Booking (Users.ID, Users.username, Users.email, Users.password, Users.username, Users.phoneno, Users.Confirmed, Booking.date, booking.time, Service.ID, Service.Type, BookingConfirmed, Comments, Service.Description, Service.Cost)

1NF

Users (UsersID*, UserName, Mail, Password, Username, PhoneNo, Confirmed)

Booking (UserID, BookingDateTime, ServiceID, ServiceType, BookingConfirmed, Comments, ServiceDescription, ServiceCost, StaffID, StaffForename, StaffSurname)

2NF

Users (UsersID*, UserName, Mail, Password, Username, PhoneNo, Confirmed)

Booking (UserID, BookingDateTime, ServiceID, BookingConfirmed, Comments, StaffID, StaffForename, StaffSurname)

Service (ServiceID*, ServiceType, ServiceDescription, ServiceCost)

3NF

Booking (UserID, ServiceID, StaffID, BookingDateTime, ServiceType, BookingConfirmed, Comments)

Users (UsersID*, UserName, Mail, Password, Username, PhoneNo, Confirmed)

Service (ServiceID*, ServiceType, ServiceDescription, ServiceCost)

Staff (StaffID*, StaffForename, StaffSurname, StaffPIN)

PRIMARY KEYS INCLUDE ASTERISKS 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 (client.username, 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.

2.3.2 Data Dictionary

Client						
Attribute	Field Purpose	Length	Type	Rules	Example Data	Validation
Username	Defines user.	25	String	letters, numbers and underscore only	skadir	Must not have duplicates.
Password	Validates authentication of user	256	String	Must contain a capital, a lowercase and a number	E\$oD%hAcs=Urm{}	Not blank
Forename	To identify the user by its Firstname.	50	String	First character is capitalised	John	Not Blank
Surname	To identify the user by its surname.	50	String	First character is capitalised	Smith	Not blank
Email	Serves as a directory to email to send forgotten passwords to.	100	String	Must contain "@" and "	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	String	letters, numbers and underscore only	skadir	Must not have duplicates.
Cost	defines the price of the service	2	Integer	must be a number	21	PHP, check if input is an integer

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.

Scheduling & Booking System

Password	Type of service	25	String	letters, numbers and underscore only	skadir	Must not have duplicates.
-----------------	-----------------	----	--------	--------------------------------------	--------	---------------------------

StaffPIN						
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.
PIN	Contains the daily pin	6	Integer	automatically generated	988812	must have 6 characters that are numbers.
Forename	To identify the user by its Firstname.	50	String	First character is capitalised	John	Not Blank
Surname	To identify the user by its surname.	50	String	First character is capitalised	Smith	Not blank

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.
UserID	Key to refer to the customer that is booking	25	String	letters, numbers and underscore only	skadir	
ServiceID	Key to link to the service that the customer has booked for	256	Integer	must be an integer	1	Not blank
ConfirmedByStaff	Boolean that Staff confirms 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	140	String	Must not contain illegal characters.	Lorpem Ipsum sit amet dor	Must be limited to 140 characters.
Date	Specifies the date and of appointment	8	Date	Y/M/D	2012/01/01	Must be a real date
Time	Specifies 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,

2.4 File Organisation 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 never exceed ten gigabytes. 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. All of the processes are also handled sever-side.

2.5 Identification of Storage Media

As the system is stored online, the customers and staff don't have to worry about losing data, (*This is discussed in further detail in 2.13 Security Measures*). Therefore, keeping the integrity of the storage is important. The physical storage media primarily consist of hard drives. The hard drives would be handled by web hosting provider. These drives should run independent of the operating system of the server, which is stored on a separate hard drive. While it seems completely arbitrary, 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. Backups are discussed in *2.13 Descriptions of Measures Planned for System Security*.

2.6 Identification of Suitable Algorithms

2.6.1 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. I have chosen to make the database a class, as I would assume that this would be recalled throughout the system. This way, querying the database would be very quick to code.

```
class database
    public function initiate
        include page_with_db_variables.php
        try
            this <- database <- new database(hostname, db_name, username, password)
        catch
            die, print error
    end function
    public function doquery (query, arguments = array())
        try
            this <- result <- this <- database <- prepare query
            this <- result <- execute(arguments)
        catch
            die, print error
        end
    public function fetch
        return this <- result <- fetch()
    end
    public function fetchall
        return this <- result <- fetchall()
    end
    public function rowcount()
        return this <- result <- rowcount()
    end
end
```

2.6.2 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 MD5, the industry standard. Originally, I had proposed the idea of creating my own form of encryption, but I heavily advised against due to the sensitivity of the data I am trying to protect. Endorsing a standard that is verified for its security is important, otherwise the program would be against the Data Protection Act. I have chosen to use the prebuilt function in PHP instead of converting a pseudo code of MD5 in order to decrease vulnerability.

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'
salt <- hash_md5(salt)
variable <- hash_md5(variable)
hash(sha256, variable + salt)
```

2.6.3 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. This utilises 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. The automation may require tinkering with the server configuration file, `php.ini`. In this file, SMTP Settings of the outgoing server would need to be inserted.

```
msg <- Get Message Form Input
email <- Get Email Form Input

mail('email','Sample Comments', msg)

exit
```

2.6.4 Regular Expressions

Regular Expressions, or Reg-ex, 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. Below is the pseudocode for a self built function.

```
function validate_email(input){
    return (bool)preg_match("/^([a-zA-Z]+[a-zA-Z]*@[a-zA-Z]+\.[a-zA-Z]+)$/", trim($input))
}
```

Another example of Regular Expressions being useful is the incorporation of a password strength meter. This way it can determine the strength of the password, based on what characters are in the string. For example, a password containing symbols will be stronger than passwords that only have numbers.

```
if password length > 7
    if password matches (/([a-z].*[A-Z])|([A-Z].*[a-z])/)
        strength <- strength + 1
    if password matches (/([a-zA-Z])/) AND password matches(/([0-9])/)
        strength <- strength + 1
    if password matches (/([a-zA-Z])/) AND password matches(/([0-9])/)
        strength <- strength + 1
    if password matches (/[^!%&@#$^*,_,~].*[^!%&@#$^*,_,~]/)
        strength <- strength + 1
```

2.6.5 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.

```
Creating Captcha:
    scramble(u) {
        IP <- U
        HASH as String
        SALT <- random text
        TIME <- Time ++ 5 Minutes
        HASH <- md5(IP + TIME + SALT)
        OUTPUT HASH
    }
Save as Cookie

User solves the captcha, submits answer:
    Scramble(Userinput)
    IF Scramble(Userinput) = Cookie
        User <- "Correct!"
    ELSE
        User <- "Incorrect"
```

2.6.6 PIN Generator

In order for the staff to connect to their part of the system, A randomly generated pin is allocated to the staff. This PIN is unique and identifies staff from others. Also, The actual PIN is not stored on the database, but the PIN hash instead. A PIN hash would be created out of conjunction of the below pseudocode and 2.6.3 *Salt and hashed Encryption*. The function itself would be recursive, and a argument will be used to contain pin hashes that are currently in use.

```
generate_pin(used_pins as array){
    new_pin <- random_number(10000,99999)
    new_pin_hash <- encrypt(new_pin)
    while (in_array(new_pin_hash, used_pins))
        generate_pin(used_pins)
    }
    return array(new_pin, pin_hash)
}
```

2.6.7 Logging Into the System

In order to make the system secure, The customer should only be able to login as themselves. This way, the system can make multiple checks depending on the user. For example, when the user logs in, the system will check whether the user has activated their account, or has not been banned. If it isn't the case then the user will be passed through, and will be able to get into the dashboard.

```
username <- get username
password <- get password

if username and password not empty
    username <- trim username
    password <- encrypt(trim password)
```

```

query <- select * from users where username = username and password = password
DoQuery(query)
result <- fetch()
if result
    if result[activated] <- 1
        if result[banned] <- 0
            log in
        else
            print error
        end
    else
        print error
    end
else
    print error
end
else
print error
end

```

2.6.8 Forgot Password

In the event that the customer forgets their password, than they can use the email that is registered with their account to recover their password. They will also need access to their email account to validate the recovery request. When the user types in their email, the system will check whether the email is set in the database. If it has, then it will send a message to their email address. Regardless of the input, a notification will be sent. This way, it prevents malicious attempts to gain password entry as they wouldn't be able to know whether an email address is registered or not.

```

if !empty(email)
    query <- select * from users where email <- email
    DoQuery(query)
    details <- fetchall()
    if !empty(details)
        code <- encrypt(random number)
        query <- update users set forgot_code <- code
        DoQuery(query)
        email(details[user], code)
    end if
    direct user to confirmation page
end

```

2.6.9 Adding Tables into the System Database

With this, the client is able to migrate the system to another server in the event that they wish to. The following pseudocode can be used to automatically generate all the tables in the database, and start adding items in the database also.

```

create_tables <- array(query1, query2, .... queryN)

for x = 0, x <= 6, x++
    if x = 0
        table_name <- a
    end if
    if x = 1
        table_name <- b
    end if
    if x = 2

```

```

    table_name <- c
end if
if x = 3
    table_name <- d
end if
if x = 4
    table_name <- e
end if
if x = 5
    table_name <- f
end if
if x = 6
    table_name <- g
end if
check_if_exists <- SHOW TABLES LIKE 'x'
doQuery(check_if_exists)
fetch <- fetch_results_from_query
if !fetch
    doquery(table[x])
end
end for

insert_base_queries <- (querya1, querya2, .... queryaN)

foreach insert_base_queries as query
    doQuery(query)
end for

```

2.6.10 Paginator

In the event that the query results are too large, It would be much better to separate them into parts that are easier to go through. Using a pagination is considered more user friendly in this case as a consequence. This can also be adapted to display in terms of letter, or number.

```

query <- select from x order by date(desc)
query <- query . where date <- 'the_date'
rows <- number of rows (get from query)
per_page <- 10
pages <- ceil(rows/per_page)
if !empty(page)
    page <- 1
else
    page <- get page
start <- (page - 1)/page
query <- query + " limit start, per page"
DoQuery(query)
num <- FetchAll()

// displaying results

while i = 0 and i < pages
    if page = i
        echo i
    else
        echo i
    end if
    i <- i + 1
end while

```

2.7 Sample of Planned SQL Queries

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

Query Name	Adding A New Customer into the database
Purpose	Used to update a customers details
Type	Update
Tables	Users
Fields	UserForename, UserSurname, UserEmail, UserPassword, UserUsername, UserPhoneNo
Criteria	VALUES(?, ?, ?, ?, ?, ?)
Display Method	Information will be displayed on the screen to the user.
SQL Statement	INSERT INTO Client(ClientForename, ClientSurname, ClientEmail, ClientPassword, ClientUsername, ClientPhoneNo) VALUES(?, ?, ?, ?, ?, ?)

Query Name	Adding A New Appointment
Purpose	Used to create a new appointment using the customers details.
Type	Insert
Tables	Booking, Users
Fields	UserForename, UserSurname, UserEmail, UserPassword, UserUsername, UserPhoneNo
Criteria	VALUES(?, ?, ?, ?, ?)
Display Method	Information will be displayed on the screen to the user.
SQL Statement	INSERT INTO Booking(ClientID, ServiceKey, DateTime, ConfirmedByStaff) VALUES(?, ?, ?, ?)

Query Name	Updating an Appointment
Purpose	Used to update a customers details
Type	Update
Tables	Booking
Fields	ClientID, ServiceKey, DateTime, ConfirmedByStaff
Criteria	VALUES(?, ?, ?, ?, ?, ?)
Display Method	Information will be displayed on the screen to the user.
SQL Statement	UPDATE Booking SET ClientID, ServiceKey, DateTime, ConfirmedByStaff

Query Name	Removing an Appointment
Purpose	Used to delete a booking
Type	Delete
Tables	Booking
Fields	DateTime
Criteria	VALUES(?, ?, ?, ?, ?, ?)
Display Method	Information will be displayed on the screen to the user.
SQL Statement	<code>DELETE * FROM Booking WHERE DateTime = ?</code>

Query Name	Logging into the Administrators Dashboard
Purpose	Used to log in
Type	SELECT
Tables	Users
Fields	Username, Password, IsAdmin
Criteria	VALUES(?, ?, ?, ?, ?, ?)
Display Method	Information will be displayed on the screen to the user.
SQL Statement	<code>SELECT * FROM users WHERE Username = ? AND Password = ? and IsAdmin = 1</code>

Query Name	Displaying All Appointments on the Database
Purpose	Used to Display current Appointments
Type	Select
Tables	Client, Booking
Fields	DateTime, ClientID, ServiceKey, ConfirmedByStaff, ClientForename, ClientSurname
Criteria	VALUES(?, ?, ?, ?, ?, ?)
Display Method	Information will be displayed on the screen to the user.
SQL Statement	<code>SELECT DateTime, ClientID, ServiceKey, ConfirmedByStaff, ClientForename, ClientSurname FROM Client, Booking WHERE DateTime LIKE ?</code>

2.8 Class Definitions

Class: Booker

```
Day as Integer
Month as Integer
Year as Integer
Selected_Date as Date
First_day as Date
Back as Date
Forward as Date

Function Database
Function Fetch
Function MakeCalendar
Function Post
Function Confirmation
Function StartBooking
Function Minutes
Function CreateDaysTable
Function DaySwitch
Function MakeBookingSlots
Function StartCalendar
Function CreateForm
Function CreateDaysArray
```

The booker class contains the calendar and the form that the customer would use to book an appointment into the system. Several functions are simply used to help speed up the programming time, such as fetch and minutes. Several functions are used linearly so that some functions won't run without running other functions first. When the customer submits the form, Post will be used to send the data to the database.

Class: Database

```
Function Initiate
Function DoQuery
Function Fetch
Function FetchAll
Function RowCount
```

The database class is based on PDO, and is used to connect the system to the database. The Initiate function establishes a link to the database. The class also contains functions that fetches data from the database.

2.9 User Interface Design 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.

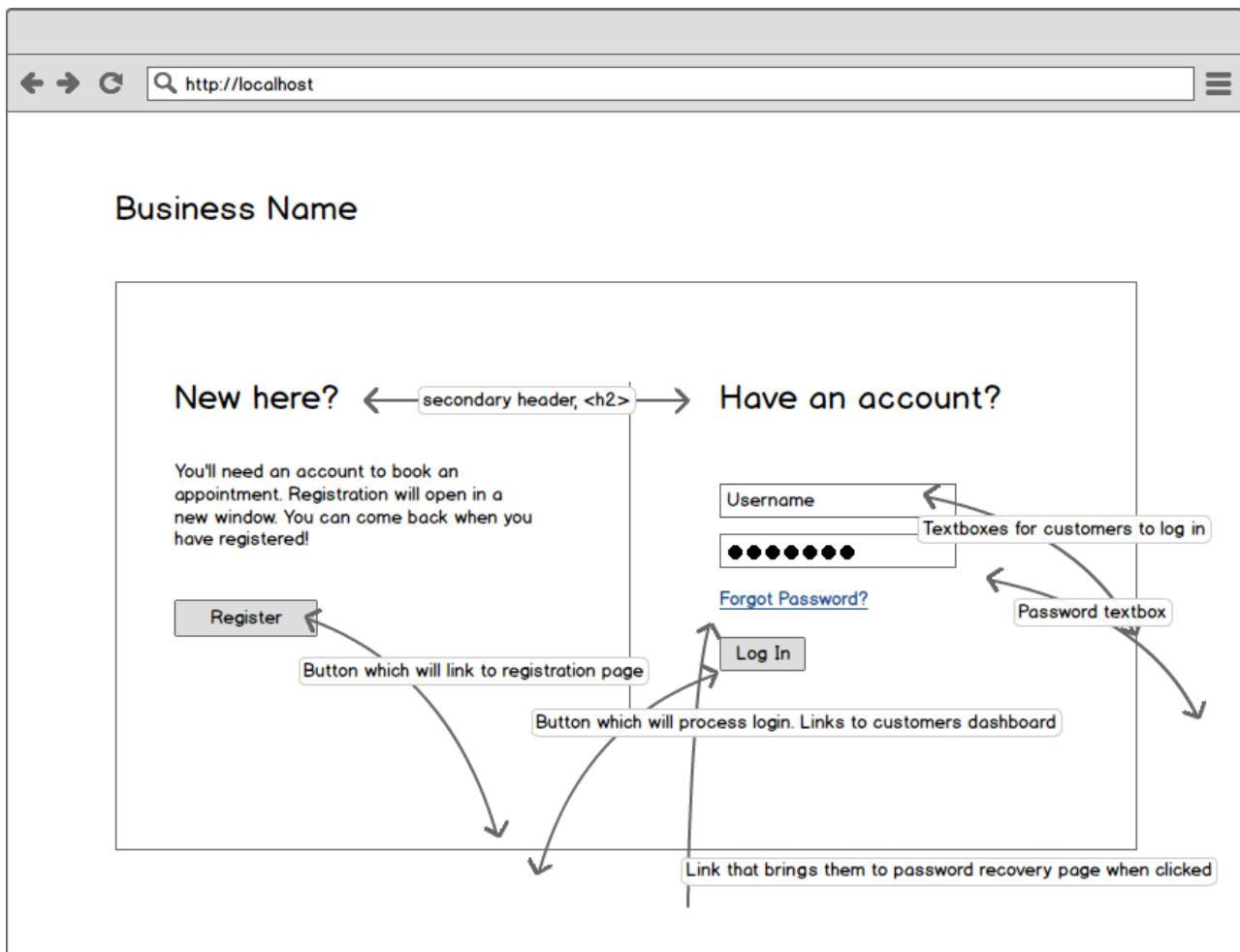
Since this is a web program, it would be pragmatic to create a user interface that people are already adjusted to. Conventions of website interface include a logo that is placed on the upper left, definitive link styling, button functionalities, and visual hierarchy.

The primary colour scheme will be shades of grey white, as the client is not concerned with the colour of the system. The content font will be black (#000), with headings and the background in a grey (#d3d5dd). Colour can play a big role in user interface design, so I have chosen to use colour keys, such as green (#a7d18d) for good and red (#d34747) for bad, 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*.

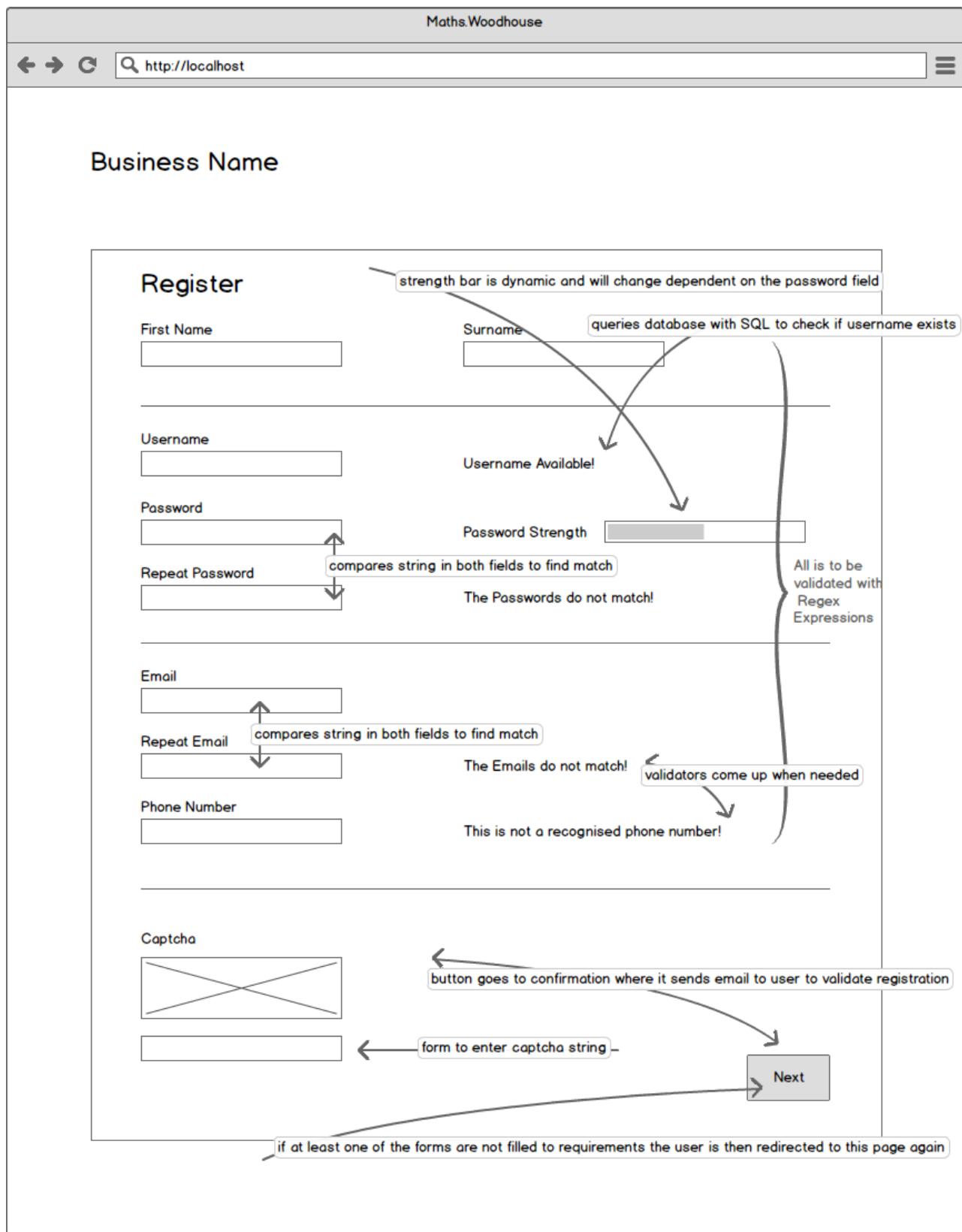
The primary font will be Helvetica, and the fall-back font will be Arial. In the special cases where neither fonts are supported, It will fall-back even further into the default sans-serif. The font size will be set in 14px, with a vertical padding between words of 1em. This is chosen for its legibility; 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 Helvetica, then Arial and eventually, 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. Link styling is made clear through the use of a differentiating colour from the content, which is black. The hyperlinks will be blue(#3585bb) and, when hovered over, will be underlined.

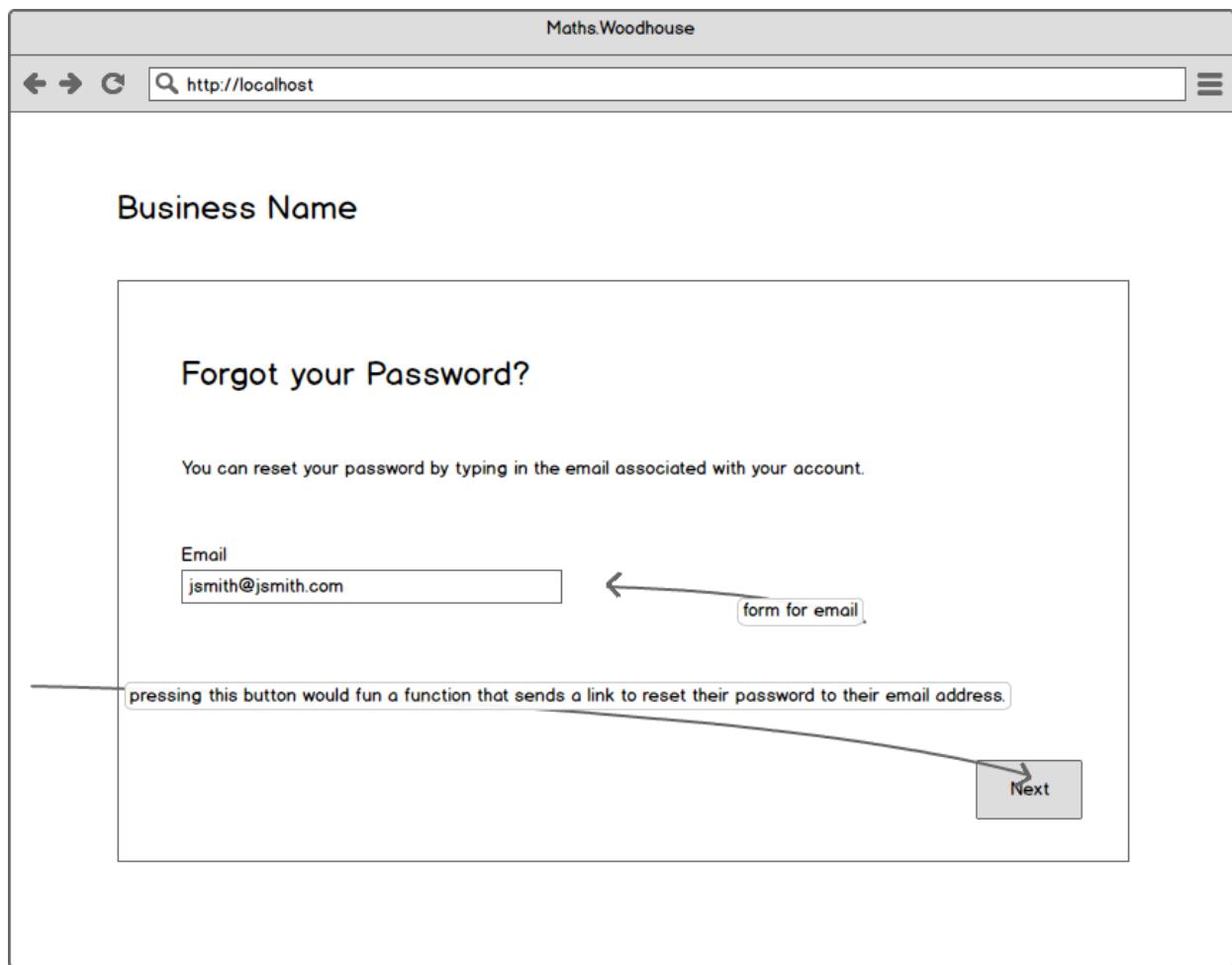
2.10 UI Sample of Planned Data



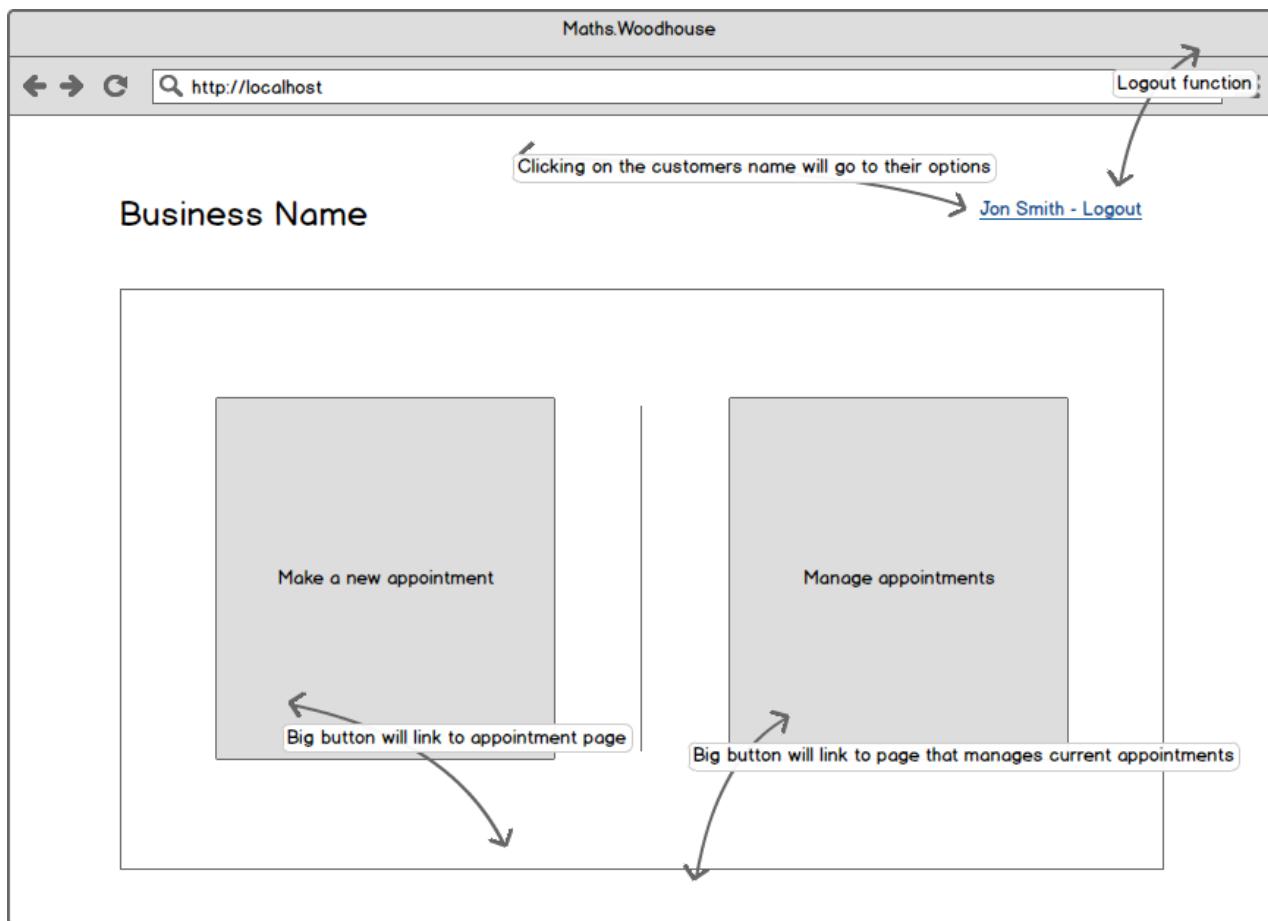
For the front page of the system, the customer is provided with the option to either log in or register a new account. Text about registration is on the left as people read from left to right. On the right, customers who already have an account will have the chance to log in. Also, in the event that the customer forgets their password, a link is shown where they can follow to generate a new password.



The above diagram shows what would be included in the registration page. Text boxes are for the customers first name, surname, username, password, their password again, email and its duplicate, their phone number and the captcha. When all of the details are correctly filled in, the customer can click next to continue with the registration process.



In the event that the customer forgets their password, than they can use the email that is registered with their account to recover their password. They will also need access to their email account to validate the recovery request. When the user types in their email, the system will check whether the email is set in the database. If it has, then it will send a message to their email address. Regardless of the input, a notification will be sent. This way, it prevents malicious attempts to gain password entry as they wouldn't be able to know whether an email address is registered or not.



This page is called the dashboard, and will be shown in the event that the customer logs in successfully. In this page the customer can click the left box to make a new appointment, click the right box to manage their current appointments or click their name on the top right to manage their settings. In the event they wish to log out they can click the “logout” link on the top right.

Maths.Woodhouse

http://localhost

[Jon Smith - Logout](#)

Business Name

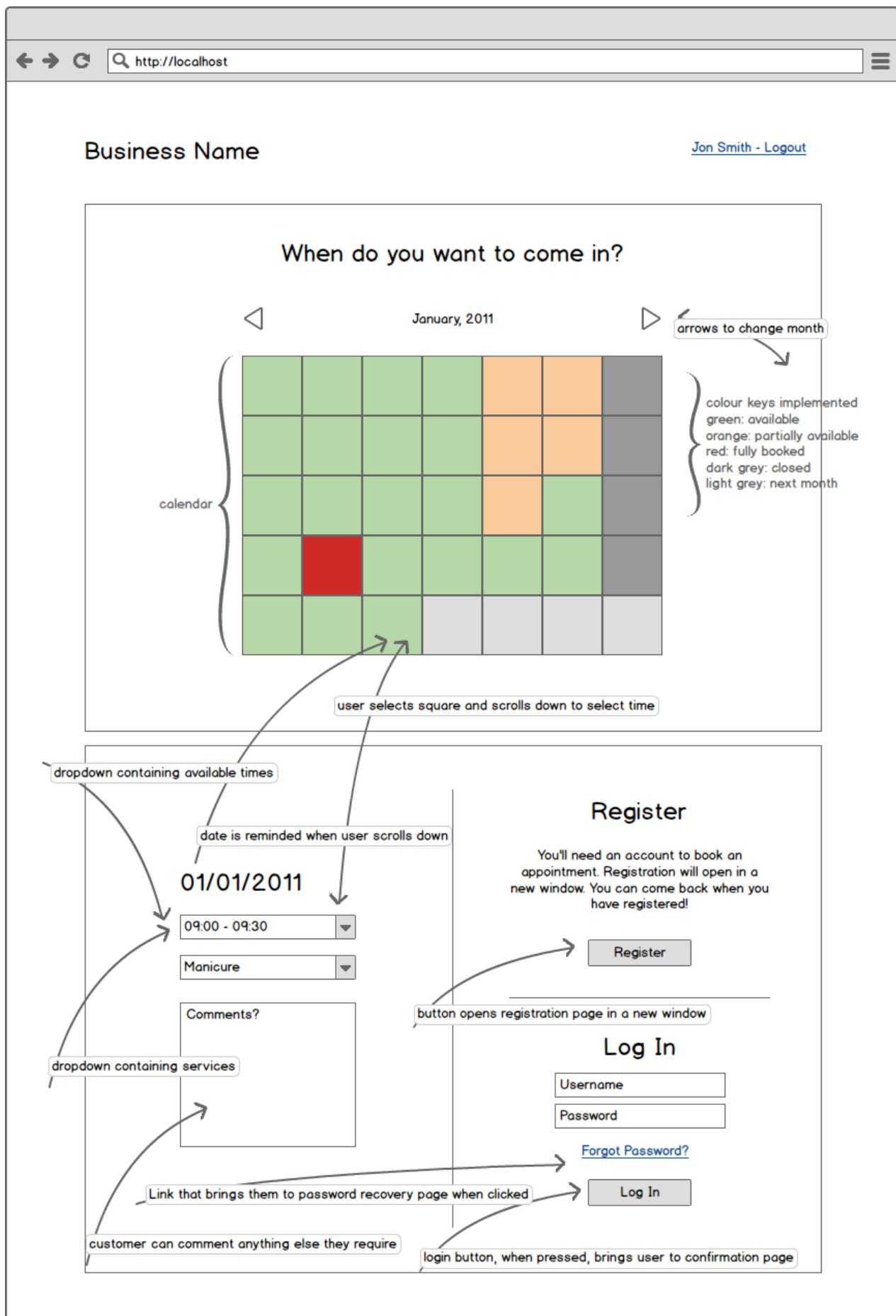
Edit Account

First Name <input type="text" value="Jon"/>	Surname <input type="text" value="Smith"/>
Most details are prefilled and loaded from the database	
Username <input type="text" value="jsmith"/>	Username Available!
Password <input type="password"/>	Password Strength <div style="width: 50%;">██████████</div>
Repeat Password <input type="password"/>	password inserted at this stage must not be the same as the previous password. The Passwords do not match!
Email <input type="text" value="jsmith@jsmith.com"/>	The Emails do not match!
Repeat Email <input type="text"/>	
Phone Number <input type="text" value="12345678901"/>	This is not a recognised phone number!
<input type="button" value="button then shows them to verify page"/> <input type="button" value="Next"/>	

Follows similar layout to registration

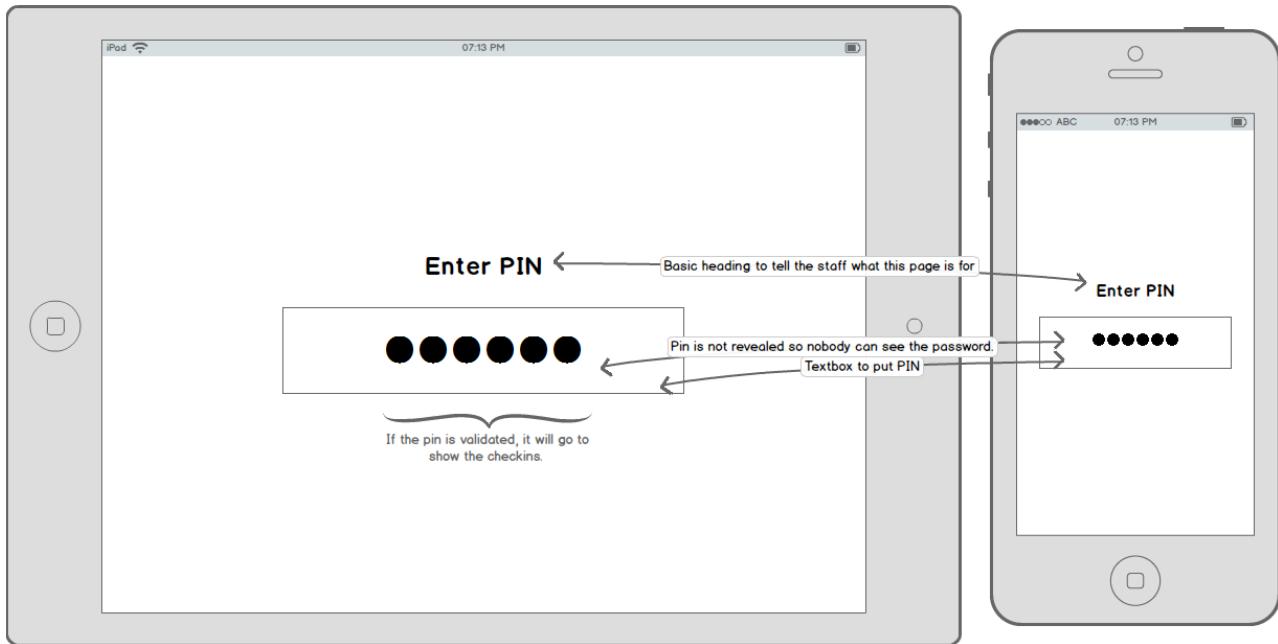
Validators remain the same and will show when needed

In this page, the customer can adjust their details. The text boxes will be pre-filled with customer data that has been stored in the database, except for the password. Customers can modify the data in the text boxes and save by clicking next.

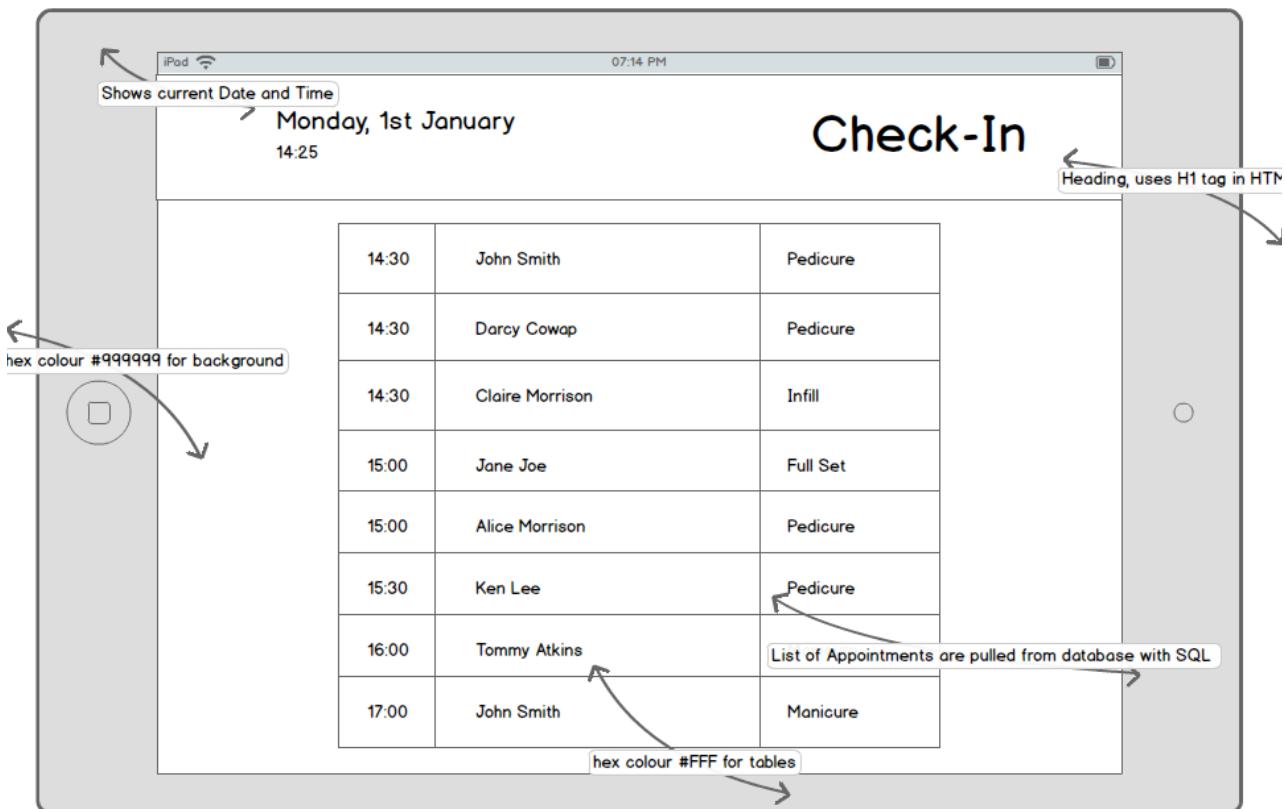


In the above mockup, the page where the customer would book is shown. Customers would start by

clicking on a date in the coloured box on the top half of the page, and then choose their date and service before booking in an appointment. This initial design includes the register and login page. When the customer logs in, the booking would be sent to the database.



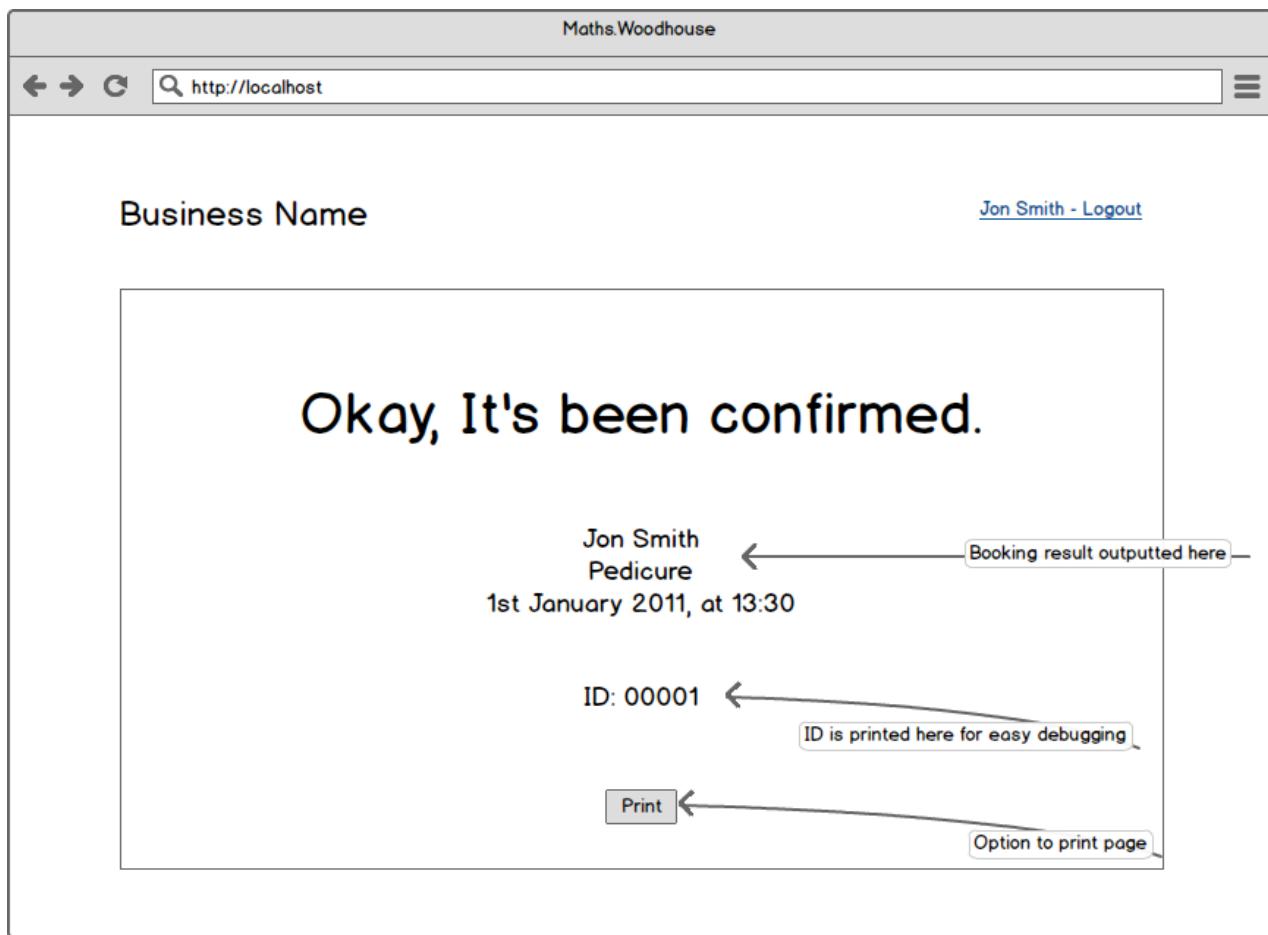
Above is a mockup for the staff login. On this page the staff would need to type in their PIN before using the system. With this mockup, the system would determine whether the PIN is correct on the entry of the last PIN number. If it is correct, they would be sent to the customer's checkin. Otherwise they would be locked back into the page and will have to try and attempt to log in again.



In this mockup, details of customers bookings are shown and the time of their booking. The top bar, which houses the date and time, alongside the “Check In” heading, is anchored at the top of the page. This means that while the staff is scrolling down the list of customers, the top bar would still be shown.

2.11 UI Sample of Planned Output Designs





The mockup is an example of a confirmation page, where the booking details such as their chosen service and date would be displayed to the customer. In this section the customer is given an option to print the booking.

Scheduling & Booking System

The mockup shows a web browser window titled "Maths.Woodhouse". The URL bar shows "http://localhost". The page header includes "Business Name" and "Jon Smith - Logout". The main content area is titled "Your Appointments" and displays a table of bookings:

Date	Time	Service	Action
01/01/11	14:30	Pedicure	Remove
01/01/11	14:30	Pedicure	Remove
01/01/11	14:30	Pedicure	Remove
01/01/11	15:00	Pedicure	Remove
01/01/11	15:00	Pedicure	Remove
01/01/11	15:30	Pedicure	Remove
01/01/11	16:00	Pedicure	Remove
01/01/11	17:00	Pedicure	Remove

Annotations explain the structure:

- A brace on the left side groups the rows, labeled "recalls bookings from database".
- Labels below the table indicate the meaning of each column: "row for date", "row for time", "row for services booked for", and "row for removing appointments".
- An arrow points from the "Remove" button in the first row to a callout box: "Clicking remove will popup window".
- A callout box for the popup window asks "Are you sure you want to remove this?" with "No" and "Yes" buttons.
- Text at the bottom right of the callout box says "clicking yes will call SQL query to delete booking".

In the above mockup is the customer manage checkin's page. Each row represents a customer's booking. A button on the right shows the possibility of removing a checkin. On the left, a column for the date and time are shown for each booking. Clicking on the remove button would show a popup, confirming them of the removal of a booking.

The mockup shows a mobile device (iPhone/iPad) displaying a "Check-In" application. The screen is split into two sections:

- Left Side (Mobile View):** Shows a table of bookings. A specific row for "John Smith" at 14:25 is highlighted in green, indicating it has been checked in. A note says "Confirmed Checkins are in green". A gesture indicator "Swipe Left to complete booking" is shown next to the row.
- Right Side (Desktop View):** Shows a larger view of the same bookings. The checked-in row for John Smith is also highlighted in green. The table includes columns for Date, Time, Customer Name, and Service Type.

Annotations provide additional details:

- "Shows current Date and Time" is noted above the desktop view.
- "Heading is anchored on the top, so it will remain in position regardless of scrolling" is noted near the top of the desktop view.
- "Service is shortened to ServiceKey on mobile due to lack of space" is noted at the bottom of the mobile view table.

The above shows a staff's customer checkin page. Sliding a booking row to the left will make that row green, indicating that the customer has been checked in.

2.12 Descriptions of Measures Planned for Security & Integrity of Data

Regular Expressions are 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.

If they choose to use a server within their business, 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. Otherwise the use of an off-shore server would have their own dedicated team of specialists who can deal with issues such as this. Also, with the use of an offshore server, backups wouldn't be necessary due to the redundancy and reliability the service provides.

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.

2.13 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 5 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, MD5, 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.

2.14 Overall Test Strategy

White box testing will be conducted to test all possible paths, seeing if the start of the program can lead me to the end of the program. This would be done from the perspective of each intended user.

Black Box testing will be used to see if the program functions as intended. Below is an example of the tests that will be conducted in order to see if the program has any errors or such. Normal, Erroneous and Boundary Data are recognised by the different shades the boxes are filled in. Below is the key for the different kinds of data that will be used to test the program for black box testing.

#	TESTING METHOD	DATA TYPE	TEST DESC	TEST DATA	PREDICTION
1	Blackbox	Typical	Customer is trying to log into the system with the correct details	customer password	System will take them to their dashboard
2	Blackbox	Erroneous	Customer is trying to log into the system with incorrect details	Customer passw0rd	System will not take them into their dashboard, and will tell them that the credentials are incorrect.
3	Blackbox	Typical	Customer is trying to recover a password by using an email associated with the account they are trying to recover.	thisis.tnguyen@gmail.com">thisis.tnguyen@gmail.com	They are notified that an email is sent to that email address, discussing how to take further action.
4	Blackbox	Typical	Customer is trying to recover a password by using an email that is not recognised by the database.	Cust0m3r@em43l.com	They are notified that an email is sent to that email address, discussing how to take further action. An email is not actually sent to that address.
5	Blackbox	Typical	Staff is attempting to log in with the correct PIN	12345	They will be directed to the staffs dashboard
6	Blackbox	Erroneous	Staff is attempting to log in with the incorrect PIN	123123	They will be told the pin is incorrect
7	Blackbox	Typical	Administrators is attempting to log in with the correct username and password.	Admin password	They are directed to the administrators dashboard
8	Blackbox	Erroneous	Administrators is attempting to log in with the correct username and an incorrect password.	Admin p4dsa23ff	They are told that the credentials are incorrect
9	Blackbox	Erroneous	Administrators is attempting to log in with an incorrect username and incorrect password	Adm3n Partawfb2@	they are told that the credentials are incorrect

Scheduling & Booking System

#	TESTING METHOD	DATA TYPE	TEST DESC	TEST DATA	PREDICTION
10	Blackbox	Typical	Customer is attempting to click on "Book an appointment" button	mouseclick	They are directed to the appointment page where they can make an appointment.
11	Blackbox	Typical	Customer is attempting to click on "Manage Appointments" button	mouseclick	They are directed to the "manage appointments" page.
12	Blackbox	Typical	Customer clicks on a fully available day	mouseclick	The day is parsed onto the system and they can continue with the booking process
13	Blackbox	Boundary	Customer clicks on a partially available day	mouseclick	The day is parsed onto the system and they can continue with the booking process
14	Blackbox	Erroneous	Customer clicks on a closed day	mouseclick	nothing happens
15	Blackbox	Erroneous	Customer clicks on a day which is on the following month	mouseclick	nothing happens
16	Blackbox	Erroneous	Customer does not select a service and attempts to book the appointment	not applicable	They are not directed on to the next page, and they are informed that they need to select a service.
17	Blackbox	Erroneous	Customer does not select a time period and attempts to book the appointment	not applicable	They are not directed on to the next page, and they are informed that they need to choose a time period of the booking.
18	Blackbox	Typical	Customer registers a username that is available	customer2	They are notified that the username is available
19	Blackbox	Erroneous	Customer registers a username that is not available	customer	They are notified that the username is not available
20	Blackbox	Typical	Customer types in a password that is eligible	passw0rD	The password is parsed onto the function that generates the password strength bar
21	Blackbox	Erroneous	Customer registers with a password that is not eligible	pass	They are notified that the password does not meet the requirements. (Password too short)
22	Blackbox	Typical	Customer types in a password that matches	passw0RD passw0RD	Nothing happens
23	Blackbox	Erroneous	Customer types in a different value in the match password textbox.	password p4q7thegh	They are told that the passwords do not match
24	Blackbox	Typical	Customer types in value in the email textbox that contains the "@" sign and a period.	john@doe.com	Nothing happens
25	Blackbox	Erroneous	Customer types in a value in the email textbox that does not contain an "@" sign	johndoe.com	They are told that this email is invalid
26	Blackbox	Erroneous	Customer types in a value in the email textbox that does not contain a period.	john@doecom	they are told that this email is invalid

Scheduling & Booking System

#	TESTING METHOD	DATA TYPE	TEST DESC	TEST DATA	PREDICTION
27	Blackbox	Typical	Customer types in a value in the verify email textbox that matches the value in the email textbox	john@doe.com john@doe.com	They told that the emails match
28	Blackbox	Erroneous	Customer types in a value in the verify email textbox that does not match the value in the email textbox.	john@doe.com sean@kingston.net	They are told that the emails do not match
29	Blackbox	Typical	Customer types in the captcha incorrectly during the booking process.	Not applicable	They are not directed on to the next page, and they are informed that they need to retype the captcha
30	Blackbox	Typical	Customer types in a number value that is 11 digits long in the phone number field.	01234567891	Nothing happens
31	Blackbox	Erroneous	Customer types in a number value that is not 11 digits long in the phone number field.	1	They are informed that they need to retype the phone number.
32	Blackbox	Typical	Staff attempts to validate a checkin by clicking (tapping) the checkin box related to the customer's name and time.	finger tap	The row bounces back to their matching columns and the row is given a green background
33	Blackbox	Typical	Staff attempts to undo a checkin by clicking (tapping) the uncheck box related to the customer's name and time.	finger tap	nothing happens
34	Blackbox	Typical	Staff types in a customers name in the search field.	Claire	A preset appointment under the name Claire Snow will display as a result.
35	Blackbox	Typical	Admin adds a new closing date with 2015-05-01	2015/5/1	Date is saved in database, is notified of change.
36	Blackbox	Boundary	Admin adds a new closing date with the current day. The day of the testing is 2015/01/28	2015/1/28	Date is saved in database, is notified of change.
37	Blackbox	Erroneous	Admin adds a new closing date.	2015/13/01	Error message saying that this is not a valid date should occur.
38	Blackbox	Erroneous	Admin selects a date in the appointments day.	2015/02/31	An error message saying that the date is invalid is displayed.
39	Blackbox	Boundary	Admin selects a date in the appointments day.	2015/1/1	Results are showed, but there are no results on that day.
40	Blackbox	Typical	Admin selects a date in the appointments day.	2015/1/5	Results will display for appointments booked on 2015/1/5
41	Blackbox	Erroneous	Admin adds a new option with numbers in the title, no Price, No description.	Acrylic Refil	Error messages will be displayed telling the admin that they need to input a price and a description.
42	Blackbox	Erroneous	Admin adds a new option with numbers in the title, a number value for price, No description.	Acrylic Refil, 18	Error messages will be displayed telling the admin that they need to input a description.
43	Blackbox	Typical	Admin adds a new option with correct values for title, a number value for price, and a description.	Acrylic Refil, 18, A maintenance service for Acrylic nails.	A message is displayed saying that the query is added.

Scheduling & Booking System

#	TESTING METHOD	DATA TYPE	TEST DESC	TEST DATA	PREDICTION
44	Blackbox	Erroneous	Admin adjusts the first slot.	25:00	An error message is displayed saying that the input is an invalid time.
45	Blackbox	Typical	Admin adjusts the first slot.	08:00	A message is displayed saying that the information is updated into the database.
46	Blackbox	Erroneous	Admin adjusts the last slot.	25:00	An error message is displayed saying that the input is an invalid time.
47	Blackbox	Typical	Admin adjusts the last slot.	20:00	A message is displayed saying that the information is updated into the database.
48	Blackbox	Erroneous	Admin adjusts the booking frequency.	20	A message is displayed saying that the information is updated into the database.
49	Blackbox	Typical	Admin adjusts the booking frequency.	-100	An error message is displayed saying that the booking frequency is either too long or too short.
50	Blackbox	Typical	Admin changes the Business name.	Business Name	A message is displayed saying that the information is updated into the database. Also, the bold header on the top left of all pages will be replaced with the new input.
51	Blackbox	Boundary	Admin changes the slogan. The test data is 123 characters long.	Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.	An error message is displayed saying that the slogan is too long.
52	Blackbox	Typical	Admin changes the slogan. The test data is 13 characters long.	Lorpem Ipsum.	A message is displayed saying that the information is updated into the database. Also, A slogan is shown underneath the top header where the company name is placed.
53	White Box	Erroneous	Customer attempts to change the day of booking. Chosen day is set to be closed.	http://projects.tnguyen.ch/calendar.php?month=01&year=2015&day=01	A message is displayed saying that the date is unavailable for booking. The message will contain a reason, of which this case it will be that the day is closed.
54	White Box	Erroneous	Customer attempts to change the day of booking. Chosen day is a sunday.	http://projects.tnguyen.ch/calendar.php?month=02&year=2015&day=01#selected_date	A message is displayed saying that the date is unavailable for booking. The message will contain a reason, of which this case it will be that the day is a sunday.
55	White Box	Erroneous	Customer attempts to change the day of booking. Chosen day is beyond the limit of schedule.	http://projects.tnguyen.ch/calendar.php?month=02&year=2016&day=01#selected_date	A message is displayed saying that the date is unavailable for booking. The message will contain a reason, of which this case it will be that the day is out of bounds of the booking threshold.
56	Whitebox	Typical	Testing the result of the hashing algorithm.	password, password2	a randomly scrambled hash will be generated.

Scheduling & Booking System

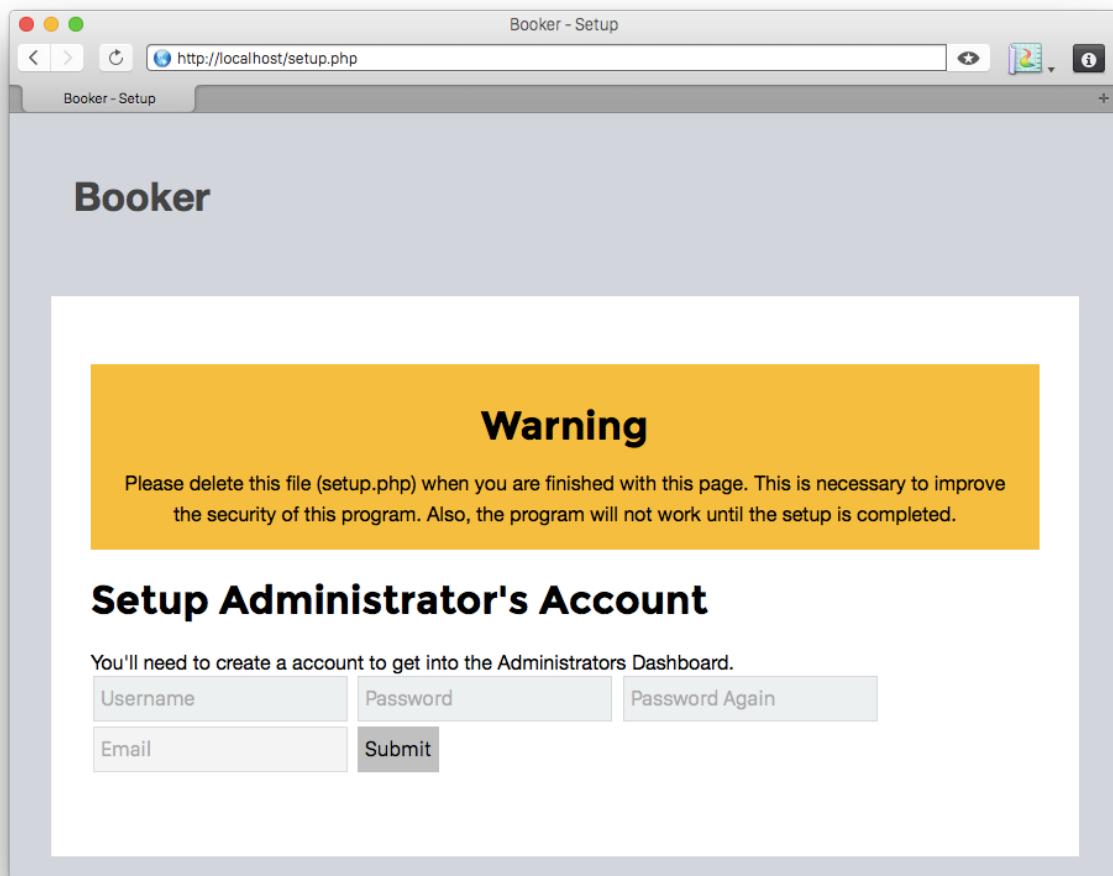
#	TESTING METHOD	DATA TYPE	TEST DESC	TEST DATA	PREDICTION
57	Whitebox	Typical	Testing the result of PIN generation. The chosen staff is originally allocated with a random PIN. The staff is then given another PIN.	mouseclick	On first click, the staff is given a PIN of 12345. (This is done to ensure that the second pin will be different to this PIN.) On the second click, the PIN would be a different to 12345.

Technical Solution

3.1 Screen Designs

Throughout the implementation, I have chosen to change some of the designs of the system. There are multiple reasons for some of the adjustments, which will be specified for each screenshot.

3.1.1 Setup



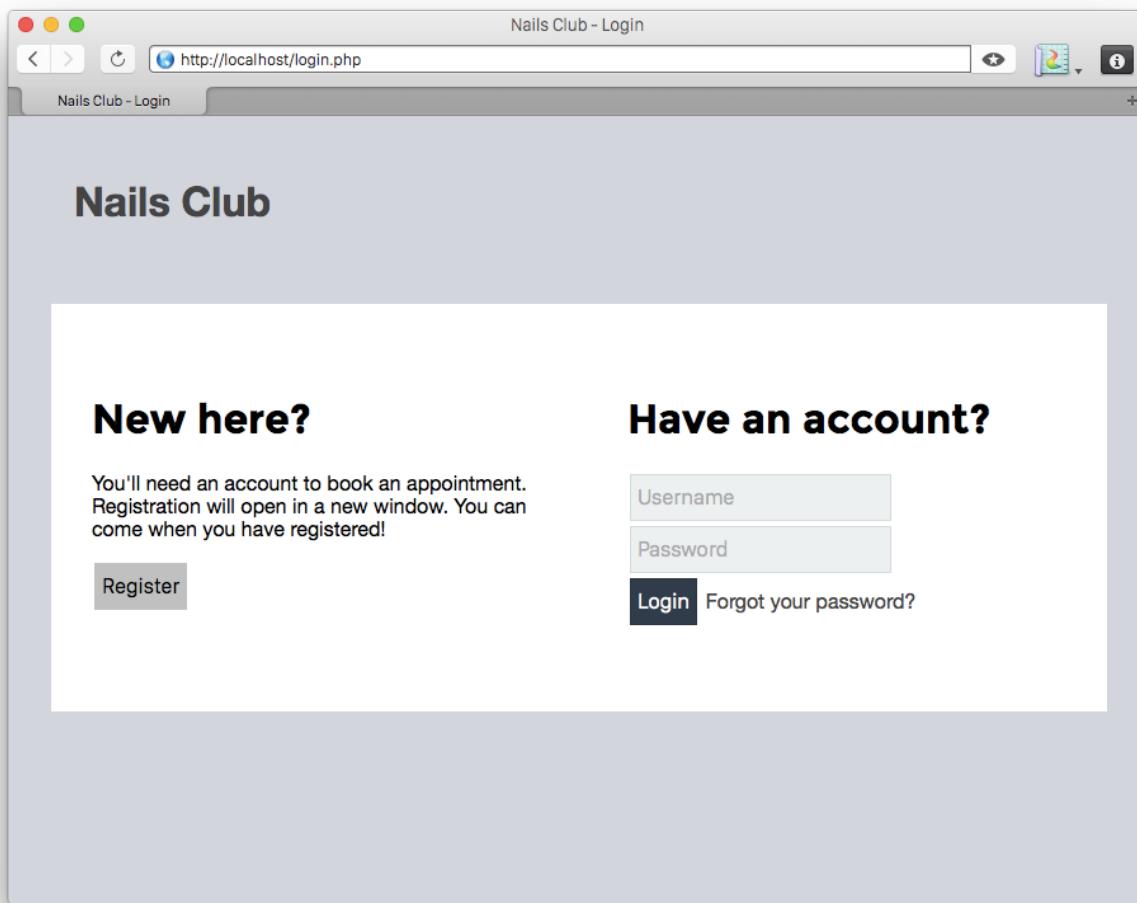
The setup page is the default page loaded when the system is run for the first time. This is handled by the core.php's redirect function if the setup file exists. At this stage the administrators are instructed beforehand to put in their database credentials inside the db file within the includes folder. A yellow notice is there to remind them to delete the file when they are finished with the setup. At this stage they will be able to register the administrators account.

In the background several events will occur after they submit their information. provided that their details are correctly parsed; the page will execute loops in the background to insert tables inside the database. Also, for certain parts of the system to run, some rows are filled at this stage as well, such as the website metadata.

Validations

- Username
- Password
- Email Address

3.1.2 Login



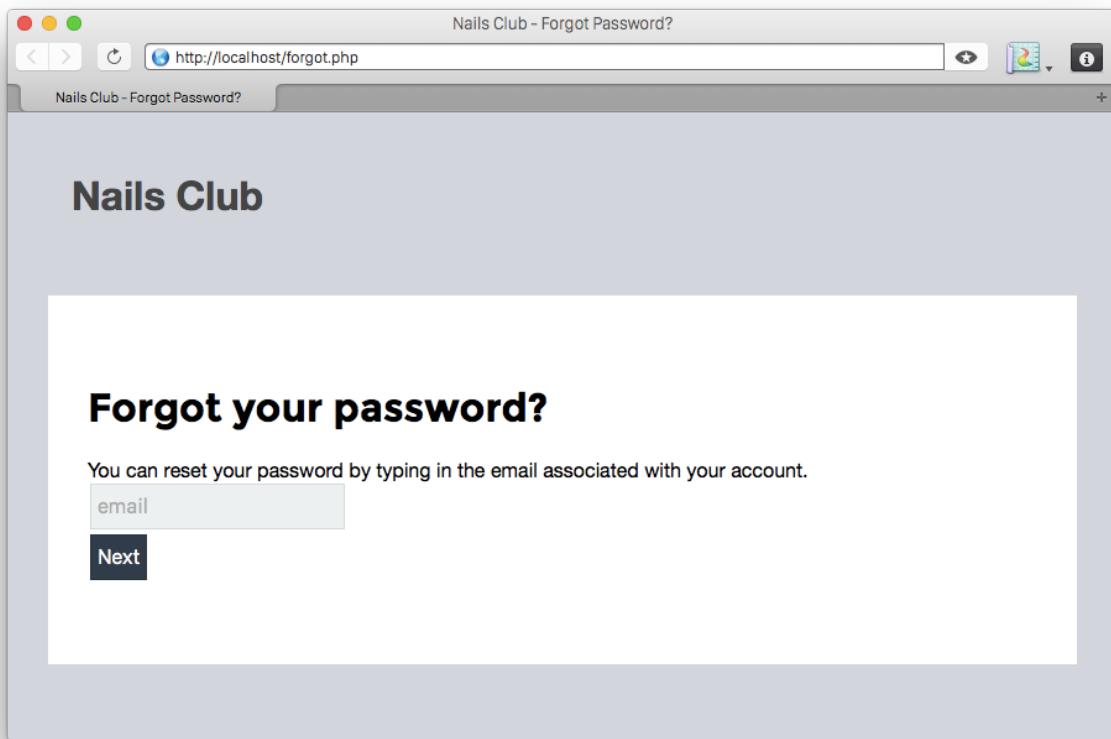
This design is very similar to the planned design. When the setup file is removed, the system is now ready to run. This page will be the de-facto page that all visitors will land on. At this stage visitors are given the option to register, or to log in. Additionally, if they have registered they can access a link to recover their password.

Clicking the register button will redirect them to the registration page. Credentials that are typed in the login form will be parsed, and checked in the database in order to see if a match occurs between the details in the database and the posted data. If there is a match the customer is then directed into the dashboard. User handling is dealt with the ID instead of the username.

Validations

- Username, Password

3.1.3 Forgot Password



Like the login, this part of the program similarly follows the footsteps of the enacted design. When a email address is correctly entered a email would be sent to them about further instructions.

Validations

- Email Address

3.1.4 Customer Registration

The screenshot shows a web browser window titled "Nails Club - Register" at the URL <http://localhost/registerphp>. The page has a header "Nails Club" and a main title "Register". The form fields include:

- Forename:
- Surname:
- Username:
- Password:
- Confirm Password:
- Email:
- Confirm Email:
- Phone Number:
- Captcha: A reCAPTCHA interface with a blurred image of a building and the number 178, a text input field containing "Type the text", and a "reCAPTCHA" button.
- Next: A large brown "Next" button at the bottom right.

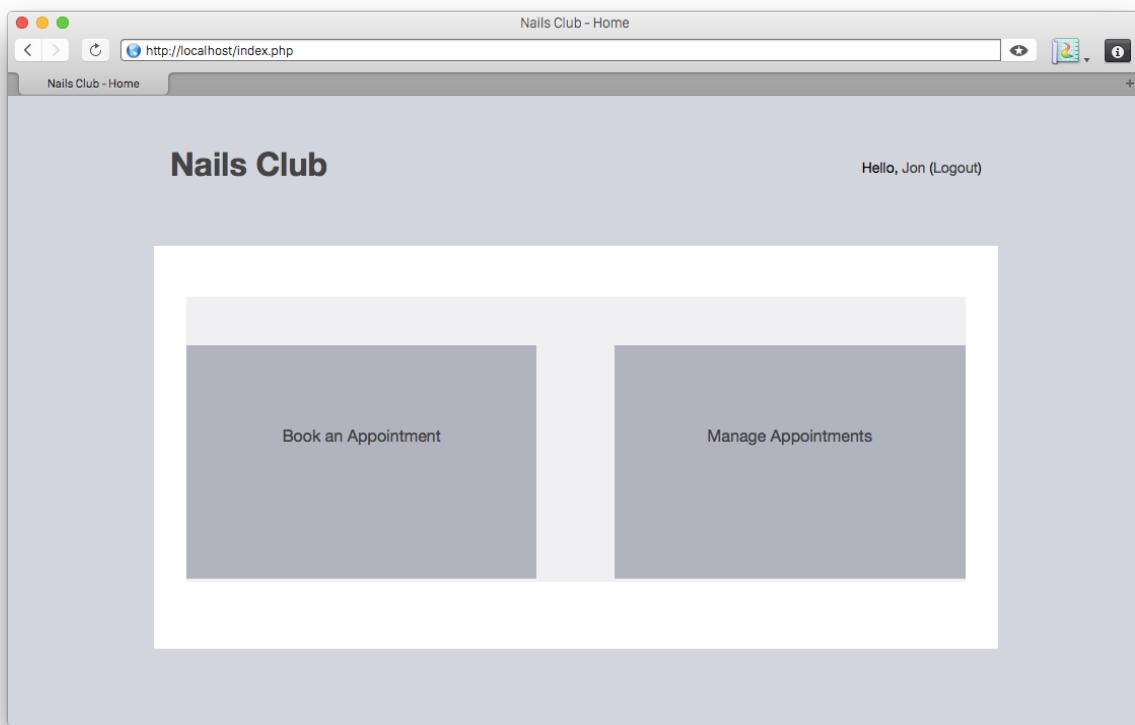
Registration is similarly represented in the design phase, except in this case the captcha is dealt with Google's ReCAPTCHA service.

jQuery is used in the handling of the input values, alongside the password strength bar. The next button is disabled unless all parameters are met. A php fallback is used in the event that the customer doesn't have javascript enabled.

Validations

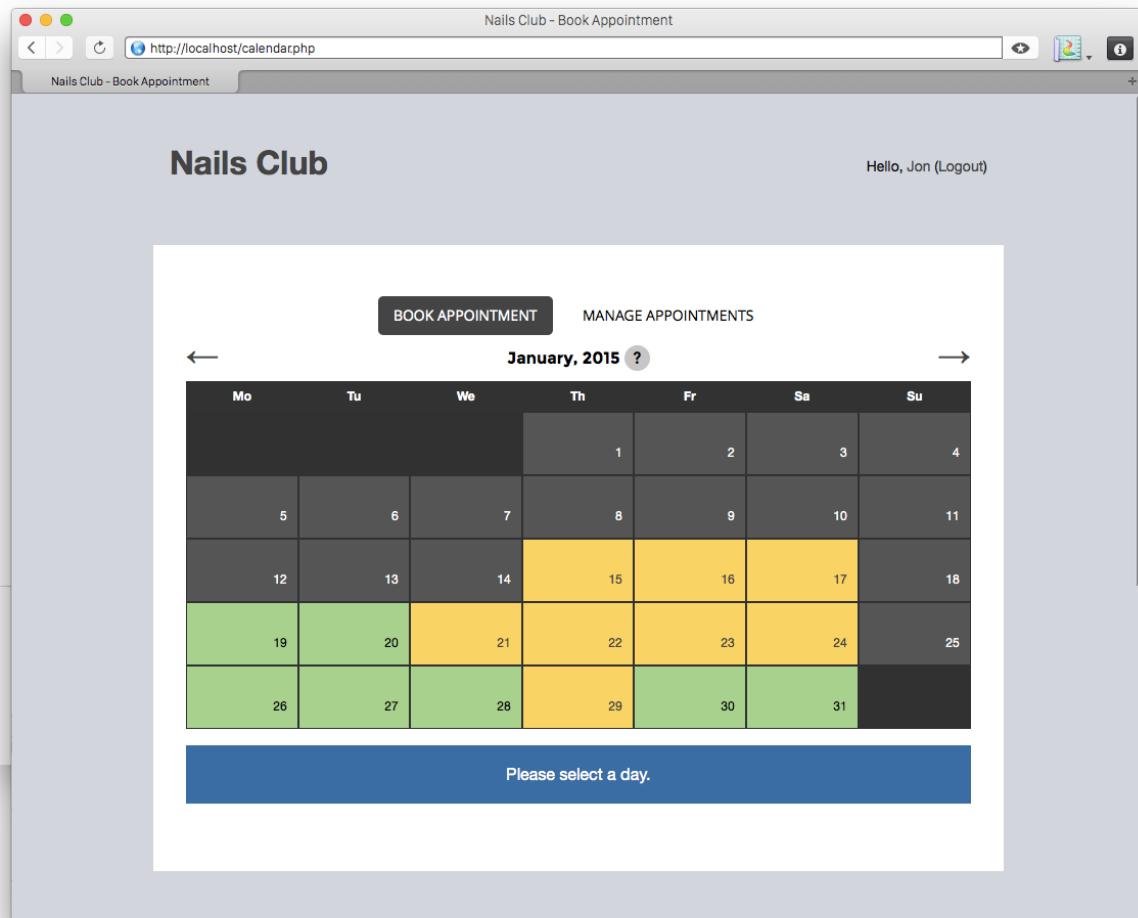
- Forename
- Surname
- Username
- Password
- Email Address
- Phone Number
- Captcha

3.1.5 Dashboard



The dashboard is accessed whenever the customer has logged in successfully. At this stage they have the option to book an appointment or manage appointments. Also, they can click on their forename on the top right to access their details. There is very little changed between the technical solution of this part of the system and the design.

3.1.6 Calendar



There has been multiple adjustments to this compared to the design.

- The technical solution includes days of the week so it is easier to see what number of the month relates to what part of the week.
- A captcha has been added to deter customers from making bulk appointments. This also ensures that they have specifically chosen to book this appointment.
- The option to register and login is removed as the navigation of the customers side of the system is adjusted. Customers are now required to login prior to making an appointment.
- This way, when the website is scaled to a mobile browser, There are less content shown on the screen making it easier for the customer to navigate through.
- A colour key has been added in the event that the customer doesn't recognise what each colour represents.

The colours have changed to reflect the layout. The background of the actual calendar is now darker to compliment the surrounding colours of the website.

Validations

- Date
- Time
- Comments
- Captcha

The screenshot shows a web application interface for a scheduling and booking system.

Top Navigation: The address bar shows <http://localhost>. The top right corner displays "Jon Smith - Logout".

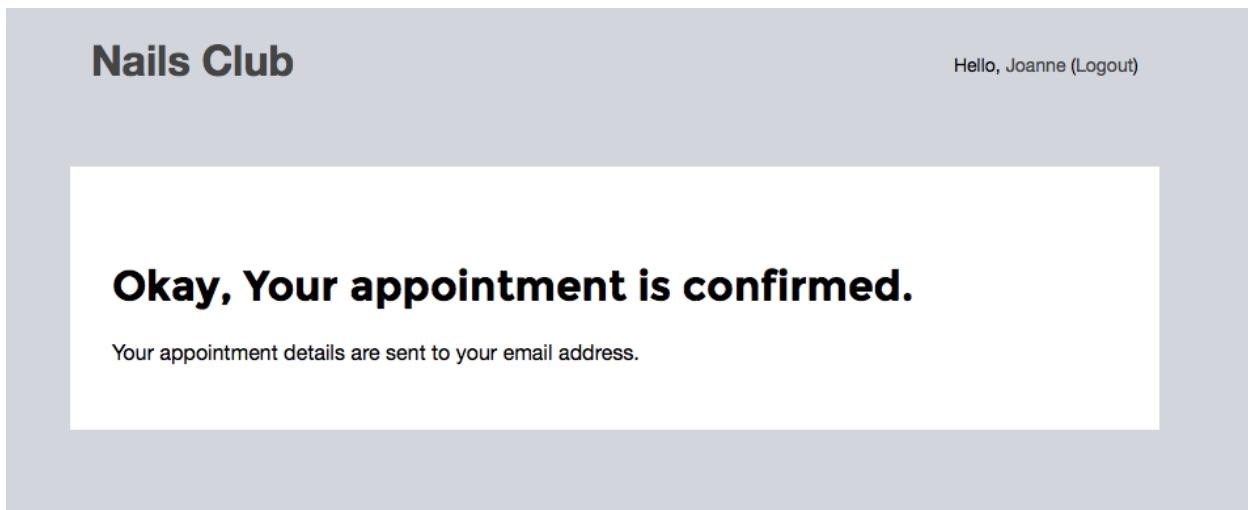
Business Name: The page title is "Business Name".

Calendar View: A grid titled "When do you want to come in?" shows a month of January 2011. The grid uses color-coding to represent availability: green for available, orange for partially available, red for fully booked, dark grey for closed, and light grey for the next month. Arrows on the right side of the grid allow users to change the month. A legend on the right explains the color keys: green (available), orange (partially available), red (fully booked), dark grey (closed), and light grey (next month). A callout indicates that the user selects a square and scrolls down to select time.

Registration Form: The "Register" section contains fields for "Comments?", a dropdown for services (e.g., Manicure), and a dropdown for available times (e.g., 09.00 - 09.30). It also includes a "Register" button and a note about creating an account for booking appointments.

Login Form: The "Log In" section contains fields for "Username" and "Password", a "Forgot Password?" link, and a "Log In" button. It also includes a note about registration opening in a new window.

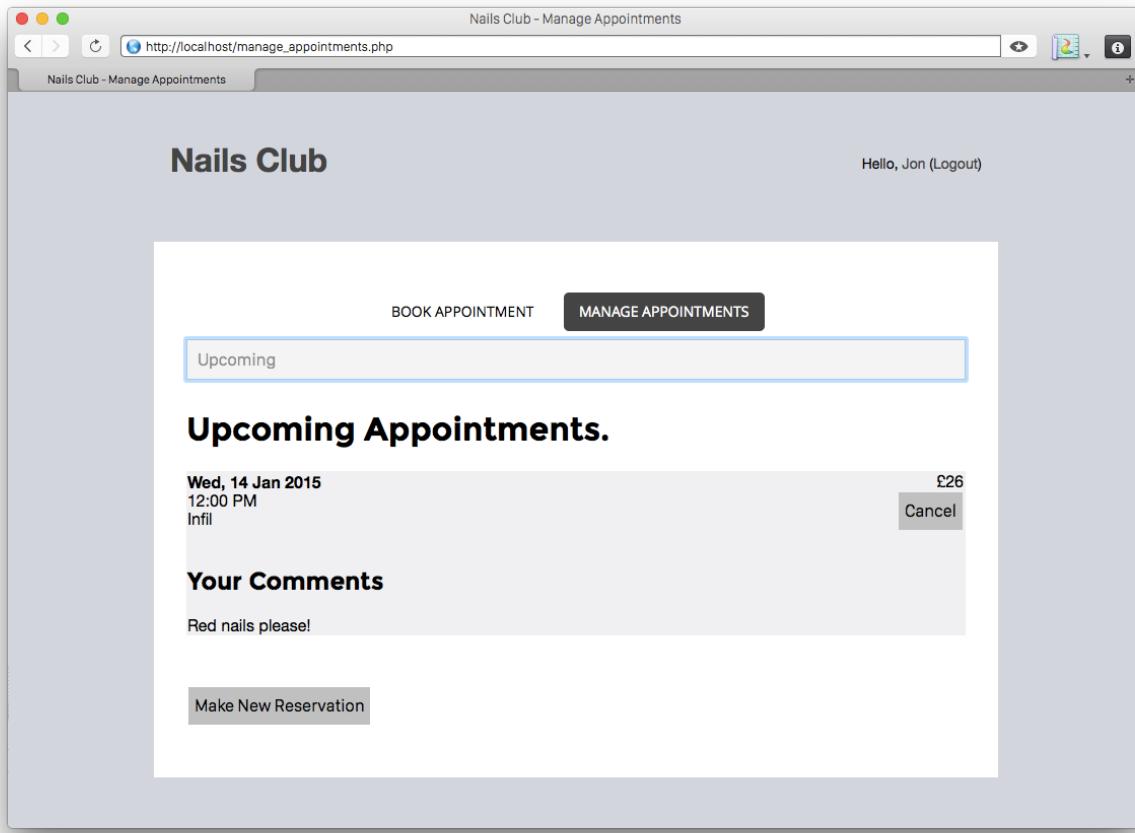
Customer Feedback: A text area at the bottom allows customers to leave comments.



3.1.7 Confirmation

This has been changed significantly. I have chosen to send the booking details to the customer through Email instead of notifying the customer of the booking afterwards. I feel that this is more convenient when the customer comes to check in their appointment. Also, booking ID is negated in its entirety to improve user friendliness. It becomes more conversationally friendly to tell the staff their name and the time they have come to book their appointment. This makes the email more useful as they can refer to such when they need to instead of remembering what was printed on the screen after they booked.

The image shows two side-by-side screenshots. On the left is a screenshot of an iPhone inbox showing an email from "robot@tnguyen.ch" with the subject "Your Appointment". The email body contains a greeting, the appointment details (date, time, service), and the business address. On the right is a screenshot of a web browser window titled "Maths.Woodhouse" showing a confirmation message. The message includes the customer's name, the service, the date and time, the booking ID, and a "Print" button. Annotations with arrows point from specific text elements to their counterparts in the email, illustrating how the web-based system provides a more direct and user-friendly way to confirm bookings.



3.1.8 Your Appointments

The Technical Solution for this part of the system has changed since the declared design decision also.

I have chosen to group each customers appointments to past, upcoming and all. This way it becomes easier to see their current appointments and less appointments show on the screen, reducing visual clutter. I have chosen to display each appointment in its entirety as a result due to the extra vertical space that is given.

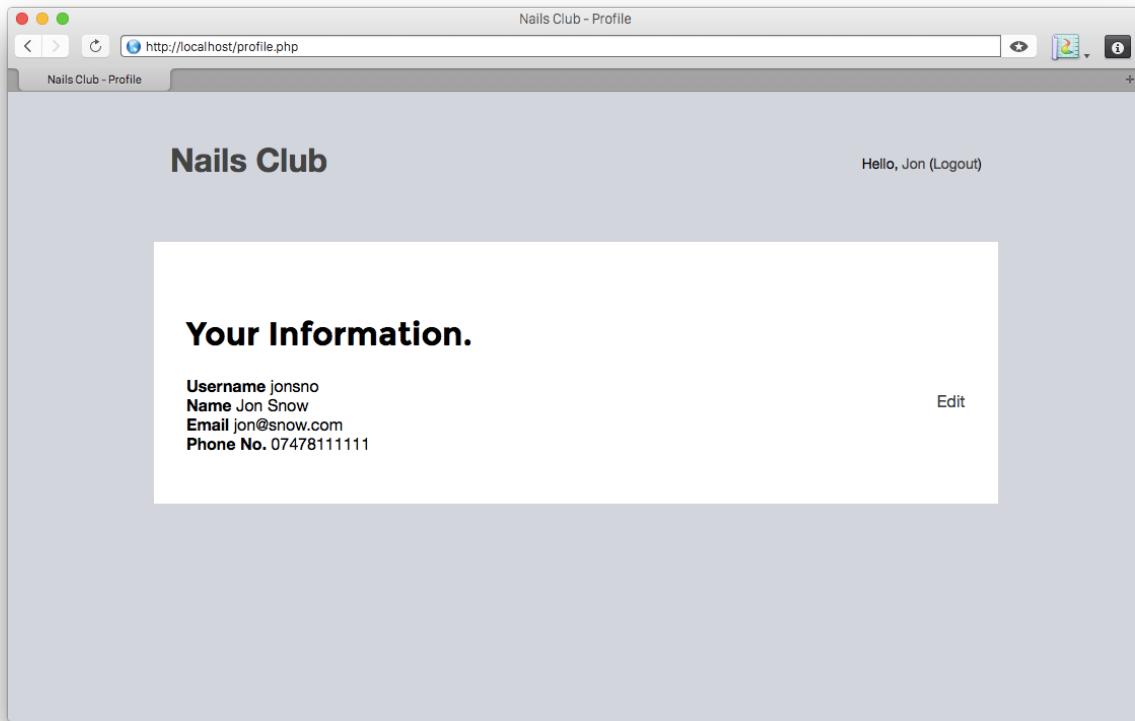
Your Appointments

Appointments from database	01/01/11	14:30	Pedicure	Remove
	01/01/11	14:30	Pedicure	Remove
	01/01/11	14:30	Pedicure	Remove
	01/01/11	15:00	Pedicure	Remove
	01/01/11	15:00	Pedicure	Remove
	01/01/11	15:30	Pedicure	Remove
	01/01/11	16:00	Pedicure	Remove
	01/01/11	17:00	Pedicure	Remove
	Clicking remove will p...			

row for date
row for time
row for services booked for
row for removing appointments

The word remove is changed to cancel to hedge the seriousness of cancelling appointments.

3.1.9 Users Information



This page is accessed whenever the customer clicks on their forename, displayed on the top right of the system. This page displays the customers current information that is stored in the database, except for the password. A link to edit the customer's information is displayed on the right. Information is recalled through the database using multiple queries.

3.1.10 Edit Details

The screenshot shows a web browser window titled 'Nails Club - Edit' with the URL 'http://localhost/edit.php'. The page header says 'Hello, Jon (Logout)'. The main content area is titled 'Your Details.' and contains the following fields:

- Forename:** Jon
- Surname:** Snow
- Password:** Password
- Password Strength:** No Data
- Confirm Password:** Password
- Email:** jon@snow.com
- Phone Number:** 0747811111
- Note:** Enter your current password to confirm changes.
- Action:** Update

This page follows a similar layout to the registration page, except the captcha is replaced with a text-box for the customers current password.

All the text-boxes, except for the passwords, are filled with data retrieved from the database. Whenever a customer updates their details with correctly sanitised data, they are redirected to another page, confirming their change. Of course, this is provided that the data is submitted into the database properly.

The username is removed and is unable to be changed.

Validations

- Forename
- Surname
- Passwords
- Email Address
- Phone Number

Maths.Woodhouse

http://localhost

Business Name [Jon Smith - Logout](#)

Edit Account

First Name Surname
Most details are prefilled and loaded from the database

Username Username Available!

Password Password Strength
 Repeat Password *password inserted at this stage must not be the same as the previous password.*
The Passwords do not match!

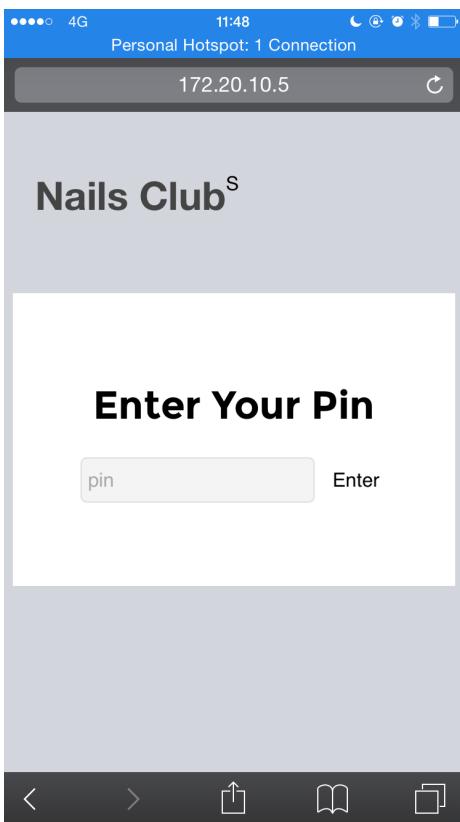
Email Repeat Email
follows similar layout to registration
validators remain the same and will show when needed
The Emails do not match!

Phone Number *This is not a recognised phone number!*

button then shows them to verify page

Next

The original design didn't include an option to confirm changes by repeating the customers original password.



3.1.11 Staff Login

This page is only accessed by staff within the local network set up in the nail shop. The staff are sent an email beforehand containing their PIN, which is inputted in the textbox. When the PIN matches they are sent to the checkins page where they can check in customers.

The UI of the staff's login page is similar to the rest of the system, and the only recognisable difference is the inclusion of the 'S' after the header. This is similar to the administrators login.

Validations

- PIN

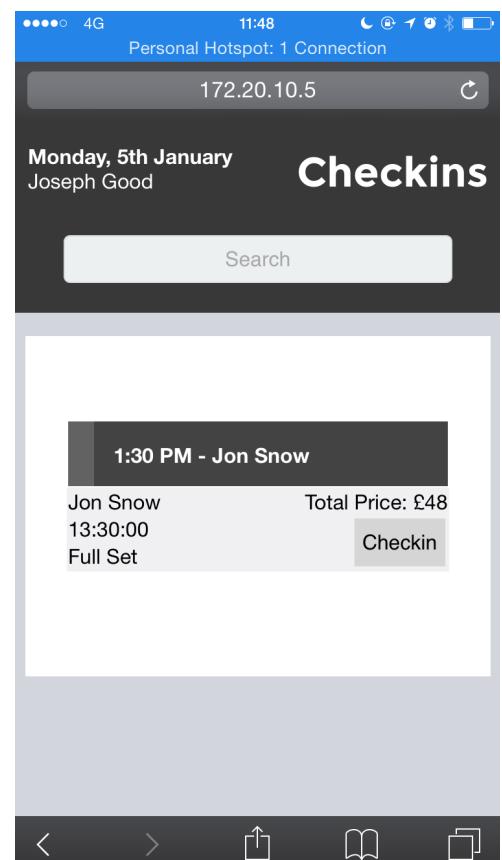
3.1.12 Staff Checkin

The Technical Solution for this part of the system has changed since the declared design decision. The original

decision was to handle checkin's using the phone's gestures but I have chosen not to do so as the mobile engines that handles touch gestures through HTML5 or Javascript is not smooth enough for me to consider it useful. For the technical solution I have chosen to put buttons instead to check in the customers.

In terms of user interface the following has been changed:

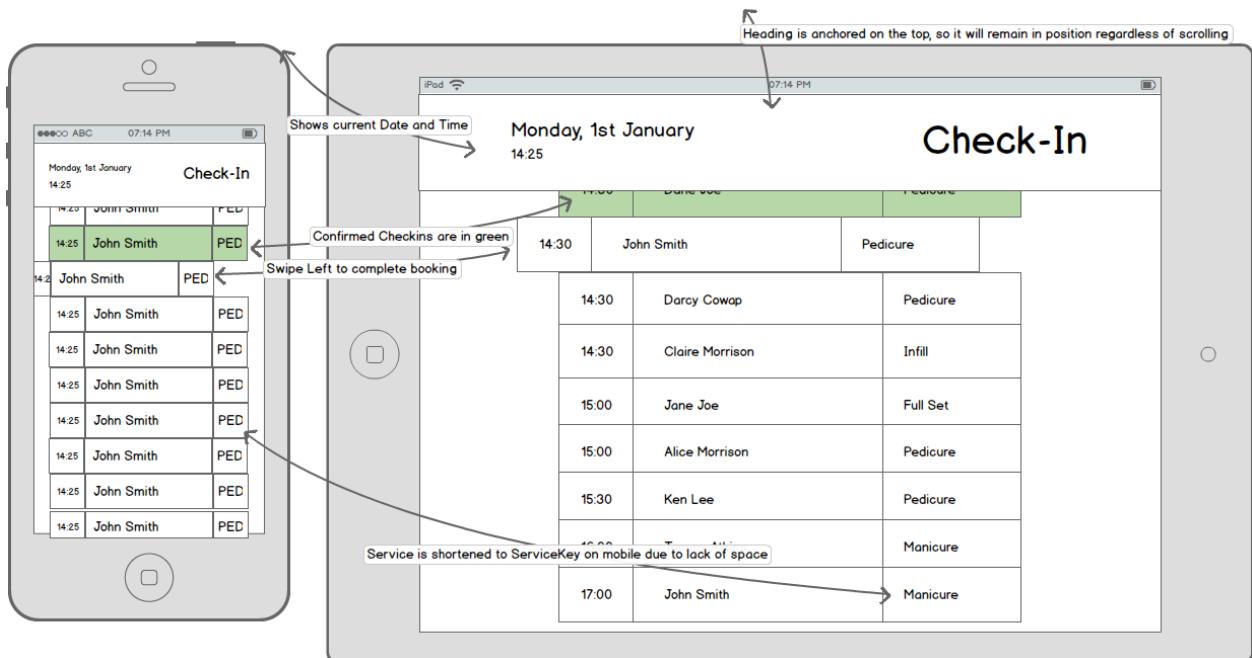
- There is now a search bar that can be used to search for surnames or forenames of the customer. This is handled with Ajax and a more detailed analysis of how this search works can be seen in the *Annotated Program Listings*.
- The time is removed as there is a time shown anyway as the phone's operating system includes such a thing, making the inclusion of the time redundant.
- A full detail of the booking is showed for every booking.
- Coloured Tabs are used to check whether the customer has checked in or not, and whether they are late.



Validations

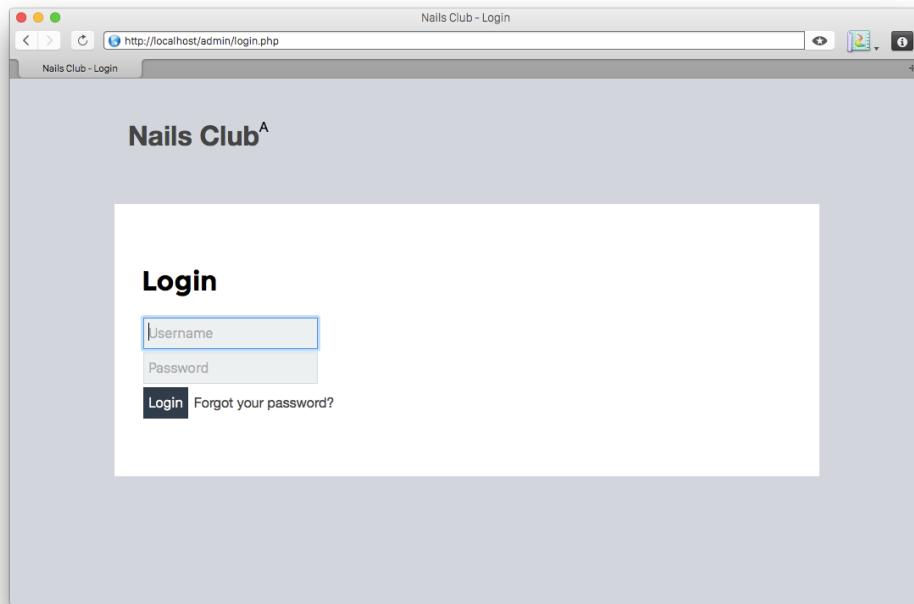
- Forename
- Surname

Scheduling & Booking System



The original design proposed the idea of checking in the customer by using phone gestures. This proved difficult in the amount of time allocated to implementing this feature in the project.

3.1.13 Administrators Login

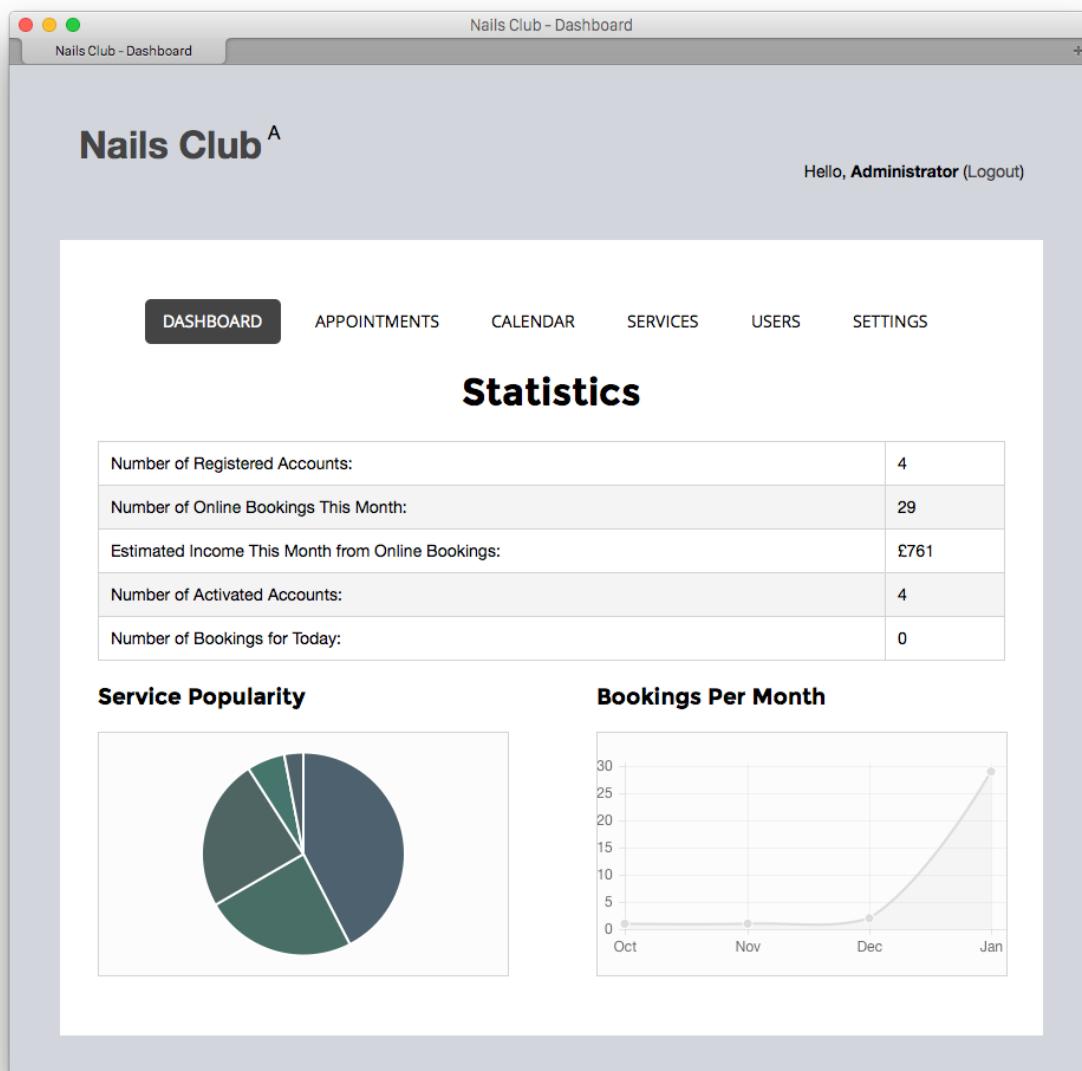


Similar to the Staff Login, The Administrators Control Panel is indicated by the letter A affixed at the end of the header. This page is used for the administrators to log into the dashboard.

Validations

- Username
- Password

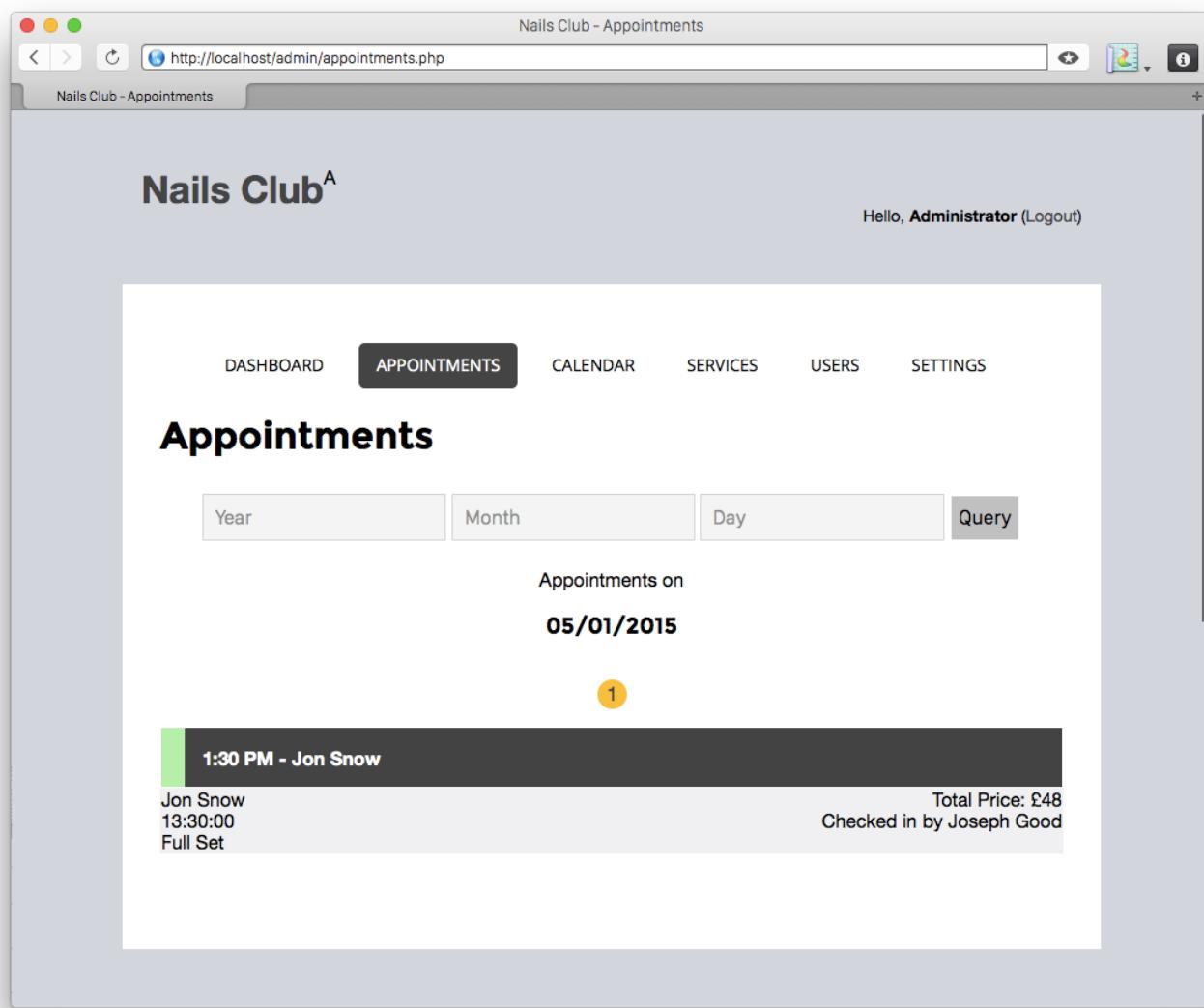
3.1.14 Administrators Dashboard



The administrators dashboard is shown in the event that the administrator logs in successfully. On this page they are given general statistics for the month. Data is pulled from the database using a variety of queries which are discussed later in this document.

They are also given a navigation bar which allows them to go through the available options that are only accessed by administrators.

At the top right, administrators can log out safely, or they are automatically timed out if they fail to do something for a set period of time. This is done to improve security measures.



3.1.15 Appointments

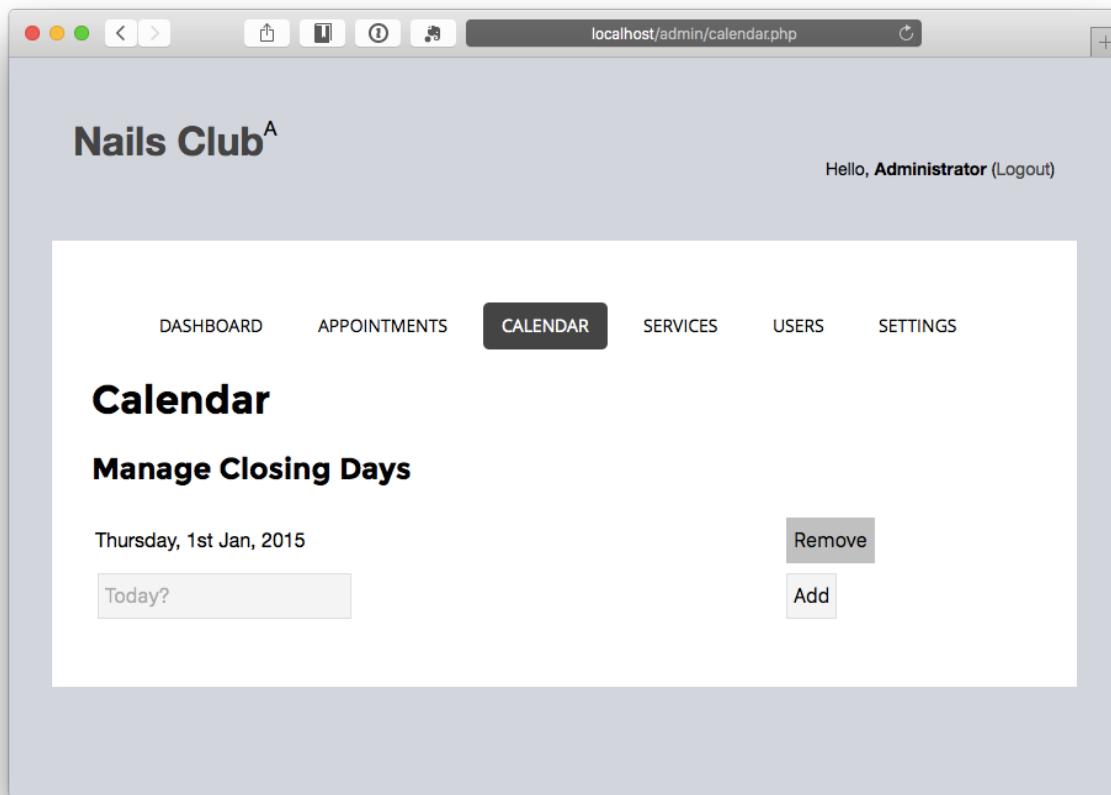
On this page administrators are given access to see appointments that have occurred on certain days. By default, it is set to show appointments occurred on the day they are accessing the website, but there are drop-downs for the year, the month and the day for which they can use to navigate through to other days. They are not given access to modify customer details or checkins.

A Pagination is used in the event that there are enough appointments that will be difficult to display without hindering the UI experience. This pagination is handled within the database query in order to preserve bandwidth and performance.

Validations

- Year
- Month
- Day

3.1.16 Calendar



In this page the administrator can choose to set closing days on the calendar. This would be saved onto the database and will be recalled when the customer loads the calendar to book an appointment. The closed days set on the database will show up as unavailable on the calendar. Days that have been set on the database will be displayed here as a result of a loop.

The administrators are also given the option to remove dates if they wish to do so.

Validations

- Date

3.1.17 Services

The screenshot shows the 'Services' management interface for the Nails Club. The top navigation bar includes links for DASHBOARD, APPOINTMENTS, CALENDAR, SERVICES (which is currently selected), USERS, and SETTINGS. The main content area is titled 'Services' and lists various nail services with their descriptions and edit/delete buttons.

Option	Value (£)	Description
Manicure	25	A luxurious beauty treatment for the fingernails and hands which includes perfectly-polished nails.
Infil	26	Description
Pedicure	29	Description
Full Set	48	Description
File & Polish	15	Description
Nail Repair	10	Description
Name	Price	Description

Selected Service Details:

Manicure £25

A luxurious beauty treatment for the fingernails and hands which includes perfectly-polished nails.

Action Buttons:

- Update
- Remove
- Add

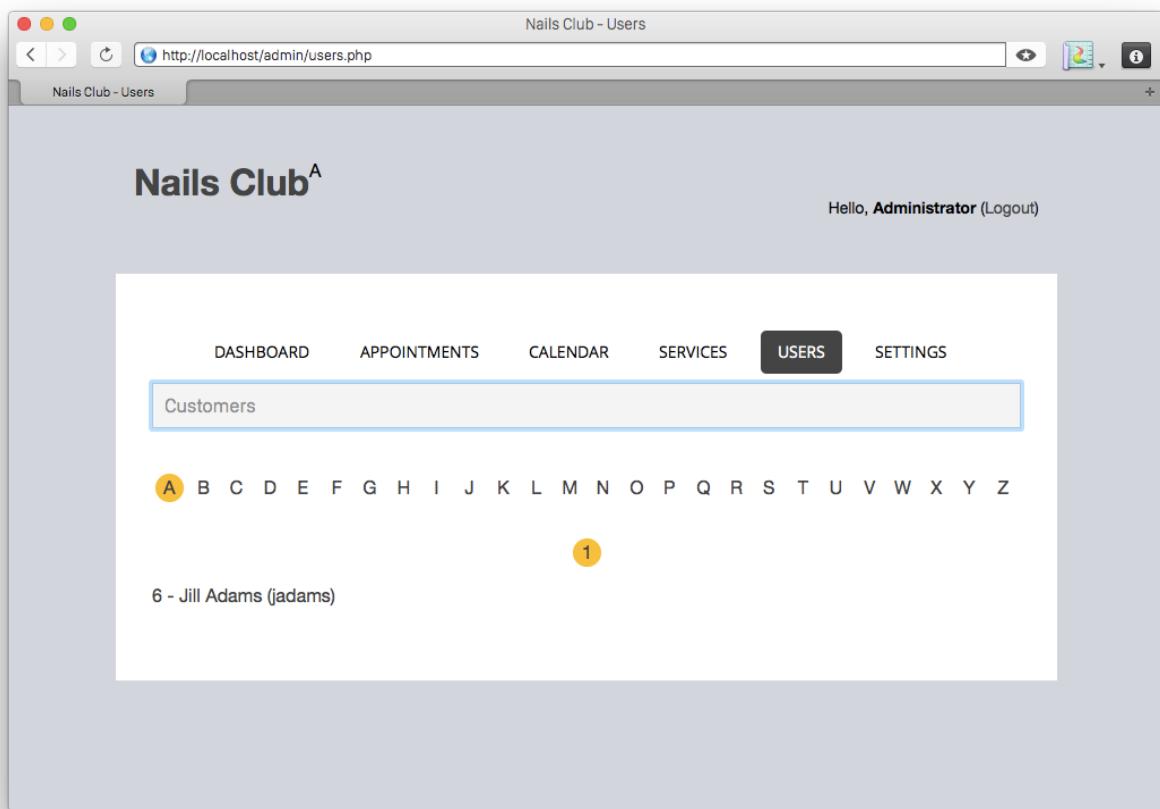
On this page the administrators can manage the current services the Nail Bar offers, allowing them to modify current ones, add new ones or remove old ones. Data is displayed as a result of a database query.

By default, the description text-box is sized to be the same height as the other text boxes, and will expand on click.

Validations

- Option (String)
- Value (Integer)
- Description (Text)

3.1.18 Users



On this page the administrator can access the different users on the site, customer or staff. These two categories are separated by the on-load and selected dropdown. By default it is set to load customers, but it can be changed to staff.

Users are displayed in order of surname in ascending order. By default, it is set to A. Pagination is also handled on this page, dividing the total number of customers loaded per page to ten.

Clicking on the customers name will direct the page to show the customers details.

Validations

- Customer/Staff Toggle (Switch)

3.1.19 Displaying User Details

The screenshot shows a web browser window titled "Nails Club - Users" with the URL "http://localhost/admin/users.php?usertype=customers&user_id=6". The page header includes "Hello, Administrator (Logout)". Below the header is a navigation menu with tabs: DASHBOARD, APPOINTMENTS, CALENDAR, SERVICES, USERS (which is selected and highlighted in dark grey), and SETTINGS. A sub-menu for "Customers" is displayed. The main content area shows a customer profile for "jadams". The profile information includes:
Name Jill Adams
Email jadams@wag1.com
Phone No. 01234567890
Statistics
Has booked a total of 0 appointments.
A small "Ban" button is visible.

On this page, Details about the customer such as the customer's phone number and their email addresses are displayed. It also shows the number of bookings that the customer has booked through the account.

Administrators are given the option to ban the customer account from booking another appointment.

3.1.20 Staff List

The screenshot shows a web browser window titled "Nails Club - Users" with the URL "http://localhost/admin/users.php?usertype=staff". The page header includes the "Nails Club - Users" logo and a "Hello, Administrator (Logout)" message. The main navigation menu at the top has tabs: DASHBOARD, APPOINTMENTS, CALENDAR, SERVICES, USERS (which is highlighted in black), and SETTINGS. Below the menu, a sub-menu for "Staff" is selected. The main content area is titled "Staff" and contains form fields for a staff member named "Joseph Good". The fields include "Forename: Joseph", "Surname: Good", and "Email: jg00d@good.com". To the right of these fields are three buttons: "Update", "Ban", and "Generate New PIN". A second staff entry is partially visible below, showing "Forename: " and buttons for "Update", "Ban", and "Generate New PIN".

Details about staff are displayed as a result of a loop that goes through each array value. On this page the administrator can add new staff or adjust current staff. The administrator is also given the option to generate a new PIN for the Staff to use. A New Pin is sent to the staff through Email.

Administrators can also ban staff who are fired or staff who have left the business. They can also be unbanned.

Validations

- Forename (String)
- Surname (String)
- Email (String)

3.1.21 Settings

The screenshot shows a web application window titled "Nails Club". The header includes a logo, the title "Nails Club", and a user greeting "Hello, Administrator (Logout)". Below the header is a navigation bar with links: DASHBOARD, APPOINTMENTS, CALENDAR, SERVICES, USERS, and SETTINGS (which is highlighted). The main content area is titled "Settings" and contains a table with six rows of configuration options:

Option	Value	Description	Action
First Slot	10:00	The time of the first slot.	<button>Update</button>
Last Slot	19:30	The time of the last slot. (Not the closing time)	<button>Update</button>
Booking Frequency	30	The amount of slots per hour, expressed in minutes.	<button>Update</button>
Booking Slots Per Day	20	The total number of slots available in one day.	<button>Update</button>
Business Name	Nails Club	The name of the business.	<button>Update</button>
Business Slogan		The slogan of the business.	<button>Update</button>

This page is used for the administrator to adjust or update settings that are set on the metadata table in the database. Each row is recalled in a loop. The administrator can set the opening and closing times, the booking frequency, the business name and the slogan.

Whenever the administrator clicks update, the row the update button is executed on will be updated, provided that the value is correctly validated.

Validations

First Slot (Time)

last Slot (Time)

Booking Frequency (Integer)

Business Name (String)

Business Slogan (String)

3.2 Annotated Program Listings

These can be found in Appendix A.

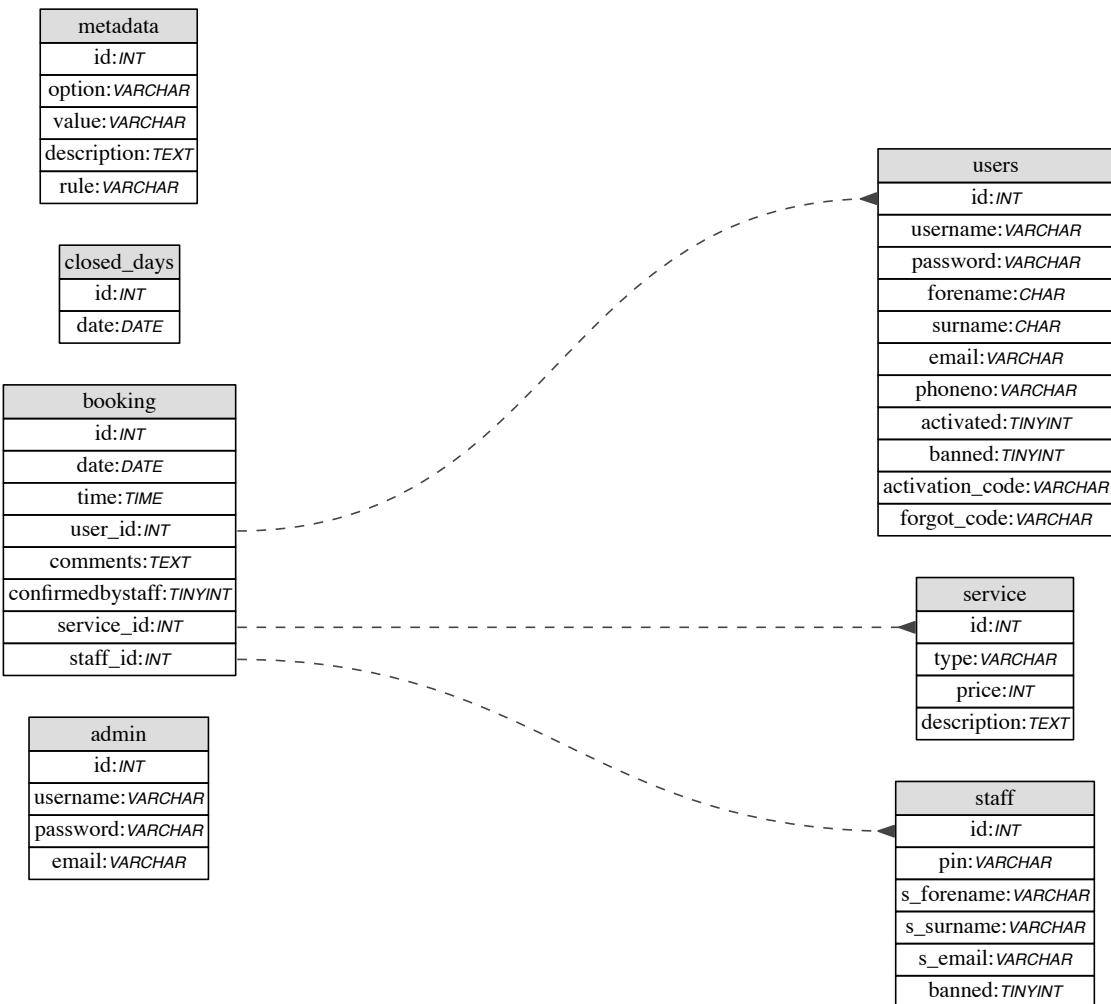
3.3 Procedure & Variable Description

for the whole list, please refer to the system maintenance.

3.4 Methods of establishing Database Connections

Please refer to the Database Class in *User Defined Classes*.

3.5 Database Tables (Final)



At this stage, there are some subtle changes to the tables. The table users have included a banned and activated column indicating extra features which will be explained later on. This contrasts with the original database layout that was indicated in the design.

3.6 How was the technical Solution Achieved?

This system was constructed over a several month period using Sublime Text. Most of the program files revolve around /includes/core.php, meaning that core.php is “included” in all php documents. Related files and functions that are also necessary for the page to function are also included. The main program’s code is included before the inclusion of /includes/header.php.

For the design, the front-end is handled by the inclusion of the system header. This file includes the links to the HTML head elements, which also include links to css and javascript libraries. Generally, the pages output is displayed after the inclusion of the header file.

A top down approach was used as I found it easier to discover the submodules that I would require in order to create the program. Also, it becomes easier to group the different program files together to make the system.

White Box testing was conducted through the progression of the system to ensure that everything works and that there are no errors as I work through the development/coding stage of the program. A more thorough testing will be described in the next section of this document.

SQL Statements were tested for results in the Terminal ensuring that results are displayed, prior to variable substitution.

Backups of the development was handled with Git. This way I am able to see a history of each file, and in the event that files are corrupt, I would be able to see a recent history of each file at each save state.

Testing

The client would be interviewed to discuss the current implementation. At this stage the client is able to discuss what they like, and what needs to be improved or added into the implementation. This is done prior to testing to ensure the client is satisfied with the system. In the *Technical Solution*, I mentioned testing techniques that were done through the progression of the system. Since that testing is undocumented, a more final and thorough testing can be enacted, and documented into this document.

Black box testing will be used to test the inputs and outputs of the system, and white box testing will be used to test the encryption function and the PIN generator function.

4.1 Test Table

#	TESTING METHOD	DATA TYPE	TEST DESC	TEST DATA	PREDICTION	OUTCOME	COMMENTS
1	Blackbox	Typical	Customer is trying to log into the system with the correct details	customer password	System will take them to their dashboard	As expected.	Please See Test Evidence 1.
2	Blackbox	Erroneous	Customer is trying to log into the system with incorrect details	Customer passw0rd	System will not take them into their dashboard, and will tell them that the credentials are incorrect.	As expected.	Please See Test Evidence 2.
3	Blackbox	Typical	Customer is trying to recover a password by using an email associated with the account they are trying to recover.	thisis.tnguyen@gmail.com">thisis.tnguyen@gmail.com	They are notified that an email is sent to that email address, discussing how to take further action.	As expected.	Please See Test Evidence 3.
4	Blackbox	Typical	Customer is trying to recover a password by using an email that is not recognised by the database.	Cust0m3r@em43l.com	They are notified that an email is sent to that email address, discussing how to take further action. An email is not actually sent to that address.	As expected.	Please See Test Evidence 4.
5	Blackbox	Typical	Staff is attempting to log in with the correct PIN	12345	They will be directed to the staffs dashboard	As expected.	Please See Test Evidence 5.
6	Blackbox	Erroneous	Staff is attempting to log in with the incorrect PIN	123123	They will be told the pin is incorrect	As expected.	Please See Test Evidence 6.
7	Blackbox	Typical	Administrators is attempting to log in with the correct username and password.	Admin password	They are directed to the administrators dashboard	As expected.	Please See Test Evidence 7.
8	Blackbox	Erroneous	Administrators is attempting to log in with the correct username and an incorrect password.	Admin p4dsa23ff	They are told that the credentials are incorrect	As expected.	Please See Test Evidence 8.
9	Blackbox	Erroneous	Administrators is attempting to log in with an incorrect username and incorrect password	Adm3n Partawfb2@	they are told that the credentials are incorrect	As expected.	Please See Test Evidence 9.
10	Blackbox	Typical	Customer is attempting to click on "Book an appointment" button	mouseclick	They are directed to the appointment page where they can make an appointment.	As expected.	Please See Test Evidence 10.
11	Blackbox	Typical	Customer is attempting to click on "Manage Appointments" button	mouseclick	They are directed to the "manage appointments" page.	As expected.	Please See Test Evidence 11.

Scheduling & Booking System

#	TESTING METHOD	DATA TYPE	TEST DESC	TEST DATA	PREDICTION	OUTCOME	COMMENTS
12	Blackbox	Typical	Customer clicks on a fully available day	mouseclick	The day is parsed onto the system and they can continue with the booking process	As expected.	Please See Test Evidence 12.
13	Blackbox	Boundary	Customer clicks on a partially available day	mouseclick	The day is parsed onto the system and they can continue with the booking process	As expected.	Please See Test Evidence 13.
14	Blackbox	Erroneous	Customer clicks on a closed day	mouseclick	nothing happens	As expected.	Please See Test Evidence 14.
15	Blackbox	Erroneous	Customer clicks on a day which is on the following month	mouseclick	nothing happens	As expected.	Please See Test Evidence 15.
16	Blackbox	Erroneous	Customer does not select a service and attempts to book the appointment	not applicable	They are not directed on to the next page, and they are informed that they need to select a service.	As expected.	Please See Test Evidence 16.
17	Blackbox	Erroneous	Customer does not select a time period and attempts to book the appointment	not applicable	They are not directed on to the next page, and they are informed that they need to choose a time period of the booking.	As expected.	Please See Test Evidence 17.
18	Blackbox	Typical	Customer registers a username that is available	customer2	They are notified that the username is available	As expected.	Please See Test Evidence 18.
19	Blackbox	Erroneous	Customer registers a username that is not available	customer	They are notified that the username is not available	As expected.	Please See Test Evidence 19.
20	Blackbox	Typical	Customer types in a password that is eligible	passw0rD	The password is parsed onto the function that generates the password strength bar	As expected.	Please See Test Evidence 20.
21	Blackbox	Erroneous	Customer registers with a password that is not eligible	pass	They are notified that the password does not meet the requirements. (Password too short)	As expected.	Please See Test Evidence 21.
22	Blackbox	Typical	Customer types in a password that matches	passw0RD passw0RD	Nothing happens	As expected.	Please See Test Evidence 22.
23	Blackbox	Erroneous	Customer types in a different value in the match password textbox.	password p4q7thegh	They are told that the passwords do not match	As expected.	Please See Test Evidence 23.
24	Blackbox	Typical	Customer types in value in the email textbox that contains the "@" sign and a period.	john@doe.com	Nothing happens	As expected.	Please See Test Evidence 24.

Scheduling & Booking System

#	TESTING METHOD	DATA TYPE	TEST DESC	TEST DATA	PREDICTION	OUTCOME	COMMENTS
25	Blackbox	Erroneous	Customer types in a value in the email textbox that does not contain an "@" sign	johndoe.com	They are told that this email is invalid	As expected.	Please See Test Evidence 25.
26	Blackbox	Erroneous	Customer types in a value in the email textbox that does not contain a period.	john@doecom	they are told that this email is invalid	As expected.	Please See Test Evidence 26.
27	Blackbox	Typical	Customer types in a value in the verify email textbox that matches the value in the email textbox	john@doe.com john@doe.com	They told that the emails match	As expected.	Please See Test Evidence 27.
28	Blackbox	Erroneous	Customer types in a value in the verify email textbox that does not match the value in the email textbox.	john@doe.com sean@kingston.net	They are told that the emails do not match	As expected.	Please See Test Evidence 28.
29	Blackbox	Typical	Customer types in the captcha incorrectly during the booking process.	Not applicable	They are not directed on to the next page, and they are informed that they need to retype the captcha	As expected.	Please See Test Evidence 29.
30	Blackbox	Typical	Customer types in a number value that is 11 digits long in the phone number field.	01234567891	Nothing happens	As expected.	Please See Test Evidence 30.
31	Blackbox	Erroneous	Customer types in a number value that is not 11 digits long in the phone number field.	1	They are informed that they need to retype the phone number.	As expected.	Please See Test Evidence 31.
32	Blackbox	Typical	Staff attempts to validate a checkin by clicking (tapping) the checkin box related to the customer's name and time.	finger tap	The row bounces back to their matching columns and the row is given a green background	As expected.	Please See Test Evidence 32.
33	Blackbox	Typical	Staff attempts to undo a checkin by clicking (tapping) the uncheck box related to the customer's name and time.	finger tap	nothing happens	As expected.	Please See Test Evidence 33.
34	Blackbox	Typical	Staff types in a customers name in the search field.	Claire	A preset appointment under the name Claire Snow will display as a result.	As expected.	Please See Test Evidence 34.
35	Blackbox	Typical	Admin adds a new closing date with 2015-05-01	2015/5/1	Date is saved in database, is notified of change.	As expected.	Please See Test Evidence 35.

Scheduling & Booking System

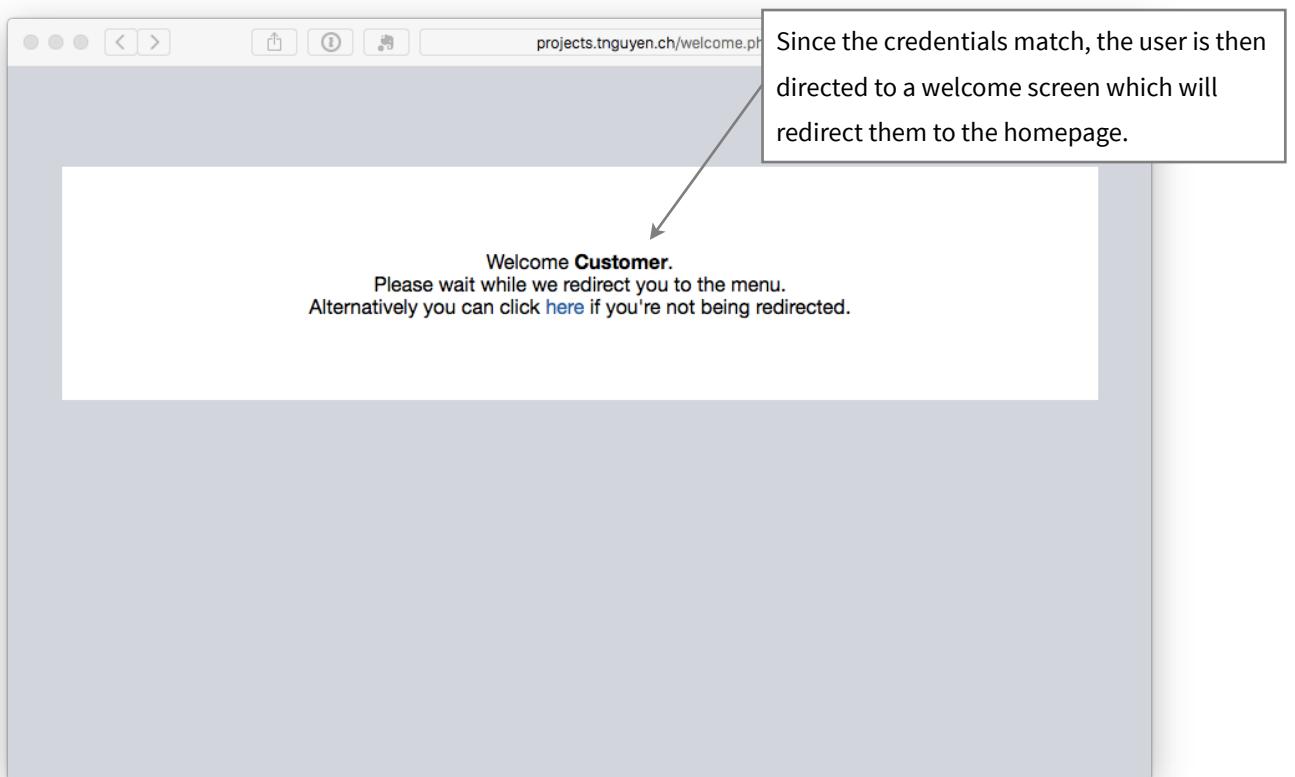
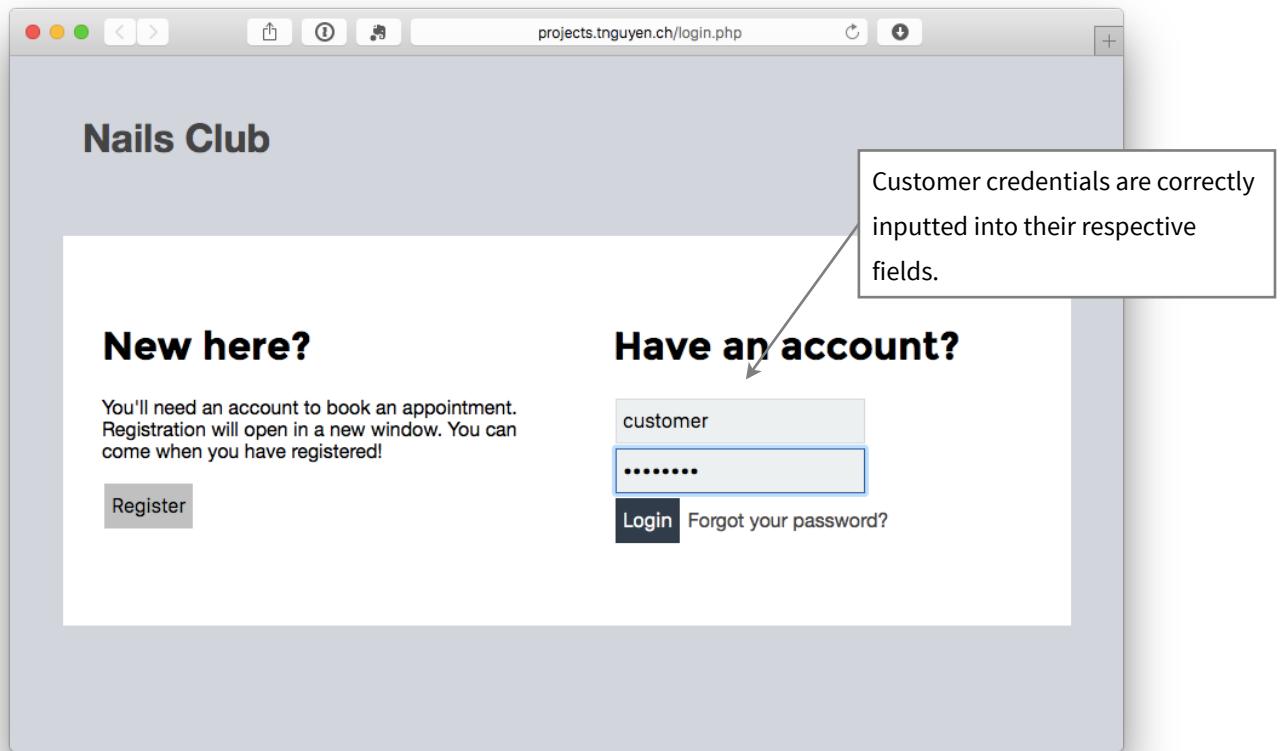
#	TESTING METHOD	DATA TYPE	TEST DESC	TEST DATA	PREDICTION	OUTCOME	COMMENTS
36	Blackbox	Boundary	Admin adds a new closing date with the current day. The day of the testing is 2015/01/28	2015/1/28	Date is saved in database, is notified of change.	As expected.	Please See Test Evidence 36.
37	Blackbox	Erroneous	Admin adds a new closing date.	2015/13/01	Error message saying that this is not a valid date should occur.	As expected.	Please See Test Evidence 37.
38	Blackbox	Erroneous	Admin selects a date in the appointments day.	2015/02/31	An error message saying that the date is invalid is displayed.	As expected.	Please See Test Evidence 38.
39	Blackbox	Boundary	Admin selects a date in the appointments day.	2015/1/1	Results are showed, but there are no results on that day.	As expected.	Please See Test Evidence 39.
40	Blackbox	Typical	Admin selects a date in the appointments day.	2015/1/5	Results will display for appointments booked on 2015/1/5	As expected.	Please See Test Evidence 40.
41	Blackbox	Erroneous	Admin adds a new option with numbers in the title, no Price, No description.	Acrylic Refil	Error messages will be displayed telling the admin that they need to input a price and a description.	As expected.	Please See Test Evidence 41.
42	Blackbox	Erroneous	Admin adds a new option with numbers in the title, a number value for price, No description.	Acrylic Refil, 18	Error messages will be displayed telling the admin that they need to input a description.	As expected.	Please See Test Evidence 42.
43	Blackbox	Typical	Admin adds a new option with correct values for title, a number value for price, and a description.	Acrylic Refil, 18, A maintenance service for Acrylic nails.	A message is displayed saying that the query is added.	As expected.	Please See Test Evidence 43.
44	Blackbox	Erroneous	Admin adjusts the first slot.	25:00	An error message is displayed saying that the input is an invalid time.	As expected.	Please See Test Evidence 44.
45	Blackbox	Typical	Admin adjusts the first slot.	08:00	A message is displayed saying that the information is updated into the database.	As expected.	Please See Test Evidence 45.
46	Blackbox	Erroneous	Admin adjusts the last slot.	25:00	An error message is displayed saying that the input is an invalid time.	As expected.	Please See Test Evidence 46.
47	Blackbox	Typical	Admin adjusts the last slot.	20:00	A message is displayed saying that the information is updated into the database.	As expected.	Please See Test Evidence 47.
48	Blackbox	Erroneous	Admin adjusts the booking frequency.	20	A message is displayed saying that the information is updated into the database.	As expected.	Please See Test Evidence 48.
49	Blackbox	Typical	Admin adjusts the booking frequency.	-100	An error message is displayed saying that the booking frequency is either too long or too short.	As expected.	Please See Test Evidence 49.

Scheduling & Booking System

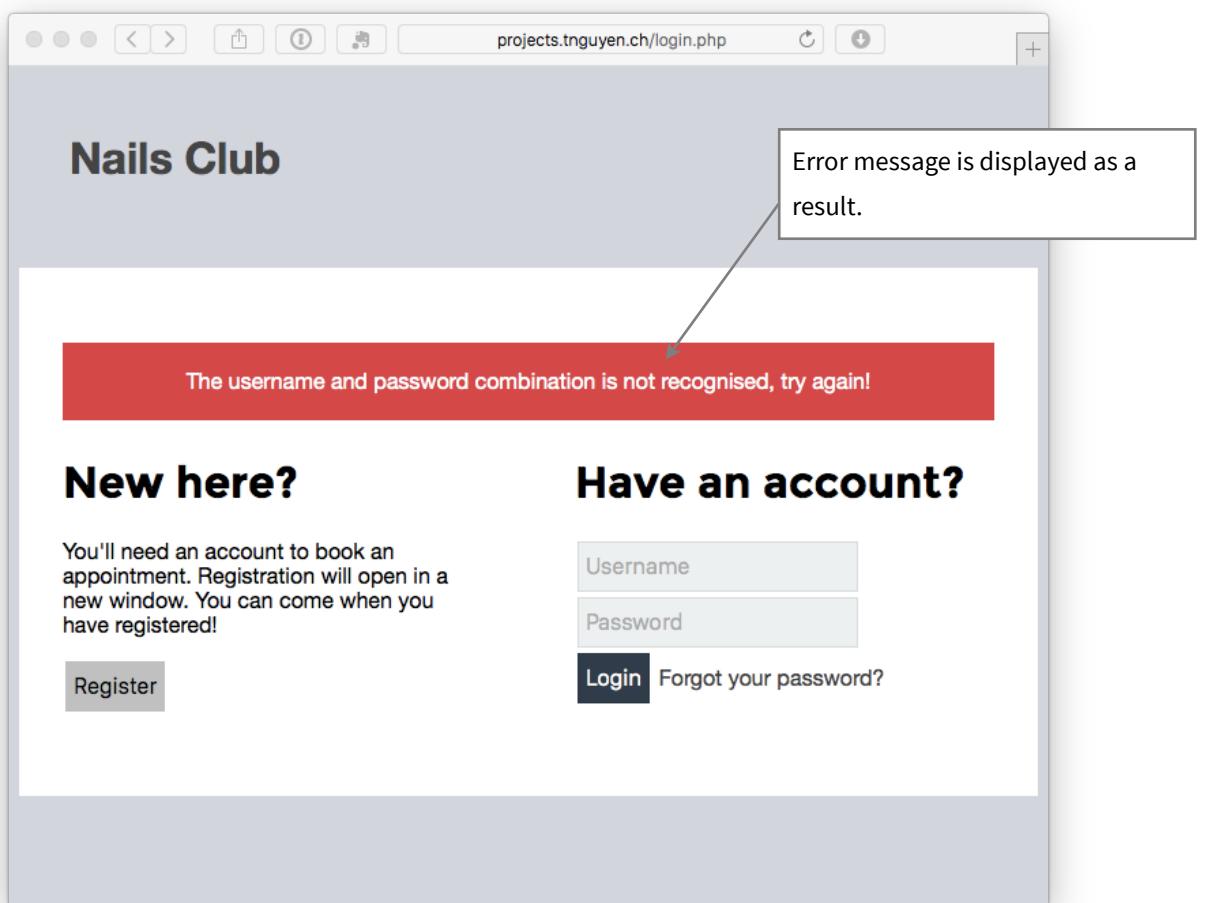
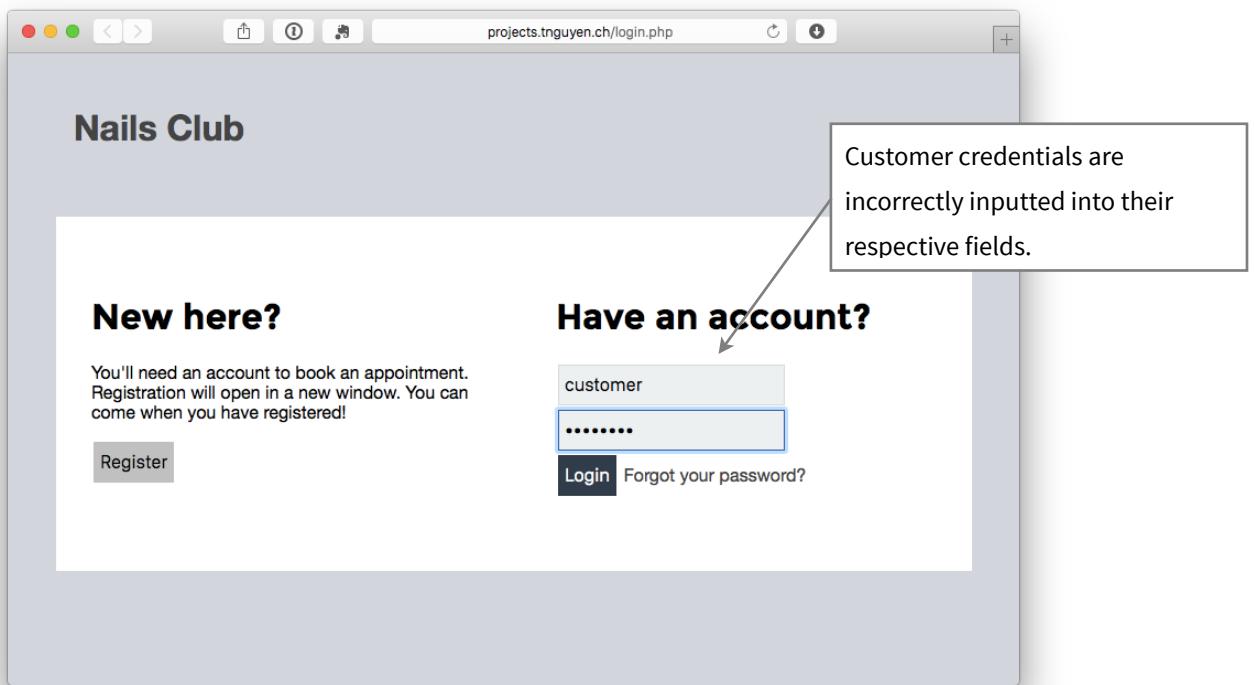
#	TESTING METHOD	DATA TYPE	TEST DESC	TEST DATA	PREDICTION	OUTCOME	COMMENTS
50	Blackbox	Typical	Admin changes the Business name.	Business Name	A message is displayed saying that the information is updated into the database. Also, the bold header on the top left of all pages will be replaced with the new input.	As expected.	Please See Test Evidence 50.
51	Blackbox	Boundary	Admin changes the slogan. The test data is 123 characters long.	Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.	An error message is displayed saying that the slogan is too long.	As expected.	Please See Test Evidence 51.
52	Blackbox	Typical	Admin changes the slogan. The test data is 13 characters long.	Lorpem Ipsum.	A message is displayed saying that the information is updated into the database. Also, A slogan is shown underneath the top header where the company name is placed.	As expected.	Please See Test Evidence 52.
53	White Box	Erroneous	Customer attempts to change the day of booking. Chosen day is set to be closed.	http://projects.tnguyen.ch/calendar.php?month=01&year=2015&day=01	A message is displayed saying that the date is unavailable for booking. The message will contain a reason, of which this case it will be that the day is closed.	As expected.	Please See Test Evidence 53.
54	White Box	Erroneous	Customer attempts to change the day of booking. Chosen day is a sunday.	http://projects.tnguyen.ch/calendar.php?month=02&year=2015&day=01#selected_date	A message is displayed saying that the date is unavailable for booking. The message will contain a reason, of which this case it will be that the day is a sunday.	As expected.	Please See Test Evidence 54.
55	White Box	Erroneous	Customer attempts to change the day of booking. Chosen day is beyond the limit of schedule.	http://projects.tnguyen.ch/calendar.php?month=02&year=2016&day=01#selected_date	A message is displayed saying that the date is unavailable for booking. The message will contain a reason, of which this case it will be that the day is out of bounds of the booking threshold.	As expected.	Please See Test Evidence 55.
56	Whitebox	Typical	Testing the result of the hashing algorithm.	password, password2	a randomly scrambled hash will be generated.	As expected.	Please See Test Evidence 56.
57	Whitebox	Typical	Testing the result of PIN generation. The chosen staff is originally allocated with a random PIN. The staff is then given another PIN.	mouseclick	On first click, the staff is given a PIN of 12345. (This is done to ensure that the second pin will be different to this PIN.) On the second click, the PIN would be a different to 12345.	As expected.	Please See Test Evidence 57.

4.2 Test Evidence

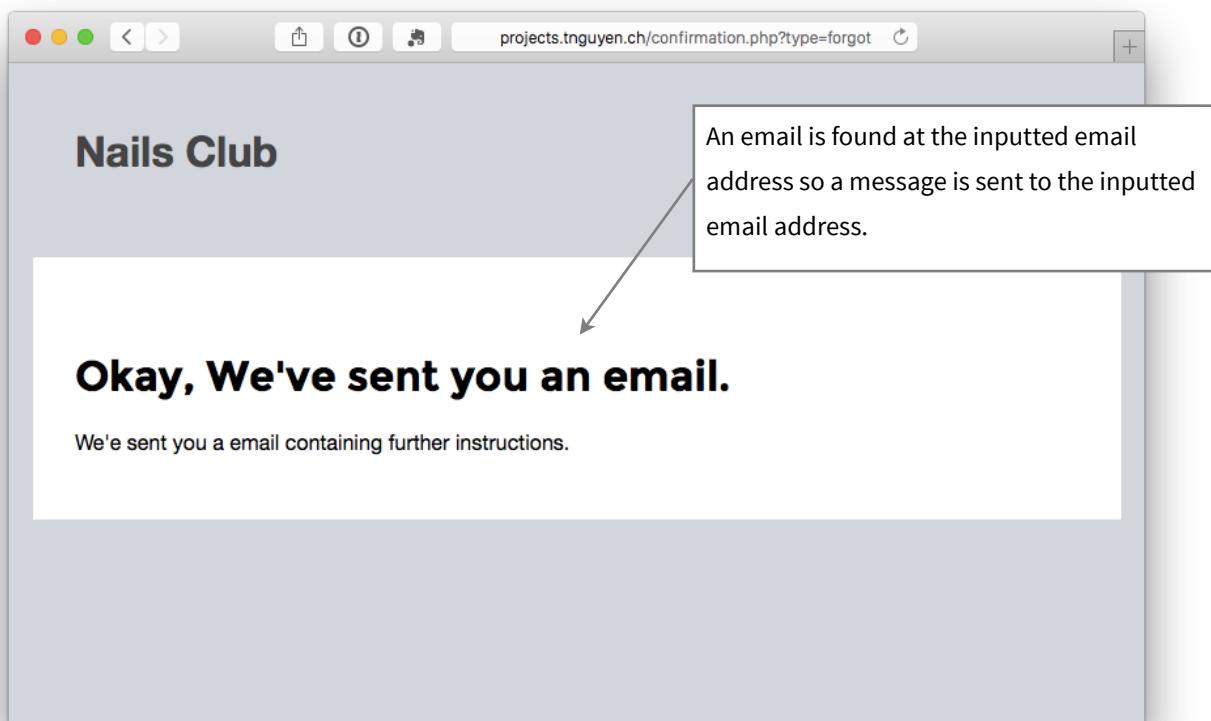
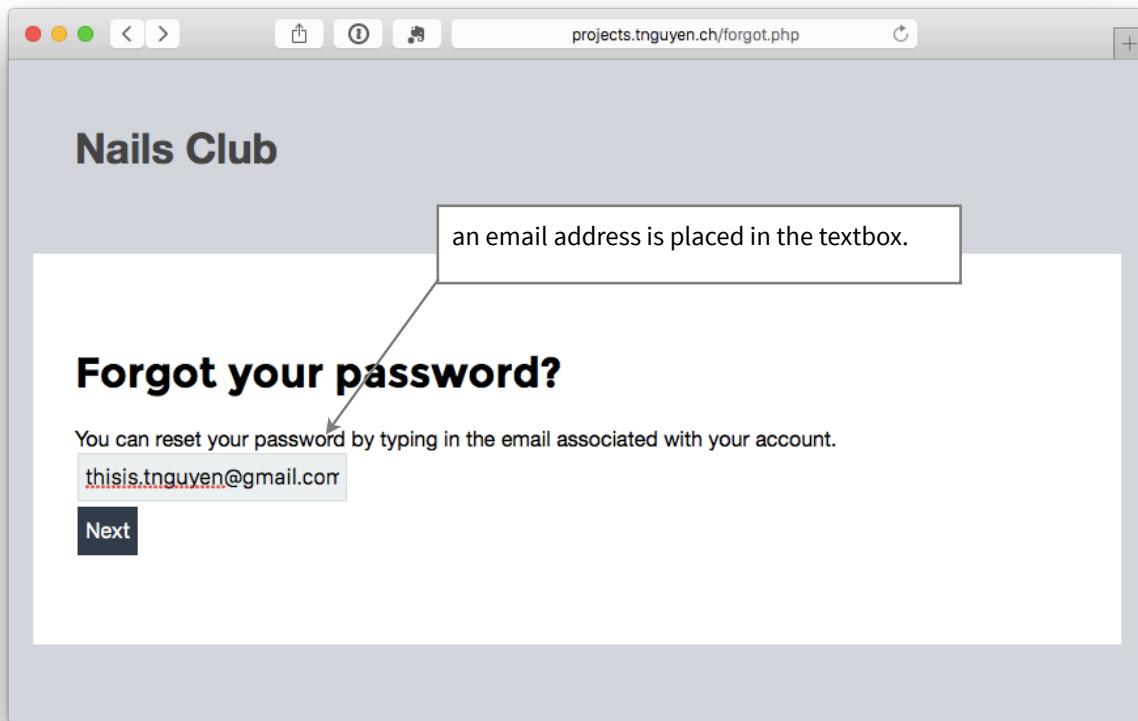
Test Evidence 1:

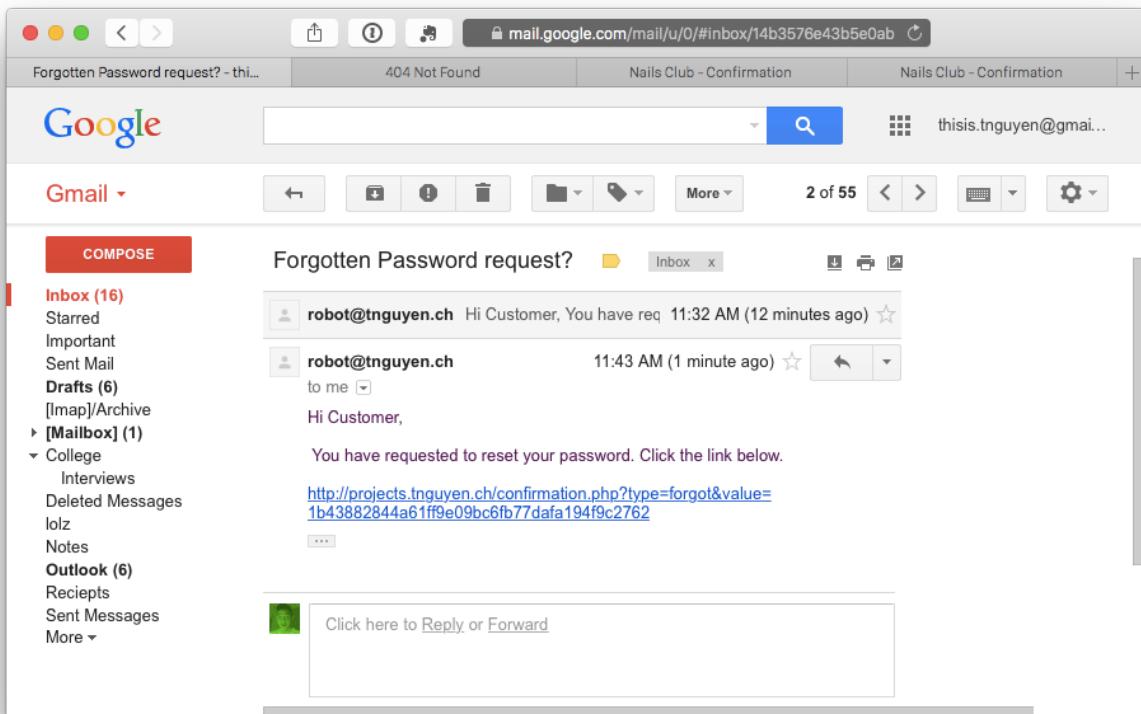


Test Evidence 2:

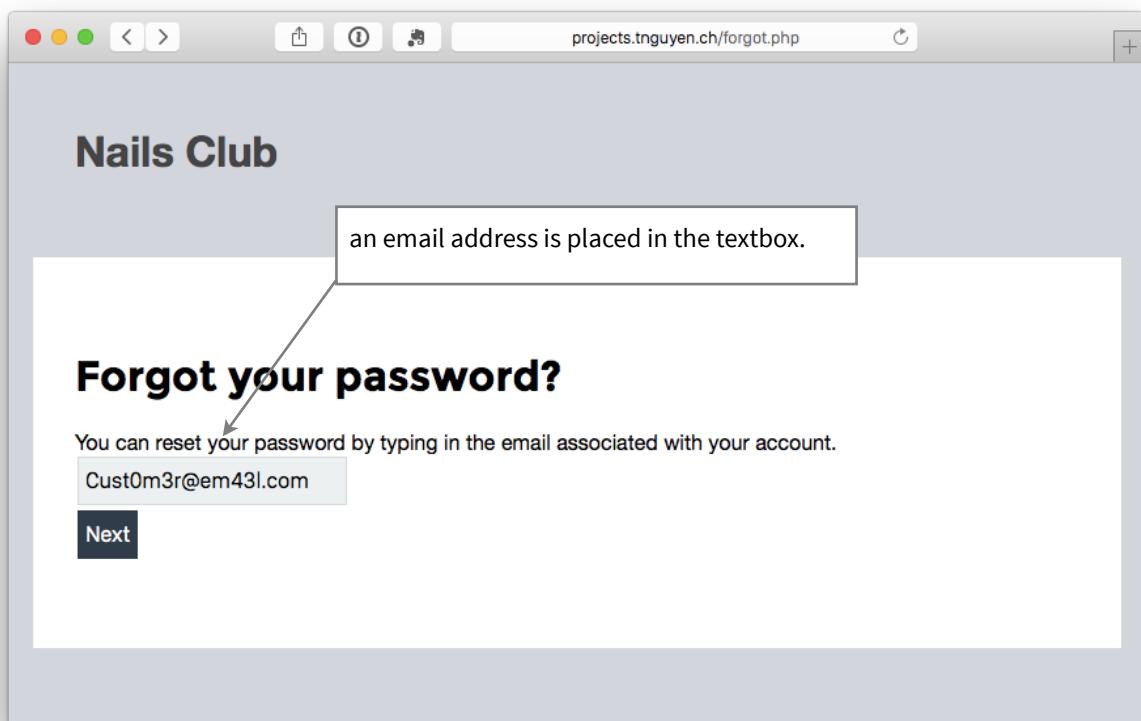


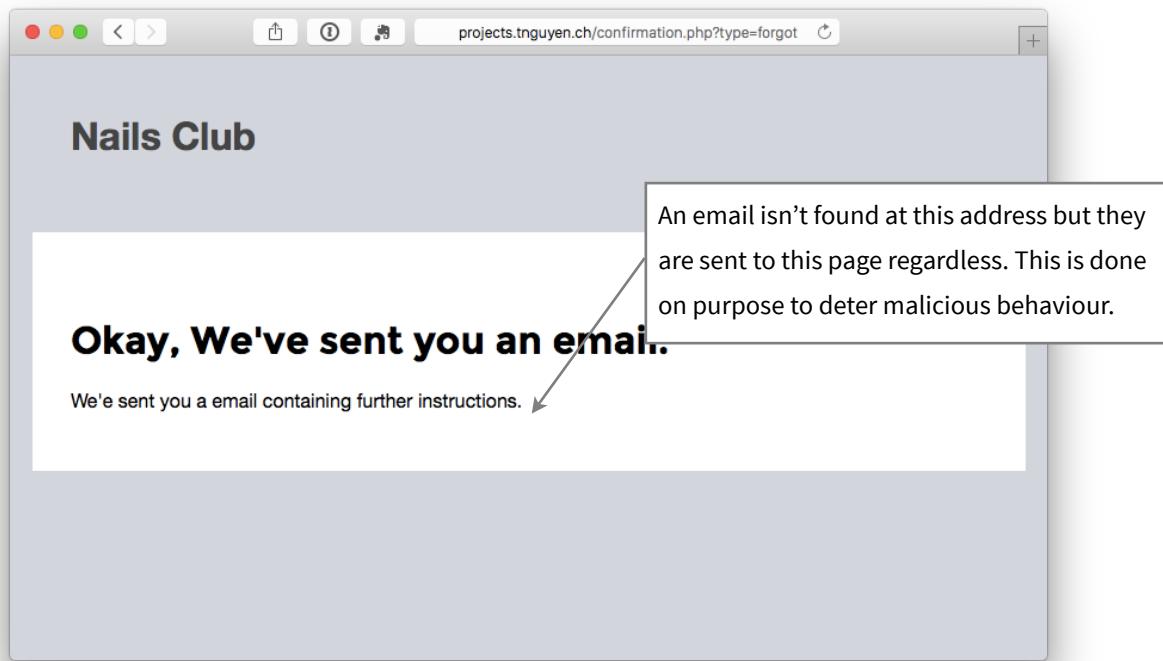
Test Evidence 3:





Test Evidence 4:





Test Evidence 5:

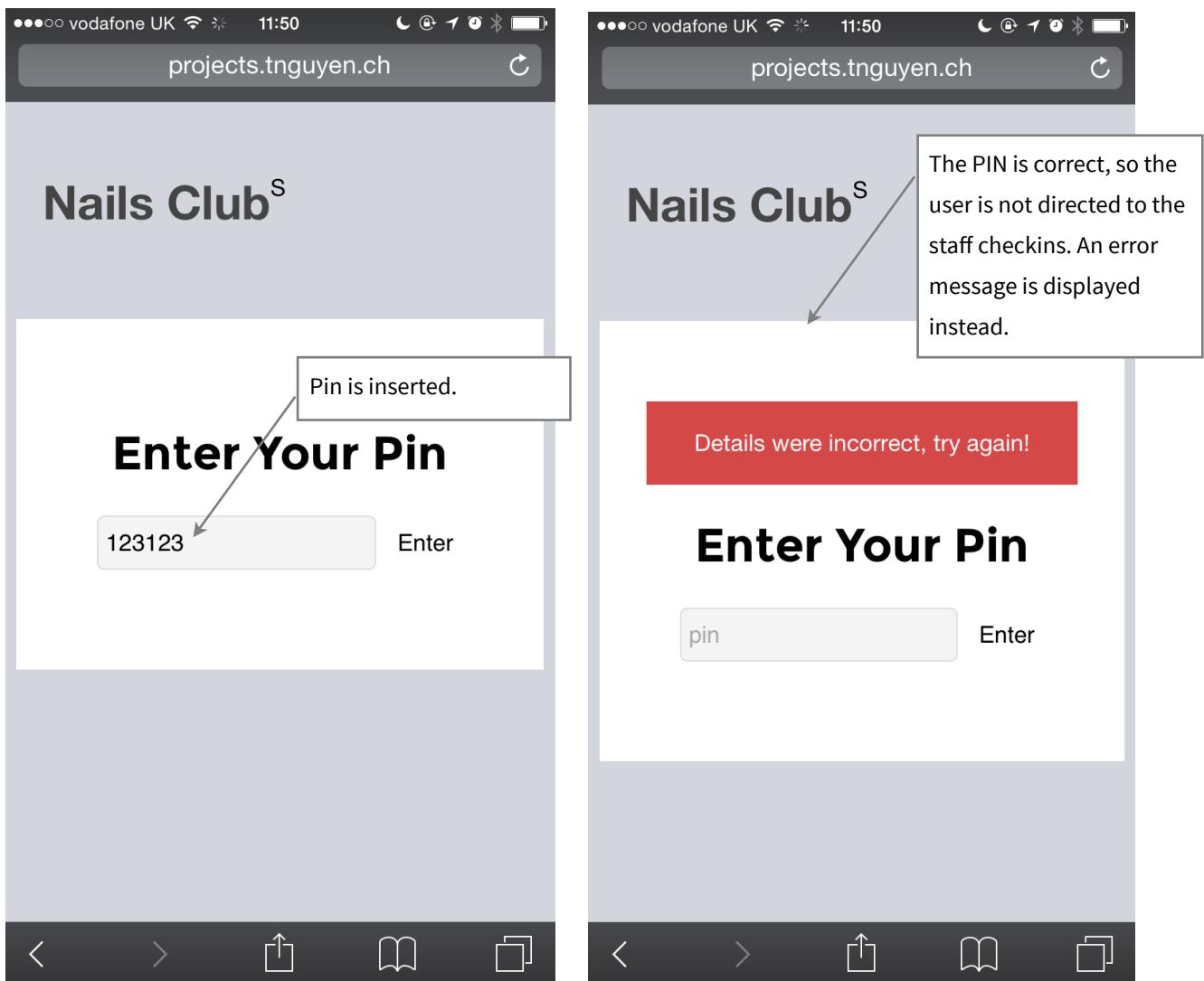
The image contains two side-by-side screenshots of mobile devices. Both screens show the URL projects.tnguyen.ch.

Left Screen (Pin Entry): The title is "Nails Club". It displays a large button labeled "Enter Your Pin". Below it is a text input field containing "12345" and a button labeled "Enter". A callout box with an arrow points to the input field, containing the text: "Pin is inserted."

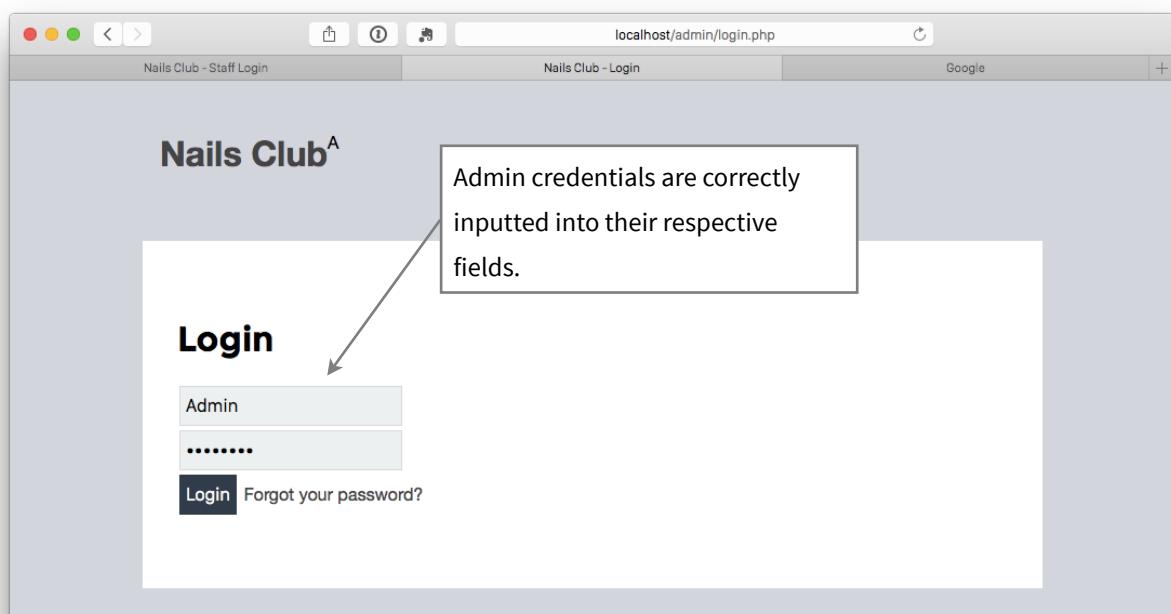
Right Screen (Checkins): The title is "Checkins". It shows a list of bookings. The first booking is for "11:30 AM - Jon Snow" with details: "Jon Snow", "11:30:00", "Pedicure", and a "Checkin" button. The second booking is for "12:00 PM - Henry Ford". Both screens have standard iOS-style navigation bars at the bottom.

A callout box with an arrow points to the "Checkin" button on the right screen, containing the text: "The PIN is correct, so the user is directed to the staff checkins."

Test Evidence 6:



Test Evidence 7:



The screenshot shows a web browser window titled "localhost/admin/index.php". The title bar also displays "Nails Club - Staff Login", "Nails Club - Dashboard", and "Google". The main content area is titled "Nails Club^A". A top navigation bar includes links for DASHBOARD, APPOINTMENTS, CALENDAR, SERVICES, USERS, and SETTINGS. Below this is a section titled "Statistics" containing a table with the following data:

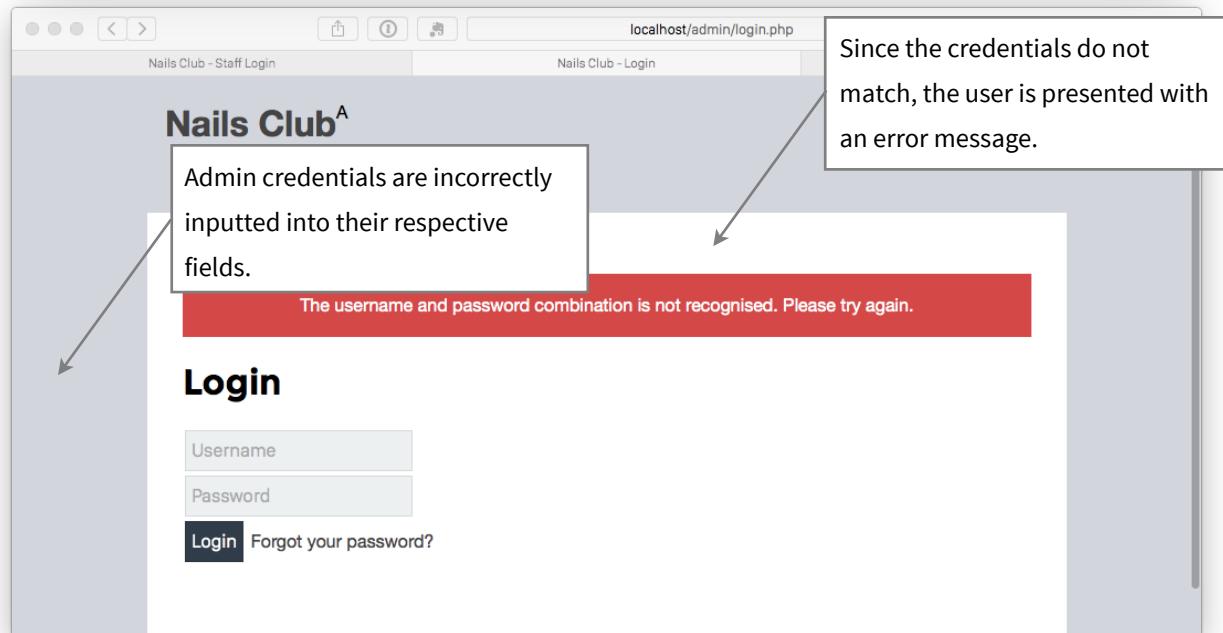
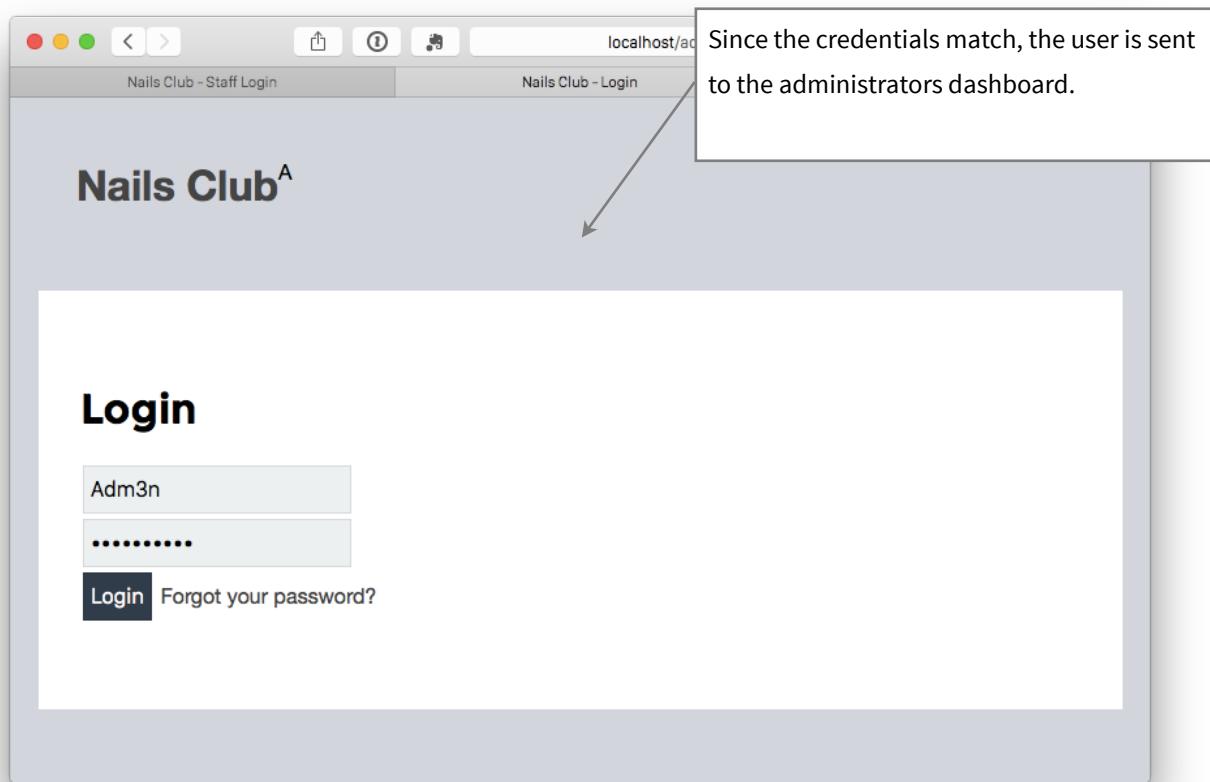
Number of Registered Accounts:	4
Number of Online Bookings This Month:	28
Estimated Income This Month from Online Bookings:	£736
Number of Activated Accounts:	4
Number of Bookings for Today:	1

Test Evidence 8:

The screenshot shows a login form for "Nails Club^A". The form has two input fields: "Username" (containing "Admin") and "Password" (containing "....."). Below the password field is a blue button labeled "Login". To the right of the "Login" button is a link "Forgot your password?".

The screenshot shows the same login form as the previous one, but with an additional red error message box at the top stating: "The username and password combination is not recognised. Please try again." The rest of the form and layout are identical to the first screenshot.

Test Evidence 9:



Test Evidence 10:

The image shows two screenshots of a web application. The top screenshot is the 'Nails Club - Home' page, which displays two main buttons: 'Book an Appointment' and 'Manage Appointments'. A callout box with a yellow border and black text points to the 'Book an Appointment' button, stating: 'Admin credentials are incorrectly inputted into their respective fields.' A large red arrow points from this callout down to the second screenshot. The bottom screenshot is the 'Nails Club - Book Appointment' page. It features a calendar for January 2015 with days numbered 1 through 18. Above the calendar, a callout box with a yellow border and black text states: 'Admin credentials do not match, user is presented with an error message.' Below the calendar, there are two buttons: 'BOOK APPOINTMENT' and 'MANAGE APPOINTMENTS'.

Admin credentials are incorrectly inputted into their respective fields.

Hello, Jon (Logout)

Book an Appointment

Manage Appointments

Nails Club

Admin credentials do not match, user is presented with an error message.

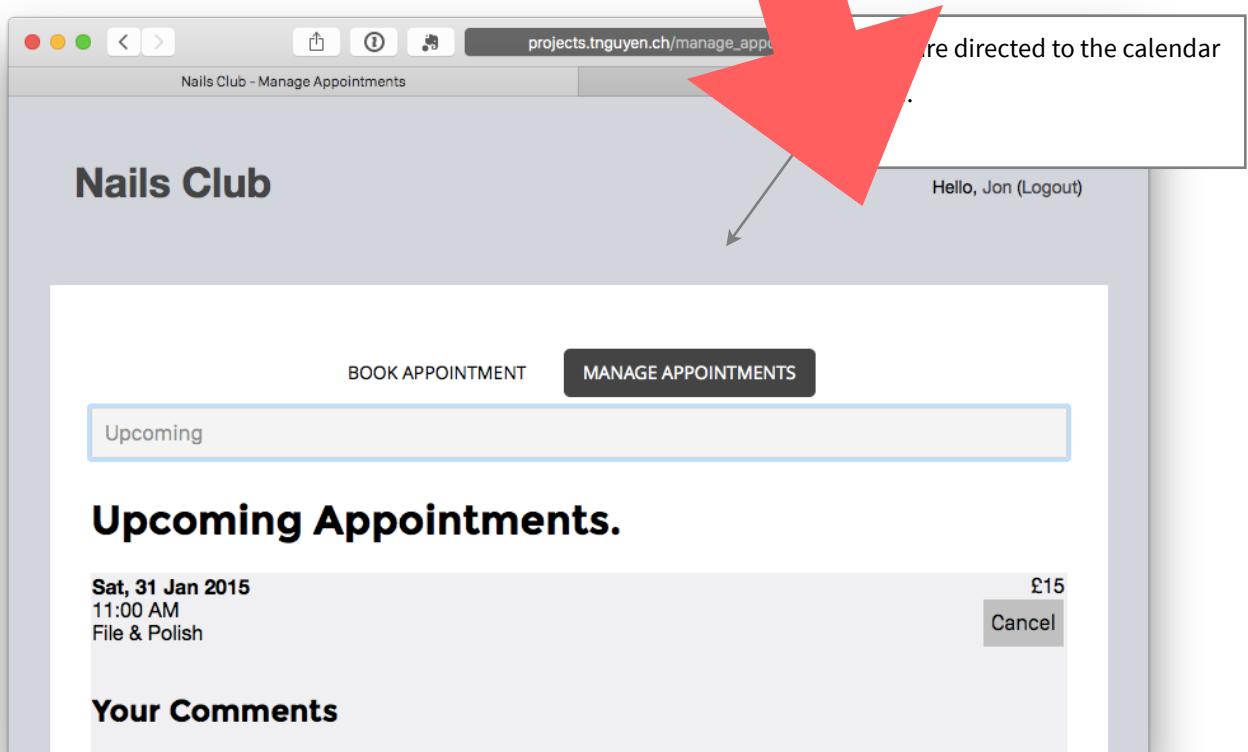
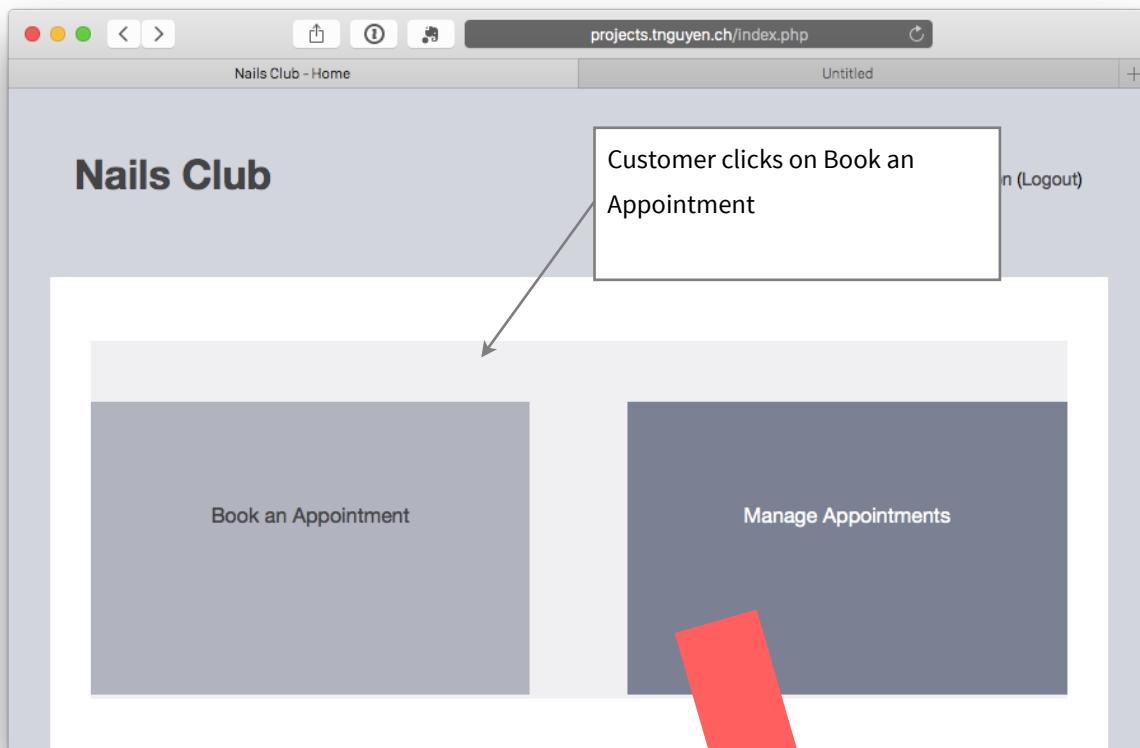
Hello, Jon (Logout)

BOOK APPOINTMENT MANAGE APPOINTMENTS

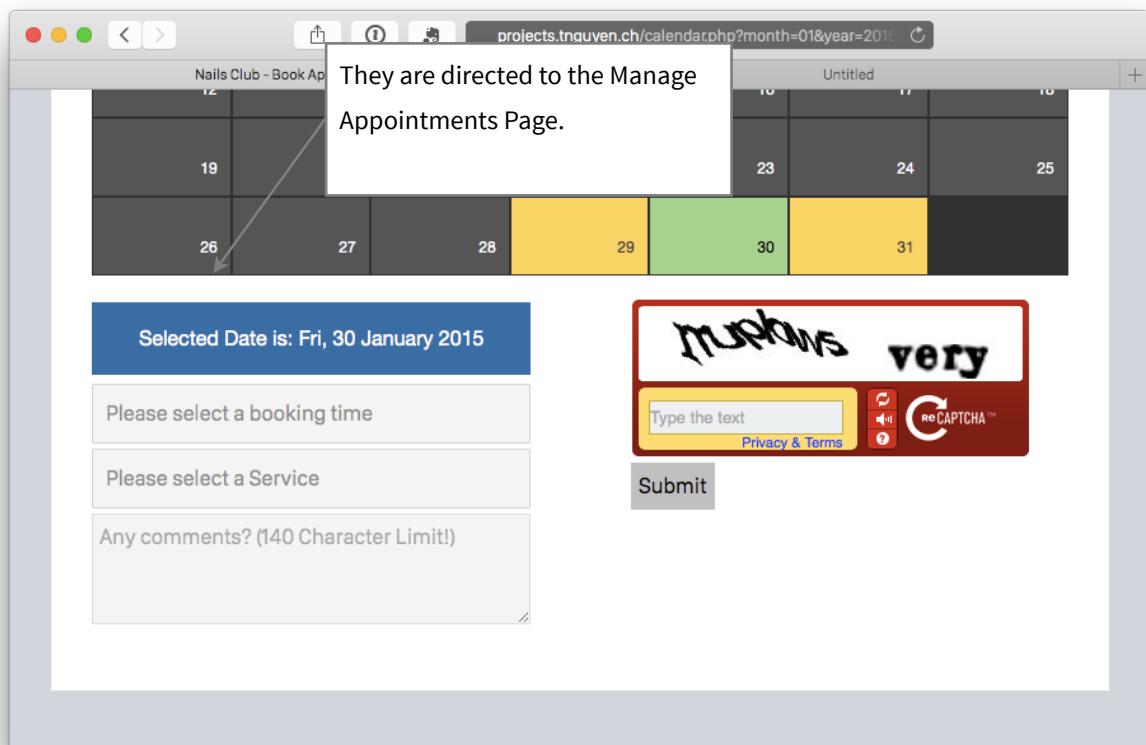
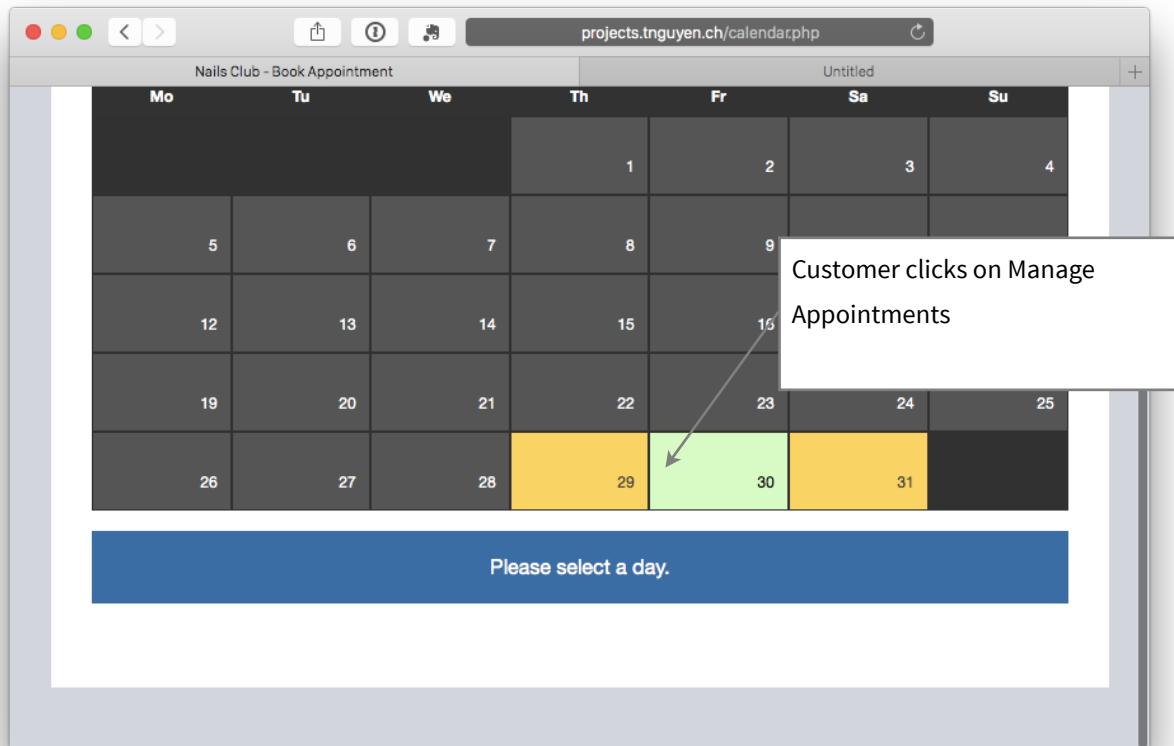
← January, 2015 →

Mo	Tu	We	Th	Fr	Sa	Su
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18

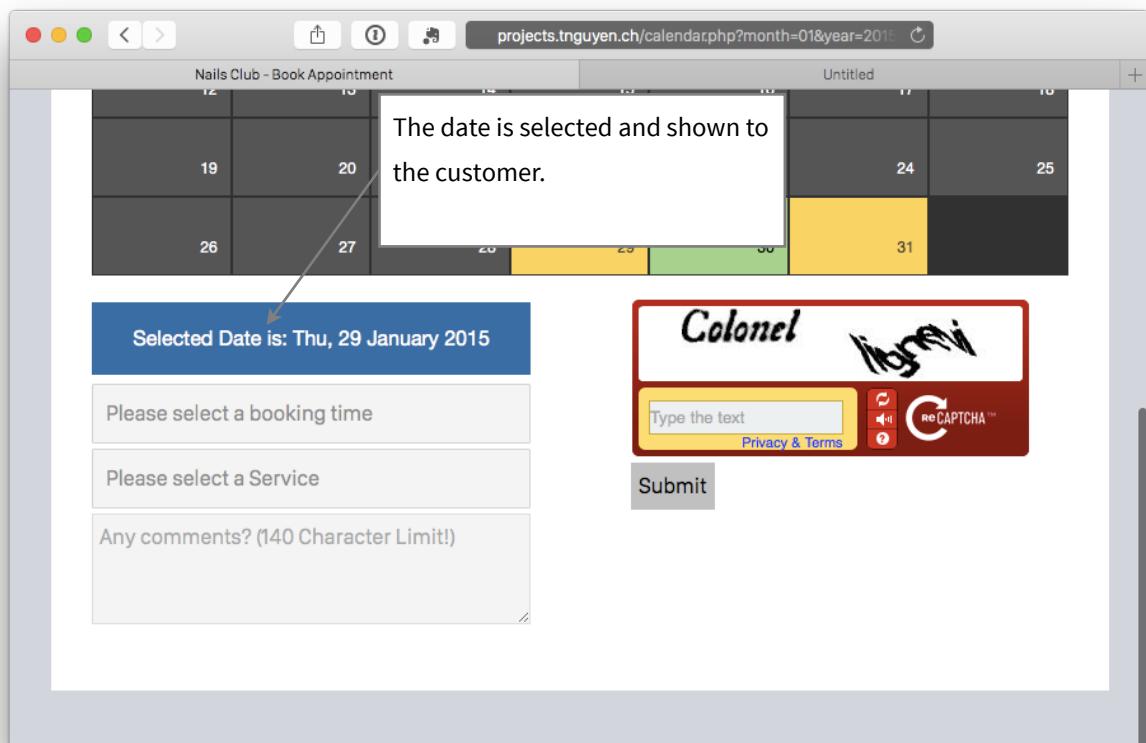
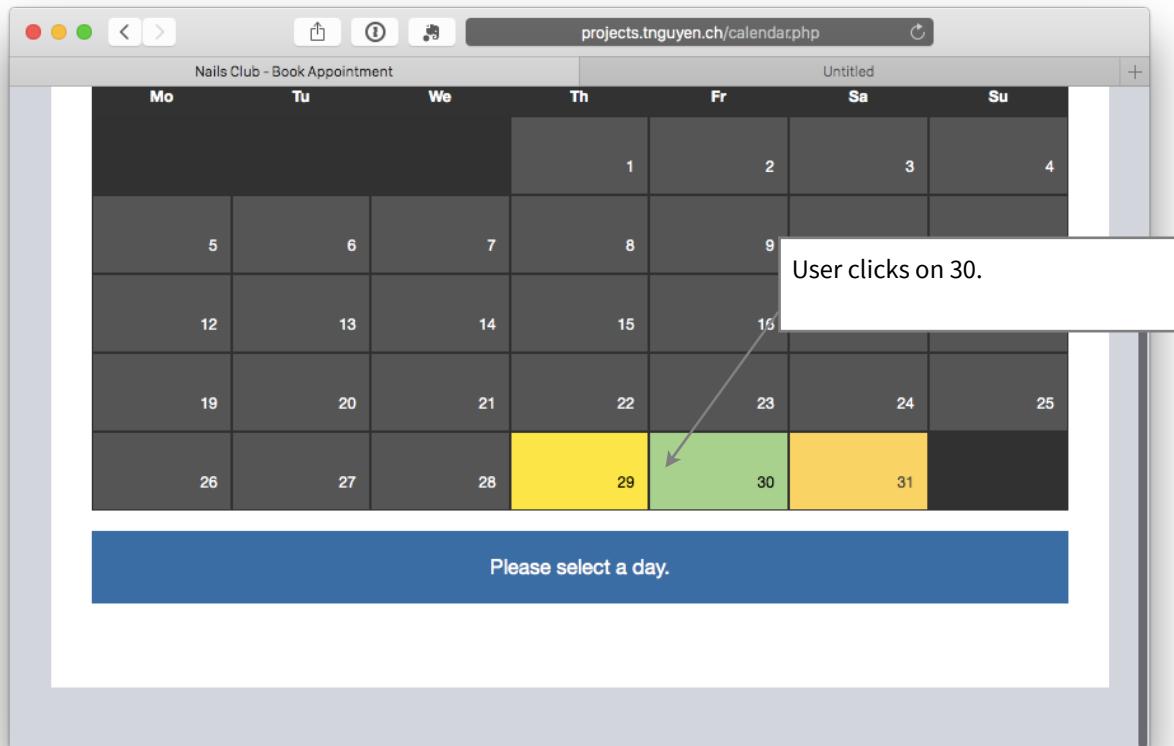
Test Evidence 11



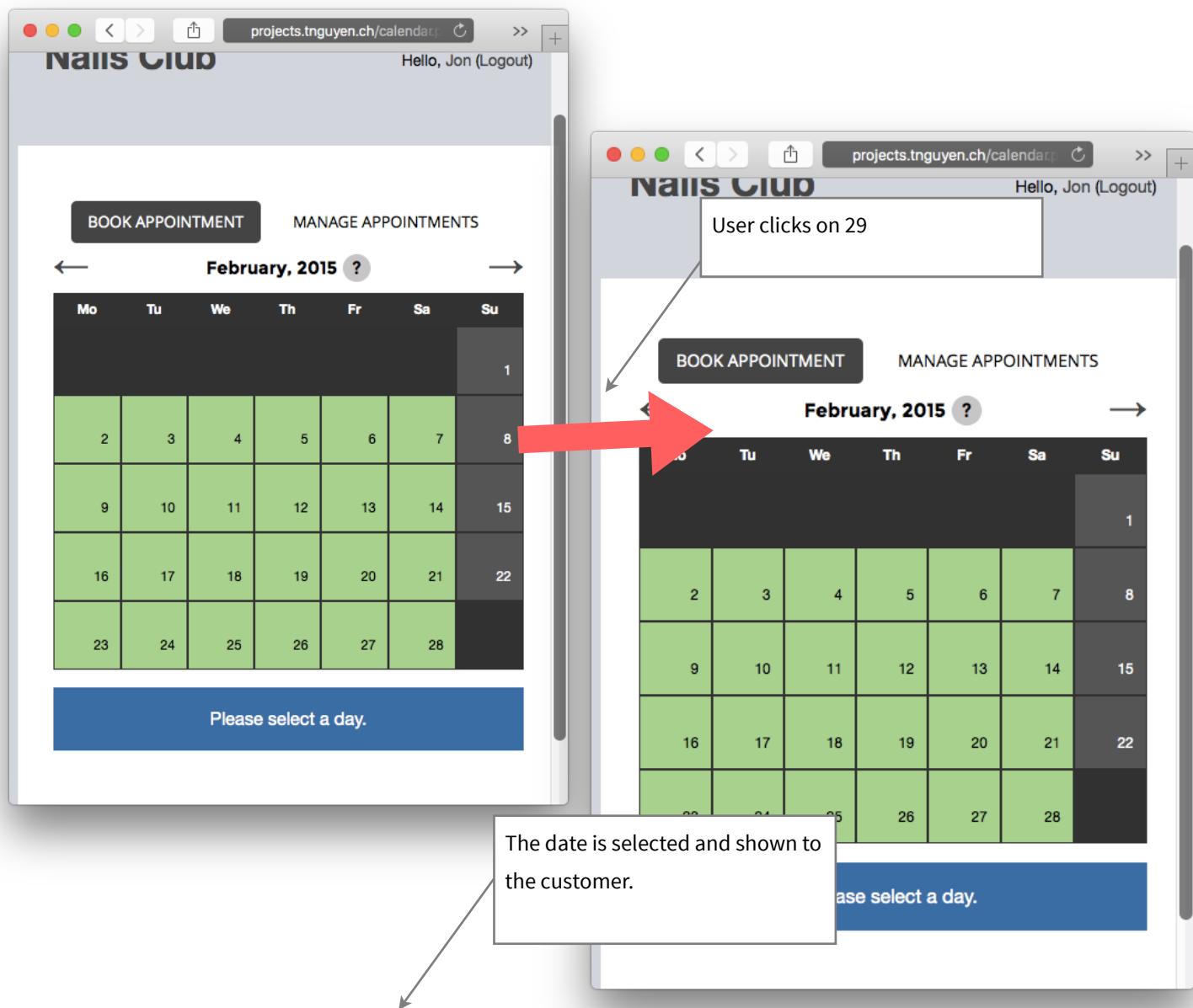
Test Evidence 12:



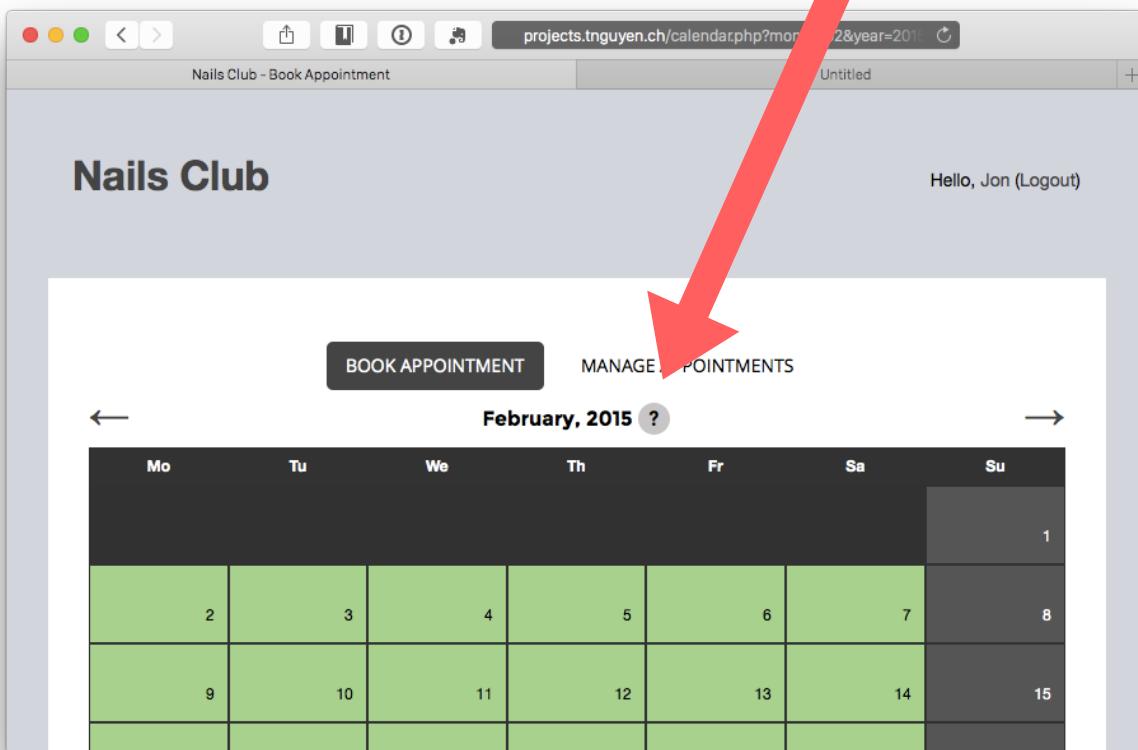
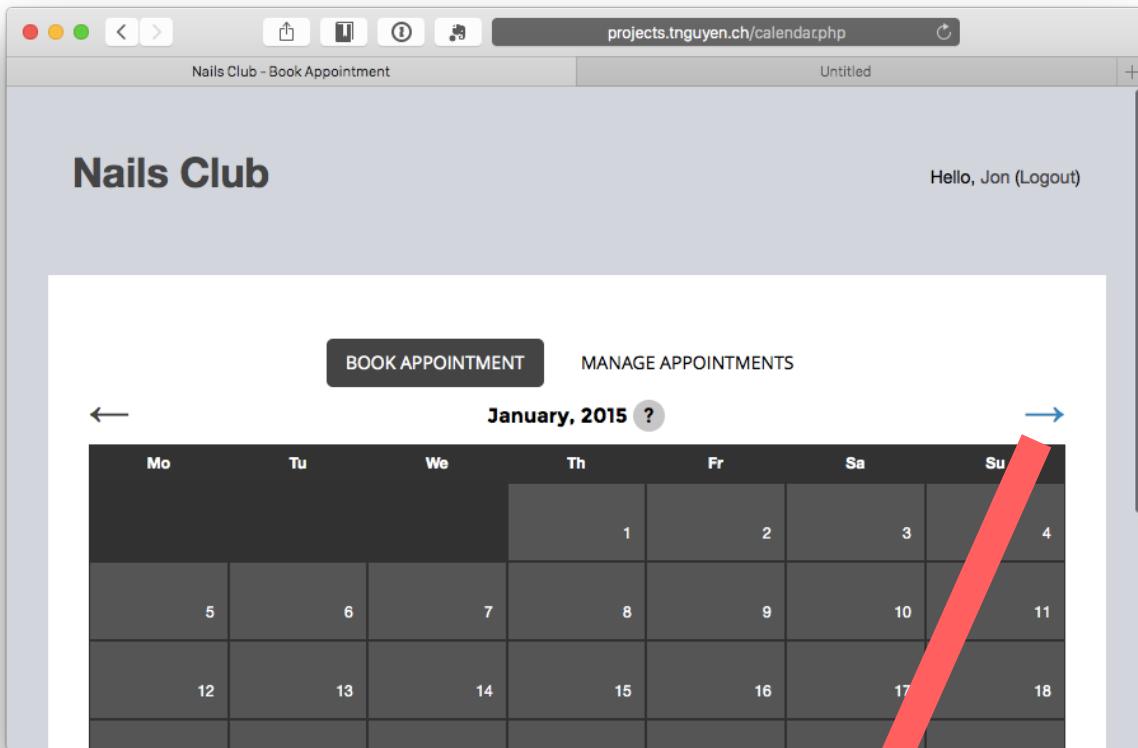
Test Evidence 13:



Test Evidence 14:



Test Evidence 15:



Test Evidence 16:

Selected Date is: Fri, 30 January 2015

12:30:00 - 13:00:00

Please select a Service

fruit iticssf

Privacy & Terms

Submit

BOOK APPOINTMENT

MANAGE APPOINTMENTS

Please select a service.

Nails Club

Hello, Jon (Logout)

January, 2015

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

Test Evidence 17:

The image displays two screenshots of a web-based scheduling and booking system. The top screenshot shows a calendar for January 2015. The date '30' is highlighted in green, and a red arrow points from the 'BOOK APPOINTMENT' button on the main page below to this date. The bottom screenshot shows the same calendar with the date '30' highlighted in green, indicating it has been selected.

Selected Date is: Fri, 30 January 2015

Please select a booking time

Info: £26

Thank you!

sfamio and

Submit

BOOK APPOINTMENT

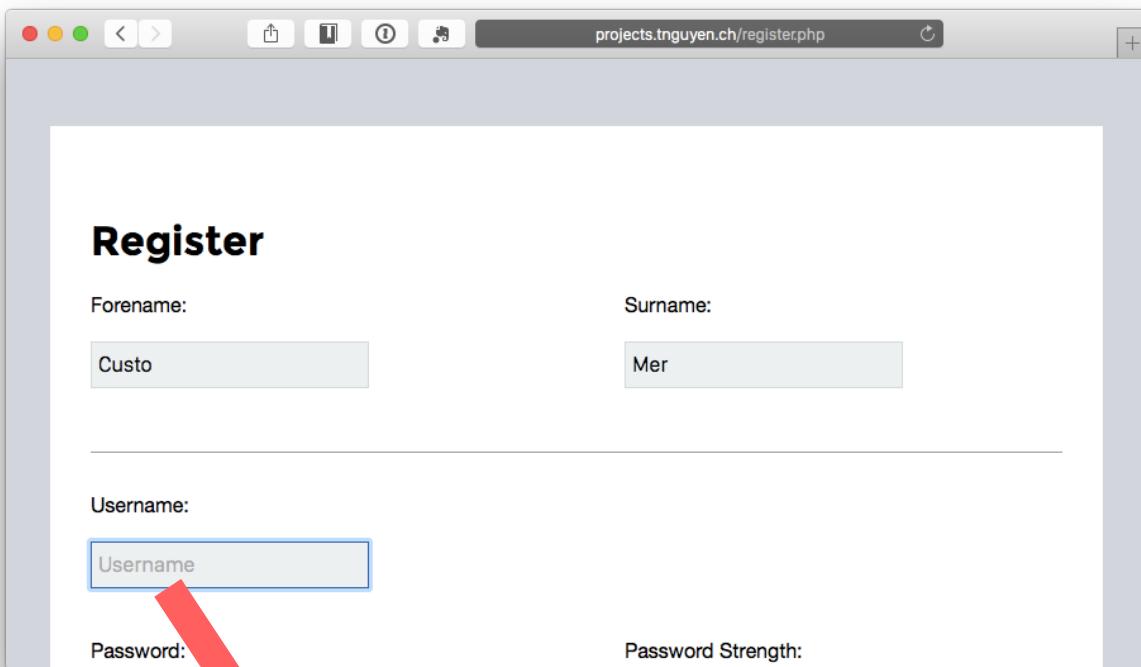
MANAGE APPOINTMENTS

Please select a booking time.

← January, 2015 →

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

Test Evidence 18:

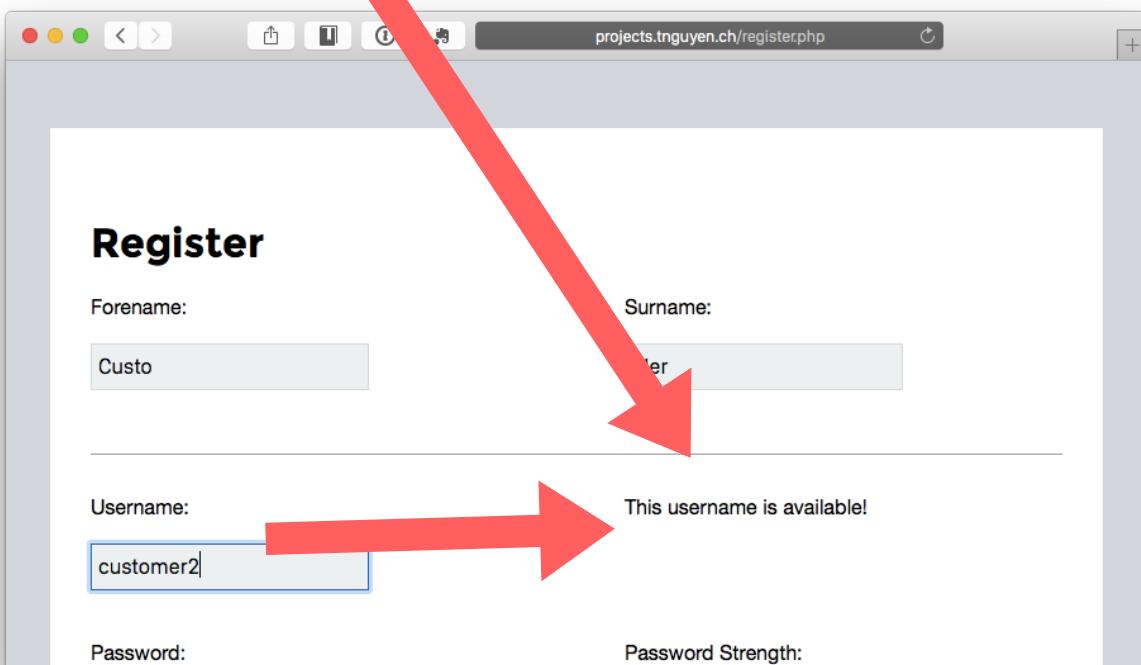


Register

Forename: Custo Surname: Mer

Username:

Password: Password Strength:



Register

Forename: Custo Surname: Mer

Username: customer2 This username is available!

Password: Password Strength:

Test Evidence 19:

The image shows two screenshots of a web browser displaying a registration form. Both screenshots have a red arrow pointing from the top one to the bottom one, specifically highlighting the 'Username' input field.

Screenshot 1 (Top):

- Page Title:** Register
- Fields:**
 - Forename: Custo
 - Surname: Mer
 - Username: (empty)
 - Password: (empty)
 - Password Strength: (empty)

Screenshot 2 (Bottom):

- Page Title:** Register
- Fields:**
 - Forename: Custo
 - Surname: Mer
 - Username: customer (highlighted by a red arrow)
 - Password: (empty)
 - Password Strength: No Data
- Feedback:** This username isn't available.

Test Evidence 20:

Username:

Password: Confirm Password:

Password Strength: No Data

Email: example@domain.com

Username:

Password: Confirm Password:

Password Strength: Strong

Email: example@domain.com

Test Evidence 21:

A screenshot of a web browser displaying a registration form. The URL is `projects.tnguyen.ch/register.php`. The form includes fields for Username, Password, Confirm Password, and Email. To the right of the Password field is a "Password Strength" indicator bar labeled "No Data". A large red arrow points from the bottom of the first screenshot to this "No Data" label.

Username:

Password:

Confirm Password:

Password Strength: No Data

Email:

A screenshot of a web browser displaying a registration form. The URL is `projects.tnguyen.ch/register.php`. The form includes fields for Username, Password, Confirm Password, and Email. The Password field contains four dots ("...."). To the right of the Password field is a "Password Strength" indicator bar labeled "Too short!". Below the Password field, a red box contains the validation message "Your password must be at least six characters long." A red arrow points from the bottom of the first screenshot to the "Too short!" label.

Username:

Password:

Confirm Password:

Password Strength: Too short!

Your password must be at least six characters long.

Test Evidence 22:

The image displays two screenshots of a web-based registration form titled "register.php" from the URL "projects.tnguyen.ch". Both screenshots show a "Password:" field containing "....." and a "Password Strength:" bar indicating "Strong". A red arrow points from the bottom screenshot to the "Confirm Password:" field in the top screenshot.

Screenshot 1 (Top):

Username:	
Password:
Confirm Password:	<input type="text" value="Password"/>
Email:	example@domain.com
Confirm Email:	example@domain.com
Phone Number:	

Screenshot 2 (Bottom):

Username:	
Password:
Confirm Password:	<input type="text" value="....."/>
Email:	example@domain.com
Confirm Email:	example@domain.com
Phone Number:	

Test Evidence 23:

The image displays two screenshots of a web application's registration page, labeled "register.php".

Screenshot 1 (Top):

- Username: [Input field]
- Password: [Input field] (containing 6 dots)
- Confirm Password: [Input field] (containing "Password")
- Password Strength: Strong (green bar)

Screenshot 2 (Bottom):

- Username: [Input field]
- Password: [Input field] (containing 6 dots)
- Confirm Password: [Input field] (containing "Password")
- >Password Strength: Strong (green bar)

A large red arrow points from the "Password" input field in the top screenshot down to the "Confirm Password" input field in the bottom screenshot, indicating a comparison between the two fields.

Validation Error: A red box highlights the message "Please enter the same value again." displayed below the "Confirm Password" field in the second screenshot.

Test Evidence 24:

The image displays two screenshots of a web browser window, likely from a Mac OS X system, showing a registration form at the URL projects.tnguyen.ch/register.php. The top screenshot shows the initial state of the form with placeholder email addresses. The bottom screenshot shows the form after the user has entered their own email address ('john@doe.com') into the 'Email:' field.

Top Screenshot (Initial State):

- Email: example@domain.com
- Confirm Email: example@domain.com
- Phone Number: (empty input field)
- Captcha: A reCAPTCHA challenge with the word "dry" and a text input field containing "antalla".

Bottom Screenshot (After User Input):

- Email: john@doe.com
- Confirm Email: example@domain.com
- Phone Number: (empty input field)
- Captcha: A reCAPTCHA challenge with the word "dry" and a text input field containing "antalla".

A large red arrow points vertically down from the 'Email:' field in the top screenshot to the same field in the bottom screenshot, indicating the progression of the user's input.

Test Evidence 25:

The image displays two screenshots of a web browser window for the URL `projects.tnguyen.ch/register.php`. Both screenshots show a registration form with fields for Email, Confirm Email, Phone Number, and Captcha.

Screenshot 1 (Top):

- Email: `example@domain.com`
- Confirm Email: `example@domain.com`
- Phone Number: `(123) 456-7890`
- Captcha:

Screenshot 2 (Bottom):

- Email: `johndoe.com`
- Confirm Email: `example@domain.com`
- Phone Number: `(123) 456-7890`
- Captcha:

A large red arrow points from the top screenshot down to the bottom screenshot, specifically highlighting the validation message in the Email field.

Validation Message:

Please enter a valid email address.

Test Evidence 26:

The image shows two screenshots of a web browser displaying a registration form at projects.tnguyen.ch/register.php.

Screenshot 1 (Top):

- Email: example@domain.com
- Confirm Email: example@domain.com
- Phone Number: (empty input field)
- Captcha:

Screenshot 2 (Bottom):

- Email: john@doecom (highlighted in red)
- Confirm Email: example@domain.com
- Phone Number: (empty input field)
- Captcha:

A large red arrow points from the top screenshot down to the bottom one, indicating a user action.

Test Evidence 27:

The image displays two screenshots of a web browser window, specifically projects.tnguyen.ch/register.php, illustrating a user registration process. Both screenshots show the same form fields: Email, Confirm Email, and Phone Number. A CAPTCHA challenge is present in both cases, featuring the word "dry" over a distorted background.

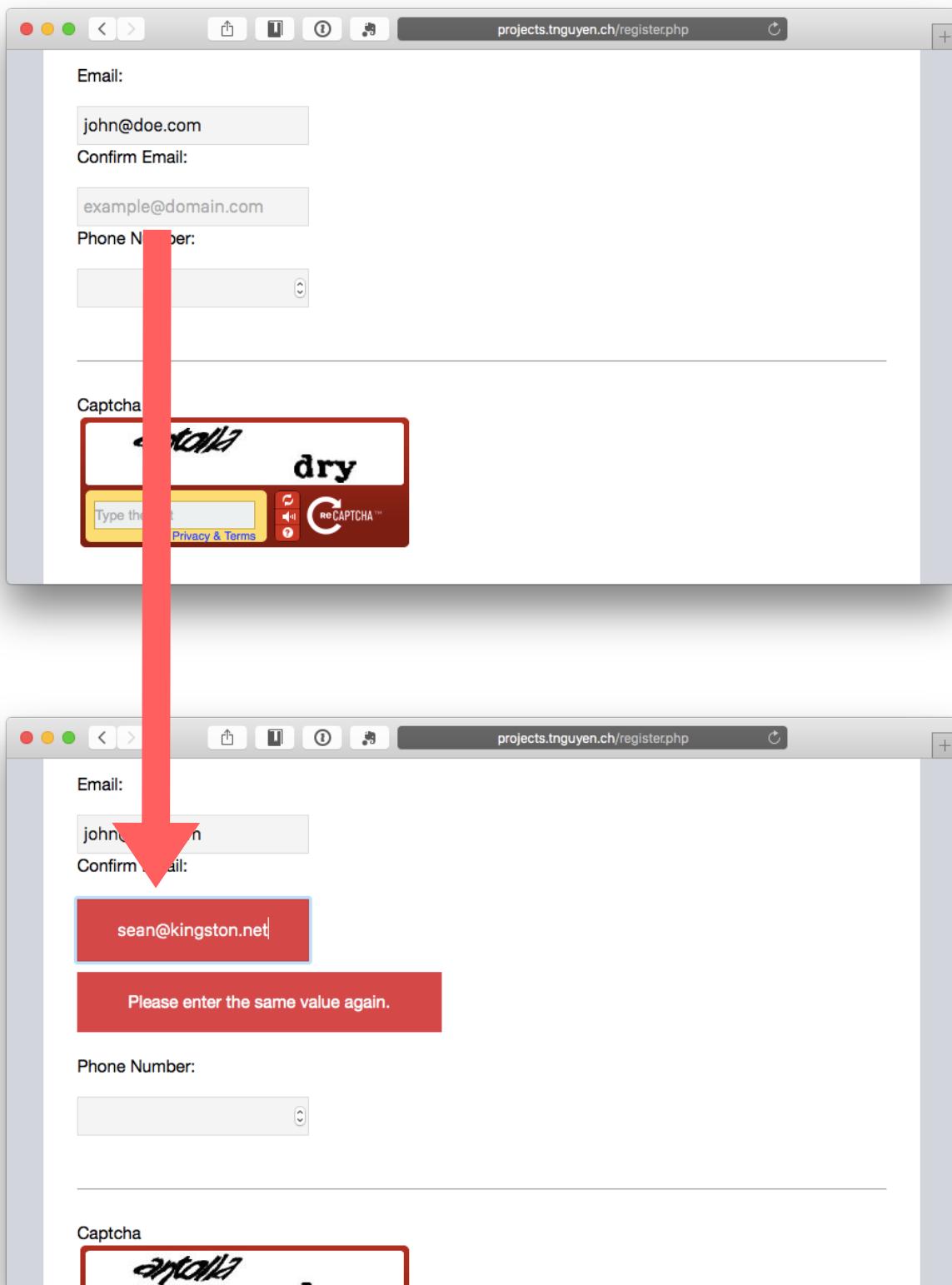
Top Screenshot (Initial State):

- Email: john@doe.com
- Confirm Email: example@domain.com
- Phone Number: (Input field)
- Captcha: dry (over a distorted background)

Bottom Screenshot (After Modification):

- Email: john@doe.com (The value 'john@doe.com' has been typed into the 'Email' field, indicated by a red arrow pointing from the top screenshot.)
- Confirm Email: john@doe.com
- Phone Number: (Input field)
- Captcha: dry (over a distorted background)

Test Evidence 28:



The screenshot shows two consecutive screenshots of a web browser window displaying a registration form at projects.tnguyen.ch/register.php.

Screenshot 1 (Top):

- Email:
- Confirm Email:
- Phone Number:
- Captcha:

Screenshot 2 (Bottom):

- Email:
- Confirm Email: (The value "sean@kingston.net" is highlighted with a red box.)
- A red error message box displays: "Please enter the same value again."
- Phone Number:
- Captcha:

A large red arrow points vertically downwards from the top screenshot to the "Confirm Email" field in the bottom screenshot, indicating a comparison between the initial input and the erroneous confirmation.

Test Evidence 29:

localhost/register.php

jon@snow.com

Phone Number:

07478691803

Captcha

east tyGnfor

captcha Privacy Terms reCAPTCHA

Next

localhost/register.php

Nails Club

The captcha is incorrect.

Register

Forename: Surname:

First Last

Username:

Test Evidence 30:

The image displays two screenshots of a web browser window, likely from a Mac OS X interface, showing a registration form at the URL `localhost/register.php`. Both screenshots show the same form fields: an email input field containing `example@domain.com`, a phone number input field, a CAPTCHA image, and a "Next" button.

Screenshot 1 (Top): The phone number input field is empty. A large red arrow points downwards from the top screenshot to the phone number field in the bottom screenshot.

Screenshot 2 (Bottom): The phone number input field now contains the value `01234567891`. The rest of the form (email, CAPTCHA, and "Next" button) remains the same.

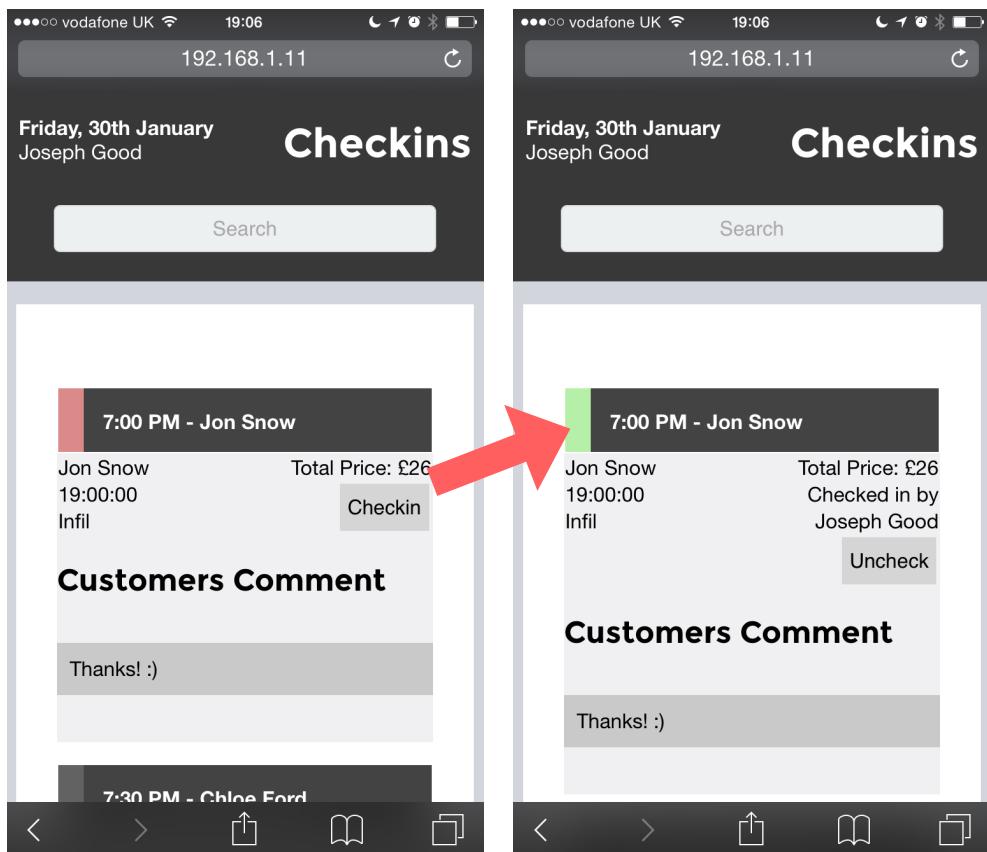
Test Evidence 31:

The image displays two screenshots of a web browser window, likely from a Mac OS X interface, showing a registration form at the URL `localhost/register.php`. Both screenshots show the same form fields: an email input field containing `example@domain.com`, a phone number input field, a CAPTCHA section, and a "Next" button.

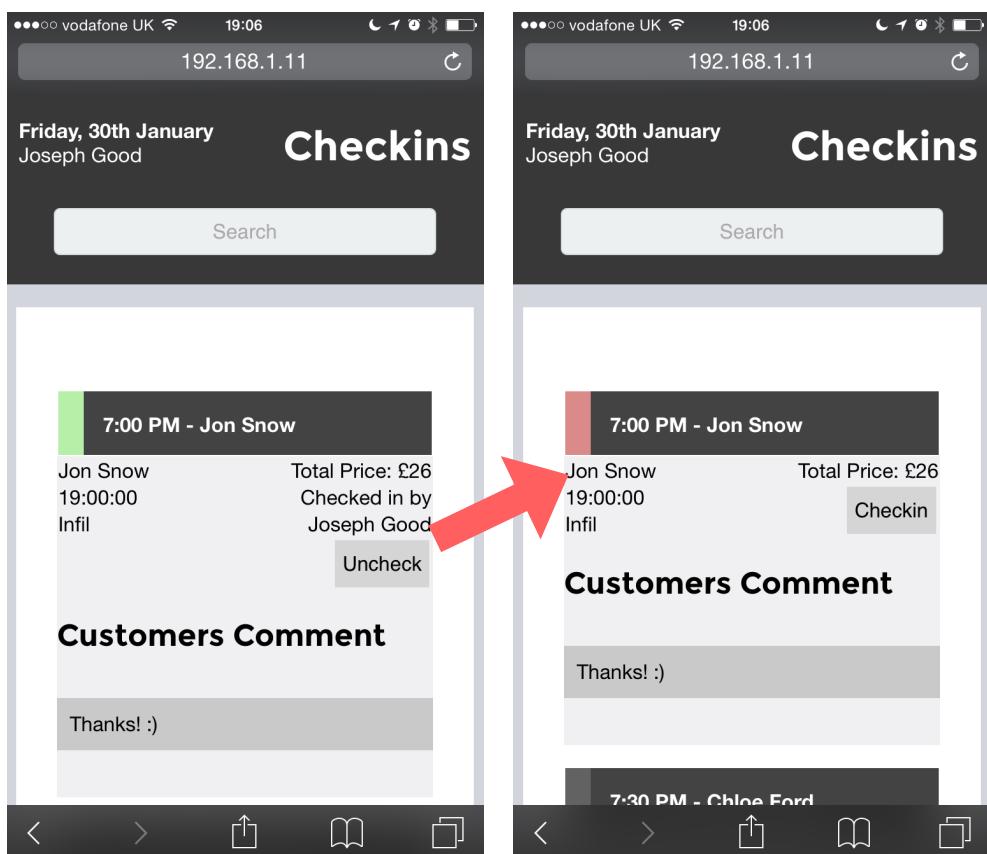
Screenshot 1 (Top): The phone number input field is empty. A red arrow points from the validation message in Screenshot 2 down to this field. The CAPTCHA text is partially visible as "part" and "tlyour".

Screenshot 2 (Bottom): The phone number input field contains the character "1". Below it, a red error message box displays the text: "This phone number is too short." The CAPTCHA text is partially visible as "part" and "tlyour".

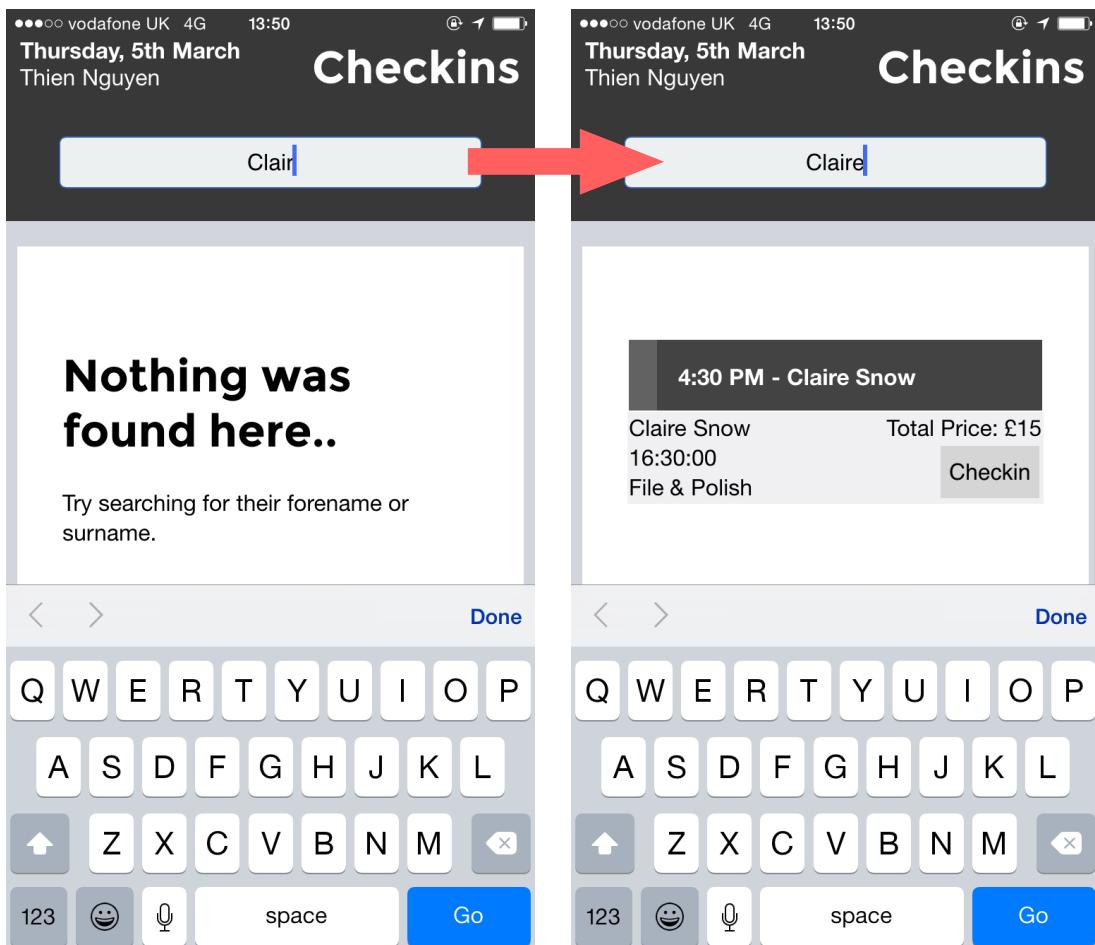
Test Evidence 32:



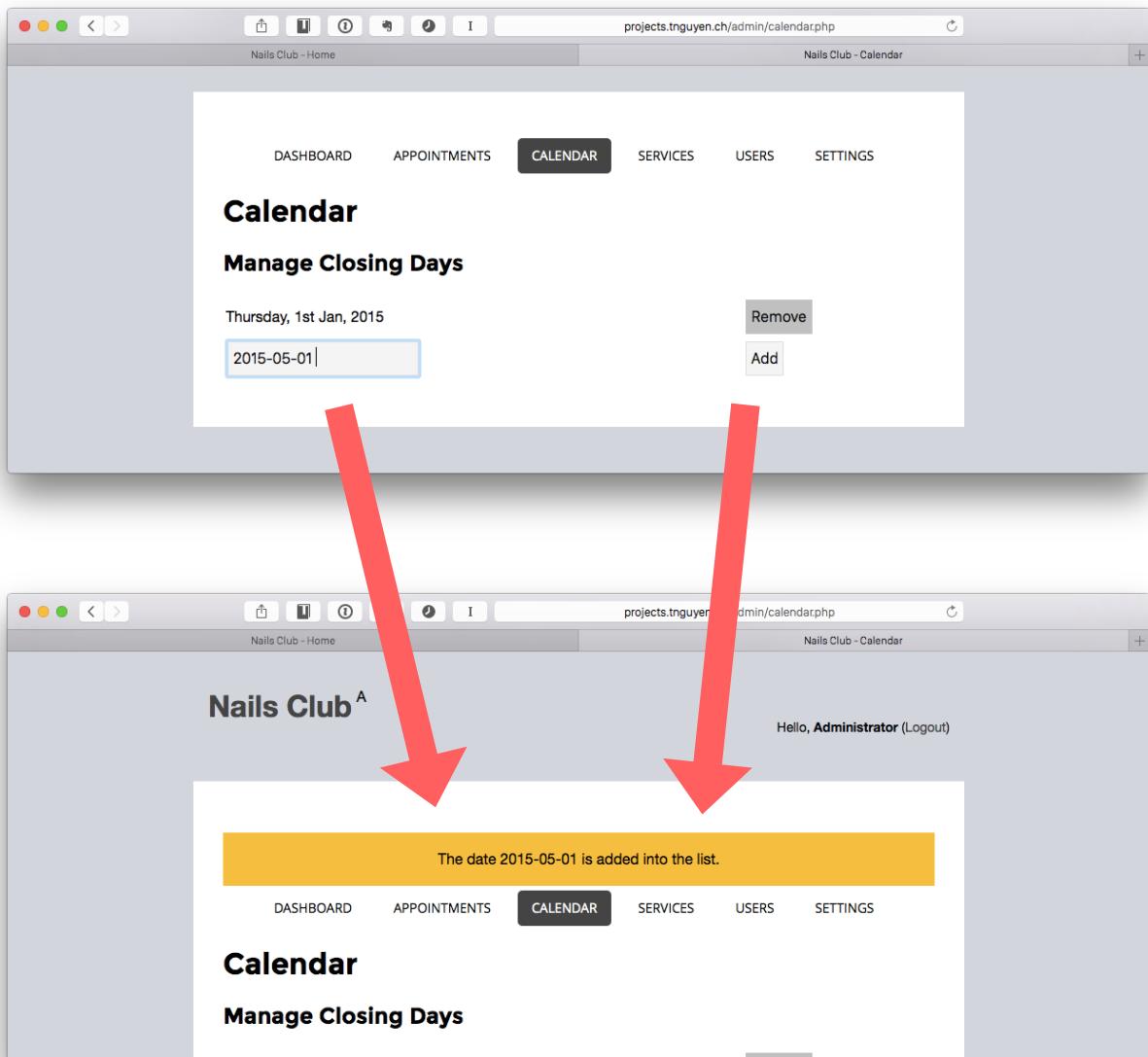
Test Evidence 33:



Test Evidence 34:



Test Evidence 35:



Test Evidence 36:

The screenshot shows two consecutive screenshots of a web application interface for 'Nails Club'. Both screenshots are from the 'CALENDAR' section, with the URL 'localhost/admin/calendar.php' visible in the browser's address bar.

Screenshot 1 (Top): This screenshot shows the 'Manage Closing Days' page. A date input field contains '2015-01-28'. To the right of the input field are two buttons: 'Remove' and 'Add'. Two large red arrows point downwards from the top of the image towards the 'Add' button and the confirmation message in the second screenshot.

Date	Action
2015-01-28	Add

Screenshot 2 (Bottom): This screenshot shows the same 'Manage Closing Days' page after the addition. A yellow success message at the top states 'The closing day is added into the database.' Below this message, the date '2015-01-28' is listed again, followed by a 'Remove' button. Further down, another date 'Wednesday, 28th Jan, 2015' is listed with its own 'Remove' button. At the bottom left is a 'Today?' button, and at the bottom right is an 'Add' button.

Date	Action
2015-01-28	Added
Wednesday, 28th Jan, 2015	Remove

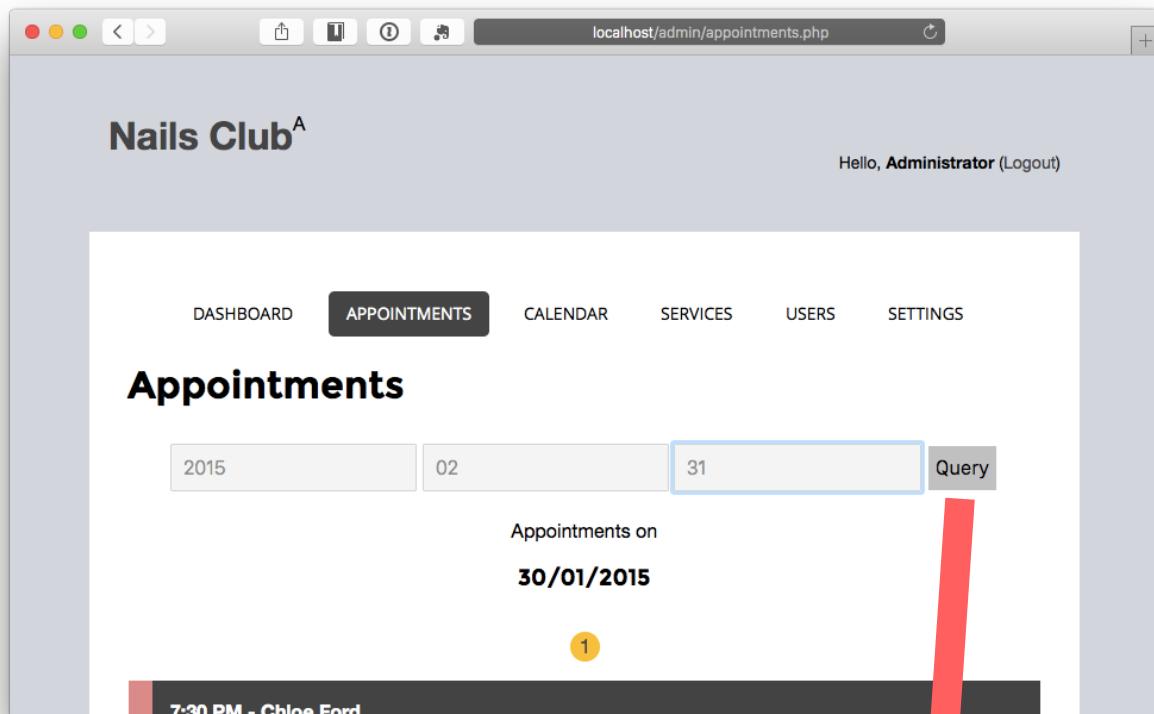
Test Evidence 37:

The image displays two screenshots of a web application interface, specifically the 'CALENDAR' section of 'Nails Club'. Both screenshots show a date input field containing '2015-02-31' and a red 'Add' button.

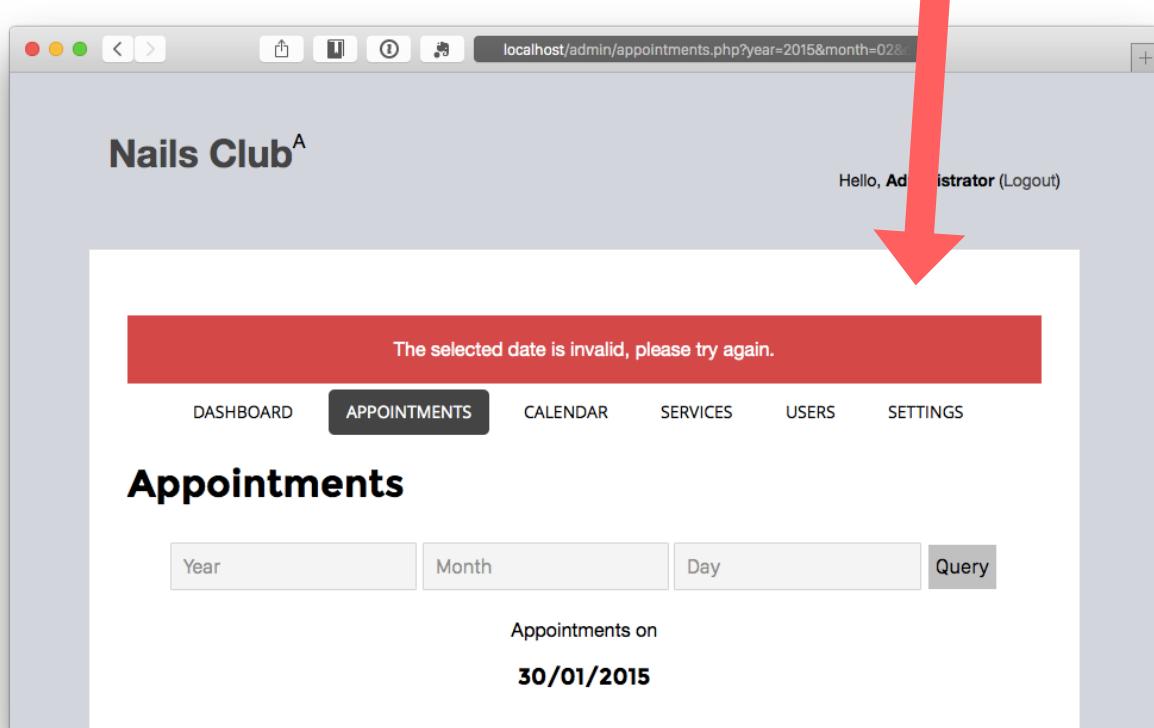
Screenshot 1 (Top): This screenshot shows a light gray background. A red arrow points from the date input field towards the bottom of the page, indicating the location of the validation message. The message is displayed in a red box at the bottom of the calendar section, stating: '2015-02-31 is an invalid date, please try again.'

Screenshot 2 (Bottom): This screenshot shows a similar interface but with a different validation result. A red arrow points from the date input field towards the bottom of the page, indicating the location of the validation message. The message is displayed in a red box at the bottom of the calendar section, stating: 'Today? is an invalid date, please try again.'

Test Evidence 38:



The screenshot shows the Nails Club Admin interface for the 'Appointments' section. The URL is `localhost/admin/appointments.php`. The top navigation bar includes links for DASHBOARD, APPOINTMENTS (which is selected and highlighted in black), CALENDAR, SERVICES, USERS, and SETTINGS. On the right, it says 'Hello, Administrator (Logout)'. Below the navigation is a search form with three input fields: 'Year' (2015), 'Month' (02), and 'Day' (31). A 'Query' button is to the right of the day field. Below the form, a message states 'Appointments on 30/01/2015' and shows a single appointment entry: '7:30 PM - Chloe Ford'. A red arrow points from the bottom of this screenshot down to the error message in the second screenshot.



The screenshot shows the same Nails Club Admin interface, but the 'Day' field in the search form contains an invalid value ('31'). A red error message box at the top of the main content area states 'The selected date is invalid, please try again.' The rest of the interface is identical to the first screenshot, including the navigation bar and the appointment entry below the search form.

Test Evidence 39:

The screenshot shows the 'Appointments' section of the Nails Club admin interface. The URL is `localhost/admin/appointments.php`. The search fields show '2015', '01', and '01'. The results indicate '1' appointment on '30/01/2015'. A red arrow points from the bottom screenshot to this result.

Nails Club^A

Hello, Administrator (Logout)

DASHBOARD APPOINTMENTS CALENDAR SERVICES USERS SETTINGS

Appointments

2015 01 01 Query

Appointments on
30/01/2015

1

7:30 PM - Chloe Ford

The screenshot shows the 'Appointments' section of the Nails Club admin interface. The URL is `localhost/admin/appointments.php?year=2015&month=01&day=01`. The search fields show 'Year', 'Month', and 'Day'. The results indicate '0' appointments on '01/01/2015'. A red arrow points from the top screenshot to this result.

Nails Club^A

Hello, Administrator (Logout)

DASHBOARD APPOINTMENTS CALENDAR SERVICES USERS SETTINGS

Appointments

Year Month Day Query

Appointment on
01/01/2015

Nothing was found here..

Try searching for their forename or surname.

Test Evidence 40:

The screenshot shows the 'Appointments' section of the Nails Club admin interface. At the top, there are navigation links: DASHBOARD, APPOINTMENTS (which is highlighted), CALENDAR, SERVICES, USERS, and SETTINGS. Below these, a date range selector shows '2015' and '01'. A dropdown menu is open, listing days from 01 to 09, with '01' selected. A red arrow points from the bottom screenshot to this dropdown menu. The main content area displays 'Appointments on 30/01/2015' with a count of 1. A detailed appointment card for '1:30 PM - Jon Snow' is shown, including service details: 'Jon Snow', '13:30:00', 'Full Set', and payment information: 'Total Price: £48' and 'Checked in by Joseph Good'.

Test Evidence 41:

The screenshot shows a web application interface for managing services. The top navigation bar includes links for DASHBOARD, APPOINTMENTS, CALENDAR, SERVICES (which is the active tab), USERS, and SETTINGS. Below the navigation is a section titled "Services" containing a table with columns for Option, Value (£), and Description. The table lists several service options with their respective values and descriptions. An "Add" button is located at the bottom right of the table. A tooltip on the "Add" button states: "The input doesn't match the criteria that's required for the data to be sent to the database." A second screenshot below shows the same interface after attempting to add a new service. A red error message box displays two errors: "Please type in a price." and "Please type in a description." The "Add" button is now disabled.

Option	Value (£)	Description	Update	Remove
Manicure	25	A luxurious beauty treatment	Update	Remove
Infil	26	Description	Update	Remove
Pedicure	29	Description	Update	Remove
Full Set	48	Description	Update	Remove
File & Polish	15	Description	Update	Remove
Nail Repair	10	Description	Update	Remove
Acrylic Refil	Price	Description	Add	

Nails Club^A

Hello, Admin

Please type in a price.

Please type in a description.

DASHBOARD APPOINTMENTS CALENDAR SERVICES USERS SETTINGS

Services

Option	Value (£)	Description	Update	Remove
Manicure	25	A luxurious beauty treatment	Update	Remove
Infil	26	Description	Update	Remove

Test Evidence 42:

The screenshot shows a web application interface for managing services. The top navigation bar includes links for DASHBOARD, APPOINTMENTS, CALENDAR, SERVICES (which is selected), USERS, and SETTINGS. The main content area is titled "Services" and displays a list of service options with their values and descriptions. An arrow points from the "Add" button in the "Acrylic Refill" row to a callout box containing an error message.

Option	Value (£)	Description	Update	Remove
Manicure	25	A luxurious beauty treatment	Update	Remove
Infil	26	Description	Update	Remove
Pedicure	29	Description	Update	Remove
Full Set	48	Description	Update	Remove
File & Polish	15	Description	Update	Remove
Nail Repair	10	Description	Update	Remove
Acrylic Refill	18	Description	Add	

The input doesn't match the criteria that's required for the data to be sent to the database.

It is not sent, and an error message is displayed to the user notifying why it hasn't been sent.

The bottom screenshot shows the same application after an attempt to add a new service. A red error message box at the top states "Please type in a description." The "Add" button is still present in the "Acrylic Refill" row.

Test Evidence 43:

The screenshot shows a web-based administrative interface for managing services. The top navigation bar includes links for DASHBOARD, APPOINTMENTS, CALENDAR, SERVICES (which is the active tab), USERS, and SETTINGS. Below the navigation is a section titled "Services" containing a table with columns for "Option", "Value (£)", and "Description". A modal window is open, prompting the user to enter a new service. The "Option" field contains "Acrylic Refil", the "Value (£)" field contains "18", and the "Description" field contains "A maintenance service for Acrylic nails." An "Add" button is visible at the bottom right of the modal.

Option	Value (£)	Description	Update	Remove
Manicure	25	A luxurious beauty treatment	Update	Remove
Infil	26	Description	Update	Remove
Pedicure	29	Description	Update	Remove
Full Set	48	Description	Update	Remove
File & Polish	15	Description	Update	Remove
Nail Repair	10	Description	Update	Remove

The input matches the criteria that's required for the data to be sent to the database.

It is sent, and a message is displayed to the user notifying that it has been sent.

A maintenance service for Acrylic nails.

Add

The query has been added.

DASHBOARD APPOINTMENTS CALENDAR SERVICES USERS SETTINGS

Services

This screenshot shows the same services management interface after the new service has been added. The table now includes a new row for "Acrylic Refil" with a value of "18". The "Description" for this service is "A maintenance service for Acrylic nails." The "Add" button is no longer visible.

Option	Value (£)	Description	Update	Remove
Manicure	25	A luxurious beauty treatment	Update	Remove
Infil	26	Description	Update	Remove
Pedicure	29	Description	Update	Remove
Full Set	48	Description	Update	Remove
File & Polish	15	Description	Update	Remove
Nail Repair	10	Description	Update	Remove
Acrylic Refil	18	A maintenance service for Acrylic nails.	Update	Remove

Name

Price

Description

Add

Test Evidence 44:

Nails Club^A

Hello, Administrator (Logout)

DASHBOARD APPOINTMENTS CALENDAR SERVICES USERS SETTINGS

Settings

Option	Value	Description
First Slot	25:00	The time of the first slot.
Last Slot	19:30	The time of the last slot. (Not closing time)
Booking Frequency	30	The interval of each booking, expressed in minutes.
Business Name	Nails Club	The name of the business.
Business Slogan		The slogan of the business.

The input doesn't match the criteria that's required for the data to be sent to the database.
It is not sent, and an error message is displayed to the user notifying why it hasn't been sent.

This is an invalid time.

DASHBOARD APPOINTMENTS CALENDAR SERVICES USERS SETTINGS

Settings

Option	Value	Description	Action
First Slot	10:00	The time of the first slot.	Update
Last Slot	19:30	The time of the last slot. (Not closing time!)	Update
Booking Frequency	30	The interval of each booking, expressed in minutes.	Update

Test Evidence 45:

Nails Club^A

Hello, Administrator (Logout)

DASHBOARD APPOINTMENTS CALENDAR SERVICES USERS SETTINGS

Settings

Option	Value	Description
First Slot	08:00	The time of the first slot.
Last Slot	19:30	The time of the last slot. (Not closing time!)
Booking Frequency	30	The interval of each booking minutes.
Business Name	Nails Club	The name of the business.
Business Slogan		The slogan of the business.

Nails Club^A

Hello, Administrator (Logout)

Your information has been updated into the database.

DASHBOARD APPOINTMENTS CALENDAR SERVICES USERS SETTINGS

Settings

Option	Value	Description
First Slot	08:00	The time of the first slot.
Last Slot	19:30	The time of the last slot. (Not closing time!)
Booking Frequency	30	The interval of each booking, expressed in minutes.

Test Evidence 46:

Nails Club^A

Hello, Administrator (Logout)

DASHBOARD APPOINTMENTS CALENDAR SERVICES USERS SETTINGS

Settings

Option	Value	Description
First Slot	10:00	The time of the first slot.
Last Slot	25:00	The time of the last slot. (Not closing time!)
Booking Frequency	30	The interval of each booking, minutes.
Business Name	Nails Club	The name of the business.
Business Slogan		The slogan of the business.

Update

The input doesn't match the criteria that's required for the data to be sent to the database.

It is not sent, and an error message is displayed to the user notifying why it hasn't been sent.

Nails Club^A

Hello, Administrator (Logout)

DASHBOARD APPOINTMENTS CALENDAR SERVICES USERS SETTINGS

Settings

This is an invalid time.

Option	Value	Description
First Slot	10:00	The time of the first slot.
Last Slot	19:30	The time of the last slot. (Not closing time!)
Booking Frequency	30	The interval of each booking, expressed in minutes.

Update

Update

Update

Test Evidence 47:

Nails Club^A

Hello, Administrator (Logout)

DASHBOARD APPOINTMENTS CALENDAR SERVICES USERS SETTINGS

Settings

Option	Value	Description
First Slot	10:00	The time of the first slot.
Last Slot	20:00	The time of the last slot. (Not closing time!)
Booking Frequency	30	The interval of each booking, expressed in minutes.
Business Name	Nails Club	The name of the business.
Business Slogan		The slogan of the business.

Nails Club^A

Hello, Administrator (Logout)

DASHBOARD APPOINTMENTS CALENDAR SERVICES USERS SETTINGS

Settings

Option	Value	Description
First Slot	10:00	The time of the first slot.
Last Slot	20:00	The time of the last slot. (Not closing time!)
Booking Frequency	30	The interval of each booking, expressed in minutes.

Your information has been updated into the database.

Test Evidence 48:

Nails Club^A

Hello, Administrator (Logout)

DASHBOARD APPOINTMENTS CALENDAR SERVICES USERS SETTINGS

Settings

Option	Value	Description
First Slot	10:00	The time of the first slot.
Last Slot	20:00	The time of the last slot. (Not closing time!)
Booking Frequency	20	The interval of each booking, expressed in minutes.
Business Name	Nails Club	The name of the business.
Business Slogan		The slogan of the business.

The input matches the criteria that's required for the data to be sent to the database.

It is sent, and a message is displayed to the user notifying that it has been sent.

Nails Club^A

Hello, Administrator (Logout)

Your information has been updated into the database.

DASHBOARD APPOINTMENTS CALENDAR SERVICES USERS SETTINGS

Settings

Option	Value	Description
First Slot	10:00	The time of the first slot.
Last Slot	20:00	The time of the last slot. (Not closing time!)
Booking Frequency	20	The interval of each booking, expressed in minutes.

Test Evidence 49:

Screenshot of the Nails Club Admin Settings page showing an invalid booking frequency input.

The screenshot shows the "Settings" page with the following table:

Option	Value	Description
First Slot	10:00	The time of the first slot.
Last Slot	20:00	The time of the last slot. (Not closing time!)
Booking Frequency	-100	The interval of each booking, expressed in minutes.
Business Name	Nails Club	The name of the business.
Business Slogan		The slogan of the business.

A callout box points to the "Booking Frequency" row with the following text:

The input doesn't match the criteria that's required for the data to be sent to the database.

Another callout box points to the error message at the bottom of the page with the following text:

It is not sent, and an error message is displayed to the user notifying why it hasn't

The error message at the bottom of the page is: "The booking frequency is either too short or too long. Try again!"

Test Evidence 50:

Nails Club^A

Hello, Administrator (Logout)

DASHBOARD APPOINTMENTS CALENDAR SERVICES USERS SETTINGS

Settings

Option	Value	Description	Action
First Slot	10:00	The time of the first slot.	Update
Last Slot	20:00	The time of the last slot. (Not closing time!)	Update
Booking Frequency	30	The interval of each booking, expressed in minutes.	Update
Business Name	Business Name	The name of the business.	Update
Business Slogan		The slogan of the business.	Update

The input matches the criteria that's required for the data to be sent to the database.

Business Name^A

Hello, Administrator (Logout)

Your information has been updated into the database.

DASHBOARD APPOINTMENTS CALENDAR SERVICES USERS SETTINGS

Settings

Option	Value	Description	Action
First Slot	10:00	The time of the first slot.	Update
Last Slot	20:00	The time of the last slot. (Not closing time!)	Update
Booking Frequency	30	The interval of each booking, expressed in minutes.	Update
Business Name	Business Name	The name of the business.	Update
Business Slogan		The slogan of the business.	Update

It is sent, and a message is displayed to the user notifying that it has been sent.

Test Evidence 51:

The screenshot shows the 'Settings' page of a scheduling and booking system. The 'Business Slogan' field contains the value 're et dolore magna aliqua.'. An arrow points from this field to a callout box containing the text: 'The input doesn't match the criteria that's required for the data to be sent to the database.'

The screenshot shows the 'Settings' page of a scheduling and booking system. A red banner at the bottom displays the message: 'The slogan is too long.' An arrow points from this banner to another callout box containing the text: 'It is not sent, and an error message is displayed to the user notifying why it hasn't'.

Option	Value	Description
First Slot	10:00	The time of the first slot.
Last Slot	20:00	The time of the last slot. (Not closing time!)
Booking Frequency	30	The interval of each booking, expressed in minutes.
Business Name	Nails Club	The name of the business.
Business Slogan	re et dolore magna aliqua.	The slogan of the business.

Test Evidence 52:

Option	Value	Description	Update
First Slot	10:00	The time of the first slot.	<button>Update</button>
Last Slot	20:00	The time of the last slot. (Not closing time!)	<button>Update</button>
Booking Frequency	30	The interval of each booking, expressed in minutes.	<button>Update</button>
Business Name	Nails Club	The name of the business.	<button>Update</button>
Business Slogan	Lorpem ipsum.	The slogan of the business.	<button>Update</button>

Since a slogan is stored in the database, it is displayed on all pages.

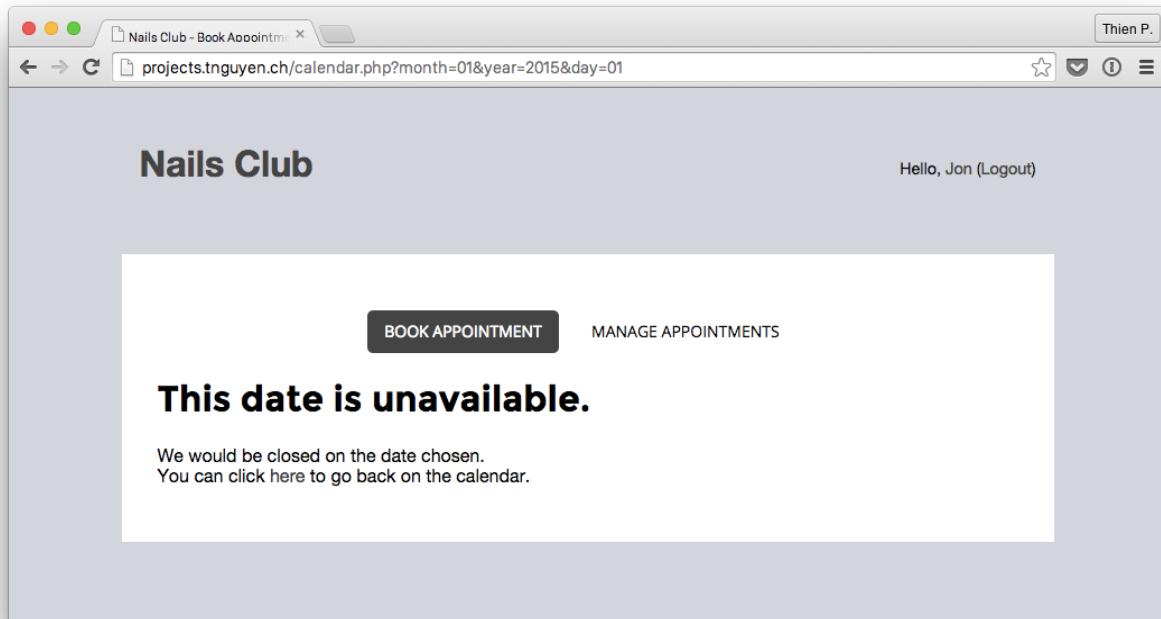
The input matches the criteria that's required for the data to be sent to the database.
It is sent, and a message is displayed to the user notifying that it has been sent.

Nails Club^A
Lorpem ipsum.

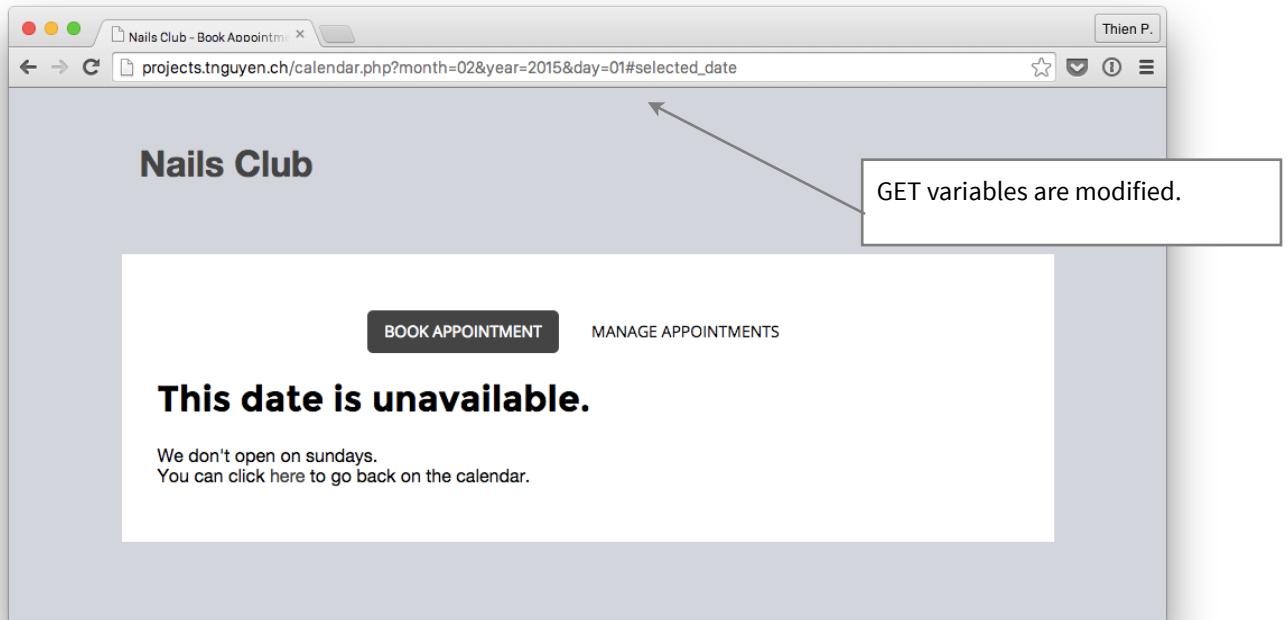
Your information has been updated into the database.

Option	Value	Description	Update
First Slot	10:00	The time of the first slot.	<button>Update</button>
Last Slot	20:00	The time of the last slot. (Not closing time!)	<button>Update</button>
Booking Frequency	30	The interval of each booking, expressed in minutes.	<button>Update</button>

Test Evidence 53:



Test Evidence 54:



Test Evidence 55:

editing date values on URL. (these will be GET variables for the PHP code.)

Hello, Jon (Logout)

Date values are out of booking threshold (within 6 months over or under the current date.)

This date is unavailable.

This date is currently beyond our booking threshold for now.
You can click here to go back on the calendar.

BOOK APPOINTMENT MANAGE APPOINTMENT

Test Evidence 56:

```
<?php
include_once('functions/encryption.php');
$string = "password";
$string2 = "password2";
echo "<pre>";
echo "Testing encryption function with string: ". $string;
echo "<br>Result: " . encrypt($string);
echo "<br><br>";
echo "Testing encryption function with string: ". $string2;
echo "<br>Result: " . encrypt($string2);
echo "</pre>";
?>
```

A new php file is used to test the function, seeing whether it works. Two variables are used.

localhost/playground.php

Testing encryption function with string: password
Result: a5b4a87e4af67e10c408e2deal92b458873556f1

Testing encryption function with string: password2
Result: db7312a1e2ad92980ca186c6b4b079606faa0acb

Variables are printed and checked to compare differences. The more varied the hashes are compared to the similarity of the strings prior to hash, the better.

Test Evidence 57:

To test that this recursive function works, modifications to the code had to be done in order to print what the process of PIN generation. Since the function itself consists of two lines of code, the original code is modified, where the recursive line is replaced with the functions contents. This way, the first call of the recursive stack can be modified.

Original Code	Modified Code
<pre>function generate_pin(\$used_pins = array()){ \$new_pin = rand(10000,99999); \$new_pin_hash = encrypt(\$new_pin); while (in_array(\$new_pin_hash, \$used_pins)){ generate_pin(\$used_pins); } return array('pin' => \$new_pin, 'pin_hash' => \$new_pin_hash); }</pre>	<pre>function generate_pin(\$used_pins = array()){ // \$new_pin = rand(10000,99999); \$new_pin = 12345; \$new_pin_hash = encrypt(\$new_pin); echo "new pin: " . \$new_pin; echo "
new pin hash: " . \$new_pin_hash; while (in_array(\$new_pin_hash, \$used_pins)){ // generate_pin(\$used_pins); \$new_pin = rand(10000,99999); \$new_pin_hash = encrypt(\$new_pin); echo "
current pin is in use, trying again with newly generated pin.
"; echo "new pin: " . \$new_pin; echo "
new pin hash: " . \$new_pin_hash; echo "
assigning new pin hash to check in array. if fails loop will iterate
"; } return array('pin' => \$new_pin, 'pin_hash' => \$new_pin_hash);</pre>

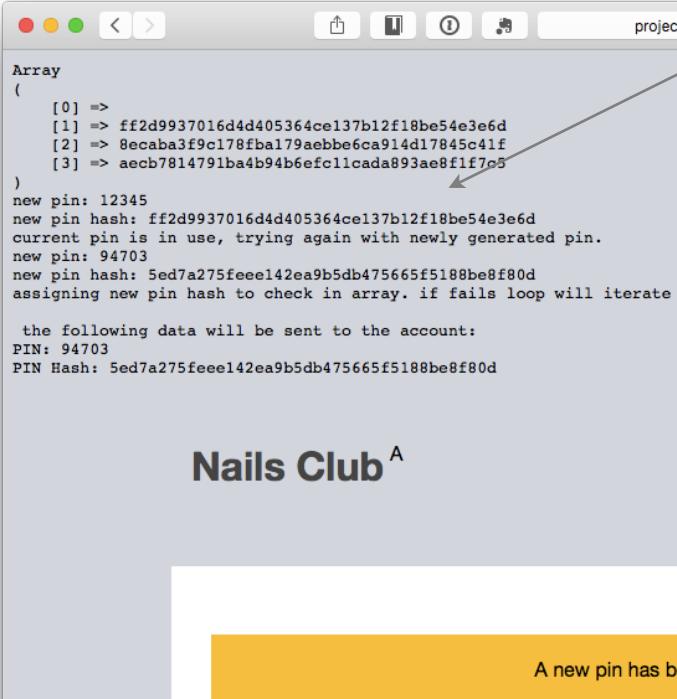
The screenshot shows a web application interface for 'Nails Club'. At the top, there are buttons for 'Update', 'Ban', and 'Generate New PIN'. The 'Generate New PIN' button is highlighted with a yellow arrow pointing from the 'Modified Code' table. Below the buttons, there's a text area showing an array of hashes and some log output. A second yellow arrow points from the same text area to the explanatory text on the right. The main content area displays a success message: 'A new pin has been set to Joseph.' Below this message, a navigation bar includes 'DASHBOARD', 'APPOINTMENTS', 'CALENDAR', 'SERVICES', 'USERS' (which is currently selected), and 'SETTINGS'. The 'USERS' section also includes a 'Staff' link.

A new pin (12345) is hashed and checked with the database to see if this hash has been already been used.

If theres no match then the hash is then sent to the database. In this case the 12345 hash isn't in the database, so it is sent.

Generate New PIN is clicked, initiating the function.

A new pin has been set to Joseph.



```

Array
(
    [0] =>
        ff2d9937016d4d405364ce137b12f18be54e3e6d
    [1] => 8ecaba3f9c178fb179aebbe6ca914d17845c41f
    [2] => aecb7814791ba4b94b6efc11cada893ae8ff7c5
)
new pin: 12345
new pin hash: ff2d9937016d4d405364ce137b12f18be54e3e6d
current pin is in use, trying again with newly generated pin.
new pin: 94703
new pin hash: 5ed7a275feee142ea9b5db475665f5188be8f80d
assigning new pin hash to check in array. if fails loop will iterate

the following data will be sent to the account:
PIN: 94703
PIN Hash: 5ed7a275feee142ea9b5db475665f5188be8f80d

```

Nails Club^A

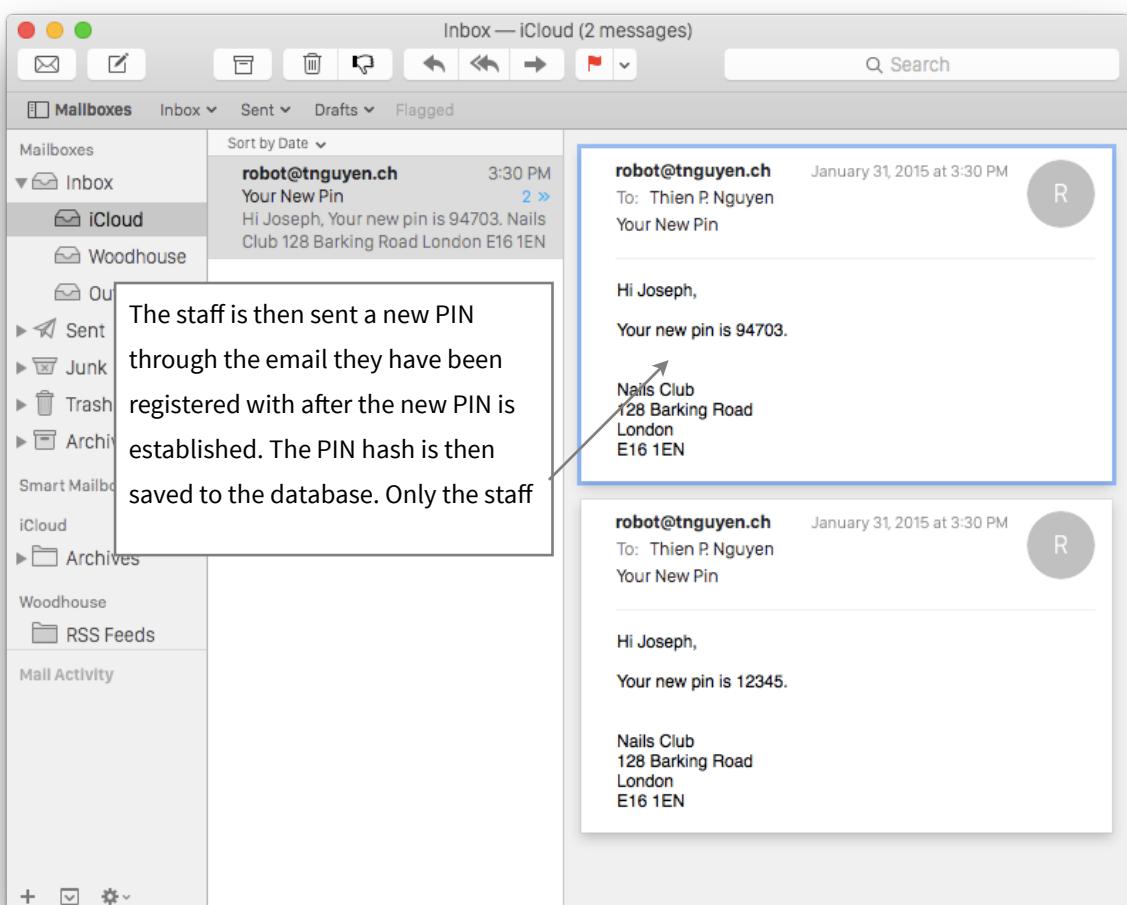
A new pin (12345) is hashed and checked with the database to see if this hash has been already been used.

If there's no match then the hash is then sent to the database. In this case the 12345 hash is already in the database, so it isn't sent.

Instead, the while loop is initiated and a new pin and its hash is generated.

This new hash is then checked with the database, if it isn't in the database it is then

A new pin has been set to Joseph.



The staff is then sent a new PIN through the email they have been registered with after the new PIN is established. The PIN hash is then saved to the database. Only the staff

Inbox — iCloud (2 messages)

Sort by Date ▾

robot@tnguyen.ch 3:30 PM Your New Pin 2 »

Hi Joseph, Your new pin is 94703. Nails Club 128 Barking Road London E16 1EN

robot@tnguyen.ch January 31, 2015 at 3:30 PM R

To: Thien P Nguyen
Your New Pin

Hi Joseph,
Your new pin is 94703.

Nails Club
128 Barking Road
London
E16 1EN

robot@tnguyen.ch January 31, 2015 at 3:30 PM R

To: Thien P Nguyen
Your New Pin

Hi Joseph,
Your new pin is 12345.

Nails Club
128 Barking Road
London
E16 1EN

Maintenance

5.1 System Overview

The resulting program is a database management system revolving around booking appointments. The system is compartmentalised for each user, such that the customer's options will be separate from the administrators and the staff, and so on. The customer will be able to book and cancel appointments, the staff will be able to check in the customer's appointments and the administrator is able to manage the system.

This program is created using four languages (programming or otherwise), PHP, 5.6.2 Javascript 1.8.5, HTML5 and CSS3. It includes libraries from jQuery, jQuery UI and Chart.js. It also includes the ReCAPTCHA library. All of which are included in the /assets folder and are embedded appropriately.

jQuery lays central to the users user interface so its necessary that the user have javascript enabled in order to fully appreciate the features set on the system. In the event that the user has it disabled, a PHP fallback is implemented.

ReCAPTCHA Requires an API key which is already set for the program to run on localhost. This doesn't need to be changed unless in the event that the domain is separate from the server.

I've chosen to use MAMP to run the MySQL Backend alongside the web-server. MAMP contains MySQL and Apache, and is free for commercial use. Alternatively you can use a similar software bundle/solution stack that will enable a computer system to run a MySQL and Web Server. You are free to edit the program using any desired text editor. Each file is written in UTF-8. It is imperative that the system is set as the root directory, such that the homepage will be directed to the users index.php, located on the root directory of the system.

There are some discrepancies between tables. For example, the admin table, metadata and closed days are as their own entities. This is done so for a few reasons. First of which is that these tables would have no direct relationship with the bookings. Also, the tables are used simply as a place to store the data.

There are no data that are to be read from files except from loading libraries, and logging into the database. Data is stored on the database, which include login details, email addresses and booking appointments. The system metadata is contained in the database also.

Form	Description
Login	To log into the system, authenticating the user in order to access their respective side of the system.
Register	Allows customers to register an account in order to use the system.
Booker	Allows customers to book an appointment within the system.
Cancel Bookings	Allows customers to cancel appointments they have made beforehand.

Form	Description
Edit User Information	Allows the users to adjust information that they have set during registration.
Staff Search	Allows the staff to search through appointments going on through the day.
Modify Closing Days	Allows the client to adjust the days that the store will be closing.
Modify Services	Allows the client to adjust the services and their price.
Administrator Settings	Allows the client to adjust system parameters.
New Staff	Allows the client to register a new staff.

Report	Description
Services	Displays different services available to the customer.
Customer List	Displays all registered customers, sorted alphabetically.
Staff Appointments	Displays current bookings on the day.
Admin Dashboard	Displays statistics related to bookings. This consists of graphs and quantified data.
Booking Calendar	Displays currently available appointments, alongside partially booked days, closed days and full days.
Staff List	Displays all registered staff, sorted in order of registration.

5.2 Annotated List of Program Code

Please refer for full annotations of the code in *Appendix*.

5.3 Procedure and Variable Lists

#	Procedures	Location	Description
1	count_instances()	Administrator's Dashboard	Returns a number of bookings per a certain month.
2	list_staff()	Administrator's User List	A Display template for staff details.
3	generate_pin()	Administrator's User List	Randomly generates a pin. Also checks if a PIN is used, upon which it will recur itself until it finds a PIN that is available.
4	database()	Booker Class	Establishing a link into the database class within the booker class.
5	QuickFetch()	Booker Class	A quicker handling for fetching data from the database, exclusive to the booker class only.
6	make_calendar()	Booker Class	Stores parameter variables into the class as variables to be used specifically within the booker class.
7	post()	Booker Class	The process of collecting booking details, validating the data before sending it to the database.

Scheduling & Booking System

#	Procedures	Location	Description
8	confirmation()	Booker Class	This procedure sends an email to the customer validating the booking process, and directs the customer to the confirmed page notifying that the booking was successful.
9	start_booking()	Booker Class	The procedure finds bookings that are already established in the database. Also establishes a link to create_days_arr subroutine.
10	minutes()	Booker Class	Converts string time into minutes.
11	create_days_table()	Booker Class	Creates the table rows where each date is stored.
12	day_switch()	Booker Class	Used to determine the property of each day. For example, whether each day is closed or partially full.
13	make_booking_slots()	Booker Class	Determines whether to establish a form or not. This is determined upon the date, where the day chosen is not 0.
14	start_calendar()	Booker Class	Creates the actual calendar design template.
15	create_form()	Booker Class	The input form that customers will use to create a booking.
16	create_days_arr()	Booker Class	Creates an array of days in the month.
17	initiate()	Database Class	Establishes a connection to the database.
18	DoQuery()	Database Class	Queries the database using a SQL statement.
19	fetch()	Database Class	Fetches the first result from the database after DoQuery().
20	fetchAll()	Database Class	fetches all the results from the database after DoQuery().
21	RowCount()	Database Class	Fetches the number of rows from the database after DoQuery().

Scheduling & Booking System

#	Variables	Location	Description
1	\$cookies	Logout	2D array representing \$_COOKIES
2	\$selected_date_timestamp	Customer's Booking	a converted timestamp in unix, of \$selected_date
3	\$current_password	Customer's Edit Details	A temporary holder containing the current password.
4	\$expired	Logout	A timestamp of the expiration time.
5	\$expiry	Logout	A timestamp of the expiration time.
6	\$fdom	Booker Class	acronym; first day of the month string.
7	\$booking_times	Booker Class	An array containing booking times.
8	\$email_parameters	Customer Registration	An array containing parameters necessary for the sending of an email.
9	\$email_params	Admin's User Details	An array containing parameters necessary for the sending of an email.
10	\$create_table	Setup	An array containing queries for the creation of tables.
11	\$query_parameters	Admin's Login	An array holding parameters for the query.
12	\$query_params	Administrator's Calendar Management Admin's Services Admin's User Details Booker Class Check Username Function Search Appointments Function Staff's Checkin Staff's Login Confirmation Customer's Edit Details Customer's Forgot Password Customer's Login Customer's Appointment Management Customer's Profile Customer Registration	An array holding parameters for the query.
13	\$serviceid	Booker Class	An unique identifier, identifying services.
14	\$pages	Admin's Booking List Admin's User Details	array containg the number of pages for a given page.
15	\$newpins	Admin's User Details	Array containing a pin and its hash.

Scheduling & Booking System

#	Variables	Location	Description
16	\$prevalue	Customer's Edit Details	array containing current user details.
17	\$staff_details	Admin's User Details	array containing details about a particular staff.
18	\$newdetails	Customer's Edit Details	Array containing details.
19	\$errors	Admin's Booking List Administrator's Calendar Management Admin's Login Admin's Services Admin's Settings Admin's User Details Booker Class System Core System Header Staff's Login Customer's Booking Customer's Edit Details Customer's Login Customer Registration Setup	Array containing errors to be printed out. This is used to prevent certain things to be carried out whenever an array is not empty.
20	\$extra	Booker Class Email Function Confirmation Customer's Forgot Password	array containing extra parameters.
21	\$insert_base_queries	Setup	Array containing queries used to insert at setup.
22	\$service	Booker Class	Array containing services found in the database.
23	\$days	Booker Class	array containing the days of the week.
24	\$results	Customer's Appointment Management	Array containing the query results.
25	\$rows	Booker Class Check Username Function Confirmation Customer's Edit Details Customer Registration	Array containing the query results.
26	\$values	Admin's Dashboard	array containing values.

#	Variables	Location	Description
27	\$num	Admin's Booking List Administrator's Calendar Management Admin's Login Admin's Services Admin's Settings Admin's User Details Appointment List Function Search Appointments Function Staff's Checkin Staff's Login Customer's Login	array holding results from an query. Multipurpose.
28	\$username_available	Check Username Function	boolean representing whether a username is available or not.
29	\$require_admin	Admin's Booking List Administrator's Calendar Management Admin's Dashboard Admin's Services Admin's Settings Admin's User Details System Core	boolean that is used to check whether an admin identification is required.
30	\$require_user	System Core System's 404 Customer's Booking Confirmation Customer's Edit Details Customer's Index Customer's Profile	boolean that is used to check whether a user identification is required.
31	\$check_if_exists	Setup	boolean that switches whenever a certain requirement is met.
32	\$day_capacity	Booker Class	Calculates the available capacity of the day.
33	\$password_again	Setup	contains a password. this is used to compare to \$password.

Scheduling & Booking System

#	Variables	Location	Description
34	\$password_confirm	Customer's Edit Details Customer Registration	contains a password. this is used to compare to \$password.
35	\$phoneno	Customer's Edit Details Customer Registration	contains a phone number.
36	\$pin	Email Function Staff's Login	contains a PIN.
37	\$password	Admin's Login System Database Database Credentials Page Customer's Edit Details Customer's Login Customer Registration Setup	contains the password or the password hash, depending on the file.
38	\$new_pin_hash	Admin's User Details	contains the PIN Hash. this is sent to the database.
39	\$new_pin	Admin's User Details	contains the PIN. This is sent to the staff.
40	\$r	Admin's Dashboard Booker Class	Counter variable.
41	\$x	Admin's Dashboard Setup	counter variable.
42	\$num_days_month	Booker Class	Counts the number of days in a month.
43	\$endmonth	Admin's Dashboard	Date of the last day of the current month.
44	\$forward	Booker Class Customer's Booking	Date which is a month forward to the date chosen.
45	\$back	Booker Class Customer's Booking	Date which is a month prior to the date chosen.
46	\$per_page	Admin's Booking List Admin's User Details	defines how many results for a page.
47	\$q	Admin's User Details Customer's Profile	defines queries. May be used as an array.

#	Variables	Location	Description
48	\$menutype	Admin's Booking List Administrator's Calendar Management Admin's Dashboard Admin's Login Admin's Services Admin's Settings Admin's User Details System Header Customer's Booking Customer's Login Customer's Appointment Management	determines the kind of navigation is used for each page.
49	\$usertype	Admin's User Details	determines the user type for each user.
50	\$output	Customer Registration	displays error results.
51	\$staff_expiry	Search Appointments Function Staff's Checkin Staff's Login	Expiry time for staff, used in \$_COOKIE.
52	\$datequery	Admin's Booking List	Extra parameter to be affixed into the main query. This contains the WHERE Date. Used optionally.
53	\$first_day	Booker Class Customer's Booking	first day of the month.
54	\$count	Admin's Booking List Admin's Dashboard Admin's User Details	Generic stepper variable
55	\$counter	Admin's Dashboard	Generic stepper variable
56	\$counts	Admin's Dashboard	Generic stepper variable
57	\$description	Admin's Services	Holder of descriptions for each metadata.

Scheduling & Booking System

#	Variables	Location	Description
58	\$query	Admin's Booking List Administrator's Calendar Management Admin's Dashboard Admin's Login Admin's Services Admin's Settings Admin's User Details Booker Class System Database Check Username Function Search Appointments Function System Header Staff's Checkin Staff's Login Confirmation Customer's Edit Details Customer's Forgot Password Customer's Login Customer's Appointment Management Customer's Profile Customer Registration Setup	holds a query which is to be sent to the database.
59	\$e	System Database	Holds error message in the event that a connection to the database is not established.
60	\$hostname	System Database Database Credentials Page	holds string containing the host name.
61	\$end	Customer's Booking	Holds the last booking time.

Scheduling & Booking System

#	Variables	Location	Description
62	\$directory	Administrator's Calendar Management Admin's Dashboard Admin's Services Admin's Settings Admin's User Details System Core System Header Staff's Checkin Customer's Splash Screen	Holds the location of the files relative to the root location.
63	\$newpassword	Email Function Confirmation	holds the new password. This is sent to the database after it has been verified.
64	\$day_count	Customer's Booking	Holds the results of a query, related to counting the number of bookings on a day.
65	\$link	Booker Class	hyperlink in the event that the customer clicks on a date; it will transfer them to the next part of the booking.
66	\$id	Administrator's Calendar Management Admin's Services Admin's Settings Admin's User Details	identifier. Used to uniquely identify certain types depending on the scenario. Multipurpose.
67	\$i	Admin's Booking List Admin's Dashboard Admin's User Details Booker Class	Incrementer variable.
68	\$char	Admin's User Details	indicate character in alphabet.
69	\$character	Admin's User Details	indicate character in alphabet.
70	\$message	Email Function System Core	message of the email is contained in this variable.

#	Variables	Location	Description
71	\$value	Admin's Settings Check Username Function System Encryption Search Appointments Function System Header Confirmation	multi purpose variable. Represents a value that is stored in this variable.
72	\$services	Admin's Dashboard Booker Class	multidimensional array containing services and their related data. Data is retrieved from a query.
73	\$option	Booker Class Customer's Appointment Management	multipurpose variable. In booker.php, it is used to contain HTML referring to links related to booking times. In the Customer's Appointment Management, it is used as a switch to determine what appointments should be shown.
74	\$database_name	System Database Database Credentials Page	Name of the database.
75	\$number_of_bookings	Admin's User Details	Number representing the number of bookings a user has made with the system.
76	\$numberofbookingsonday	Customer's Booking	Number representing the number of bookings on a day.
77	\$registered_accounts	Admin's Dashboard	number specifying the number of registered accounts.
78	\$fetch	Setup	placeholder for query results.
79	\$fetch_value	Booker Class	placeholder for query results.
80	\$_SESSION	Admin's Login Customer's Login	Predefined PHP Variable. Similarly used in the same style as \$_COOKIE.

#	Variables	Location	Description
81	\$_POST	Administrator's Calendar Management Admin's Login Admin's Services Admin's Settings Admin's User Details Booker Class Check Username Function Search Appointments Function Staff's Checkin Staff's Login Customer's Edit Details Customer's Forgot Password Customer's Login Customer's Appointment Management Customer Registration Setup	Predefined PHP Variable. Used to define variables that are sent through forms.
82	\$_GET	Admin's Booking List Admin's Login Admin's User Details Staff's Login Customer's Booking Confirmation Customer's Login Customer's Appointment Management Setup	Predefined PHP Variable. Used to get/define variables from the URL.

Scheduling & Booking System

#	Variables	Location	Description
83	\$_COOKIE	Booker Class Appointment List Function System Core System Header Staff's Checkin Staff's Login Customer's Edit Details Customer's Index Logout Customer's Appointment Management Customer's Profile Customer's Splash Screen	Predefined PHP Variable. Used to handle cookies set by the program and accessed only by the browser. Used mainly as a 2D Array.
84	\$_SERVER	System Captcha Booker Class System Database System Core System Header Staff's Checkin Customer's Booking Customer's Edit Details Logout Customer Registration Setup	Predefined PHP Variable. variable itself is an array containing information such as headers. Generally used in this program for directories.
85	\$privatekey	System Captcha Booker Class Customer Registration	Private key to be used for ReCAPTCHA.
86	\$publickey	System Captcha Booker Class Customer Registration	public key to be used for ReCAPTCHA.
87	\$order	Admin's Booking List	query part that defines the order of the query.
88	\$freq	Customer's Booking	query string to be sent to the database. Query is related to fetching the frequency from metadata.

Scheduling & Booking System

#	Variables	Location	Description
89	\$closed_days_query	Booker Class Customer's Booking	Query to be sent to the database related to finding closed days.
90	\$booking_date	Booker Class Email Function	refers to the date of the booking.
91	\$booking_service	Booker Class Email Function	refers to the service of the booking.
92	\$booking_time	Booker Class Email Function	refers to the time of the booking.
93	\$title	Admin's Booking List Administrator's Calendar Management Admin's Dashboard Admin's Login Admin's Services Admin's Settings Admin's User Details System Header Staff's Checkin Staff's Login System's 404 Customer's Booking Confirmation Customer's Edit Details Customer's Forgot Password Customer's Index Customer's Login Customer's Appointment Management Customer's Profile Customer Registration Setup Customer's Splash Screen	Refers to the title of the page. This is printed at the top of each page.
94	\$dtA	Customer's Appointment Management	Represents date and time of Appointment.

Scheduling & Booking System

#	Variables	Location	Description
95	\$date_number	Booker Class	Represents the day number of a date.
96	\$daynumber	Booker Class	Represents the day number of a date.
97	\$openhr	Admin's Dashboard	Represents the opening hour query.
98	\$username	Admin's Login System Database Database Credentials Page Customer's Edit Details Customer's Login Customer Registration Setup	represents username of a given user.
99	\$fullybooked	Customer's Booking	result of calculation related to whether the date is fully booked.
100	\$frequency	Customer's Booking	results of \$freq query is placed here.
101	\$stamp	Administrator's Calendar Management	Segments of a exploded date.
102	\$parts	Logout	Segments of a exploded string.
103	\$smonth	Admin's Booking List	select month; used to display months in a select (dropdown).
104	\$syear	Admin's Booking List	select year; used to display years in a select (dropdown).
105	\$selected_date	Booker Class Customer's Booking	Selected date is contained, ready to be parsed into the query.
106	\$sday	Admin's Booking List	Shortened for select day. Used in a loop to display days in a dropdown.
107	\$price	Admin's Services	specifies the price of a given service.
108	\$count_rows	Admin's Booking List Admin's User Details	stepper variable for counting rows in an array,
109	\$company_name	System Header	Stores Company Name.
110	\$code	Email Function Customer's Forgot Password	stores PIN and hashes for the user to see.
111	\$slogan	System Header	String containing a slogan if set in the database.
112	\$headers	Email Function	String containing email header.
113	\$salt	System Encryption	string containing the salt, which is used for encryption.

#	Variables	Location	Description
114	\$forename	Admin's User Details Email Function Customer's Edit Details Customer's Login Customer Registration	string that holds forename of user
115	\$msg	System Core	temporary variable used for loops. (In this case printing all values in array)
116	\$row	Administrator's Calendar Management Admin's Services Admin's Settings Admin's User Details Booker Class Appointment List Function Customer's Appointment Management	temporary variable used in foreach loop.
117	\$t	Admin's Dashboard	temporary variable, used in a foreach loop.
118	\$j	Booker Class	temporary variable, used on for loops.
119	\$item	Booker Class Customer's Booking	temporary variable; used in loops.
120	\$page	Admin's Booking List Admin's User Details	temporary variable, used in loops related to printing each page as a link.
121	\$d	Booker Class	Unix timestamp for parameters.
122	\$bookings_on_day	Booker Class	used as a counter for the number of bookings on a certain day.
123	\$cookie	Logout	used as part of an while loop.
124	\$box	Booker Class	used for the calendar; contains a day and its corresponding colour.
125	\$calendar	Customer's Booking	used to create object at runtime (booker class)
126	\$db	Check Username Function Search Appointments Function System Core	used to create object at runtime (database class)

#	Variables	Location	Description
127	\$db-	Admin's Booking List Administrator's Calendar Management Admin's Dashboard Admin's Login Admin's Services Admin's Settings Admin's User Details Check Username Function Search Appointments Function System Core System Header Staff's Checkin Staff's Login Customer's Booking Confirmation Customer's Edit Details Customer's Forgot Password Customer's Login Customer's Appointment Management Customer's Profile Customer Registration Setup	used to create object at runtime (database class)
128	\$arguments	System Database	Used to refer to input variables declared in the subroutine.
129	\$subject	Email Function	variable containing the subject in an email.
130	\$padding_end	Booker Class	variable containing HTML padding.
131	\$period	Booker Class	variable containing the date of booking.
132	\$table_name	Setup	variable containing the name of a table in the database.
133	\$monthly_bookings	Admin's Dashboard	Variable holding a query related to counting the number of bookings for the month.

Scheduling & Booking System

#	Variables	Location	Description
134	\$monthly_revenue	Admin's Dashboard	Variable holding a query related to counting the sum of bookings for the month.
135	\$error	System Captcha Booker Class Staff's Login Customer's Booking Customer's Forgot Password Customer Registration	variable holding an error message.
136	\$daily_bookings	Admin's Dashboard	Variable holding query related to daily bookings.
137	\$month	Admin's Booking List Administrator's Calendar Management Admin's Dashboard Booker Class Customer's Booking	variable holding the month of a date. Month is generally stored as an integer.
138	\$months	Admin's Dashboard	variable holding the month of a date. Month is generally stored as an integer.
139	\$start	Admin's Booking List Admin's User Details Customer's Booking	Variable related to first booking time.
140	\$startmonth	Admin's Dashboard	variable representing the first day of the month.
141	\$surname	Admin's User Details Customer's Edit Details Customer's Login Customer Registration	variable representing the surname of a user.
142	\$finish_time	Booker Class	variable that holds the finish time.
143	\$pattern	Check Username Function	variable to be used with ReCAPTCHA.

#	Variables	Location	Description
144	\$date	Admin's Booking List Administrator's Calendar Management Admin's Dashboard Booker Class Customer's Appointment Management Customer's Profile	variable to hold a date.
145	\$email	Admin's User Details Booker Class Email Function Customer's Edit Details Customer Registration Setup	variable to hold an email address.
146	\$email_confirm	Customer Registration	variable to hold an email address. Used to compare to \$email.
147	\$day	Admin's Booking List Administrator's Calendar Management Booker Class Customer's Booking	variable to hold the day.
148	\$activated_accounts	Admin's Dashboard	variable to store query results related to activated accounts.
149	\$closed_days	Booker Class Customer's Booking	variable to store query results related to closed days.
150	\$year	Admin's Booking List Administrator's Calendar Management Booker Class Customer's Booking	variable which represents a year.
151	\$string	Booker Class	variable used in a switch within the function day-switch.
152	\$time	Booker Class Customer's Booking	variable containing the time.
153	\$timee		
154	\$timeg		

Scheduling & Booking System

#	Variables	Location	Description
155	\$timeout	Admin's Login System Core Customer's Login	expiry time for users. (This is different to staff and admins, hence a separate timestamp for them).
156	\$today	Admin's Dashboard	variable containing today's date.
157	\$total	Admin's Dashboard	array containing all dates. This is pulled from the database using a query immediately declared prior to this variable.
158	\$totalrow	Admin's Dashboard	total number of services - 1, since the program counts from one.
159	\$totalrows	Admin's Dashboard	variable that stores a query result.
160	\$type	Admin's Services Email Function Confirmation	Variable that is used to define the action the program should do.
161	\$update	Administrator's Calendar Management Admin's Services Admin's Settings Admin's User Details System Core System Header Customer's Appointment Management Setup	array containing messages to be displayed to the user.
162	\$used_pins	Admin's User Details	Arary containing used pins in the database.
163	\$usedpin	Admin's User Details	individual pin that is used. This is used in a loop related to \$used_pins
164	\$usedpins	Admin's User Details	Arary containing used pins in the database.
165	\$user		
166	\$user_id	Admin's User Details Booker Class Email Function Customer's Edit Details Customer's Login	unique identifier for each user. This variable is a number which is used to identify a certain user in the database.
167	\$userdetails	Confirmation Customer's Forgot Password	array containing user's details.

5.4 Samples of Algorithm Design

Forgot	
Location	Customer's Password Recovery
Description	In the event that the customer forget their password, the system can send the customer an email of a new password. This is done after the customer validates that they have forgotten a password, by clicking on the link sent to their email after they have filled a form out.
<pre> if !empty(email) query = select * from users where email = email DoQuery(query) details = fetchall() if !empty(details) code = encrypt(random number) query = update users set forgot_code = code DoQuery(query) email(details[user], code) end if direct user to confirmation page end </pre>	<pre> if (isset(\$_POST['email'])){ \$query = "SELECT * FROM users WHERE email = :email"; \$query_params = array(':email' => \$_POST['email']); \$db->DoQuery(\$query, \$query_params); \$.userdetails = \$db->fetchAll(); if (\$userdetails) { \$code = encrypt(rand(99341, 1102400)); \$query = "UPDATE users SET forgot_code = :code WHERE email = :email"; \$query_params = array(':code' => \$code, ':email' => \$_POST['email']); \$db->DoQuery(\$query, \$query_params); \$extra = array(':code' => \$code); email(\$_POST['email'], \$userdetails[0] ['username'], \$userdetails[0]['forename'], "forgotten_password", \$extra); } header('Location: confirmation.php? type=forgot'); exit(); } </pre>

Adding Tables into the database	
Location	Setup
Description	This is used in the event that the administrator is migrating to a new system. This way, they only need to set up the connection to the database. The system can deal with the rest of the database counterparts, where it will create table names and insert values to make the system ready to use.
<pre> create_tables = array(query1, query2, queryN) for x=0, x<=6, x++ if x = 0 table_name = a end if if x = 1 table_name = b end if if x = 2 table_name = c end if if x = 3 table_name = d end if if x = 4 table_name = e end if if x = 5 table_name = f end if if x = 6 table_name = g end if check_if_exists = SHOW TABLES LIKE 'x' doQuery(check_if_exists) fetch <- fetch_results_from_query if !fetch doquery(table[x]) end end for insert_base_queries = (querya1, querya2, queryaN) foreach insert_base_queries as query doQuery(query) end for </pre>	<pre> if (count(\$errors) == 0) { \$password = encrypt(\$password); \$create_table = array(...); for (\$x = 0; \$x <= 6; \$x++) { switch (\$x) { case 0: \$table_name = "admin"; break; case 1: \$table_name = "booking"; break; case 2: \$table_name = "closed_days"; break; case 3: \$table_name = "metadata"; break; case 4: \$table_name = "service"; break; case 5: \$table_name = "staff"; break; case 6: \$table_name = "users"; break; } \$check_if_exists = "SHOW TABLES LIKE '\$table_name'"; \$db->doQuery(\$check_if_exists); \$fetch = \$db->fetch(); if (!\$fetch) { // echo \$table_name . " does not exist.
"; \$db->doQuery(\$create_table[\$x]); // echo \$table_name . " has been built for you.
"; } // else { // echo \$table_name . " exists
"; // } } \$insert_base_queries = array(...); foreach (\$insert_base_queries as \$query) { \$db->doQuery(\$query); } header('Location: setup.php?done'); } </pre>

Logins	
Location	Customer's and Administrator's Login
Description	In order to log into their respective systems, the user will need to verify that they are indeed them by inserting their username and password in the login form. If it is correct the system will send them to their respective dashboards.
<pre> username <- get username password <- get password if username and password not empty username <- trim username password <- encrypt(trim password) query <- select * from users where username = username and password = password DoQuery(query) result = fetch() if result if result[activated] = 1 if result[banned] = 0 log in else print error end else print error end else print error end else print error end </pre>	<pre> if (isset(\$_POST['username']) && (isset(\$_POST['password']))) { \$username = trim(\$_POST['username']); \$password = encrypt(trim(\$_POST['password'])); \$query = "SELECT * FROM admin WHERE username = :username AND password = :password"; \$query_parameters = array(':username' => \$username, ':password' => \$password); \$db->DoQuery(\$query, \$query_parameters); \$num = \$db->fetchAll(); if (\$num) { // //user entered correct details setcookie('admin[loggedin]', TRUE, \$admin_expiry, '', '', TRUE); header('Location: index.php'); exit(); } else { //user entered incorrect details array_push(\$errors, 'The username and password combination is not recognised. Please try again.'); } } </pre>

Database	
Location	Database Class
Description	The system revolves around a database, so it would be logical to create a class which will assist in writing the program code. This way, it will be quicker to write a statement that would connect to the database and to query the database.
<pre>class database public function initiate include page_with_db_variables.php try this->database = new database(hostname, db_name, username, password) catch die, print error end function public function doquery (query, arguments = array()) try this->result = this->database->prepare query this-> result = execute(arugments) catch die, print error end public function fetch return this->result->fetch() end public function fetchall return this->result->fetchall() end public function rowcount() return this->result->rowcount() end end </pre>	<pre><?php class database { public function initiate() { include(\$_SERVER['DOCUMENT_ROOT'].'/includes/ db.php'); try { \$this->database = new PDO("mysql:host={\$hostname};dbname={\$database_na me}", \$username, \$password); } catch(PDOException \$e) { die(\$e->getMessage()); } \$this->database- >setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION); } public function DoQuery(\$query, \$arguments = array()) { try { \$this->result = \$this->database- >prepare(\$query); \$this->result->execute(\$arguments); } catch(PDOException \$e) { die(\$e->getMessage()); } } public function fetch() { return \$this->result->fetch(); } public function fetchAll() { return \$this->result->fetchAll(); } public function RowCount() { return \$this->result->RowCount(); } ?></pre>

Email Function	
Location	Email
<pre>function email(\$email, \$userid, \$forename, \$type, \$extra = array()) { \$headers = "From: robot@tnguyen.ch"; if (\$type == "confirm_registration") { if (isset(\$extra["pin"])) { \$pin = \$extra["pin"]; \$subject = "Subject"; \$message = \$message . \$pin; } else { echo "No pin is inserted."; } } if (\$type == "forgotten_password") { if (isset(\$extra["pin"])) { \$pin = \$extra["pin"]; \$subject = "Subject"; \$message = \$message . \$pin; } else { echo "No pin is inserted."; } } ... mail(\$email, \$subject, \$message, \$headers); }</pre>	<p>The system will send emails to the user in response to certain actions, such as registration, bookings and recovery.</p> <pre>function email(\$email, \$user_id, \$forename, \$type, \$extra = array()) { \$headers = "From: robot@tnguyen.ch"; if (\$type == "confirm_registration") { if(isset(\$extra[":pin"])) { \$pin = \$extra[":pin"]; \$subject = 'Time to confirm your email'; \$message = "Hi \$forename, \n\n You need to activate your account. Click the link below. \n\n"; \$message .= "http://projects.tnguyen.ch/ comp4/confirmation.php?type=registration&value= \$pin";} else { echo "No pin is inserted."; } } ... // Send \$message .= "\n\n\n"; \$message .= "Nails Club\n"; \$message .= "128 Barking Road\n"; \$message .= "London\n"; \$message .= "E16 1EN\n"; mail(\$email, \$subject, \$message, \$headers); } ?></pre>

Password Strength	
Location	Customer's Registration & Customer's Edit
Description	In order to improve the security of the system, Customer's are shown how strong their password is. Using a combination of simple mathematics and regular expressions, a strength of a password can be deduced by how many regular expression statements it matches.
<pre> if password.length > 7 if password.matches(/([a-z].*[A-Z]) ([A- Z].*[a-z])/) strength = strength + 1 if password.matches(/([a-zA-Z])/) && password matches(/([0-9])/) strength = strength + 1 if password.matches(/([a-zA-Z])/) && password matches(/([0-9])/) strength = strength + 1 if password.matches(/(.*[!%,&,@,#, \$,^,*,?,_,~].*[!%,&,@,#,\$,^,*,?,_,~])/) strength = strength + 1 </pre>	<pre> function password_validator(password){ var strength = 0 if (password.length < 5) { \$('#password_str').removeClass() \$('#password_str').addClass('short') return 'Too short!' } if (password.length > 7) strength += 1 if (password.match(/([a-z].*[A-Z]) ([A- Z].*[a-z])/)) strength += 1 if (password.match(/([a-zA-Z])/) && password.match(/([0-9])/)) strength += 1 if (password.match(/([a-zA-Z])/) && password.match(/([0-9])/)) strength += 1 if (password.match(/(.*[!%,&,@,#, \$,^,*,?,_,~].*[!%,&,@,#,\$,^,*,?,_,~])/)) strength += 1 ... } </pre>

Encryption	
Location	Encryption
Description	Cryptographic hashing methods are useful in securing data as it simply doesn't keep the actual input. This way, data can be stored in the database but it can't be used to obtain data.
<pre> function encrypt(value) salt = '}B8->{9#3*4L3t98d9QF3)&#9j?tKf' salt = md5(salt) value = hash(salt, md5(value)) return value end </pre>	<pre> function encrypt(\$value) { \$salt = '}'B8->{9#3*4L3t98d9QF3)&#9j?tKf'; \$salt = md5(\$salt); \$value = md5(\$value); \$value = hash('ripemd160', \$salt . \$value); return \$value; } </pre>

Paginator	
Location	Paginator
<pre> query = select from x order by date(desc) query = query . where date = 'the_date' rows = number of rows (get from query) per_page = 10 pages = ceil(rows/per_page) if !empty(page) page = 1 else page = get page start = (page - 1)/page query = query + " limit start, per page" DoQuery(query) num = FetchAll() // displaying results while i = 0 and i < pages if page = i echo i else echo i end if i = i + 1 end while </pre>	<pre> \$query = "SELECT booking.id, booking.date, users.forename, .. service.id"; \$order = "ORDER BY DATE(booking.date) DESC, booking.time DESC"; \$count_rows = "SELECT count(*) FROM booking"; if (isset(\$_GET)) if (isset(\$_GET['year']) & isset(\$_GET['month'])& isset(\$_GET['day'])) { if (!empty(\$_GET['year']) & empty(\$_GET['month'])& !empty(\$_GET['day'])) { if (checkdate(\$_GET['month'], \$_GET['day'], \$_GET['year']) == TRUE){ \$date = \$_GET['day'].("/"); \$_GET['month']. "/" . \$_GET['year']; \$datequery = " WHERE date = '". \$_GET['year'] ."-". \$_GET['month'] ."-". \$_GET['day'] ."' "; \$count_rows = \$count_rows.\$datequery; \$query = \$query.\$datequery.\$order; } else { array_push(\$errors, "The selected date is invalid, please try again."); \$query = \$query . \$order; } } } \$db->DoQuery(\$count_rows); \$count = \$db->fetch(); \$per_page =10; \$pages = ceil(\$count[0]/\$per_page); if(!isset(\$_GET['page'])){ \$page="1"; }else{ \$page=\$_GET['page']; } \$start = (\$page - 1) * \$per_page; \$query = \$query . " LIMIT \$start, \$per_page"; \$db->DoQuery(\$query); \$num = \$db->fetchAll(); //Show page links for (\$i = 1; \$i <= \$pages; \$i++){ if (\$page == \$i){ echo '<li id="active">' . \$i . ''; } else { echo '' . \$i . ''; } } } </pre>

PIN Generator	
Location	Customer's Password Recovery
Description	In the event that the customer forget their password, the system can send the customer an email of a new password. This is done after the customer validates that they have forgotten a password, by clicking on the link sent to their email after they have filled a form out.
<pre>function generate_pin(used_pins as array) new_pin = random_number(10000,99999) new_pin_hash = encrypt(new_pin) while new_pin_hash is in used_pins array generate_pin(used_pins as array) end while return array pin = new_pin pin_hash = new_pin_hash end</pre>	<pre>function generate_pin(\$used_pins = array()){ \$new_pin = rand(10000,99999); \$new_pin_hash = encrypt(\$new_pin); while (in_array(\$new_pin_hash, \$used_pins)){ generate_pin(\$used_pins); } return array('pin' => \$new_pin, 'pin_hash' => \$new_pin_hash); }</pre>

Appraisal

6.1 Objectives

6.1.1 Specific Objectives

Input

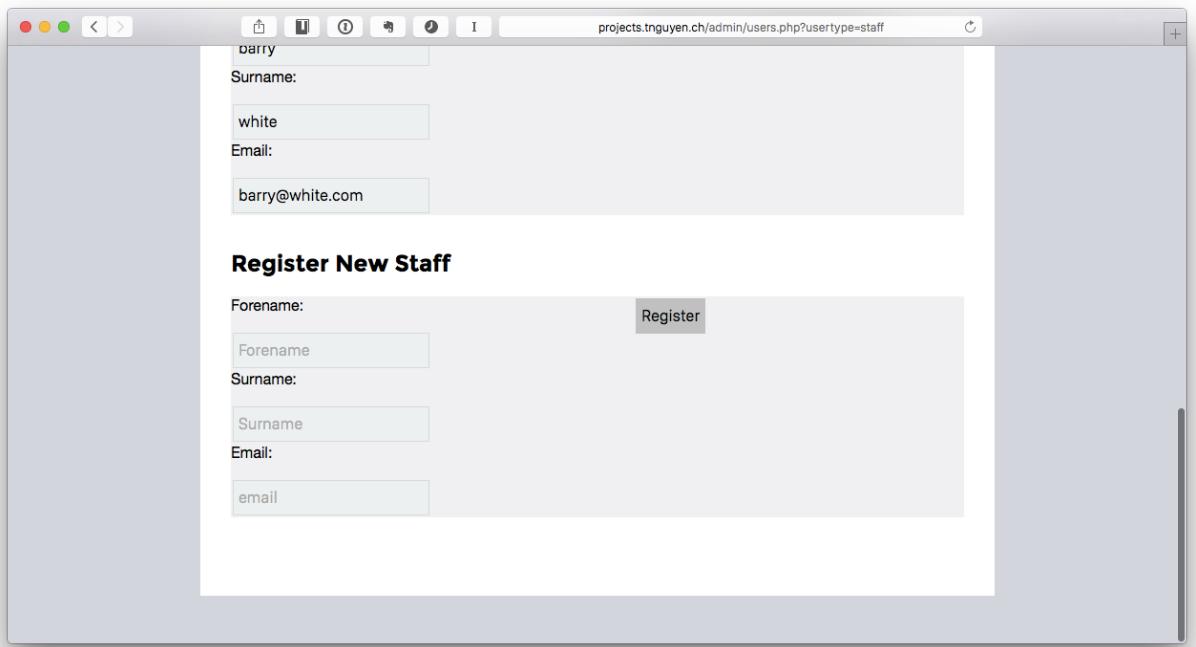
1. Inputting a new staff

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.

Objective: Met ✓

Evidence:

Administrators can create a new staff and modify their details in the users section of the administrators dashboard.



The screenshot shows a web browser window with the following details:

- Address Bar:** projects.tnnguyen.ch/admin/users.php?usertype=staff
- Form Fields (Visible in Browser):**
 - Forename: barry
 - Surname: white
 - Email: barry@white.com
- Form Title:** Register New Staff
- Form Fields (Visible in Modal):**
 - Forename: [Placeholder: Forename]
 - Surname: [Placeholder: Surname]
 - Email: [Placeholder: email]
- Buttons:** A "Register" button is located next to the form fields.

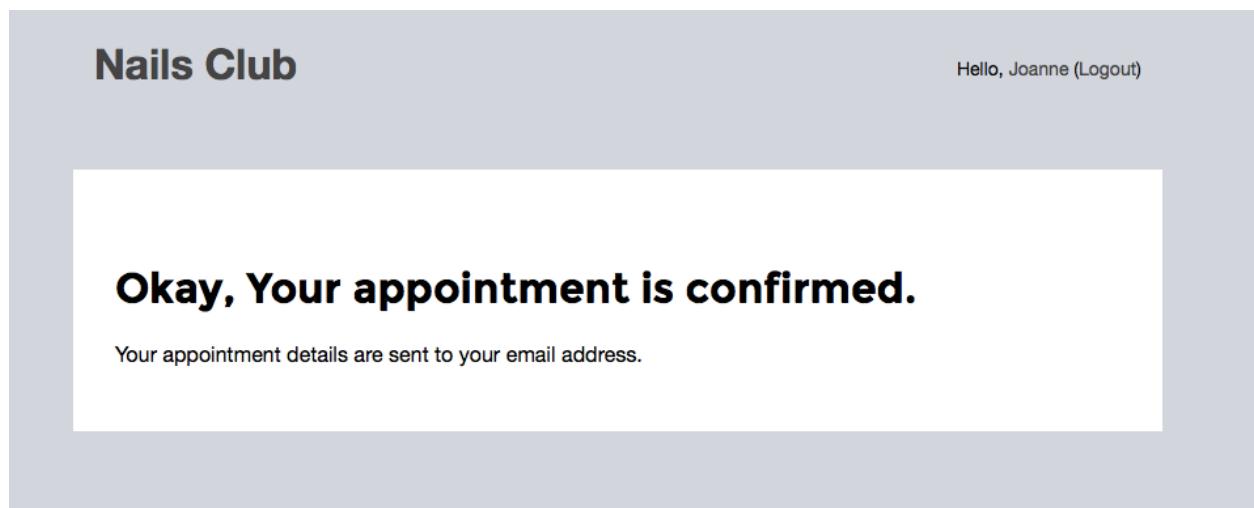
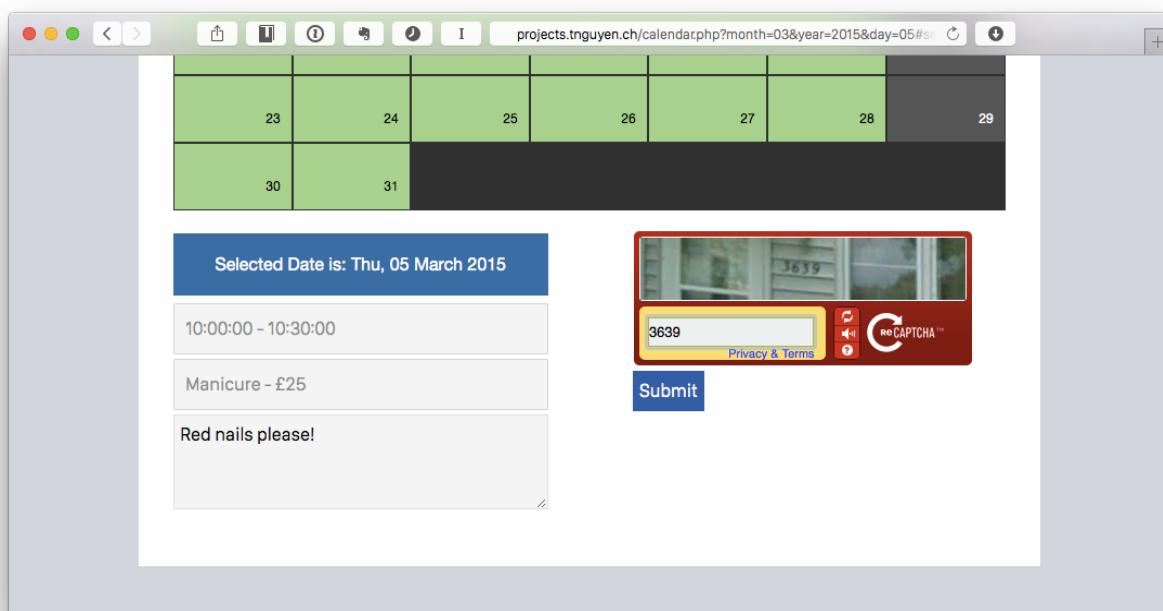
2. Set new appointment

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.

Objective: Met ✓

Evidence:

Customers are able to book an appointment when they have logged into the system. They are able to book an appointment by clicking their desired date, time and service.



3. Updating customer details

The customer should be able to update their credentials, such as their name and password.

Objective: Met ✓

Customers are able to modify their details on the edit page.

The screenshot shows a web browser window titled "Nails Club - Edit" with the URL "http://localhost/edit.php". The page is titled "Nails Club" and displays a "Hello, Jon (Logout)" message. The main content is a form titled "Your Details." with the following fields:

- Forename:
- Surname:
- Password:
- Confirm Password:
- Password Strength:
- Email:
- Phone Number:
- Enter your current password to confirm changes:
- Update button (highlighted in red)

4. Inputting customer details

The customer should be able to register an account using their name, phone, email and password.

Objective: Met ✓

Evidence:

Customers can register an account by clicking on the register button when they access the homepage.

The screenshot shows a web browser window titled "Nails Club - Register" with the URL "http://localhost/register.php". The page has a header "Nails Club" and a main title "Register". The form fields are as follows:

- Forename: First (text input)
- Surname: Last (text input)
- Username: Username (text input)
- Password: Password (text input)
- Confirm Password: Password (text input)
- Email: example@domain.com (text input)
- Confirm Email: example@domain.com (text input)
- Phone Number: (text input)
- Captcha: A reCAPTCHA interface with a blurred image and a text input field containing "178". It includes a "Type the text" label, a "Privacy & Terms" link, and a "reCAPTCHA" logo.

A "Next" button is located at the bottom right of the form area.

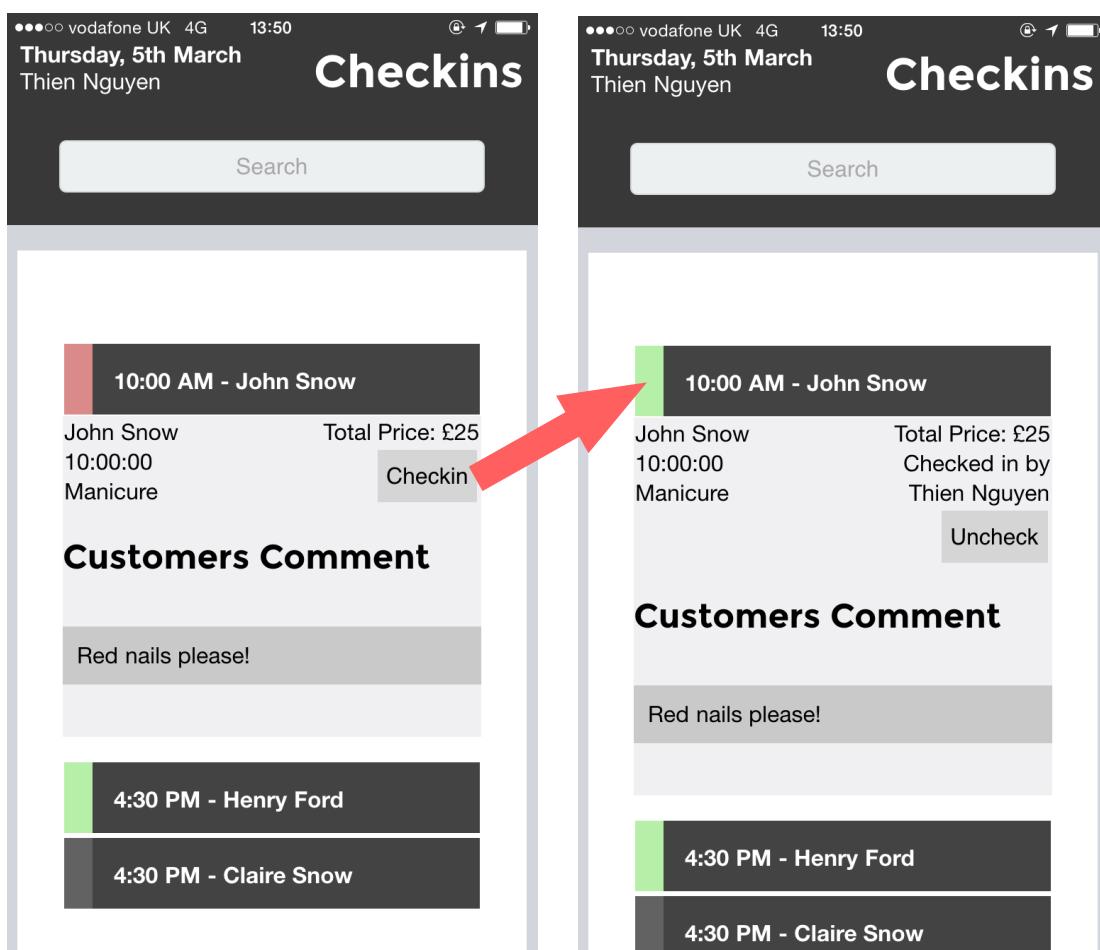
5. Checking-in customer

The staff must be able to check in the customer in order to validate the booking process.

Objective: Met ✓

Evidence:

Staff can check in customers by using their side of the system. In the system they can find customers and check them in.

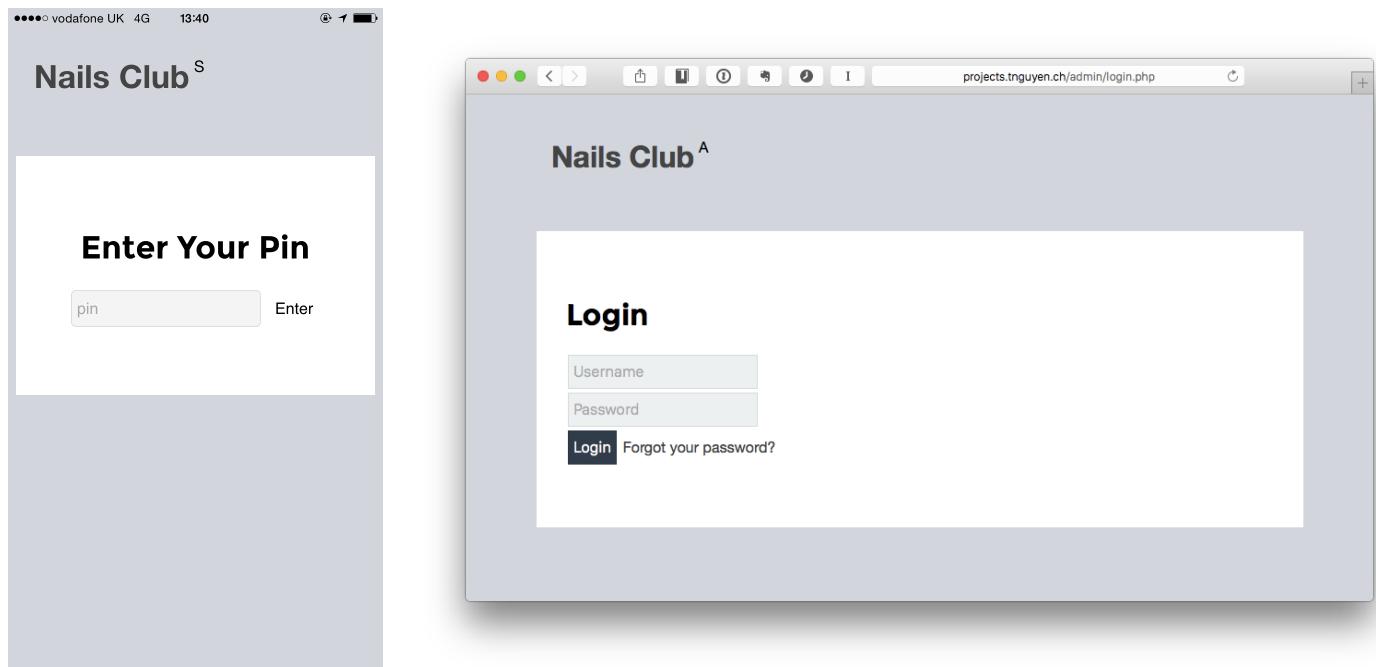
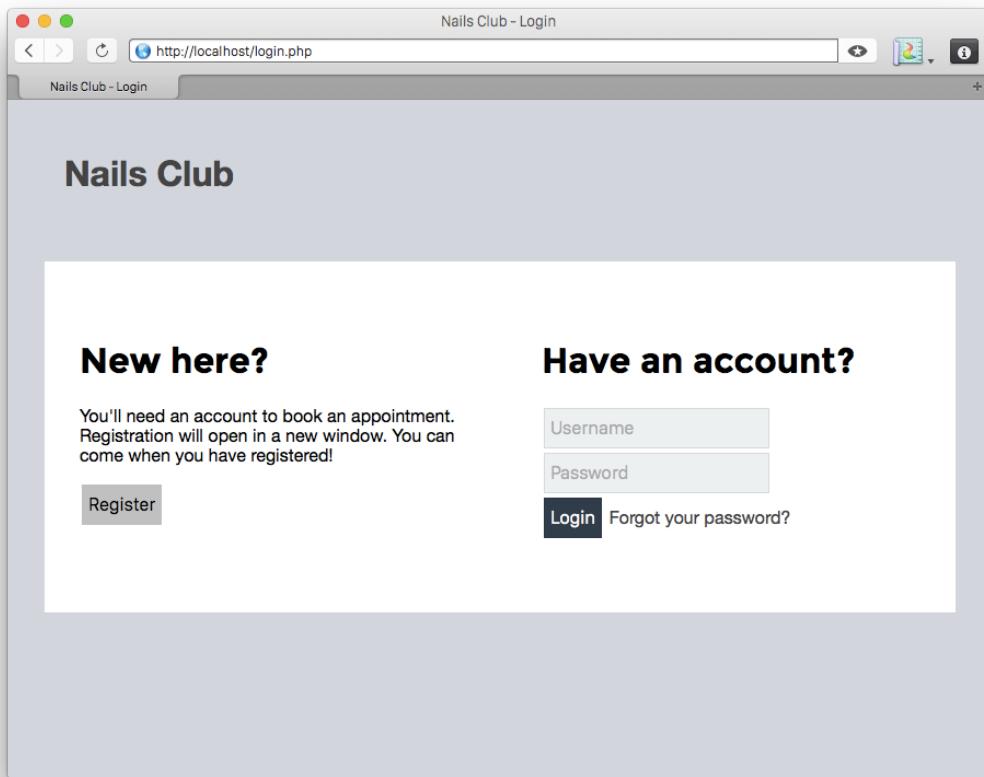


6. Login

The staff and the customer must be able to log into their respective control panels.

Objective: Met ✓

Evidence: the users have their own respective login pages which will lead to their side of the system.



Output

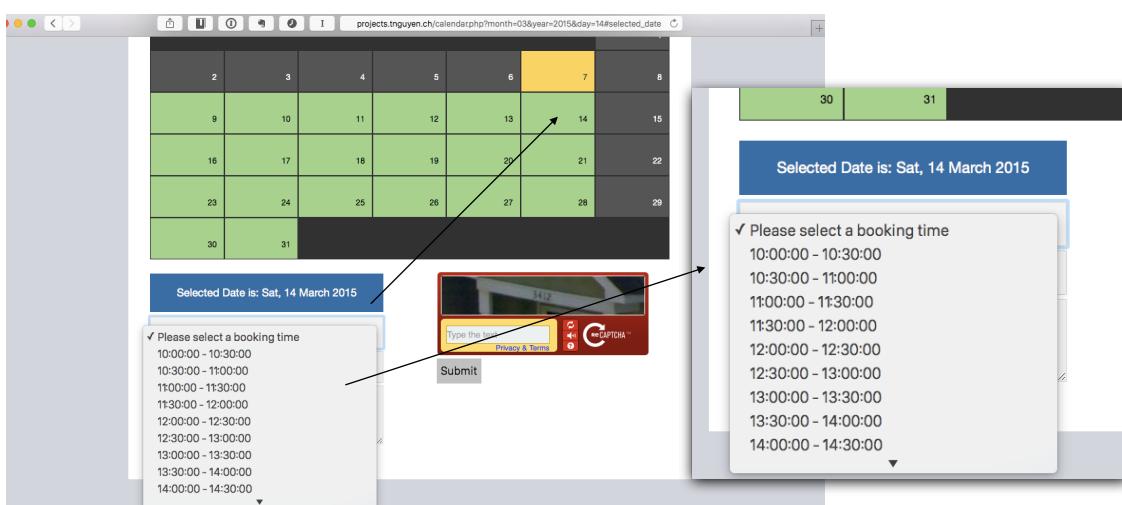
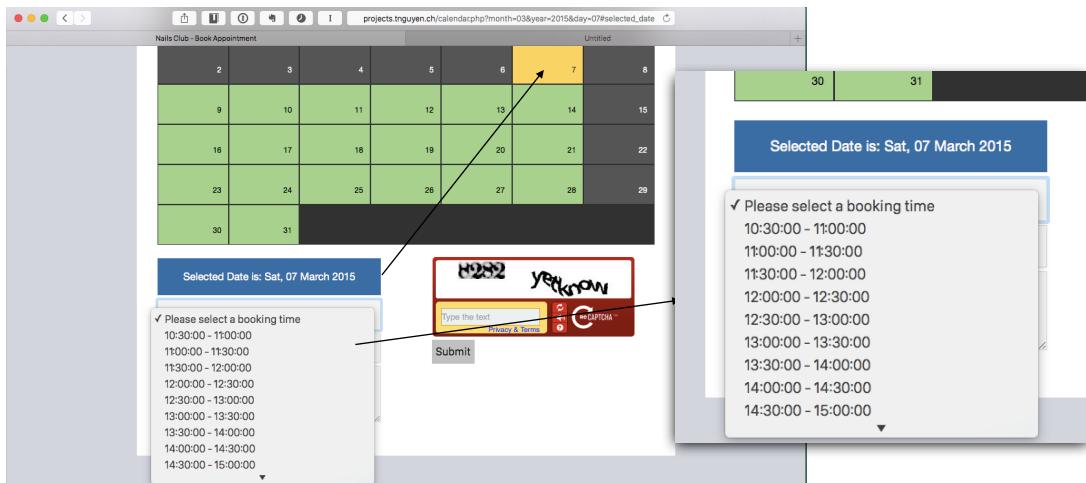
7. Load available timings for appointments

The system should be able to get information from the database, calculate availability, and display it for the secondary user to see.

Objective: Met ✓

Evidence:

Days that are partially booked are shown and times that are used are omitted in the selection.



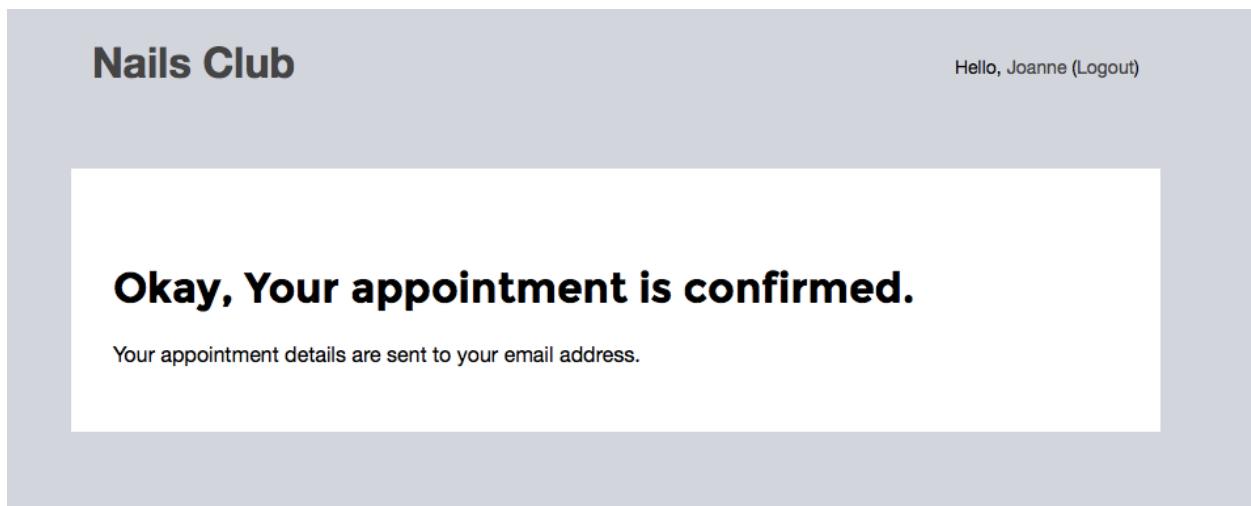
Days that are shown in yellow have bookings already set on that date. In this example, on the 7th of March the day is shown yellow, and an appointment for 10:00 to 10:30 is missing. But on the 14th March the slot for 10:00 to 10:30 is shown available for taking.

8. Confirmation Message

The customer must be notified that the booking has processed successfully.

Objective: Met ✓

Evidence: Confirmation messages are shown whenever the user has successfully completed a process.



A screenshot of an email inbox. The sidebar shows a list of folders: "Inbox" (highlighted in red), "Starred", "Important", "Sent Mail", "Drafts (6)", "[Imap]/Archive", "[Mailbox]", "College", "Interviews", "Deleted Messages", "lolz", and "Notes". The main area shows an email from "robot@tnguyen.ch" to "me" at "2:01 PM (9 minutes ago)". The email subject is "Your Appointment" and the body contains the appointment details: "You have an appointment at 2015-03-05, 10:00:00 for a Manicure." Below the email, the appointment details are repeated: "Nails Club", "128 Barking Road", "London", "E16 1EN". The interface is a standard web-based email client.

9. Password Validation

When registering, a password strength meter shows how strong the password is.

Objective: Met ✓

Evidence: The password strength meter shows how strong the password is depending on what is in the string.

Username:

Password: Password Strength: Good

Confirm Password:

Email:

Username:

Password: Password Strength: Too short!

Confirm Password:

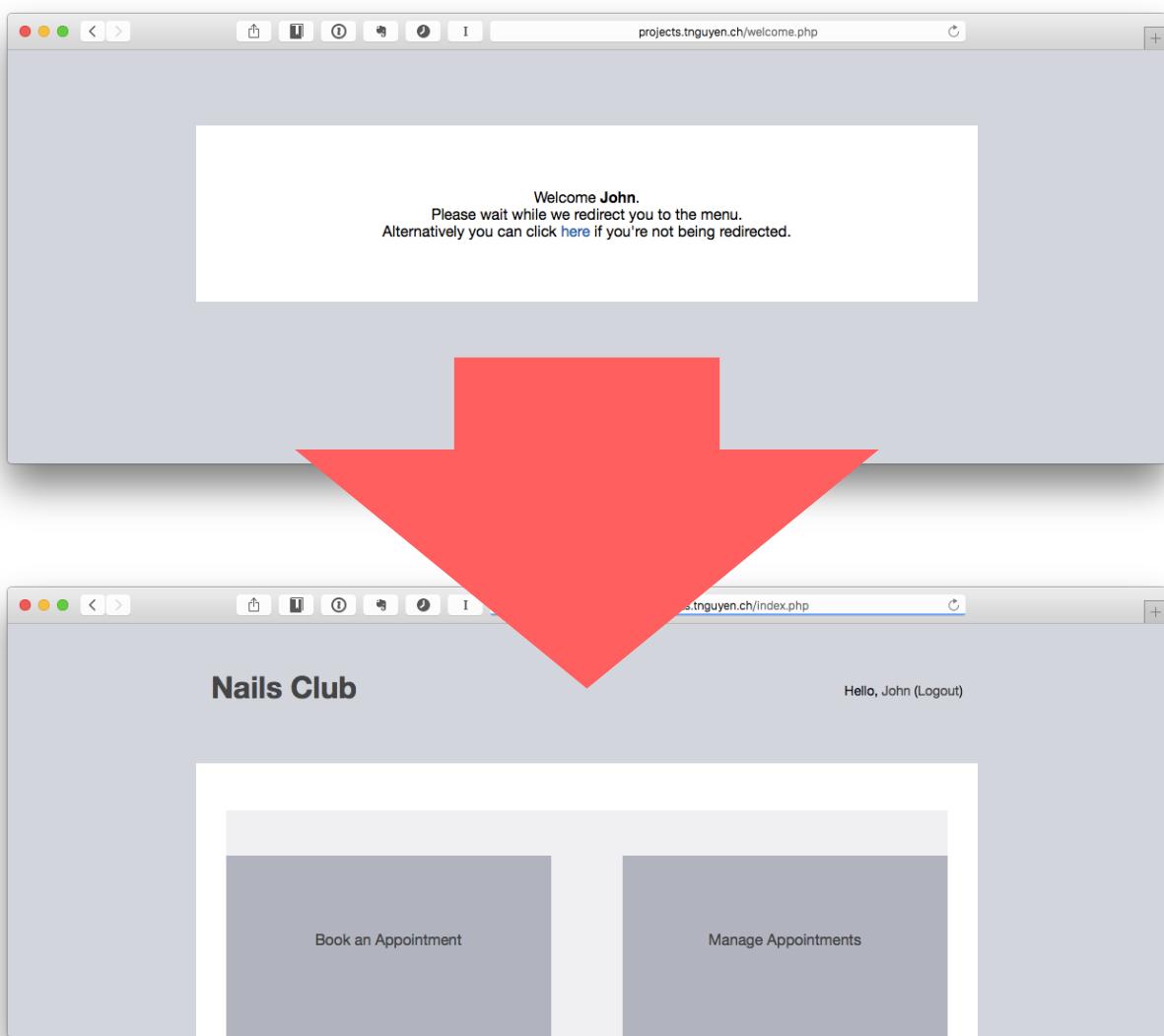
Email:

10. Display User on Login (Dashboard)

The user credentials should be displayed when the primary user logs into the dashboard.

Objective: Met ✓

Evidence: When the customer logs in, they are greeted with a message before being redirected to the dashboard.

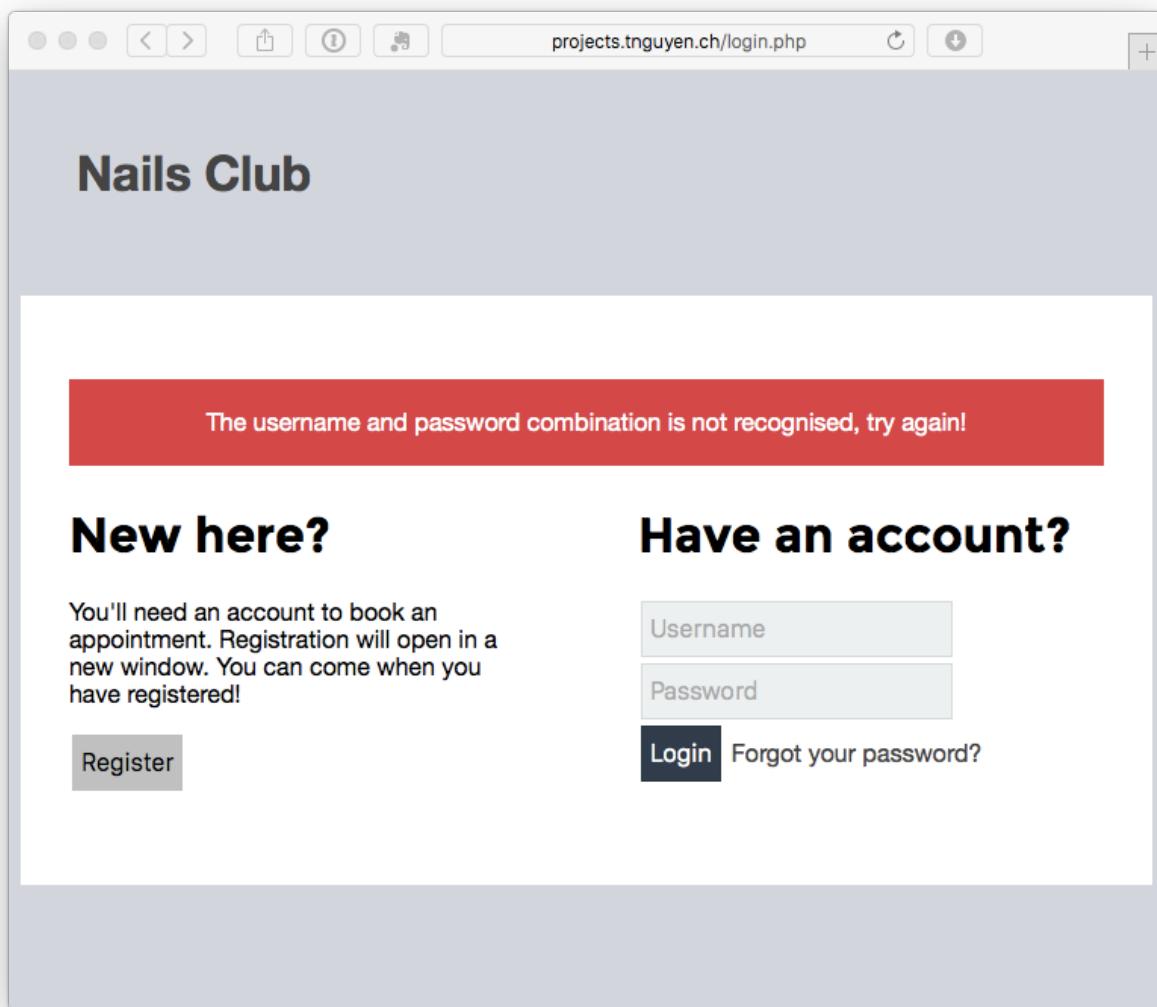


11. Error Messages

The system should output error messages relating to the problem that has occurred, such as being unable to log in or notifying the customer for invalid input values.

Objective: Met ✓

Evidence:



A screenshot of a web browser window displaying a user registration form. The URL in the address bar is `projects.tnnguyen.ch/register.php`. The form includes fields for 'Username' (empty), 'Password' (containing four asterisks, highlighted with a blue border), and 'Confirm Password' (empty). To the right of the password field, a 'Password Strength' indicator shows a red box with the text 'Too short!'. Below the password field, a red box displays the validation message: 'Your password must be at least six characters long.'

Username:

Username

Password:

>Password Strength:

Too short!

Your password must be at least six characters long.

Confirm Password:

Password

A screenshot of a web browser window showing an appointments page for 'Nails Club'. The URL in the address bar is `localhost/admin/appointments.php?year=2015&month=02&day=30`. The page title is 'Nails Club^A'. The top navigation bar includes links for DASHBOARD, APPOINTMENTS (which is currently selected and highlighted in dark grey), CALENDAR, SERVICES, USERS, and SETTINGS. A welcome message 'Hello, Administrator (Logout)' is visible on the right. A red banner at the top of the main content area displays the error message: 'The selected date is invalid, please try again.' Below the banner, there are input fields for 'Year' (2015), 'Month' (02), 'Day' (30), and a 'Query' button. The text 'Appointments on' is followed by the date '30/01/2015'.

Nails Club^A

Hello, Administrator (Logout)

The selected date is invalid, please try again.

DASHBOARD APPOINTMENTS CALENDAR SERVICES USERS SETTINGS

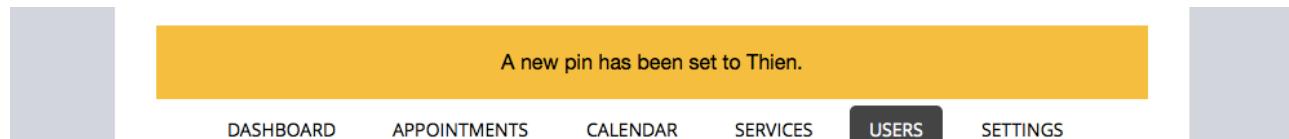
Appointments on
30/01/2015

12. Updated Information

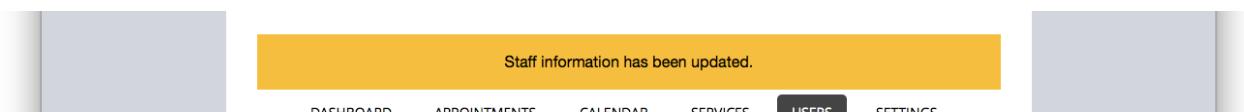
Updated information should be displayed and the user should be notified of a change.

Objective: Met ✓

Evidence:



A screenshot of the "Nails Club" application showing a customer profile for "jonsno". The top navigation bar is identical to the previous screenshot. A yellow message box says "Jon is now banned.". Below it, the customer details show "Name Jon Snow", "Email thien.nguyen@me.com", and "Phone No. 0747811111". A section titled "Statistics" indicates "Has booked a total of 31 appointments." with a "Unban" button.



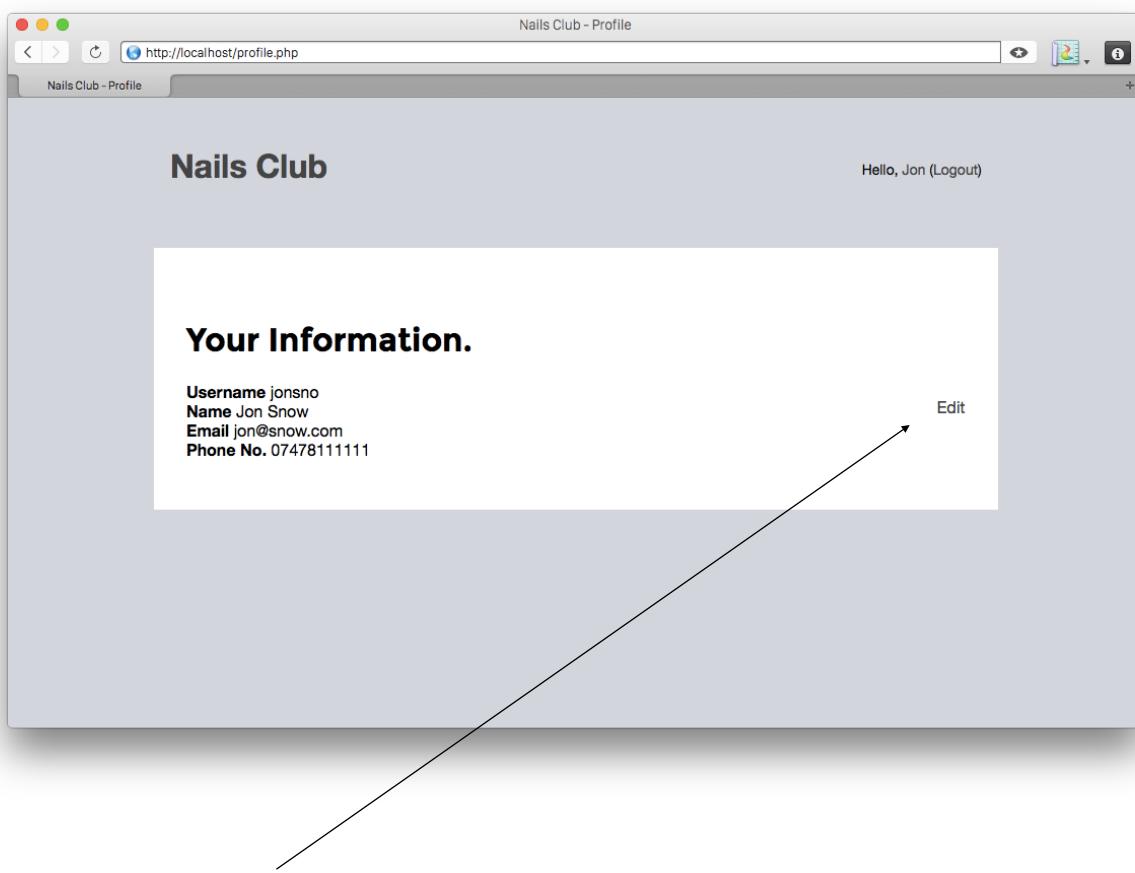
A screenshot of the "Nails Club" application showing a staff profile. The top navigation bar is identical. A yellow message box says "Thien has been included into the database. A PIN is sent to his email address at mint@null.net." Below it, the staff details show "Name Thien Nguyen", "Email mint@null.net", and "Phone No. 0747811111".

13. Showing User Information

The system should be able to display user information on a page that allows them to change credentials.

Objective: Met ✓

Evidence:



Clicking on this link takes them to a page where they are able to edit their credentials.

Nails Club - Edit

Nails Club - Edit

Hello, Jon (Logout)

Nails Club

Your Details.

Forename:

Surname:

Password:

Confirm Password:

Password Strength: No Data

Email:

Phone Number:

Enter your current password to confirm changes:

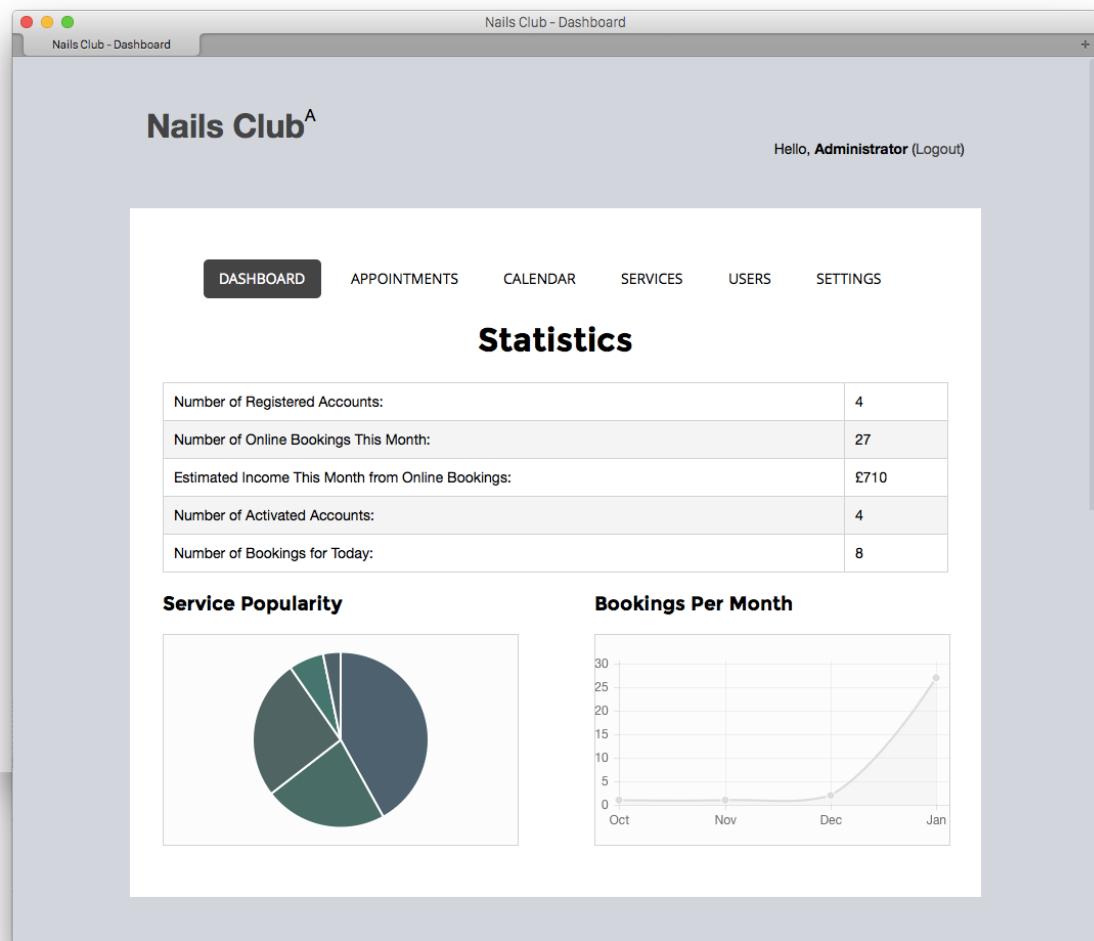
14. Show Query Results

The system must be able to display the sorted results of queries. The queries must have the option to access multiple tables in unison. The list of queries include:

- Showing system statistics related to the bookings.
- Displaying all bookings.
- Displaying all users.

Objective: Met ✓

Evidence:



Scheduling & Booking System

The screenshot shows a web browser window titled "Nails Club - Appointments" at the URL <http://localhost/admin/appointments.php>. The page header includes the "Nails Club" logo and a "Hello, Administrator (Logout)" message. A navigation bar at the top has tabs for DASHBOARD, APPOINTMENTS (which is selected), CALENDAR, SERVICES, USERS, and SETTINGS. Below the navigation is a search bar with fields for Year, Month, Day, and a "Query" button. The main content area displays "Appointments on 05/01/2015". A single appointment is listed: "1:30 PM - Jon Snow". The appointment details show "Jon Snow", "13:30:00", "Full Set", "Total Price: £48", and "Checked in by Joseph Good".

The screenshot shows a web browser window titled "Nails Club" at the URL <localhost/admin/users.php?char=S>. The page header includes the "Nails Club" logo and a "Hello, Administrator (Logout)" message. A navigation bar at the top has tabs for DASHBOARD, APPOINTMENTS, CALENDAR, SERVICES, USERS (which is selected), and SETTINGS. Below the navigation is a search bar with a dropdown menu showing "Customers". The main content area displays a list of users starting with the letter "S": "3 - Jon Snow (jonsno)" and "5 - Claire Snow (clairesno)".

The screenshot shows a web-based administrative interface for a nail salon. The title bar reads "Nails Club - Users" and the URL is "http://localhost/admin/users.php?usertype=staff". The top navigation menu includes links for DASHBOARD, APPOINTMENTS, CALENDAR, SERVICES, USERS (which is highlighted in a dark box), and SETTINGS. A welcome message "Hello, Administrator (Logout)" is displayed. Below the menu, a section titled "Staff" is shown. It contains four input fields: "Forename" (Joseph), "Surname" (Good), "Email" (jg00d@good.com), and a "Staff" dropdown menu which is currently set to "Staff". To the right of these fields are three buttons: "Update", "Ban", and "Generate New PIN". Below this section, there is another row with a "Forename" field containing "jg00d@good.com" and the same three buttons ("Update", "Ban", "Generate New PIN").

Processing

15. Identify customers uniquely

The system must be able to uniquely identify each customer.

Objective: Met ✓

Evidence: Customers data are stored in the database uniquely in the users table, with an ID for each entry. Duplicate data is removed due to normalisation.

16. Identify bookings uniquely

The system must be able to uniquely identify each booking.

Objective: Met ✓

Evidence: Booking data are stored in the database uniquely in the booking table, with an ID for each entry. Duplicate data is removed due to normalisation.

17. Identify staff uniquely

The system must be able to uniquely identify each staff.

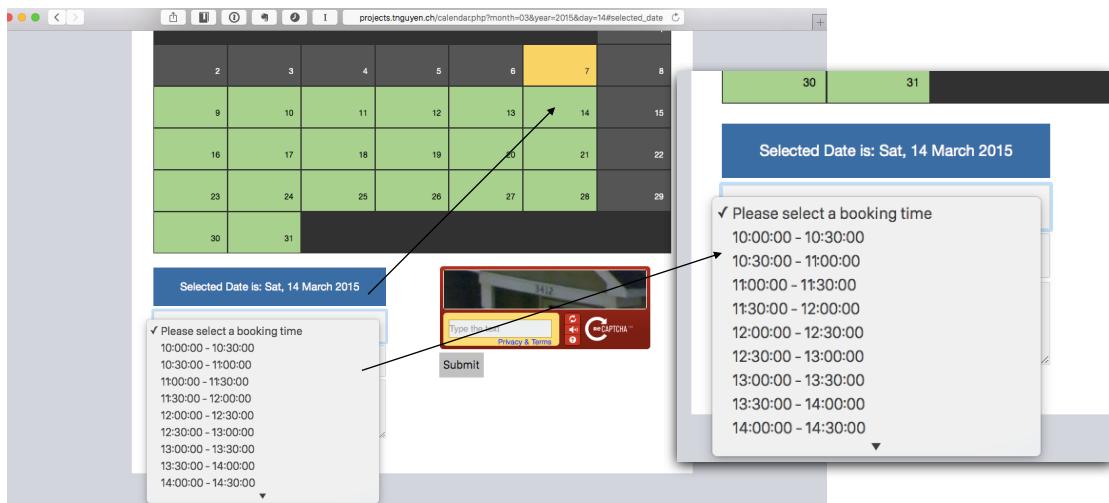
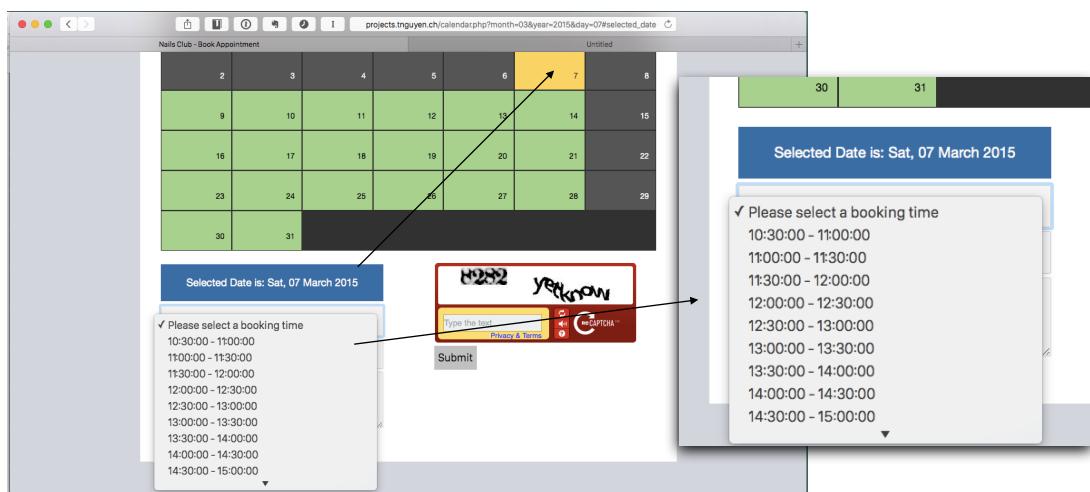
Objective: Met ✓

Evidence: Staff data are stored in the database uniquely in the staff table, with an ID for each entry. Duplicate data is removed due to normalisation.

18. Check Availability

The system must be able to check if a period is free for an appointment.

Objective: Met ✓



Evidence: Availability is done through the booker class. The system goes through all appointments that are on the database already for the date the customer has chosen. All of the appointments that are already stored will be removed from the list of available times that are shown to the customer.

```

if ($this->count >= 1) {
    foreach ($this->prior_bookings as $row) { // Check for bookings and remove any
previously booked slots
        foreach ($booking_times as $i => $r) {
            if ($row['start'] == $r && $row['date'] == $this->year . '-' . $this->month .
'-' . $this->day) {
                unset($booking_times[$i]);
            }
        }
    }
    foreach ($booking_times as $booking_time) {
        $finish_time = strtotime($booking_time) + $this->booking_frequency * 60; // Calculate
finish time
        $option .= "<option value='" . $booking_time . "'>" . $booking_time . " - " .
date("H:i:s", $finish_time) . "</option>";
    }
}

```

In the above code, an extract of the `create_form()` subroutine is shown. In this code, the prior bookings are unset from the array, before displaying the available times.

19. Cookies & User Sessions

The system must be able to separate the cookies depending on the user using data structures (such as 2D arrays) and use them for user sessions.

Objective: Met ✓

Evidence: Details about the different users are stored in different parts of the cookie array. The cookie variable is in it self, is an two-dimensional array so I have used it so to cater to all users. `$_COOKIE['userdata']`, `$_COOKIE['admin']`, and `$_COOKIE['staff']` are used for their respective users and contain session times which expire in the event of inactivity.

20. Password Validation

When registering, the system must be able to display the strength of the password.

Objective: Met ✓

Evidence: The validation of the strings are processed using regular expressions. The password goes through a list of statements, incrementing the strength counter for each requirement it matches. The strength of the password is then displayed back to the customer through a bar.

21. Sort Bookings

The system must be able to sort bookings depending on what each part of the system requires.

Objective: Met ✓

Evidence: Queries are used to sort the results prior to getting them. For example, the customer can sort bookings to upcoming, past and all, the staff will get bookings based on the current day and the admin can access bookings after choosing a date.

22. Hash Password

In order to protect the passwords of customers and administrators, the system must be able to hash passwords.

Objective: Met ✓

Evidence: Passwords are hashed using the encrypt() subroutine and their plaintext passwords are never kept in the system. Only the hashes are stored in the system.

```
function encrypt($value) {  
    $salt = '████████'; *  
    $salt = md5($salt);  
    $value = md5($value);  
    $value = hash('ripemd160', $salt . $value);  
    return $value;  
}
```

username	password
customer	e7f817c283a8d2f303599ba24095...
jonsno	9175dae505a6fbc31c8b7eaa92...
henryford	3287dft78sda79dyvasd70ya9df96...
clairesno	f97790f8g7sd89fg7s0uvjscadshfa...
tnguyen	83cd26aee7c1879262ea4d4ale7...

*The salt is censored for security purposes.

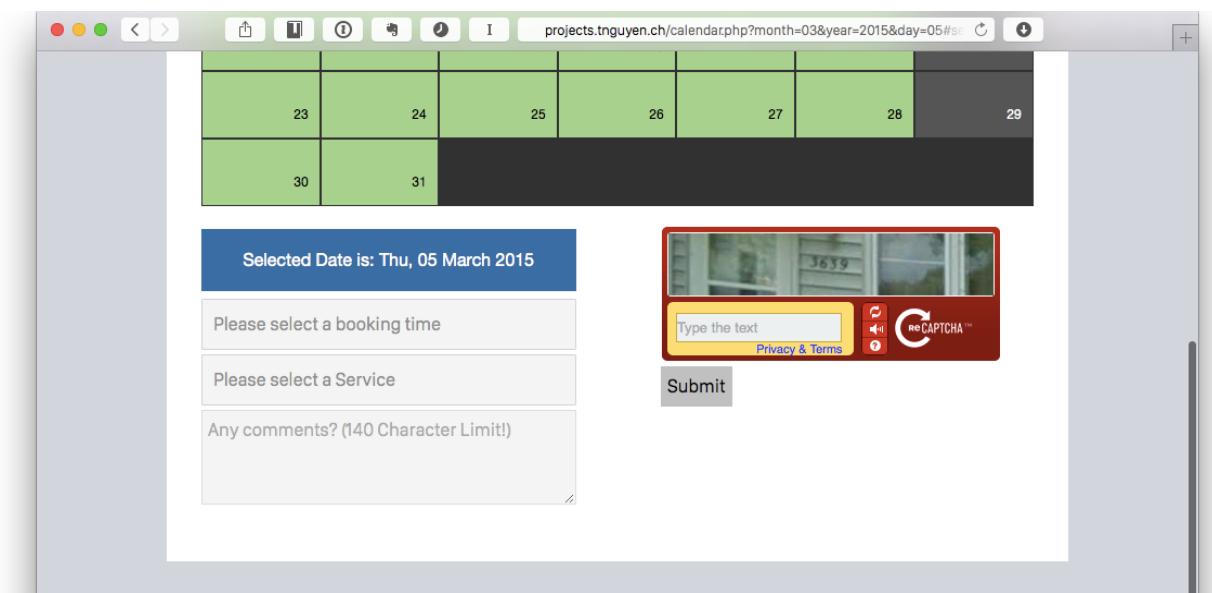
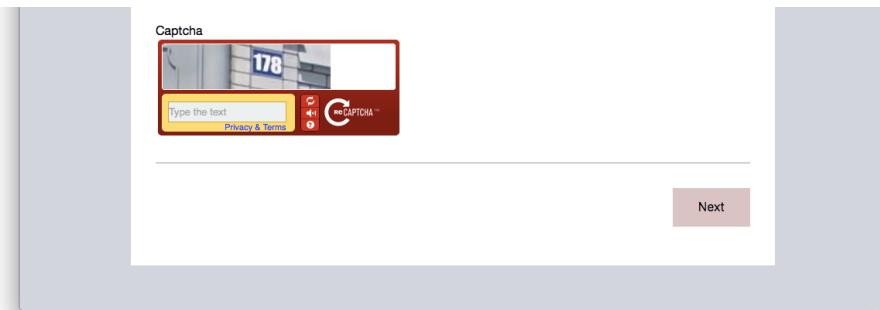
23. Ensure that the customer is a real person

Scheduling & Booking System

The system must be able to ensure that the customer is indeed a customer by using a CAPTCHA.

Objective: Met ✓

Evidence: A captcha is in place in the customers side of the system and it is used in registration and booking an appointment.



24. String Matching

The system must be able to validate data using pattern matching sequences.

Objective: Met ✓

Evidence: Similar to password match, strings are verified using regular expressions. Regular expressions are used throughout the system, and are used to verify inputs such as email addresses, names,

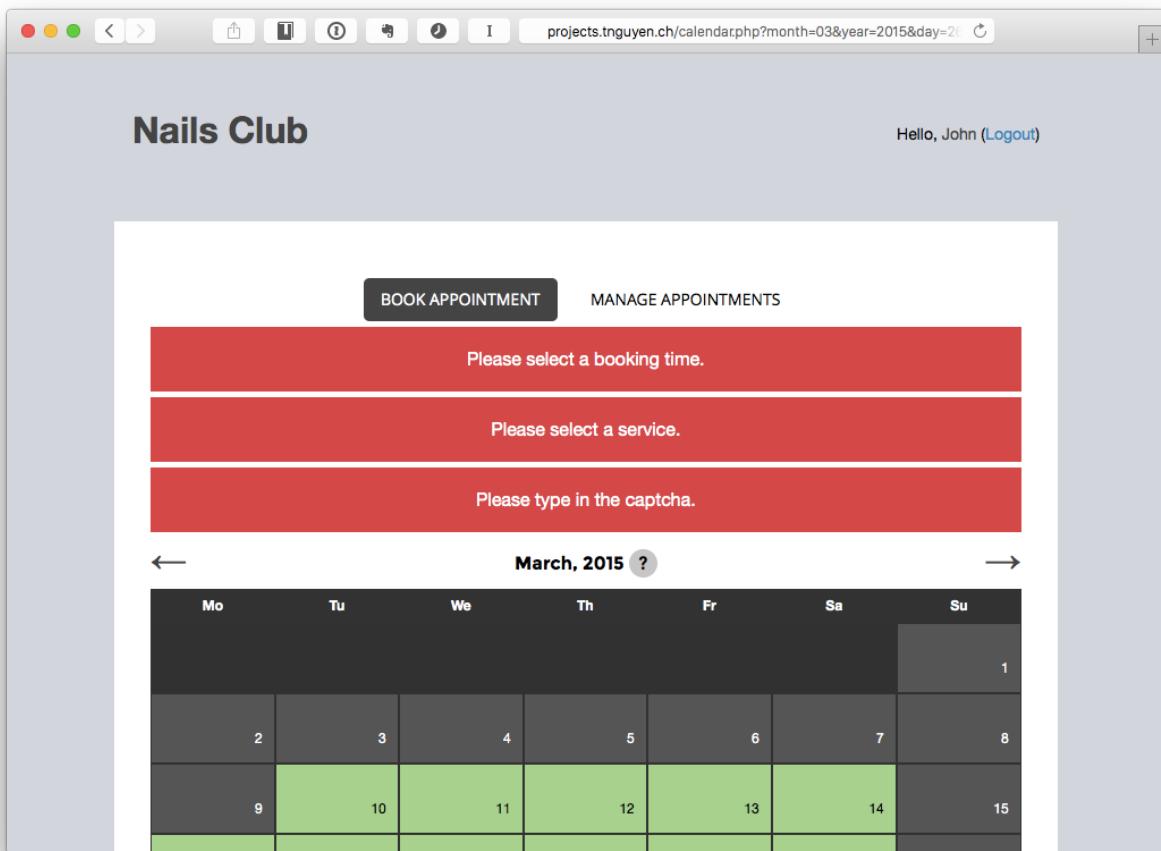
passwords and phone numbers.

25. Queue Error Messages

The system must be able to queue up all related error messages before displaying them in the order they were sent.

Objective: Met ✓

Evidence: Error messages are queued in the event that something wrong occurred.



26. Error Labels

In the event that the user misses out on something, they are notified with a friendly reminder to finish the inputs before continuing.

Objective: Met ✓

Evidence:

The screenshot shows a web browser window with a registration form. The URL in the address bar is `projects.tnguyen.ch/register.php`. The form has several input fields with associated error messages:

- Forename:** The input field contains "First" and has a red error message below it: "Please enter your forename."
- Surname:** The input field contains "Last" and has a red error message below it: "Please enter your surname."
- Username:** The input field contains "Username" and has a red error message below it: "Please enter a username."
- Password:** The input field contains "Password" and has a red error message below it: "Please provide a password."
- Confirm Password:** This field is partially visible at the bottom left.
- Password Strength:** To the right of the password field, there is a grey bar labeled "No Data".

27. Email

The system will automatically notify users for the following actions:

- *Booking an Appointment*
- *Finalising their Registration*
- *Resetting their Password*
- *Informing them of a new Password*
- *Informing them of a new PIN (Staff)*

Objective: Met ✓

Evidence:

A screenshot of a Gmail inbox. The message subject is "Your Appointment" and it is from "robot@tnguyen.ch". The message body says: "Hi Jon, You have an appointment at 2015-03-05, 10:00:00 for a Manicure. Nails Club 128 Barking Road London E16 1EN". Below the message is a reply/forward button.

A screenshot of an Apple Mail inbox titled "Inbox (243 messages)". It shows several messages from "robot@tnguyen.ch":

- 11:39 AM Time to confirm your email. To: Thien Nguyen
- 11:26 AM You need to activate your account. Click the link below. <http://projects.tnguyen.ch/confirmation.php?type=registration&value=1243da3c80ee9c3f4292ed6fd1ded8efbd80273>
- 10:50 AM Your New Password. To: John Rubinstein
- 10:49 AM Forgotten Password. To: robot@tnguyen.ch
- 10:02 AM eBay

28. Mathematical Calculations

The system should be able to use mathematical calculations to enhance the usability of the system. This will include the following:

- Check the booking thresholds
- Calculate the Booking Slots

Objective: Met ✓

Evidence:

Calculating Booking Slots

```
for ($i = strtotime($this->booking_start_time); $i <= strtotime($this->booking_end_time); $i =
    $i + $this->booking_frequency * 60) {
    $booking_times[] = date("H:i:s", $i);
}

for slot = first_slot, slot <= last_slot, slot = slot + booking_frequency * 60
    booking_time[] = slot
end
```

The above code is an extract from the booker class. In this extract, a loop is made starting at the first slot (`$this->booking_start_time`) and increments with every booking frequency. With each increment a slot is included into the list of booking times. This is repeated until the increment matches the last slot (`$this->booking_end_time`).

Calculating Booking Threshold

```
$day_capacity = ((minutes($this->booking_end_time) + 30) - minutes($this->booking_start_time)) /
    $this->booking_frequency;
```

The above code is an extract from `create_days_table()` subroutine from the booker class. In this extract, the capacity of the day, or the number of slots in a day is calculated using mathematical knowledge. The first slot and last slot (`$this->booking_start_time` and `$this->booking_end_time` respectively) are converted into minutes before taking away from each other. This is then divided by the booking frequency, which gives us the number of slots available.

29. Recursive PIN Generator

The system must be able to generate a pin that recurs in the event that the pin generated is already used.

Objective: Met ✓

Evidence:

A screenshot of a web application titled "Nails Club". At the top, there are buttons for "Update", "Ban", and "Generate New PIN". The "Generate New PIN" button is highlighted in blue. Below the buttons, there is some code output:

```

Array
(
    [0] =>
    [1] => d67595207b97a3397056e5ba1ff94da4cad4bcff
    [2] => 8ecaba3f9c178fba179aebe6ca914d17845c41f
    [3] => aecb7814791ba4b94b6efc11cada893ae8ff7c5
)
new pin: 12345
new pin hash: ff2d9937016d4d405364ce137b12f18be54e3e6d
the following data will be sent to the account:
PIN: 12345
PIN Hash: ff2d9937016d4d405364ce137b12f18be54e3e6d

```

Below the code, a yellow banner displays the message: "A new pin has been set to Joseph." The navigation bar at the bottom includes links for DASHBOARD, APPOINTMENTS, CALENDAR, USERS (which is selected), and SETTINGS. A staff member's name, "Staff", is visible on the left.

Annotations on the left side of the screenshot state: "Generate New PIN is clicked, initiating the function."

A new pin (12345) is hashed and checked with the database to see if this hash has been already been used.

If theres no match then the hash is then sent to the database. In this case the 12345 hash is already in the database, so it isn't sent.

Instead, the while loop is initiated and a new pin and its hashed is generated.

This new hash is then checked with the database, if it isn't in the database it is then sent.

A screenshot of the "Nails Club" application showing the recursive generation of a new PIN. The terminal output shows:

```

Array
(
    [0] =>
    [1] => ff2d9937016d4d405364ce137b12f18be54e3e6d
    [2] => 8ecaba3f9c178fba179aebe6ca914d17845c41f
    [3] => aecb7814791ba4b94b6efc11cada893ae8ff7c5
)
new pin: 12345
new pin hash: ff2d9937016d4d405364ce137b12f18be54e3e6d
current pin is in use, trying again with newly generated pin.
new pin: 94703
new pin hash: 5ed7a275feee142ea9b5db475665f5188be8f80d
assigning new pin hash to check in array. if fails loop will iterate

the following data will be sent to the account:
PIN: 94703
PIN Hash: 5ed7a275feee142ea9b5db475665f5188be8f80d

```

Annotations on the left side of the screenshot state: "A new pin (12345) is hashed and checked with the database to see if this hash has been already been used."

A new pin (12345) is hashed and checked with the database to see if this hash has been already been used.

If theres no match then the hash is then sent to the database. In this case the 12345 hash isn't in the database, so it is sent.

Hello, Administrator

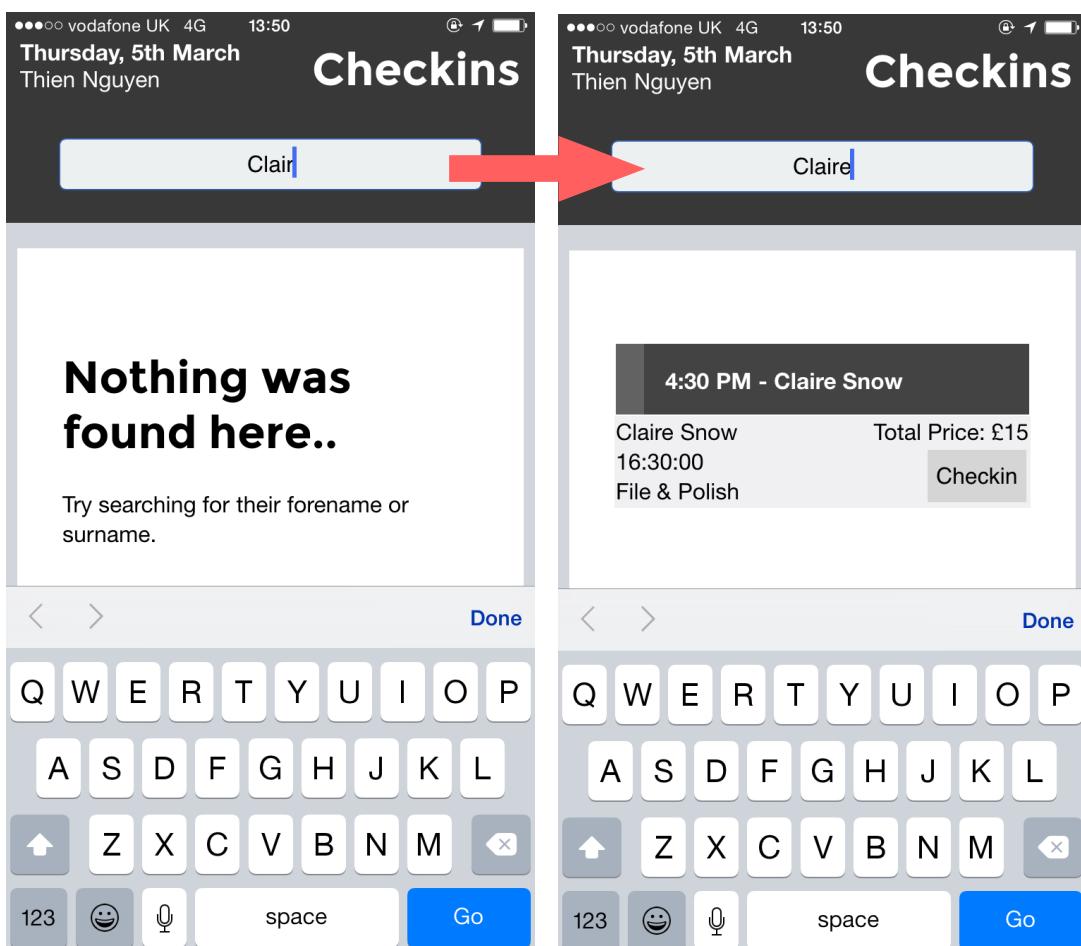
A new pin has been set to Joseph.

30. Staff Search

The staff must be able to search for customers by their surname or forename.

Objective: Met ✓

Evidence:



Storage

31. Store Customer Details

The database must be capable of storing customer credentials.

Objective: Met ✓

Evidence: Customer credentials are stored in the database under the users table. Each row in the table contains details about a specific user. This row contains their username, forename and surname,

password hash, email, phone number and whether they have validated their registration.

32. Load Customer Details

The system must be able to load customer details.

Objective: Met ✓

Evidence: Customer details are available to load from the database. All that is required is their ID, and the system can pick up all related data out of that ID.

33. Store Staff Details

The system must be capable of storing staff details, such as their forename and surname.

Objective: Met ✓

Evidence: Staff details are available to load from the database. All that is required is their ID, and the system can pick up all related data out of that ID.

34. Store Bookings

The database must be capable of storing bookings and their associated variables.

Objective: Met ✓

Evidence: Bookings are stored in the bookings table. Each row in the table contains details about a specific booking. This row contains the booking ID, Customers ID, Service ID, Date, Time, Comments, Verification from Staff, and Staff ID.

35. Store Services

The database must be capable of storing services, and their related information.

Objective: Met ✓

Evidence: Services are stored in the services table. Each row in the table contains details about a specific service. Each row contains a service ID, a service name, a description and a price.

Security

36. Encrypting Important Details

All passwords and Staff PIN's will be securely encrypted using hashes alongside modern cryptographic methods such as MD5 or SHA.

Objective: Met ✓

Evidence: in the subroutine encrypt() MD5 is used in the subroutine to encrypt the salt and the original input. RIPEMD160 is then used to encrypt the the salt and input hashes together, making the resulting hash more difficult to decrypt.

37. Password Strength

The system will attempt to insist users to use a strong password in order to improve the security of the system.

Objective: Met ✓

Evidence: While not necessarily a requirement of the system, Integrity of the security is shared with the users also. Using a password strength helps enhance the security of the system and deters hackers from attempting to use malicious methods to gain access to the system. This is done through the password strength meter, which helps users determine what kind of password is safe.

38. Timeouts

In the event that the user is inactive for a long period of time the system must be able to time out the user, making them log in again. For the staff, it is in 60 seconds, the administrator 10 minutes, and the customer 10 minutes.

Objective: Met ✓

Evidence: Timeouts are dealt with the cookie and are updated every time the user makes an action, such as loading a new page or querying the system.

39. Preventing Code Injection

The system must be able to sanitise inputs in order to prevent unauthorised or malicious methods to be used in effect against the system, such as SQL injections. This is done in order to maintain the integrity of the database.

Objective: Met ✓

Evidence: 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. Preventing SQL Injections is possible by preparing statements beforehand and by using parameterised queries. The system will use the latest version of PHP also.

40. Access Rights

The administrator must be able access:

- Viewing the Booking Statistics
- Managing the Appointments
- Managing the Calendar Settings
- Managing the Customers
- Managing the Staff

The staff must be able to access:

- Customer's Checkins

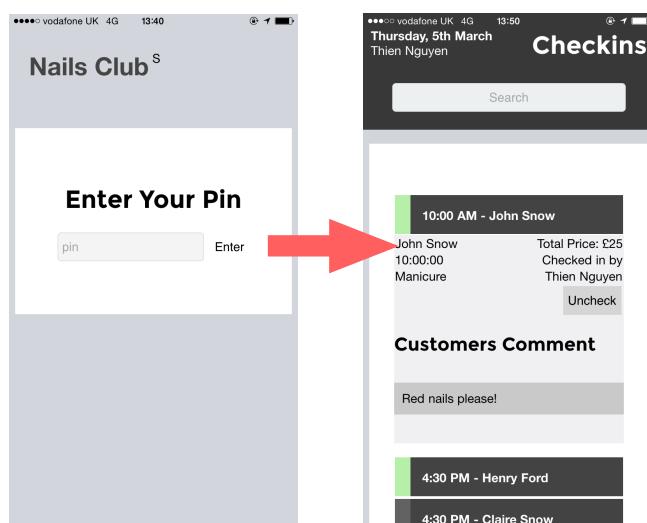
The customer must be able to access:

- Booking an Appointment
- Viewing their current appointments
- Managing their details

Objective: Met ✓

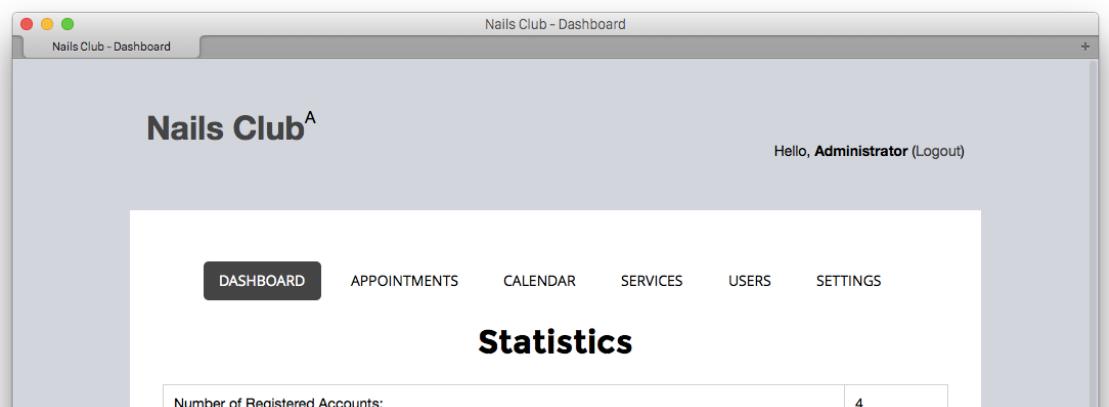
Evidence:

Users are separated by design. With the system, the customer's are unable to access the staff's side of the system or the administrator's system. Administrators likewise, are unable to access the staff's section or the customer's section but they can make accounts for themselves to get in the staff or customer section of the system. Staff are limited to their own part of the system.



Scheduling & Booking System

Staff can access the customers checkin's when they have successfully logged in with their PIN.



Administrators can access the booking statistics, appointments, calendar settings, customers and staff, alongside the system settings through the navigation bar.

The image contains four screenshots of the Nails Club application:

- Top Left:** "Nails Club - Profile" window showing "Your Information" with fields for Username (jorano), Name (Jon Snow), Email (jon.snow.com), and Phone No. (0747811111). An "Edit" button is highlighted with a red arrow.
- Top Right:** "Nails Club" window showing "Hello, Jon (Logout)" and three buttons: "Book an Appointment", "Manage Appointments", and "Past Appointments".
- Bottom Left:** "Nails Club - Book Appointment" window showing a calendar for January 2015. A red arrow points to the "Hello, Jon (Logout)" text at the top right of the calendar.
- Bottom Right:** "Nails Club" window showing "Hello, Jon (Logout)" and a list of "Past Appointments". One appointment is listed: "Mon, 29 Dec 2014 6:00 PM Pedicure" with note "I'd like it red please!" and ID "C29". Another appointment is listed: "Sat, 03 Jan 2015 11:30 AM I'm" with ID "C26". A red arrow points to the "Hello, Jon (Logout)" text at the top right of the appointment list.

Customers are able to access the calendar, where they can book an appointment by click on the left button on the dashboard. They can also view their current appointment by clicking on the right. Additionally, they can manage their details by clicking on their name on the top right.

6.2.1 General Objectives

- The new system should be an online booking system, which is accessible anywhere by anyone, as long as there is an Internet connection.**

Objective: Met ✓

The system is stored on a server which is accessible by all users who are able to access the internet.

- The system must be able to be accessed by not only the client, but their customers and staff.**

Objective: Met ✓

Customers, staff and the business owners have separate entities within the system they are able to log into.

- The new system must be better to use than the current system.**

Objective: Met ✓

The new system not only incorporates the features of the current system, but it automates the system and provides more features, such as automated email messages, making it more intuitive for the client to use.

- The system must be able to allow the customer to book an appointment.**

Objective: Met ✓

Customers can book an appointment after registering an account with the system.

- The system must be able to allow the staff to check in the customers when they arrive.**

Objective: Met ✓

The staff section of the system allows them to check in customers when they arrive on the day of their booking.

- The system must be user friendly.**

Objective: Met ✓

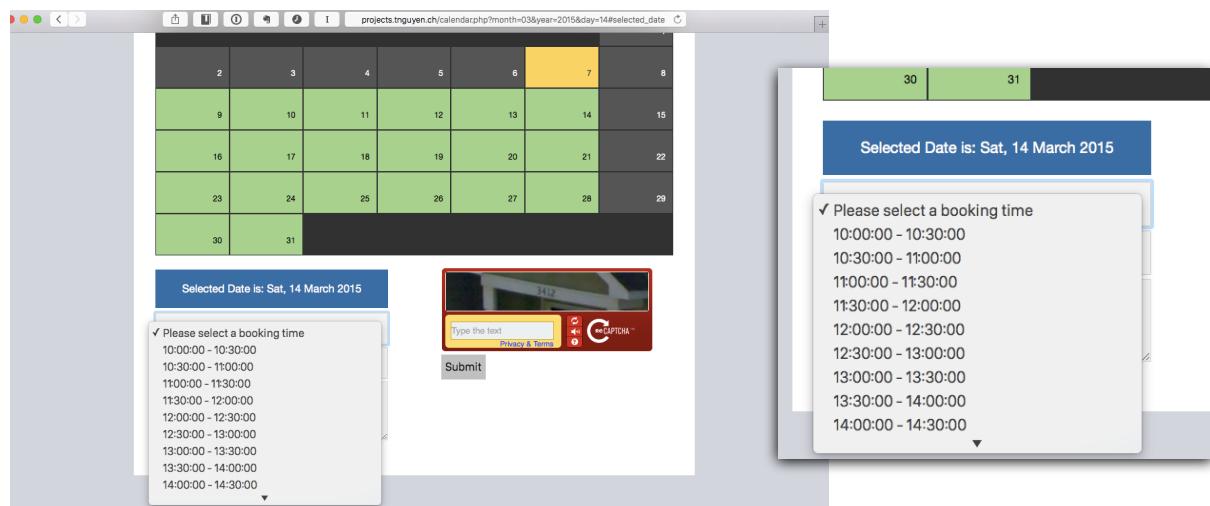
User friendliness is achieved using error messages and taking measures in the event that the user does not do something the system was intended for.

- The customer must be able to see what periods are available for them to book an appointment.**

Objective: Met ✓

Scheduling & Booking System

Customers can see the available slots while they are booking an appointment.

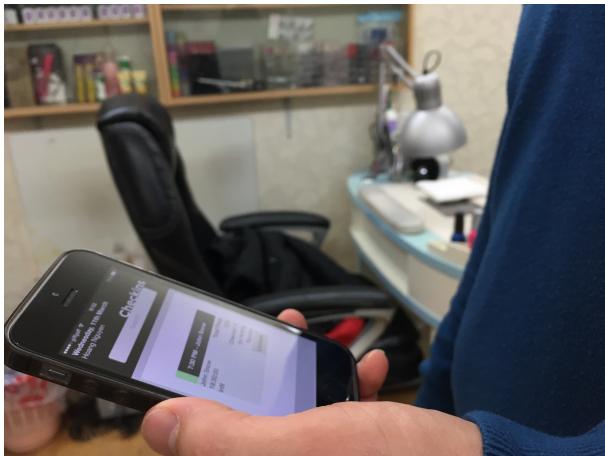


8. The system must be able to be accessed on a phone.

The user interface of the system is scaled dynamically in order to cater to mobile users in mind.

The three screenshots show the mobile application interface for Nails Club. The first screenshot shows the main landing page with "Nails Club" and "Hello, John (Logout)" at the top, and "BOOK APPOINTMENT" and "MANAGE APPOINTMENTS" buttons. Below is a calendar for March 2015 with dates 1 through 15 visible. The second screenshot shows a "Book an Appointment" screen with a large central area for appointment details and "Manage Appointments" at the bottom. The third screenshot shows a "Past Appointments" screen listing past appointments for "Mon, 29 Dec 2014" and "Sat, 03 Jan 2015", each with a note and price.

6.2 User Feedback by Assessor



With the program and user manual finished and ready, I showed the new system to my client, Hoang and Hanh, and demonstrated the system on how it works. I then asked Hoang to try the system out with customers and email me in a few days about how well the system went, whether the system met their objectives, how their customers and staff found it alongside how they found it, and their opinions on improvements.

Dear Thien

Thank you for finishing the booking system. We have started using the system since March 9 on Monday. This allowed us to test out the system and teach our staff how the new system would work.

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

Me and Hanh liked how our side of the system worked. Getting to different parts of our dashboard was very easy.

The staff found it very easy to use the system. The customer will need to get used to the new system as we have been using the appointment book for a long time. We did have some customers book with the new system so I am sure that this will be better over time.

- How does the system meet their objectives

Most of the objectives have been met. We found it easy to use and it clearly helped. The system works on phones and I can see charts for the bookings!

- How easy was the system to set up

The system is easy to set up and I found the user manual to be very useful.

- Criticisms of what they don't like

The layout is not best. I don't like the colour style. Where is the services description?

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

I would like an option to change the colour of the system for different times, like making it red for Easter and Christmas. I would also like the option to put discount codes if we wish to start making offers. Maybe this could be improved.

We are happy with your system. I can see that this can be put to good use with our business. Thank you for everything!

Sincerely

Hoang Nguyen

*The original email can be found in the appendix.

The feedback was very positive. Overall, My client agrees that I have met their objectives outlined in the analysis. The system is now ready to be fully deployed with Hoang and Hanh's Business.

6.3 Analysis of User Feedback

Feedback	Analysis
<i>Me and Hanh liked how our side of the system worked. Getting to different parts of our dashboard was very easy.</i>	The layout of the system was designed to be easy to use, as it adheres to the general conventions of websites. This way it makes it intuitive and the user can easily recognise what should be where.
<i>The staff found it very easy to use the system. The customer will need to get used to the new system as we have been using the appointment book for a long time. We did have some customers book with the new system so i am sure that this will be better over time.</i>	Limiting the number of things the staff can do benefits the staff as they are restricted in terms of possible problems that may arise. Although the system works, the current system has been in place for a long time. Customers who come on a regular may be put off due to the new system as they normally didn't have to deal with booking an appointment through other means. This may pose a problem in the short-run but terms of long-term this would be a better strategy for all in the event that the business gets busier.
<i>Most of the objectives has been met. We found it easy to use and it clearly helped. The system works on phones and I can see charts for the bookings!</i>	The client hasn't specified what objectives hasn't been met, so it's not certain to judge on how well the system has worked. Although there are praises with the new system, criticism is definitely useful in order to make a better system. The client praises the systems ability to be accurately portrayed on a mobile device, which was always considered throughout each part of the system. The use of graphs is easily comprehensible in terms of looking at statistics, but I feel that more can be done in terms of statistics.
<i>The system is easy to set up and I found the user manual to be very useful.</i>	The user manual is cohesive and is written specifically for the client. This way it makes it easier for the client to use, which is the case here. I think that it can be better done with a more thorough walkthrough of the system.
<i>I would like an option to change the colour of the system for different times, like making it red for Easter and Christmas.</i>	While the system is successful in functionality, the actual aesthetics of the system, whilst following the conventions of standard websites, seems to have fallen behind. This may be related to less focusing on the design rather than the technical features of the system. This feature however can be implemented in reasonable time. But in the current allocated time for the project, this was unreasonable.
<i>I would also like the option to put discount codes if we wish to start making offers. Maybe this could be improved.</i>	Discount codes are a considerably important part of a booking system and should have been implemented in the new system. However, due to time constraints it wouldn't have been fully fledged, hence the omission. In the future where more time can be invested into the system, This feature can be implemented.

6.4 Possible Extensions/Improvements

The feedback was very positive. Overall, My client agrees that I have met their objectives outlined in the analysis. The system is now ready to be fully deployed with Hoang and Hanh's Business.

While the new system has garnered positive feedback, Improvements are still useful. One of the big criticisms of the system i feel, is the absence of the discount codes. With the way the new system is set up for the time being, It is not made to process transactions, so processing discount codes wouldn't work as effectively. Working on this however, would take a significant amount of time and would require more time allocated in the time constraints.

One of the objectives were not met; the ability to show descriptions of the services they offer. There are several reasons behind this, one of which that the time constrained what could be put in the system, second of which is that the system merely contrives of serving a purpose to book an appointment. This system would be used in conjunction with their own website which would have additional information about their business. The client still wishes to have this information included with the new system, and preparations have been made in doing so, where the client can add a service description in their dashboard. So in the event that more time can be allocated, The inclusion of descriptions of services would be finalised.

It is clear that design updates may be useful for the system. The client has mentioned the fact that upgrades to the colour scheme should be considered. Of course, for the time being, the easiest change would be to modify the style sheet. This comes with its own difficulty as the client wouldn't know what to modify. The best option would be to create a theme modifier that allows the client to create templates or use pre-made colour templates for the system. This could be accessed in the administrators side of the system.

Next Objectives

- Implement New Discount Code System
- Create a Theme Changer
- Implement a method to display descriptions for the services
- Allow the Administrator to see an individual customer's bookings
- Implement a Online Payment System

Appendix

7.1 Interviews & Responses

7.1.1 First Interview Report

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 separately 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 efficient 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?

Scheduling & Booking System

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.

7.1.2 Second Interview

Second Interview

20th September 2014

Are there any specific requirements that you would like the system to include?

The users must be able to have their own account instead of letting anybody book an appointment. It should be implemented in a way that makes the booking long enough that it would deter any possible bots or time wasters but short enough that it doesn't give a potential customer a hard time in booking.

It is preferred if the system would be able to give me statistics on the bookings such as the amount of bookings in the week.

The system needs to be better than the current one definitely. The staff should be able to access customers bookings easily.

It is preferred it that the system would be able to be accessed on a phone, and that the system is easy to use.

Are there any other ideas that you wish to consider?

They would like the services to have descriptions so the customer can see what they're booking.

What about any graphical ideas? How would you like the design?

As long as the design is clean, everything else is open to adjustment. The fonts must be clear and modern.

Are there any other ideas that you wish to consider?

None.

7.1.3 Client's Feedback

Re: Comment on Booking System - Google Chrome
<https://buzz.woodhouse.ac.uk/owa/?ae=Item&a=Open&t=IPM.Note&id=RgAAAC9Ws%2b088NtQ7xq%2biLGVC36BwAa0uEvXkl8Sb>

Reply Reply All Forward ⌂ ⌃ ⌄ ⌅ ⌆ ⌇ ⌈ ⌉ ⌊ ⌋ ?

Re: Comment on Booking System

Hoang Nguyen [hv.nguyen@icloud.com]

To: Thien Nguyen 11 March 2015 20:51

Dear Thien

Thank you for finishing the booking system. We have started using the system since March 9 on Monday. This allowed us to test out the system and teach our staff how the new system would work.

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

Me and Hanh liked how our side of the system worked. Getting to different parts of our dashboard was very easy.

The staff found it very easy to use the system. The customer will need to get used to the new system as we have been using the appointment book for a long time. We did have some customers book with the new system so i am sure that this will be better over time.

- How does the system meet their objectives
Most of the objectives has been met. We found it easy to use and it clearly helped. The system works on phones and I can see charts for the bookings!

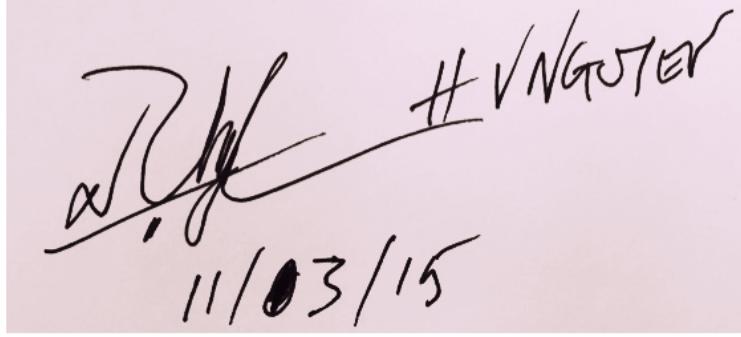
- How easy was the system to set up
The system is easy to set up and I found the user manual to be very useful.

- Criticisms of what they don't like
The layout is not best. I don't like the colour style. Where is the services description?

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

I would like an option to change the colour of the system for different times, like making it red for Easter and Christmas. I would also like the option to put discount codes if we wish to start making offers. Maybe this could be improved.

- Signature and date



We are happy with your system. I can see that this can be put to good use with our business. Thank you for everything!

Sincerely

Hoang Nguyen

7.2 Annotated Program Listings

7.2.1 Queries

7.2.1.1 Queries with Multiple Tables (Inner Joins)

```
$query = "SELECT booking.id, booking.date, users.forename,
users.surname, booking.time, booking.comments, booking.confirmedbystaff,
booking.staff_id, service.type, service.price, staff.s_forename, staff.s_surname
FROM booking INNER JOIN users ON booking.username = users.username
INNER JOIN staff ON booking.staff_id = staff.id
INNER JOIN service ON booking.service_id = service.id";
$order = "ORDER BY DATE(booking.date) DESC, booking.time DESC";
$count_rows = "SELECT count(*) FROM booking";
```

Inner Joins on Staff and Service

```
if (isset($_GET))
    if (isset($_GET['year']) & isset($_GET['month'])& isset($_GET['day'])) ){
        if (!empty($_GET['year']) & !empty($_GET['month'])& !empty($_GET['day'])) {
            if (checkdate($_GET['month'], $_GET['day'], $_GET['year']) == TRUE){
```

Original Queries (Seperated)

```
        $datequery = " WHERE date = '".$_GET['year']."' ."-".$_GET['month']."' ."-".$_GET['day']."' ";
        $count_rows = $count_rows.$datequery;
        $query = $query.$datequery.$order;
    } else {
        array_push($errors, "The selected date is invalid, please try again.");
        $query = $query . $order;
    }
} else {
    array_push($errors, "Please fill in all the criteria for the date.");
}
}

$db->DoQuery($count_rows);
$count = $db->fetch();
```

Extra Parameters affixed into query.

```
//Add following after it
$per_page =10;//define how many games for a page
$page = ceil($count[0]/$per_page);
```

```
if(!isset($_GET['page'])){
$page="1";
}else{
$page=$_GET['page'];
}
$start = ($page - 1) * $per_page;
$query = $query . " LIMIT $start, $per_page";
$db->DoQuery($query);
$num = $db->fetchAll();
```

Sending Query to Database

Above is an extract from the administrators appointments.php file. The final query is divided into several parts that are affixed procedurally as the code goes through from top to bottom. In the series of if statements the date query is managed whether or not the date variable is correctly chosen (from the customer). At the end the parts of the query are joined together.

Scheduling & Booking System

```
$query = "SELECT booking.id, booking.date, users.forename,
users.surname, booking.time, booking.comments, booking.confirmedbystaff,
booking.staff_id, service.type, service.price, staff.s_forename, staff.s_surname
FROM booking
INNER JOIN users ON booking.username = users.username
INNER JOIN staff ON booking.staff_id = staff.id
INNER JOIN service ON booking.service_id = service.id
WHERE booking.date = CURDATE()
ORDER BY booking.time ASC";
$db->DoQuery($query);
$num = $db->fetchAll();
```

Inner Joins on Staff, Users and Service and are affixed from foreign keys to primary keys.

Above is an extract from the staff's appointments.php file. A query is sent to the database, requiring results from the booking table. However, the booking table comprises of foreign keys from other tables. In this example, shown highlighted that tables that needs to be joined together (using MYSQL's INNER JOIN) include the users, staff and service tables.

```
<?php
$title = 'Manage Appointments';
$menutype = "user_dashboard";
$morethantoday = 1;
include_once("includes/core.php");
$date = date("Y-m-d");
$query = "SELECT booking.id, booking.date, booking.time, booking.comments,
service.type, service.price FROM booking
INNER JOIN service ON booking.service_id=service.id
WHERE booking.user_id = :user_id";
$query_params = array(
    ':user_id' => $_COOKIE['userdata']['user_id']
);
if (isset($_GET['option'])){
    $option = $_GET['option'];
} else {
    $option = 'upcoming';
}
$db->DoQuery($query, $query_params);
$results = $db->fetchAll();
if (isset($_POST['id_delete'])) {
    $query = "DELETE FROM booking WHERE id = :id";
    $query_params = array(':id' => $_POST['id_delete']);
    $db->DoQuery($query, $query_params);
    array_push($update, "Your appointment for is now cancelled.");
}
include 'includes/header.php';
?>
```

Above is an extract from the Customer's manage_appointments.php file. In this section, the database is queried from the booking table, with the service table being joined by service_id. This serves as the primary key for the latter, and a foreign key for the former.

7.2.2 Deleting

```

if (isset($_POST['new'])) {
    $query = "INSERT INTO closed_days (date) VALUES (:date)";
    $query_params = array(
        ':date' => $date
    );
    $db->DoQuery($query, $query_params);
    array_push($update, 'The closing day is added into the database.');
} elseif(isset($id)){
    $query = "DELETE FROM closed_days WHERE id = :id";
    $query_params = array(
        ':id' => $id
    );
    $db->DoQuery($query, $query_params);
    array_push($update, 'The closing day has been removed from the database.');
}

```

The above query, shown highlighted shows a request to delete from the “closed_days” table where the ID matches the variable that is being parsed through in the array \$query_params.

```

} elseif(isset($_POST['service_id_delete'])){
    $query = "DELETE FROM service WHERE id = '$id'";
    $db->DoQuery($query);
    array_push($update, 'The service has been deleted.');
}

```

The above query, shown highlighted shows a request to delete from the “service” table where the ID matches the variable that is being parsed within the query.

7.2.3 Selecting Customers (By Letter)

```

if(!isset($_GET['char'])){
    $char="A";
} else{
    $char=$_GET['char'];
}
$query = "SELECT * FROM users WHERE surname LIKE '".$char."%' ORDER BY surname ASC";
$count_rows = "SELECT count(*) FROM users WHERE surname LIKE '".$char."%'";
$db->DoQuery($count_rows);
$count = $db->fetch();

```

7.2.4 Selecting Staff (By Order of ID)

```

if ($usertype == 'staff'){
    $query = "SELECT * FROM staff ORDER BY id ASC";
    $db->DoQuery($query);
    $num = $db->fetchAll();
}

```

The above two select statements are issued on the users section of the administrators dashboard where the administrators are able to look through the customers and staff in their system. As the business

grows more customers would be registered in their database, meaning there must be an easier way to sort the customers.

7.2.5 Updating

```

if ($forename != $prevalue['forename']){
    $txt = "UPDATE users SET forename = '$forename' WHERE id = '$user_id'";
    $db->DoQuery($txt);
}
if ($surname != $prevalue['surname']){
    $txt = "UPDATE users SET surname = '$surname' WHERE id = '$user_id'";
    $db->DoQuery($txt);
}
if ($password !== NULL AND $password !== $prevalue['password']){
    $txt = "UPDATE users SET password = '$password' WHERE id = '$user_id'";
    $db->DoQuery($txt);
}
if ($email != $prevalue['email']){
    $txt = "UPDATE users SET email = '$email' WHERE id = '$user_id'";
    $db->DoQuery($txt);
}
if ($phoneno != $prevalue['phoneno']){
    $txt = "UPDATE users SET phoneno = '$phoneno' WHERE id = '$user_id'";
    $db->DoQuery($txt);
}

```

Updating users information were queried one at a time. This way, it ensures that blank results are not included in the database.

7.2.6 Inserting

```

$query = "INSERT INTO users (username, password, forename, surname, email, phoneno, activated,
activation_code) VALUES (:username, :password, :forename, :surname, :email, :phoneno, :activated, :activation_code)";
$query_params = array(
    ':username' => $username,
    ':password' => $password,
    ':forename' => $forename,
    ':surname' => $surname,
    ':email' => $email,
    ':phoneno' => $phoneno,
    ':activated' => '0',
    ':activation_code' => encrypt($username)
);
$db->DoQuery($query, $query_params);
header("Location: confirmation.php?type=registration");
}

```

Customer details are inserted into the database using a query conjoined with a parameter that contains the customers details. An activation code is sent alongside the data, which will be used to activate their account.

```

$query = "INSERT INTO staff (s_forename, s_surname, s_email, pin) VALUES
(:forename, :surname, :email, :pin)";
$query_params = array(
    ':forename' => $forename,
    ':surname' => $surname,
    ':email' => $email,
    ':pin' => $pin
);

```

```

':email' => $email,
':pin' => $newpins['pin_hash']
);
$db->DoQuery($query, $query_params);
array_push($update, "$forename has been included into the database. A PIN is sent to his email
address at $email.");
}

```

The above statement is related to making new staff in administrators users settings. When the query is finished, a message would be shown to the administrator confirming of the new staff being registered in the system.

7.2.2 User Defined Classes/Objects

Database

```

<?php
class database {
public function initiate() {
    include($_SERVER['DOCUMENT_ROOT'].'/includes/db.php');
    try {
        $this->database = new PDO("mysql:host={$hostname};dbname={$database_name}", $username,
$password);
    } catch(PDOException $e) {
        die($e->getMessage());
    }
    $this->database->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
}

public function DoQuery($query, $arguments = array()) {
    try {
        $this->result = $this->database->prepare($query);
        $this->result->execute($arguments);
    } catch(PDOException $e) {
        die($e->getMessage());
    }
}

public function fetch() {
    return $this->result->fetch();
}

public function fetchAll() {
    return $this->result->fetchAll();
}

public function RowCount() {
    return $this->result->RowCount();
}
?>

```

Variables are taken from db.php

Initiates connection to database.

Runs queries into database, arrays are used in the event that parameters have to be used. Prepare is used to sanitise query again.

Fetches results similar to how PDO handles fetching results.

I have chosen to make the database a class of its own to simplify the process of extracting data through MySQL. The implementation is heavily reliant on the database so simplifying the process essentially saves time and headache at the end.

Using Database Object at Runtime

```
include($directory . "classes/database.php");
$db = new database;
$db->initiate();
```

Above is an extract from core.php. This part of the code initiates a connection to the database, ready to be used by queries on other pages. Core.php is included on every code file.

```
if (isset($_POST['username']) && (isset($_POST['password']))) {
    $username = trim($_POST['username']);
    $password = encrypt(trim($_POST['password']));
    $query = "SELECT * FROM admin WHERE username = :username AND password = :password";
    $query_parameters = array(
        ':username' => $username,
        ':password' => $password
    );
    $db->DoQuery($query, $query_parameters);
    $num = $db->fetchAll();
    if ($num) {
        // user entered correct details
        setcookie('admin[loggedin]', TRUE, $admin_expiry, '', '', '', TRUE);
        header('Location: index.php');
        exit();
    } else {
        // user entered incorrect details
        array_push($errors, 'The username and password combination is not recognised. Please try again.');
    }
}
```

The code above is an extract from the administrators login.php file. At this stage inputs are collected with PHP's \$_POST variables from a form that includes a text-box for the username and password. \$query, \$query_parameters, \$db and \$num are the most important part of this code as it is used in relation to the database. \$query and \$query_parameters are the queries that would be sent to the database. This is done so using the predetermined database class mentioned earlier. The results of the query is then stored into \$num.

Booker

```
<?php
class booker {
    public $day, $month, $year, $selected_date, $first_day, $back, $forward;
    function database() {
        $this->db = new database();
        $this->db->initiate();
    }
    function QuickFetch($query) {
        $this->db->DoQuery($query);
        $fetch_value = $this->db->fetch();
        return $fetch_value[0];
    }
    function minutes($time) {
        $time = explode(':', $time);
        return (($time[0] * 3600) + ($time[1] * 60)) / 60;
    }
    function make_calendar($selected_date, $first_day, $back, $forward, $day, $month, $year) {
        $this->database();
        $this->day = $day;
        $this->month = $month;
        $this->year = $year;
        $this->selected_date = $selected_date;
        $this->first_day = $first_day;
        $this->back = $back;
        $this->forward = $forward;
        $this->booking_start_time = $this->QuickFetch("SELECT value FROM metadata WHERE id = '1'");
        $this->booking_end_time = $this->QuickFetch("SELECT value FROM metadata WHERE id = '2'");
        $this->booking_frequency = $this->QuickFetch("SELECT value FROM metadata WHERE id = '3'");
        $this->start_booking($year, $month);
    }
    function post($month, $day, $year) {
        //error lists
        $errors = array();
        include('assets/recaptcha_values.php');
        include_once('assets/recaptcha.php');

        if (isset($_POST['booking_time']) && $_POST['booking_time'] == 'selectvalue') {
            array_push($errors, "Please select a booking time.");
        }
        if (strlen($_POST['comments']) >= 140) { // In the event the html limit is bypassed
            array_push($errors, "You have exceeded the character limit for the comments. Please shorten it!");
        }
        if (isset($_POST['booking_service']) && $_POST['booking_service'] == 'selectvalue') {
            array_push($errors, "Please select a service.");
        }
        if (strlen($_POST["recaptcha_response_field"]) == 0) {
            array_push($errors, "Please type in the captcha.");
        }
        if ($_POST["recaptcha_response_field"]) {
            $resp = recaptcha_check_answer($privatekey, $_SERVER["REMOTE_ADDR"],
                $_POST["recaptcha_challenge_field"], $_POST["recaptcha_response_field"]);
            if (!$resp->is_valid) {
                array_push($errors, "The captcha is incorrect. Please try again!");
            }
        }
        if (!empty($errors)) {
            display_errors($errors); //no errors, continue with saving into the database.
        } else {
            $this->database(); //declared again as post will go to this function instead of loading the calendar.
        }
    }
}
```

Database class has to be initiated in the booker class as this is run without the system core

Quick Query is used in this class only to improve the writing of the code.

On run, variables are initiated before running the first function. This is called on calendar.php

start_booking() is called at the end of initiating variables

When the customer has finished their booking and click next, this function is then executed

```

$booking_date    = date("Y-m-d", mktime(0, 0, 0, $month, $day, $year));
$booking_time   = $_POST['booking_time'];
$booking_service = $_POST['booking_service'];
$query          = "INSERT INTO booking (date, time, user_id, comments,
confirmedbystaff, service_id, staff_id) VALUES
(:booking_date, :booking_time, :user_id, :comments, :service_id, :staff_id)";
$query_params    = array(
    ':booking_date' => $booking_date,
    ':booking_time' => $booking_time,
    ':service_id' => $booking_service,
    ':user_id' => $_COOKIE['userdata']['user_id'],
    ':comments' => $_POST['comments'],
    ':confirmed' => 0,
    ':staff_id' => 1
);
$this->db->DoQuery($query, $query_params);
$this->confirmation($booking_date, $booking_time, $booking_service, $_COOKIE['userdata']
['user_id']);
}
}

function confirmation($date, $time, $serviceid, $user_id) {
    include('functions/email.php');
    $query = "SELECT type FROM service WHERE id = '$serviceid'";
    $this->db->DoQuery($query);
    $service = $this->db->fetch();
    $query = "SELECT email FROM users WHERE id = '$user_id'";
    $this->db->DoQuery($query);
    $email = $this->db->fetch();
    $extra = array(
        ':bookingday' => $date,
        ':bookingtime' => $time,
        ':bookingservice' => $service[0]
    );
    email($email['email'], $user_id, $_COOKIE['userdata']['forename'], "appointment", $extra);
    echo "<meta http-equiv='refresh' content='0; url=confirmation.php?type=appointment'>";
}

function start_booking($year, $month) {
    $period = $year . '-' . $month . '%';
    $query = "SELECT * FROM booking WHERE date LIKE '$period'";
    $this->db->DoQuery($query);
    $this->count = $this->db->RowCount();
    while ($rows = $this->db->fetch()) {
        $this->prior_bookings[] = array(
            "date" => $rows['date'],
            "start" => $rows['time']
        );
    }
    $this->create_days_arr($year, $month);
}

function create_days_arr($year, $month) {
    $num_days_month = cal_days_in_month(CAL_GREGORIAN, $month, $year);
    for ($i = 1; $i <= $num_days_month; $i++) {
        $d = mktime(0, 0, 0, $month, $i, $year);
        $this->days[] = array(
            "daynumber" => $i,
            "dayname" => date("l", $d)
        );
    }
    for ($j = 1; $j <= $this->first_day; $j++) { // Add blank elements @ start if first day of
the month != monday
        array_unshift($this->days, '0');
    }
    $padding_end = 7 - (count($this->days) % 7); // add blanks if required
    if ($padding_end < 7) {
        for ($j = 1; $j <= $padding_end; $j++) {
            array_push($this->days, NULL);
        }
    }
}

```

initiates variables to be used on email function

send email to customer

```

        }
    }
    $this->start_calendar();
}
function create_days_table() {
    $day_capacity = ((minutes($this->booking_end_time) + 30) - minutes($this->booking_start_time)) / $this->booking_frequency;
    $fdm = $this->year . "-" . $this->month . "-" . "01"; //first day of month
    $closed_days_query = "SELECT date FROM closed_days WHERE YEAR(date) = YEAR('$fdm') AND MONTH(date) = MONTH('$fdm')";
    $this->db->DoQuery($closed_days_query);
    $closed_days = $this->db->fetchAll();
    $i = 0;
    foreach ($this->days as $row) {
        $box = '';
        if ($i % 7 == 0) {
            echo "</tr><tr>";
        }
        if (isset($row['daynumber']) && $row['daynumber'] != 0) {
            echo "<td class='days'>";
            if ($this->count > 0) {
                $bookings_on_day = 0;
                foreach ($this->prior_bookings as $booking_date) {
                    if ($booking_date['date'] == $this->year . '-' . $this->month . '-' . sprintf("%02s", $row['daynumber'])) {
                        $bookings_on_day++;
                    }
                }
            }
            $this_day = $this->year . "-" . $this->month . "-" . $row['daynumber'];
            foreach ($closed_days as $item) { // It's a closed day, set from the database.
                if (strtotime($item['date']) == strtotime($this_day)) {
                    $box = 2;
                }
            }
            if (mktime(0, 0, 0, $this->month, sprintf("%02s", $row['daynumber']) + 1, $this->year) < strtotime("now")) {
                $box = 2; // Past Day / Unavailable
            }
            if ($row['dayname'] == 'Sunday') {
                $box = 2; // It's a Sunday
            }
            if ($bookings_on_day >= $day_capacity && $box == '') {
                $box = 3; // Fully Booked
            }
            if ($bookings_on_day > 0 && $box == '') {
                $box = 1; // Part booked day.
            }
            echo $this->day_switch($box, $row['daynumber']) . "</td>";
        } else {
            echo "<td class='days'><div class='box' id='key_null'></div></td>";
        }
        $i++;
    }
    echo "</tr></table>";
    $this->make_booking_slots();
}
function day_switch($box, $daynumber) {
    $date_number = "<p>" . str_replace(NULL, ' ', $daynumber) . "</p>";
    $link = "<a href='calendar.php?month=" . $this->month . "&year=" . $this->year . "&day=" . sprintf("%02s", $daynumber) . "#selected_date' . '>'";
    switch ($box) {
        case (1):
            $text = $link . "<div class='box' id='key_partbooked'>" . $date_number . "</div></a>";
            break;
    }
}

```

modulus is used to give table row

```

        case (3):
            $text = "<div class='box' id='key_fullybooked'>" . $date_number . "</div>";
            break;
        case (2):
            $text = "<div class='box' id='key_unavailable'>" . $date_number . "</div>";
            break;
        default:
            $text = $link . "<div class='box' id='key_available'>" . $date_number . "</div></
a>";
            break;
    }
    return $text;
}

function make_booking_slots() {
    if ($this->day == 0) {
        echo "<form id='calendar_form' method='post' action=''>";
        echo "<div class='status' id='selected_date'>Please select a day.</div>";
    } else {
        $this->create_form();
    }
}

function start_calendar() {
    echo "<table cellpadding='0' id='calendar'>
        <div id='week'>";
    echo "<div class='buttons' align='left'>";
    if ($this->selected_date > strtotime('-6 months')) {
        echo "<a href='?month=" . date("m", $this->back) . "&year=" . date("Y", $this->back)
. "'>";
    } else {
        echo "<a class='invisible'>";
    };
    echo "&#8592;</a></div>";
    echo "<div id='center_date'>" . date("F, Y", $this->selected_date) . "<b class='keyguide'>
        <i>?</i>
        <ul class='keylist'>
            <h3>Key</h3>
            <li id='key_fullybooked'>Fully Booked</li>
            <li id='key_partbooked'>Free (Partially)</li>
            <li id='key_available'>Free</li>
            <li id='key_unavailable'>Not Available</li>
        </ul>
    </div></b>";
    echo "<div class='buttons' align='right'>";
    if ($this->selected_date < strtotime('+6 months')) {
        echo "<a href='?month=" . date("m", $this->forward) . "&year=" . date("Y", $this-
>forward) . "'>";
    } else {
        echo "<a class='invisible'>";
    };
    echo "&#8594;</a></div>";
    echo "</div>
<tr>";
    $days = array("Mo", "Tu", "We", "Th", "Fr", "Sa", "Su");
    for ($i = 0; $i < 7; $i++) {
        echo '<th>' . $days[$i] . '</th>';
    }
    if (empty($this->prior_bookings)) {
        $this->create_days_table($this->days, $this->month, $this->year);
    } else {
        $this->create_days_table($this->days, $this->prior_bookings, $this->month, $this->year);
    }
}
function create_form() {
    $this->db->DoQuery("SELECT * FROM service");
    $services = $this->db->fetchAll();
}

```

date is not chosen if day = 0

```

        for ($i = strtotime($this->booking_start_time); $i <= strtotime($this->booking_end_time); $i
= $i + $this->booking_frequency * 60) {
            $booking_times[] = date("H:i:s", $i);
        }
        echo "<form id='calendar_form' method='post' action='>";
        echo "<div id='left'>";
        echo "<div class='status' id='selected_date'>Selected Date is: " . date("D, d F Y",
mktime(0, 0, 0, $this->month, $this->day, $this->year)) . "</div>";
        $option = "<select id='select' name='booking_time'><option value='selectvalue'>Please select
a booking time</option>";
        if ($this->count >= 1) {
            foreach ($this->prior_bookings as $row) {
                foreach ($booking_times as $i => $r) {
                    if ($row['start'] == $r && $row['date'] == $this->year . '-' . $this->month .
'-' . $this->day) {
                        unset($booking_times[$i]); ← Checks for bookings and removes any previously
removed slots
                    }
                }
            }
        }
        foreach ($booking_times as $booking_time) {
            $finish_time = strtotime($booking_time) + $this->booking_frequency * 60;
$option .= "<option value='" . $booking_time . "'>" . $booking_time . " - " . date("H:i:s",
$finish_time) . "</option>";
        }
        echo $option . "</select>";
        echo "<select id='select' name='booking_service'>";
        echo "<option value='selectvalue'>Please select a Service</option>";
        foreach ($services as $row) {
            $text = $row[1] . " - &pound;" . $row[2];
            echo "<option value='" . $row[0] . "'>" . $text . "</option>";
        }
        echo '</select>';
        echo "<textarea rows='3' cols='30' maxlength='140' name='comments' placeholder='Any
comments? (140 Character Limit!)></textarea>";
        echo "</div><div id='right'>";
        include('assets/recaptcha_values.php');
        include_once('assets/recaptcha.php');
        echo recaptcha_get_html($publickey, $error);
        echo "<button type='submit'>Submit</button></div></form>";
    }
}
?>

```

Creation of Calendar at Runtime

```
<?php
$calendar = new booker($link);

if (isset($_GET['month'])) { $month = $_GET['month']; }
else { $month = date("m");}
if (isset($_GET['year'])) { $year = $_GET['year']; } else {
$year = date("Y"); }
if (isset($_GET['day'])) { $day = $_GET['day']; } else { $day = 0; }
$selected_date = $year."-".$month."-".$day;
$selected_date_timestamp = mktime(0, 0, 0, $month, 01, $year);
$first_day = date("N", $selected_date_timestamp) - 1;
$back = strtotime("-1 month", $selected_date_timestamp);
$forward = strtotime("+1 month", $selected_date_timestamp);
..

if($_SERVER['REQUEST_METHOD'] == 'POST') {
    $calendar->post($month, $day, $year);
}

if (empty($errors)) {
$calendar->make_calendar($selected_date_timestamp,
$first_day, $back, $forward, $day, $month, $year);
} else {
    echo "<h1>This date is unavailable.</h1>";
    foreach ($errors as $error) {
        echo $error . "<br>";
```

Collecting variables for the calendar.

if the customer has filled an appointment, brings date variables to post function; other variables are stored within post anyway.

Load calendar from calendar class. Variables are inserted as parameters.

7.2.3 Communication Protocols

Email

All emails will have this header.

```
function email($email, $user_id, $forename, $type, $extra = array()) {
    $headers = "From: robot@tnguyen.ch";
    if ($type == "confirm_registration") {
        if(isset($extra[":pin"])) {
            $pin = $extra[":pin"];
            $subject = 'Time to confirm your email';
            $message = "Hi $forename, \n\n You need to activate your account. Click the link below.\n";
            $message .= "http://projects.tnguyen.ch/comp4/confirmation.php?type=registration&value=";
            $pin";
        }
        else {
            echo "No pin is inserted.";
        }
    }
    if ($type == "forgotten_password") {
        if(isset($extra[":code"])) {
            $code = $extra[":code"];
            $subject = 'Forgotten Password request?';
            $message = "Hi $forename, \n\n You have requested to reset your password. Click the
link below. \n";
            $message .= "http://projects.tnguyen.ch/comp4/confirmation.php?type=forgot&value=";
            $code";
        }
    }
    if ($type == "password_reset_confirmed") {
        if(isset($extra[":newpassword"])) {
            $newpassword = $extra[":newpassword"];
            $subject = 'Your New Password.';
```

Scheduling & Booking System

```
$message = "Hi $forename, \n\n Your new password is:  
\n\n";  
$message .= "$newpassword";  
}  
}  
if ($type == "booking") {  
    $subject = 'Time to confirm your booking';  
    $message = "Hi $forename, \n\n You need to activate your  
account. Click the link below. \n\n";  
    "  
}  
if ($type == "appointment") {  
    if(isset($extra[:bookingday]) & isset($extra[:bookingtime"]) &  
isset($extra[:bookingservice"])) {  
        $bookingdate = $extra[:bookingday];  
        $bookingtime = $extra[:bookingtime"];  
        $bookingservice = $extra[:bookingservice];  
        $subject = 'Your Appointment';  
        $message = "Hi $forename, \n\n You have an appointment at $bookingdate, $bookingtime  
for a $bookingservice.";  
    }  
}  
if ($type == "new_pin") {  
    if(isset($extra[:pin])) {  
        $pin = $extra[:pin];  
        $subject = 'Your New Pin';  
        $message = "Hi $forename,  
                    Your new pin is $pin.";  
    }  
}  
if ($type == "new_staff") {  
    if(isset($extra[:pin])) {  
        $pin = $extra[:pin];  
        $subject = 'Welcome!';  
        $message = "Hi $forename, \n\n Welcome to our company. Your new pin is $pin.";  
    }  
}  
  
// Send  
$message .= "\n\n\n";  
$message .= "Nails Club\n";  
$message .= "128 Barking Road\n";  
$message .= "London\n";  
$message .= "E16 1EN\n";  
mail($email, $subject, $message, $headers);  
}  
?>
```

email function being used to send a hash to the customer's email, in order to activate the customers account.

all messages will have this affixed at the end of the message.

Using PHP's email function (only works provided that the program is set up with a domain with a working SMTP server attached to it. This is not a issue for this project.

Email In Action

```
// If no errors were found, proceed with storing the user input  
if (count($errors) == 0) {  
    $password = encrypt($password);  
  
    $email_parameters = array(  
        ':pin' => encrypt($username)  
    );  
    email($email, $username, $forename, "confirm_registration", $email_parameters);
```

7.2.4 Extensive Range of I/O & Processing

Registration Inputs

```

if($_POST) {

    //sanitize variables
    $username = strtolower(trim($_POST["username"]));
    $username = filter_var($username, FILTER_SANITIZE_STRING, FILTER_FLAG_STRIP_LOW | FILTER_FLAG_STRIP_HIGH);
    $password = trim($_POST['password']);
    $password_confirm = trim($_POST['password_confirm']);
    $forename = trim(ucfirst($_POST['forename']));
    $surname = trim(ucfirst($_POST['surname']));
    $email = strtolower(trim($_POST['email']));
    $email_confirm = strtolower(trim($_POST['email_confirm']));
    $phoneno = trim($_POST['phoneno']);
    $errors = array();

    // check if username is available
    $query = ("SELECT username FROM users WHERE username = :username");
    $query_params = array(':username' => $username);
    $db->DoQuery($query, $query_params);
    $rows = $db->fetch();
    if ($rows) {
        array_push($errors, "This username is already chosen. Please choose another username.");
    }

    if (!filter_var($email, FILTER_VALIDATE_EMAIL))
        array_push($errors, "Please specify a valid email address");

    if (filter_var($email, FILTER_VALIDATE_EMAIL) !== filter_var($email, FILTER_VALIDATE_EMAIL))
        array_push($errors, "The email addresses do not match");

    // check if email is available
    $query = ("SELECT email FROM users WHERE email = :email");
    $query_params = array(':email' => $email);
    $db->DoQuery($query, $query_params);
    $rows = $db->fetch();
    if ($rows) {
        array_push($errors, "This email is already associated with an account. Please choose another email.");
    }

    if (strlen($username) == 0)
        array_push($errors, "Please enter a username.");

    if (!preg_match("/^[\a-zA-Z ]*$/",$forename)) {
        array_push($errors, "Your forename has invalid characters.");
    }

    if (!preg_match("/^[\a-zA-Z ]*$/",$surname)) {
        array_push($errors, "Your surname has invalid characters.");
    }

    if (strlen($password) < 5)
        array_push($errors, "Please enter a password. Passwords must contain at least 5 characters.");

    if ($_POST["recaptcha_response_field"]) {
        $resp = recaptcha_check_answer ($privatekey, $_SERVER["REMOTE_ADDR"],
        $_POST["recaptcha_challenge_field"], $_POST["recaptcha_response_field"]);
        if (!$resp->is_valid) {
            array_push($errors, "The captcha is incorrect.");
        }
    }
}

```

```

        }

}

// If no errors were found, proceed with storing the user input
if (count($errors) == 0) {
    $password = encrypt($password);

    $email_parameters = array(
        ':pin' => encrypt($username)
    );
    email($email, $username, $forename, "confirm_registration", $email_parameters);
    $query = "INSERT INTO users (username, password, forename, surname, email, phoneno, activated, activation_code) VALUES (:username, :password, :forename, :surname, :email, :phoneno, :activated, :activation_code)";
    $query_params = array(
        ':username' => $username,
        ':password' => $password,
        ':forename' => $forename,
        ':surname' => $surname,
        ':email' => $email,
        ':phoneno' => $phoneno,
        ':activated' => '0',
        ':activation_code' => encrypt($username)
    );
    $db->DoQuery($query, $query_params);
    header("Location: confirmation.php?type=registration");
}
}
}

```

Redirect to confirmation page.

There is a plethora of Input and Outputs, especially when the customer is registering an account with the booking system. In this code, The user's input values are then trimmed and sanitised, and are checked again for any errors. The first round of errors are handled through jQuery's validator library, which are more intuitive and normally sieves through most of the possible errors prior to submitting the details. Errors are pushed into a array that is declared in core.php. When everything is verified (recognised by the fact that the error array is empty,) the password is then encrypted (More detail on encryption is shown later in the document) and a PIN is generated through its username, which is used to verify the registrant's email. The registrant is then sent an email and their information is stored in the database under the users table.

At this stage, the user is then directed to another page telling them further instructions.

Calendar Select

```

<form action="appointments.php" method="get" class="select" id="date">
<select name="year">
    <option value="">Year</option>
    <?php
        for ($syear = date('Y'); $syear > date('Y')-5; $syear--) {
            echo '<option value="'.$syear.'">'.$syear.'</option>';
        }
    ?>
</select>
<select name="month">
    <option value="">Month</option>
    <?php for ($smonth = 1; $smonth <= 12; $smonth++) {
        if (strlen($smonth)==1) {
            echo '<option value="0'.$smonth.'">0'.$smonth.'</option>';
        } else {
            echo '<option value="'.$smonth.'">'.$smonth.'</option>';
        }
    }
    ?>

```

Year loops counting down to 5 years ago.

Another loop for month; appends 0 in front if length of string is 1.

Scheduling & Booking System

```
}

?>
</select>

<select name="day">
    <option value="">Day</option>
    <?php for ($sday = 1; $sday <= 31; $sday++) {
        if (strlen($sday)==1) {
            echo '<option value="0' . $sday . '>0' . $sday . '</option>';
        } else {
            echo '<option value="' . $sday . '">' . $sday . '</option>';
        }
    }
?>
</select>

<button type="submit">Query</button>
</form>
```

Another loop for day; appends 0 in front if length of string is 1.

7.2.5 Graph

```
<?php
// Pie Chart Statistics

$query = "SELECT COUNT(*) FROM service";
$db->DoQuery($query);
$totalrows = $db->fetch();
$totalrow = $totalrows[0] - 1;

$query = "SELECT id, type FROM service";
$db->DoQuery($query);
$services = $db->fetchAll();

for ($x = 0; $x <= ($totalrows[0]-1); $x++){
    $query = "SELECT COUNT(*) FROM booking WHERE service_id = '".$services[$x]['id']."' ";
    $db->DoQuery($query);
    $counter = $db->fetch();
    array_push($services[$x], $counter[0]);
    array_push($services[$x], mb_substr(bin2hex($services[$x]['type']), 0, 6));
}
// Line graph Statistics

function count_instances($os_values, $month){
    $count = 0;
    foreach($os_values as $i)
        if(strpos($i, $month)!== FALSE)
            $count++;
    return $count;
}

$months = array();
$os_values = array();
$query = "SELECT date FROM booking WHERE date > DATE_SUB(now(), INTERVAL 6 MONTH) ORDER BY date ASC";
$db->DoQuery($query);
$total = $db->fetchAll();

foreach ($total as $counts) {
    $x = 0;
    $date = date("Y-m", strtotime($counts[$x]));
    $x++;
    if (!in_array($date, $months)){
        array_push($months, $date); //creation of months array
    }
}

foreach ($total as $t){
    array_push($os_values, $t[0]);
}
?>
<script>
<?php
//Piechart

echo "var piechart = [ ";
for ($x = 0; $x <= ($totalrow); $x++){
    echo "{ ";
    echo "value: ".$services[$x][2].", ";
    echo 'color:"#'.$services[$x][3].'", ';
    echo 'highlight: "#'.dechex(hexdec($services[$x][3])+(pow(2,13))).'", ';
    echo 'label: "'.$services[$x][1].'" ';
    if ($x == $totalrow){
        echo "}";
    } else {
        echo "}, ";
}
```

counts number of things per service in a given month

atomically generates a colour based on the name of the service

```

        }
    }
    echo "]; ";
}

//Graph

echo "var linechart = { ";
echo "labels : [";
for ($x = 0; $x <= (count($months)-1); $x++){
    if ($x == count($months)-1){
        echo "'".date("M", strtotime($months[$x]))."', ";
    } else {
        echo "'".date("M", strtotime($months[$x])).'", ';
    }
}
echo "datasets : [ { ";
echo 'label: "Bookings Per Month",
    fillColor : "rgba(220,220,220,0.2)",
    strokeColor : "rgba(220,220,220,1)",
    pointColor : "rgba(220,220,220,1)",
    pointStrokeColor : "#fff",
    pointHighlightFill : "#fff",
    pointHighlightStroke : "rgba(220,220,220,1)", ';
    echo "data : [";
for ($x = 0; $x <= (count($months)-1); $x++){
    if ($x == count($months)-1){
        echo count_instances($os_values, $months[$x]);
    } else {
        echo count_instances($os_values, $months[$x]). ",";
    }
}
echo "] } ] }";
?>

window.onload = function(){
    var ctx = document.getElementById("service_popularity").getContext("2d");
    window.myPie = new Chart(ctx).Pie(piechart, {
        responsive : true
    });
    var ctx2 = document.getElementById("graph").getContext("2d");
    window.myLine = new Chart(ctx2).Line(linechart, {
        responsive: true
    });
};

</script>

```

7.2.7 Regular Expressions.

Forename & Surname Validator

```
if (!preg_match("/^[\a-zA-Z ]*$/",$forename)) {
    array_push($errors, "Your forename has invalid characters.");
}

if (!preg_match("/^[\a-zA-Z ]*$/",$surname)) {
    array_push($errors, "Your surname has invalid characters.");
}
```

Using PHP's preg_match function to check if the specified RegEx matches the variables.

Phone Number Validator

```
if (!preg_match("/^(\+44\s?7\d{3}|\(07\d{3}\)\s?)\s?\d{3}\s?\d{3}$/",$phoneno)){
    array_push($errors, "Please specify a valid phone number.");
}
```

Using PHP's preg_match function to check if the specified RegEx matches the variables.

7.2.8 Preventing SQL Injections

```
if (isset($_POST['username']) && (isset($_POST['password'])) {
    $username = trim($_POST['username']);
    $password = encrypt(trim($_POST['password']));
    $query = "SELECT * FROM users WHERE username = :username AND password = :password";
    $query_params = array(
        ':username' => $username,
        ':password' => $password
    );
    $db->DoQuery($query, $query_params);
    $num = $db->fetch();
```

Input values are not inserted directly into the query.

Due to the nature of PDO, using prepared statements essentially deter most if not all of the possible modern SQL Injection methods due to how PDO manages queries. PDO generally prepares statements with parameters in the array, and supplies values when the query is executed.

PDO automatically takes care of escaping quotes and other characters in the values. For all values that are manually inserted, such as the username and password text-boxes, they are not put in directly into the query.

Other methods of protection include using the latest version of PHP, and forcing character encoding in UTF8 and LATIN1.

7.2.9 Encryption

```
function encrypt($value) {
    $salt = '}B8>){9#3*4L3t98d9QF3)?#9j?tKf';
    $salt = md5($salt);
    $value = md5($value);
    $value = hash('ripemd160', $salt . $value);
    return $value;
```

```

}
```

Encrypt is a simple function that is recalled whenever the page requires it. It consists of using the predetermined function string md5() in php multiple times which, in itself, increases the difficulty of a brute force exponentially. A md5 hash of the parameter is then used in conjunction with a salt that is not seen across the program other than the function itself. Both the salt and the hashed value are then hashed using another pre-programmed function RIPEMD-160, which should resultantly return a hash that should thwart most attempts of backtracking in the event that a perpetrator gains access to the database.

```

$query_params = array(
    ':email' => $_POST['email']
);
$db->DoQuery($query, $query_params);
$userdetails = $db->fetchAll();
if ($userdetails) {
    $code = encrypt(rand(99341, 1102400));
    $query = "UPDATE users SET forgot_code = :code WHERE email = :email";
    $query_params = array(
        ':code' => $code,
        ':email' => $_POST['email']
    )
}

```

Above is an example of the function being used. This extract shows the encryption function being used to hash a randomly generated scramble of numbers. This code is used to reset the users password.

7.2.10 Paginator

```

$query = "SELECT booking.id, booking.date, users.forename,
users.surname, booking.time, booking.comments, booking.confirmedbystaff,
booking.staff_id, service.type, service.price, staff.s_forename, staff.s_surname
FROM booking INNER JOIN users ON booking.username = users.username
INNER JOIN staff ON booking.staff_id = staff.id
INNER JOIN service ON booking.service_id = service.id";
$order = "ORDER BY DATE(booking.date) DESC, booking.time DESC";
$count_rows = "SELECT count(*) FROM booking";

if (isset($_GET))
    if (isset($_GET['year']) & isset($_GET['month']) & isset($_GET['day'])) {
        if (!empty($_GET['year']) & !empty($_GET['month']) & !empty($_GET['day'])) {
            if (checkdate($_GET['month'], $_GET['day'], $_GET['year']) == TRUE){
                $date = $_GET['day'] . "/" . $_GET['month'] . "/" . $_GET['year'];
                $datequery = " WHERE date = '" . $_GET['year'] . "-" . $_GET['month'] . "-" . $_GET['day'] . "' ";
                $count_rows = $count_rows . $datequery;
                $query = $query . $datequery . $order;
            } else {
                array_push($errors, "The selected date is invalid, please try again.");
                $query = $query . $order;
            }
        } else {
            array_push($errors, "Please fill in all the criteria for the date.");
        }
    }
$db->DoQuery($count_rows);
$count = $db->fetch();

```

```

$per_page = 10; //define how many games for a page
$pages = ceil($count[0]/$per_page); ← Calculates number of pages needed,
//Show page links                                         ceil rounds up so all results can be
//Accommodated.

if(!isset($_GET['page'])){
$page="1";
} else{
$page=$_GET['page'];
}
$start = ($page - 1) * $per_page;
$query = $query . " LIMIT $start, $per_page";
$db->DoQuery($query);
$num = $db->fetchAll(); ← Sets limits to show specific range of
//Results                                                 results depending on the page,
//Append extra information to query variable.

<ul id="pagination"> ← Final resulting query that is sent to
//Show page links                                         the database to show what is on that
//Specific page.

for ($i = 1; $i <= $pages; $i++) ← Displays results.
{
    if ($page == $i){
        echo '<li id="active"><a href="Appointments.php?page='.$i.'">'.$i.'</a></li>';
    } else {
        echo '<li><a href="Appointments.php?page='.$i.'">'.$i.'</a></li>';
    }
}
?>
</ul>

```

7.2.11 Alphabet Sort

```

if(!isset($_GET['char'])){
$char="A";
} else{
$char=$_GET['char'];
}
$query = "SELECT * FROM users WHERE surname LIKE '".$char."%' ORDER BY surname ASC";
$count_rows = "SELECT count(*) FROM users WHERE surname LIKE '".$char."%'";
$db->DoQuery($count_rows);
$count = $db->fetch(); ← Alphabet Sort works similarly to how
//Prints alphabet                                         pagination is done; through database
//Loop to print links for each letter in the           queries.

Loop to print links for each letter in the
alphabet. Indicates an active if the letter
matches the page's letter.

...
//list alphabet
echo '<ul id="pagination">';
foreach (range('A', 'Z') as $chara) {
    if ($char == $chara){
        echo '<li id="active"><a href="users.php?char='.$chara.'">'.$chara.</a></li>
\n';
    } else {
        echo '<li><a href="users.php?char='.$chara.'">'.$chara.</a></li>\n';
    }
}
echo '</ul>';

echo '<ul id="pagination">';
//Show page links
for ($i = 1; $i <= $pages; $i++){

```

```

if ($page == $i){
    echo '<li id="active"><a href="users.php?char='.$char.'&page='.$i.'">'.$i.'</a></li>';
} else {
    echo '<li><a href="users.php?char='.$char.'&page='.$i.'">'.$i.'</a></li>';
}
echo '</ul>';

if (!count($num) < 1){
    foreach ($num as $row) {
        echo '<p><a href="users.php?usertype=customers&user_id='.$row['id'].'">';
        echo $row['id']. " - ".$row['forename']. " ".$row['surname']. "(".
        $row['username']. ")" . '</a></p>';
    }
} else {
    echo '<h1>There\'s no customers here..</h1>';
    echo 'Try the other letters?';
}

```

Print results according to letter and page.

7.2.12 PIN Generator

```

function generate_pin($used_pins = array()){
    $new_pin = rand(10000,99999);
    $new_pin_hash = encrypt($new_pin);
    while (in_array($new_pin_hash, $used_pins)){
        generate_pin($used_pins);
    }
    return array(
        'pin' => $new_pin,
        'pin_hash' => $new_pin_hash
    );
}

```

generated pin is a number between 10000 and 99999 meaning 89999 possible values.

outputs an array, containing the pin and its hashed value.

...

```

$query = "SELECT pin FROM staff";
$db->DoQuery($query);
$used_pins = $db->fetch();
if (!empty($used_pins)){
    $newpins = generate_pin($used_pins);
} else {
    $newpins = generate_pin();
}

```

outputs an array, containing the pin and its hashed value.

7.2.13 Data Structures

2D Arrays

```
<?php
$title = 'Home';
if (!$_COOKIE['userdata']['loggedin'] == 1) {
    header('Location: login.php');
};
```

Cookies are used with multidimensional arrays to differentiate between users.

```
function list_appointments($num = array()) {
if(!empty($num)) {
    echo "<form method='post' action='appointments.php'>";
    echo "<ul class='accordion'>";
    foreach ($num as $row) {
        echo '<li>';
        echo '<div id="topbox" class="base">';
        if ($row['confirmedbystaff'] == 1) {
            echo '<div id="indicator" class="checkedin"></div>';
        } elseif ($row['confirmedbystaff'] == 0) {
            if (strtotime($row['date'] . " " . $row['time']) <= time() - 300) {
                echo '<div id="indicator" class="missing"></div>';
            } else {
                echo '<div id="indicator" class="notcheckedin"></div>';
            }
        }
    }
}
```

First Dimension in Array

Second Dimension in Array

Above is an extract from /functions/list_appointment_results.php, when

```
$db->DoQuery($query, $query_params);
$num = $db->fetchAll();
```

```
include("../functions/list_appointment_results.php");
list_appointments($num);
```

Query Results are Multidimensional.

Above is how this array is used. The function would be called with the parameters being the results of the query. The query results consist of a two dimensional array which are handled with the function.

Queues

Queues are automatically implemented into the program whenever the database is queried for data.

Queries are added in the array in an order.

```
//Calculating Number of Bookings Monthly
$startmonth = date("Y-m") . "-01";
$endmonth = date("Y-m") . "-31";
$r = "SELECT COUNT(*) FROM booking WHERE date >='".$startmonth' AND date <='".$endmonth."'";
$db->DoQuery($r);
$monthly_bookings = $db->fetch();

//Calculating Number of Bookings Daily
$today = date("Y-m-d");
$query = "SELECT COUNT(*) FROM booking WHERE date ='$today'";
$db->DoQuery($query);
$daily_bookings = $db->fetch();

//Calculating Revenue from monthly bookings
$startmonth = date("Y-m") . "-01";
$endmonth = date("Y-m") . "-31";
$query = "SELECT sum(service.price) FROM booking INNER JOIN service ON booking.service_id=service.id
WHERE date >='".$startmonth' AND date <='".$endmonth."'";
$db->DoQuery($query);
$monthly_revenue = $db->fetch();

// Calculating Number of Accounts
$query = "SELECT COUNT(*) FROM users";
$db->DoQuery($query);
$registered_accounts = $db->fetch();

// Calculating Number of Activated Accounts
$query = "SELECT COUNT(*) FROM users WHERE activated = '1'";
$db->DoQuery($query);
$activated_accounts = $db->fetch();

// Calculating Opening Day
$query = "SELECT value FROM metadata WHERE id = '1'";
$db->DoQuery($query);
$openhr = $db->fetch();
```

Another example of queues being implemented are for error messages. Errors that have occurred are pushed into an array which will be the queue. When the code is finished being read, the page will print the errors in the order they were pushed in the array, First in First Out.

```
// check if username is available
$query = ("SELECT username FROM users WHERE username = :username");
$query_params = array(':username' => $username);
$db->DoQuery($query, $query_params);
$rows = $db->fetch();
if ($rows) {
    array_push($errors, "This username is already chosen. Please choose another username.");
}

if (!filter_var($email, FILTER_VALIDATE_EMAIL))
array_push($errors, "Please specify a valid email address");

if (filter_var($email, FILTER_VALIDATE_EMAIL) !== filter_var($email, FILTER_VALIDATE_EMAIL))
array_push($errors, "The email addresses do not match");

// check if email is available
$query = ("SELECT email FROM users WHERE email = :email");
$query_params = array(':email' => $email);
$db->DoQuery($query, $query_params);
$rows = $db->fetch();
if ($rows) {
    array_push($errors, "This email is already associated with an account. Please choose another
email.");
}

if (!preg_match("/^(\+44\s?7\d{3}|\(?07\d{3}\)\?)\s?\d{3}\s?\d{3}$/", $phoneno)){
    array_push($errors, "Please specify a valid phone number.");
}

if (!preg_match("/^[\a-zA-Z ]*$/", $forename)) {
    array_push($errors, "Your forename has invalid characters.");
}

if (!preg_match("/^[\a-zA-Z ]*$/", $surname)) {
    array_push($errors, "Your surname has invalid characters.");
}

if (strlen($password) < 5){
    array_push($errors, "Please enter a password. Passwords must contain at least 5 characters.");
}

...

function display_errors($errors = array()) {
    if (!empty($errors)){
        foreach ($errors as $msg){
            echo '<div class="error">';
            echo $msg;
            echo "</div>";
        }
    }
}
```



display_errors is used in the system core.php

Recursion

```
function generate_pin($used_pins = array()){
    $new_pin = rand(10000, 99999);
    $new_pin_hash = encrypt($new_pin);
    while (in_array($new_pin_hash, $used_pins)){
        generate_pin($used_pins);
    }
    return array(
        'pin' => $new_pin,
        'pin_hash' => $new_pin_hash
    );
}
```

Recursion is used in a pin generation function. This function works by checking whether the hashes in the array arguments are the same as a newly generated pin code. If there is a match, then the code will recall itself, repeating until the program ends up with a pin and its resulting hash, unique from the other PIN hashes in the arguments.

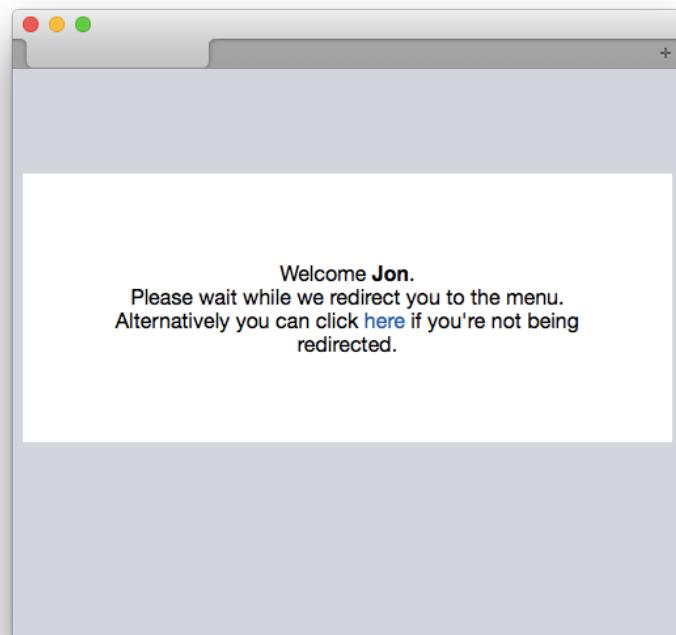
7.2.14 Creation of Objects at Runtime

Welcome Screen

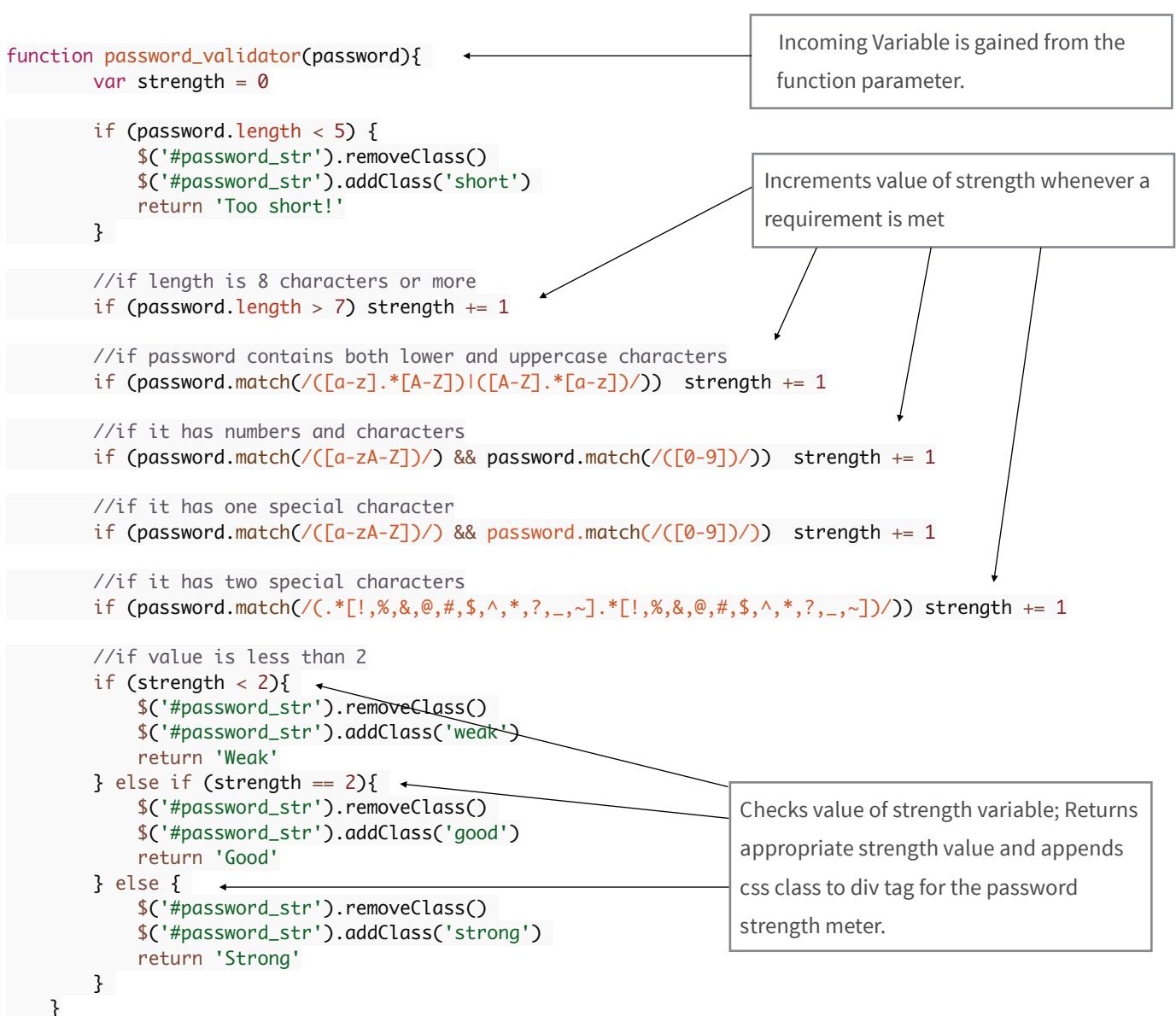
```
<?php
$title = 'Welcome';
if (!$_COOKIE['userdata']['loggedin'] == 1) {
    header('Location: login.php');
} else {
    header("refresh: 3; url=index.php");
}
include_once('includes/core.php');
echo '<link rel="stylesheet" href="'.$_COOKIE['directory'].'assets/style.css'.'"><br><br><br><br>';
echo '<div id="container"><center><p>';
echo "Welcome <b>" . $_COOKIE['userdata']['forename'] . "</b>." ;
echo "<br>Please wait while we redirect you to the menu." ;
echo "<br>Alternatively you can click <a href='index.php'>here</a> if you're not being redirected." ;
echo '</p></center></div>';
?>
```

refresh will direct the customer to the index page in three seconds automatically.

When the user has successfully logged into the system they are sent to the welcome screen, which will then transfer them to the index page.



Password Strength



	Password:	Confirm Password:	Password Strength:	
	<input type="text" value="****"/>		<div style="width: 10%;">Too short!</div>	
	<input type="text" value="*****"/>		<div style="width: 20%; background-color: #ffcccc;">Weak</div>	
	<input type="text" value="***** "/>		<div style="width: 40%; background-color: #ccffcc;">Good</div>	
	<input type="text" value="***** "/>	<input type="text" value="***** "/>	<div style="width: 100%; background-color: #32cd32;">Strong</div>	

7.3 Source Code

admin/appointments.php

```

<?php
$title = "Appointments";
$menutype = "admin_dashboard";
$require_admin = true;
include_once("../includes/core.php");
include("../functions/list_appointment_results.php");

$query = "SELECT booking.id, booking.date, users.forename,
users.surname, booking.time, booking.comments, booking.confirmedbystaff,
booking.staff_id, service.type, service.price, staff.s_forename, staff.s_surname
FROM booking INNER JOIN users ON booking.user_id = users.id
INNER JOIN staff ON booking.staff_id = staff.id
INNER JOIN service ON booking.service_id = service.id";
$order = " ORDER BY booking.date DESC, booking.time DESC";
$count_rows = "SELECT count(*) FROM booking";

$date = date("d/m/Y");
$datequery = " WHERE date = '".date("Y/m/d")."' ";

if (isset($_GET))
    if (isset($_GET['year']) & isset($_GET['month']) & isset($_GET['day']))
        $year = $_GET['year'];
        $month = $_GET['month'];
        $day = $_GET['day'];
        if (!empty($year) & !empty($month) & !empty($day))
            if (checkdate($month, $day, $year) !== true)
                array_push($errors, "The selected date is invalid, please try again.");
            else {
                $date = $day."/".$month."/".$year;
                $datequery = " WHERE date = '".$year."-".$month."-".$day."'";
            }
        } else {
            array_push($errors, "Please fill in all the criteria for the date.");
        }
    }
$count_rows = $count_rows.$datequery;
$query = $query.$datequery.$order;

$db->DoQuery($count_rows);
$count = $db->fetch();

//Add following after it
$per_page = 10;//define how many results per page.
$pages = ceil($count[0]/$per_page);

if(!isset($_GET['page'])){
$page="1";
} else{
$page=$_GET['page'];
}
$start = ($page - 1) * $per_page;
$query = $query . " LIMIT $start, $per_page";
$db->DoQuery($query);
$num = $db->fetchAll();

include '../includes/header.php';
?>

<h1>Appointments</h1>

```

Scheduling & Booking System

```
<form action="appointments.php" method="get" class="select" id="date">
<select name="year">
    <option value="">Year</option>
    <?php
        for ($syear = date('Y'); $syear > date('Y')-5; $syear--) {
            echo '<option value="'. $syear .'">' . $syear . '</option>';
        }
    ?>
</select>
<select name="month">
    <option value="">Month</option>
    <?php for ($smonth = 1; $smonth <= 12; $smonth++) {
        if (strlen($smonth)==1) {
            echo '<option value="0' . $smonth . '">0' . $smonth . '</option>';
        } else {
            echo '<option value="'. $smonth .'">' . $smonth . '</option>';
        }
    }
    ?>
</select>

<select name="day">
    <option value="">Day</option>
    <?php for ($sday = 1; $sday <= 31; $sday++) {
        if (strlen($sday)==1) {
            echo '<option value="0' . $sday . '">0' . $sday . '</option>';
        } else {
            echo '<option value="'. $sday .'">' . $sday . '</option>';
        }
    }
    ?>
</select>

<button type="submit">Query</button>
</form>
<?php
if (isset($date)){
    echo '<center>Appointments on <h3>' . $date . '</h3></center>';
}
?>

<ul id="pagination">
<?php
//Show page links

for ($i = 1; $i <= $pages; $i++)
{
    if ($page == $i){
        echo '<li id="active"><a href="Appointments.php?page=' . $i . '">' . $i . '</a></li>';
    } else {
        echo '<li><a href="Appointments.php?page=' . $i . '">' . $i . '</a></li>';
    }
}
?>
</ul>

<?php
list_appointments($num);
include '../includes/footer.php';
?>
```

```
<script>
$( ".accordion" ).accordion();
$(".pinToggles").click(function(event){
```

Scheduling & Booking System

```
    event.stopPropagation();
});
</script>
admin/calendar.php
<?php
$title = "Calendar";
$menutype = "admin_dashboard";
$require_admin = true;
include_once("../includes/core.php");

if (isset($_POST)) {

    if (isset($_POST['closing_date'])) {
        $date = $_POST['closing_date'];
    }
    if (isset($_POST['id'])) {
        $id = $_POST['id'];
    }

    if (isset($_POST['new'])) {

        if(strtotime($date) < strtotime('today')){
            array_push($errors, "This date is too old, try again.");
        } else {
            $stamp = explode("-", $date);
            $day = $stamp[2];
            $month = $stamp[1];
            $year = $stamp[0];
            if(checkdate($month, $day, $year) !== true){
                array_push($errors, $date." is an invalid date, please try again.");
            }
        }
    }

    if (empty($errors)){
        $query = "INSERT INTO closed_days (date) VALUES (:date)";
        $query_params = array(
            ':date' => $date
        );
        $db->DoQuery($query, $query_params);
        array_push($update, 'The date '.$date.' is added into the list.');
    }
} elseif(isset($id)){
    $query = "DELETE FROM closed_days WHERE id = :id";
    $query_params = array(
        ':id' => $id
    );
    $db->DoQuery($query, $query_params);
    array_push($update, 'The closing day has been removed from the database.');
}

}

include($directory . '/includes/header.php');
$query = "SELECT * FROM closed_days";
$db->DoQuery($query);
$num = $db->fetchAll(PDO::FETCH_NUM);
?>
<h1>Calendar</h1>

<h2>Manage Closing Days</h2>
```

```

<?php

echo "<table id='mytable' style='width:100%'>";
foreach ($num as $row) {
    echo '<form action="" method="post" autocomplete="off">';
    echo '<tr>';
    echo '<td>' . date("l, js M, Y", strtotime($row['date'])) . '</td>';
    echo '<td><button value="' . $row[0] . '" name="id">Remove</button></td>';
    echo '</tr>';
    echo '</form>';
}
?>
</form>
<form action="" method="post" autocomplete="off">
<tr>
<td><input type="date" name="closing_date" placeholder="Today?" /></td>
<td><input type="submit" name="new" value="Add"></td>
</tr>
</form>
</table>

```

admin/index.php

```

<?php
$title = "Dashboard";
$menutype = "admin_dashboard";
require_admin = true;
include_once("../includes/core.php");
include($directory . "/functions/encryption.php");
include($directory . '/includes/header.php');

//Calculating Number of Bookings Monthly
$startmonth = date("Y-m") . "-01";
$endmonth = date("Y-m") . "-31";
$r = "SELECT COUNT(*) FROM booking WHERE date >= '$startmonth' AND date <= '$endmonth'";
$db->DoQuery($r);
$monthly_bookings = $db->fetch();

//Calculating Number of Bookings Daily
$today = date("Y-m-d");
$query = "SELECT COUNT(*) FROM booking WHERE date = '$today'";
$db->DoQuery($query);
$daily_bookings = $db->fetch();

//Calculating Revenue from monthly bookings
$startmonth = date("Y-m") . "-01";
$endmonth = date("Y-m") . "-31";
$query = "SELECT sum(service.price) FROM booking INNER JOIN service ON booking.service_id = service.id WHERE date >= '$startmonth' AND date <= '$endmonth'";
$db->DoQuery($query);
$monthly_revenue = $db->fetch();

if (empty($monthly_revenue[0])){
    $monthly_revenue[0] = 0;
}

// Calculating Number of Accounts
$query = "SELECT COUNT(*) FROM users";
$db->DoQuery($query);
$registered_accounts = $db->fetch();

// Calculating Number of Activated Accounts
$query = "SELECT COUNT(*) FROM users WHERE activated = '1'";
$db->DoQuery($query);

```

Scheduling & Booking System

```
$activated_accounts = $db->fetch();

// Calculating Opening Day
$query = "SELECT value FROM metadata WHERE id = '1'";
$db->DoQuery($query);
$openhr = $db->fetch();
?>

<div>

<center><h1>Statistics</h1></center>
<table id='statistics'>
<?php
echo "<tr><td>Number of Registered Accounts: </td><td>" . $registered_accounts[0] . "</td></tr>";
echo "<tr><td>Number of Online Bookings This Month: </td><td>" . $monthly_bookings[0] . "</td></tr>";
echo "<tr><td>Estimated Income This Month from Online Bookings: </td><td>" . "&pound;" . $monthly_revenue[0] . "</td></tr>";
echo "<tr><td>Number of Activated Accounts: </td><td>" . $activated_accounts[0] . "</td></tr>";
echo "<tr><td>Number of Bookings for Today: </td><td>" . $daily_bookings[0] . "</td></tr>";
?>
</table>
</div>

<script src="/assets/chart.min.js"></script>
<script src="/assets/chart.doughnut.js"></script>
<script src="/assets/chart.line.js"></script>

<div id="left">
    <h3>Service Popularity</h3>
    <div id="canvas-holder">
        <canvas id="service_popularity"/></div>
</div><div id="right">
<h3>Bookings Per Month</h3>
<div id="canvas-holder">
    <canvas id="graph"/></div>
</div>

<?php
// Pie Chart Statistics

$query = "SELECT COUNT(*) FROM service";
$db->DoQuery($query);
$totalrows = $db->fetch();
$totalrow = $totalrows[0] - 1;

$query = "SELECT id, type FROM service";
$db->DoQuery($query);
$services = $db->fetchAll();

for ($x = 0; $x <= ($totalrow); $x++){
    $query = "SELECT COUNT(*) FROM booking WHERE service_id = '" . $services[$x]['id'] . "'";
    $db->DoQuery($query);
    $counter = $db->fetch();
    array_push($services[$x], $counter[0]);
    array_push($services[$x], mb_substr(bin2hex($services[$x]['type']), 0, 6));
}
// Line graph Statistics

function count_instances($values, $month){
    $count = 0;
    foreach($values as $i)
        if(strpos($i, $month) !== FALSE)
            $count++;
}
```

Scheduling & Booking System

```

        $count++;
    return $count;
}

$months = array();
$values = array();
$query = "SELECT date FROM booking WHERE date > DATE_SUB(now(), INTERVAL 6 MONTH) ORDER BY date ASC";
$db->DoQuery($query);
$total = $db->fetchAll();

foreach ($total as $counts) {
    $x = 0;
    $date = date("Y-m", strtotime($counts[$x]));
    $x++;
    if (!in_array($date, $months)){
        array_push($months, $date); //creation of months array
    }
}

foreach ($total as $t){
    array_push($values, $t[0]);
}
?>
<script>
<?php
//Piechart

echo "var piechart = [ ";
for ($x = 0; $x <= ($totalrow); $x++){
    echo "{ ";
    echo "value: ".$services[$x][2].", ";
    echo 'color:"#'.$services[$x][3].'", ';
    echo 'highlight: "#'.dechex(hexdec($services[$x][3])+(pow(2,13))).'", ';
    echo 'label: "'.$services[$x][1].'" ';
    if ($x == $totalrow){
        echo "}";
    } else {
        echo "}, ";
    }
    echo "]";
};

//Graph

echo "var linechart = { ";
echo "labels : [";
for ($x = 0; $x <= (count($months)-1); $x++){
    if ($x == count($months)-1){
        echo "'".date("M", strtotime($months[$x])).'";
    } else {
        echo "'".date("M", strtotime($months[$x])).'";
    }
}
echo "datasets : [ { ";
echo 'label: "Bookings Per Month",
    fillColor : "rgba(220,220,220,0.2)",
    strokeColor : "rgba(220,220,220,1)",
    pointColor : "rgba(220,220,220,1)",
    pointStrokeColor : "#fff",
    pointHighlightFill : "#fff",
    pointHighlightStroke : "rgba(220,220,220,1)", ';
    echo "data : [";
for ($x = 0; $x <= (count($months)-1); $x++){
    if ($x == count($months)-1){
        echo count_instances($values, $months[$x]);
    }
}

```

```

} else {
    echo count_instances($values, $months[$x]) . ",";
}
echo "] } ] }";
?>

window.onload = function(){
    var ctx = document.getElementById("service_popularity").getContext("2d");
    window.myPie = new Chart(ctx).Pie(piechart, {
        responsive : true
    });
    var ctx2 = document.getElementById("graph").getContext("2d");
    window.myLine = new Chart(ctx2).Line(linechart, {
        responsive: true
    });
};

</script>
admin/login.php
<?php
$title = "Login";
$menutype = NULL;
include_once("../includes/core.php");
include("../functions/encryption.php");

if (isset($_GET['timeout'])){
    array_push($errors, 'You were logged out automatically for being inactive.');
}

if (isset($_SESSION['logged_in'])) {
    header('Location: index.php');
} else {
    if (isset($_POST['username']) && (isset($_POST['password']))) {
        $username = trim($_POST['username']);
        $password = encrypt(trim($_POST['password']));
        $query = "SELECT * FROM admin WHERE username = :username AND password = :password";
        $query_parameters = array(
            ':username' => $username,
            ':password' => $password
        );
        $db->DoQuery($query, $query_parameters);
        $num = $db->fetchAll();
        if ($num) {
            // user entered correct details
            setcookie('admin[loggedin]', TRUE, $timeout, '', '', '', TRUE);
            header('Location: index.php');
            exit();
        } else {
            //user entered incorrect details
            array_push($errors, 'The username and password combination is not recognised.
Please try again.');
        }
    }
    include('../includes/header.php');
?>

        <h1>Login</h1>
        <form action="login.php" method="post" autocomplete="off">
            <input type="text" name="username" placeholder="Username" /><br>
            <input type="password" name="password" placeholder="Password" /><br>
            <input type="submit" value="Login" id="submit"/>
            <a href="/forgot.php" class="login-link">Forgot your password?</a>
        </form>

```

</div>

</div>

<?php

}

?>

admin/logout.php

<?php

```
$expiry = time() - 60 * 60;
$expired = time() - 99999999;
unset($_COOKIE);
setcookie('admin[loggedin]', "", $expired);
header('Location: login.php');
```

?>

admin/services.php

<?php

```
$title = "Services";
$menutype = "admin_dashboard";
$require_admin = true;
```

```
include_once("../includes/core.php");
```

```
if (isset($_POST)) {
```

```
    if (isset($_POST['service_type'])) {
        $type = $_POST['service_type'];
        if (strlen($type) < 1) {
            array_push($errors, "Please type in a type.");
        }
    }
    if (isset($_POST['service_price'])) {
        $price = $_POST['service_price'];
        if (strlen($price) < 1) {
            array_push($errors, "Please type in a price.");
        }
    }
    if (isset($_POST['service_description'])) {
        $description = $_POST['service_description'];
        if (strlen($description) < 1) {
            array_push($errors, "Please type in a description.");
        }
    }
    if (isset($_POST['service_id_update'])) {
        $id = $_POST['service_id_update'];
    }
    if (isset($_POST['service_id_delete'])) {
        $id = $_POST['service_id_delete'];
    }

    if (isset($_POST['new'])) {
        if (count($errors) == 0) {
            $query = "INSERT INTO service (type, price, description) VALUES (:type, :price, :description)";
            $query_params = array(
                ':type' => $type,
                ':price' => $price,
                ':description' => $description
            );
            $db->DoQuery($query, $query_params);
            array_push($update, 'The query has been added.');
        }
    }
}
```

```

    }
} elseif(isset($_POST['service_id_update'])) {
    if (count($errors) == 0) {
        $query = "UPDATE service SET type = :type, description = :description, price = :price WHERE id = :id";
        $query_params = array(
            ':type' => $type,
            ':description' => $description,
            ':price' => $price,
            ':id' => $id
        );
        $db->DoQuery($query, $query_params);
        array_push($update, 'The service has been updated.');
    }
} elseif(isset($_POST['service_id_delete'])) {

    $query = "SELECT service_id FROM booking WHERE service_id = '$id'";
    $db->DoQuery($query);
    $num = $db->fetchAll();
    if (!empty($num)){
        array_push($errors, 'This service is used by a customer. It is no longer able to be deleted.');
    } else {
        $query = "DELETE FROM service WHERE id = '$id'";
        $db->DoQuery($query);
        array_push($update, 'The service has been deleted.');
    }
}

include($directory . '/includes/header.php');
$query = "SELECT * FROM service";
$db->DoQuery($query);
$num = $db->fetchAll(PDO::FETCH_NUM);

?>
<h1>Services</h1>

<?php

echo "<table id='mytable' style='width:100%>";
echo "<tr>
    <th>Option</th>
    <th>Value (€)</th>
    <th>Description</th>
</tr>";
foreach ($num as $row) {
    echo '<form action="services.php" method="post" autocomplete="off">';
    echo '<tr>';
    echo '<td><input type="text" name="service_type" placeholder="Name" value="'. $row['type'] .'></td>';
    echo '<td><input id="price" type="number" name="service_price" placeholder="value" value="'.$row['price'].'></td>';
    echo '<td><textarea id="description" type="text" name="service_description" placeholder="Description"/>'. $row['description'] .'</textarea></td>';
    echo '<td><button value="'. $row[0] .'" name="service_id_update">Update</button></td>';
    echo '<td><button value="'. $row[0] .'" name="service_id_delete">Remove</button></td>';
    echo '</tr>';
    echo '</form>';
}
?>
</form>

```

```

<form action="services.php" method="post" autocomplete="off">
<tr>
<td><input type="text" name="service_type" placeholder="Name"/></td>
<td><input id="price" type="number" name="service_price" size="4" placeholder="Price"/></td>
<td><textarea id="description" type="text" name="service_description" placeholder="Description"/></td>
<td><input type="submit" name="new" value="Add"></td>
</tr>
</form>
</table>

```

admin/settings.php

```

<?php
$title = "Settings";
$menutype = "admin_dashboard";
require_admin = true;
include_once("../includes/core.php");
if (isset($_POST['service_value']) && isset($_POST['service_id'])){
    $value = $_POST['service_value'];
    $id = $_POST['service_id'];

    if ($id == 1 OR $id == 2){
        if (strlen($value) <> 5){
            array_push($errors, "Please enter a correctly formatted time. (HH:MM)");
        } else {
            if (!preg_match('/(2[0-3]|01)[0-9]:[0-5][0-9]/', $value)){
                array_push($errors, "This is an invalid time.");
            }
        }
    }
    if ($id == 3){
        if (strlen($value) <> 2){
            array_push($errors, "The booking frequency is either too short or too long. Try again!");
        } else {
            if (!preg_match("/^0-9]+$/", $value)){
                array_push($errors, "This is an invalid interval.");
            }
        }
    }
    if ($id == 5){
        if (strlen($value) > 50){
            array_push($errors, "The business name is too long.");
        }
        if (strlen($value) <= 1){
            array_push($errors, "The business name is too short.");
        }
    }
    if ($id == 6){
        if (strlen($value) > 50){
            array_push($errors, "The slogan is too long.");
        }
    }
}

if (empty($errors)){
    $query = "UPDATE metadata SET value = '$value' WHERE id = '$id'";
    $db->DoQuery($query);
    array_push($update, 'Your information has been updated into the database.');
}
include($directory . '/includes/header.php');
$query = "SELECT * FROM metadata";

```

Scheduling & Booking System

```
$db->DoQuery($query);
$num = $db->fetchAll();

?>
<h1>Settings</h1>

<?php

echo "<table style='width:100%'>";
echo "<tr>
    <th>Option</th>
    <th>Value</th>
    <th>Description</th>
</tr>";
foreach ($num as $row) {
    echo '<form action="" method="post">';
    echo '<tr>';
    echo '<td>' . $row['rule'] . '</td>';
    echo '<td><input id='forms' name='service_value' value='"' . $row['value'] . "'></td>';
    echo '<td>' . $row['description'] . '</td>';
    echo '<td><button value='"' . $row['id'] . "' name='service_id'>Update</button></td>';
    echo '</tr>';
    echo '</form>';
}
?>
</form>

</table>
```

admin/users.php

```
<?php
$menutype = "admin_dashboard";
$title = "Users";
$require_admin = true;
include_once("../functions/encryption.php");
include_once("../functions/email.php");
include_once("../includes/core.php");

function list_staff($staff_details = array()){
    echo '<form action="" method="post" autocomplete="off">';
    echo '<div id="group">';
    echo '<div id="left">';
    // echo $row['s_forename'] . " " . $row['s_surname'];
    echo '<label>Forename:</label><br>';
    if (isset($staff_details['s_forename'])){
        echo '<input type="text" name="forename" placeholder="Forename" value="'.
$staff_details['s_forename'].'"/>';
    } else {
        echo '<input type="text" name="forename" placeholder="Forename"/>';
    }
    echo '<label>Surname:</label><br>';
    if (isset($staff_details['s_surname'])){
        echo '<input type="text" name="surname" placeholder="Surname" value="'.
$staff_details['s_surname'].'"/>';
    } else {
        echo '<input type="text" name="surname" placeholder="Surname"/>';
    }
    echo '<label>Email:</label><br>';
    if (isset($staff_details['s_email'])){
        echo '<input type="text" name="email" placeholder="email" value="'.
$staff_details['s_email'].'"/>';
    } else {
        echo '<input type="text" name="email" placeholder="email"/>';
    }
    echo '</div>';
```

Scheduling & Booking System

```
echo '<div id="right">';
if (isset($staff_details[0])){
    echo '<button value="'. $staff_details[0].'" name="id_update">Update</button>';
    if ($staff_details['banned'] == 0){
        echo '<button value="'. $staff_details[0].'" name="id_ban">Ban</button>';
    } else {
        echo '<button value="'. $staff_details[0].'" name="id_unban">Unban</button>';
    }
    echo '<button value="" .
$staff_details[0].'" name="new_pin_request">Generate New PIN</button>';
} else {
    echo '<button type="submit" value="true" name="register">Register</button>';
}
echo '</div>';
echo '</form>';
echo '</div>';
}

function generate_pin($used_pins = array()){
    $new_pin = rand(10000, 99999);
    $new_pin_hash = encrypt($new_pin);
    while (in_array($new_pin_hash, $used_pins)){
        generate_pin($used_pins);
    }
    return array(
        'pin' => $new_pin,
        'pin_hash' => $new_pin_hash
    );
}

if (isset($_GET['usertype'])){
    $usertype = $_GET['usertype'];
} else {
    $usertype = 'customers';
}
if (isset($_GET['user_id'])){
    $user_id = $_GET['user_id'];
    $query = "SELECT * FROM users WHERE id = '$user_id'";
    $db->DoQuery($query);
    $sq = $db->Fetch();
    if (empty($q)){
        array_push($errors, "The userid ".$user_id." doesn't exist.");
    }
}
if ($usertype == 'customers'){

    if (!isset($_GET['char'])){
        $char="A";
    }else{
        $char=$_GET['char'];
    }
    $query = "SELECT * FROM users WHERE surname LIKE '".$char."%' ORDER BY surname ASC";
    $count_rows = "SELECT count(*) FROM users WHERE surname LIKE '".$char."%'";
    $db->DoQuery($count_rows);
    $count = $db->fetch();

    // echo $count[0];
    //Add following after it
    $per_page = 10;//define how many games for a page
    $pages = ceil($count[0]/$per_page);

    if (!isset($_GET['page'])){
        $page="1";
    }else{
        $page=$_GET['page'];
    }
}
```

```

$start = ($page - 1) * $per_page;
$query = $query . " LIMIT $start, $per_page";
$db->DoQuery($query);
$num = $db->fetchAll();

}

if ($usertype == 'staff'){
    $query = "SELECT * FROM staff ORDER BY id ASC";
    $db->DoQuery($query);
    $num = $db->fetchAll();
}

if (isset($_POST)){
    if (isset($_POST['forename'])){
        $forename = trim($_POST['forename']);
        if (strlen($forename) < 1){
            array_push($errors, "Please type in a forename.");
        }
    }
    if (isset($_POST['surname'])){
        $surname = trim($_POST['surname']);
        if (strlen($surname) < 1){
            array_push($errors, "Please type in a surname.");
        }
    }
    if (isset($_POST['email'])){
        $email = trim($_POST['email']);
        if (!filter_var($email, FILTER_VALIDATE_EMAIL)){
            array_push($errors, "Please specify a valid email address.");
        }
    }
}

if (count($errors) == 0) {
    if (isset($_POST['register'])){

        $usedpins = array();
        $query = "SELECT pin FROM staff";
        $db->DoQuery($query);
        $used_pins = $db->fetchAll();
        foreach ($used_pins as $usedpin){
            array_push($usedpins, $usedpin['pin']);
        }
        if (!empty($usedpins)){
            $newpins = generate_pin($usedpins);
        } else {
            $newpins = generate_pin();
        }

        $query_params = array(
            ':pin' => $newpins['pin']
        );
        email($email, "Staff", $forename, "new_staff", $query_params);

        $query = "INSERT INTO staff (s_forename, s_surname, s_email, pin) VALUES (:forename, :surname, :email, :pin)";
        $query_params = array(
            ':forename' => $forename,
            ':surname' => $surname,
            ':email' => $email,
            ':pin' => $newpins['pin_hash']
        );
        // print_r($query_params);
        $db->DoQuery($query, $query_params);
        array_push($update, "$forename has been included into the database. A PIN is sent to his email address at $email.");
    }
}

```

```

    }

    if (isset($_POST['id_update'])){
        $id = $_POST['id_update'];
        $query = "UPDATE staff SET s_forename = :forename, s_surname = :surname, s_email = :email WHERE id = :id";
        $query_params = array(
            ':forename' => $forename,
            ':surname' => $surname,
            ':email' => $email,
            ':id' => $id
        );
        // print_r($query_params);
        $db->DoQuery($query, $query_params);
        array_push($update, "Staff information has been updated.");
    }

    if (isset($_POST['id_ban'])){
        if ($usertype == 'customers'){
            $id = $_POST['id_ban'];
            $query = "UPDATE users SET banned = true WHERE id = $id";
            $db->DoQuery($query);
            array_push($update, $q['forename']." is now banned.");
        }
        if ($usertype == 'staff'){
            $id = $_POST['id_ban'];
            $query = "UPDATE staff SET banned = true WHERE id = $id";
            $db->DoQuery($query);
            array_push($update, $_POST['forename']." is now banned from the Staff List.");
        }
    }

    if (isset($_POST['id_unban'])){
        if ($usertype == 'customers'){
            $id = $_POST['id_unban'];
            $query = "UPDATE users SET banned = false WHERE id = $id";
            $db->DoQuery($query);
            array_push($update, $q['forename']." is now unbanned.");
        }
        if ($usertype == 'staff'){
            $id = $_POST['id_unban'];
            $query = "UPDATE staff SET banned = false WHERE id = $id";
            $db->DoQuery($query);
            array_push($update, $_POST['forename']." is now unbanned from the Staff List.");
        }
    }

    if (isset($_POST['new_pin_request'])){
        $id = $_POST['new_pin_request'];

        $usedpins = array();
        $query = "SELECT pin FROM staff";
        $db->DoQuery($query);
        $used_pins = $db->fetchAll();
        foreach ($used_pins as $usedpin){
            array_push($usedpins, $usedpin['pin']);
        }

        if (!empty($used_pins)){
            $newpins = generate_pin($usedpins);
        } else {

```

```

        $newpins = generate_pin();
    }
    include_once("../functions/email.php");

    $email_params = array(
        ':pin' => $newpins['pin']
    );
    $query_params = array(
        ':pin' => $newpins['pin_hash'],
        ':id' => $id
    );
    email($email, "Staff", $forename, "new_pin", $email_params);

    $query = "UPDATE staff SET pin = :pin WHERE id = :id";
    $db->DoQuery($query, $query_params);

    array_push($update, "A new pin has been set to ".$forename.".");
}
}

include($directory . '/includes/header.php');
?>

<select id="navigator" autofocus onchange="location = this.options[this.selectedIndex].value;">
    <option value="?usertype=customers" <?
    php if($usertype == "customers") {echo 'selected="selected"';}?>>Customers</option>
    <option value="?usertype=staff" <?
    php if($usertype == "staff") {echo 'selected="selected"';}?>>Staff</option>
</select>

<?php

if ($usertype == 'customers'){

    if (!empty($q)){
        $query = "SELECT COUNT(*) FROM booking WHERE user_id = '$user_id'";
        $db->DoQuery($query);
        $number_of_bookings = $db->Fetch();

        echo "<form method='post' action=''>";
        echo '<div id="details">';
        echo '<div class="left">';
            echo '<h2>' . $q['username'] . '</h2><br>';
            echo '<b>Name</b> ' . $q['forename'] . ' ' . $q['surname'] . '<br>';
            echo '<b>Email</b> ' . $q['email'] . '<br>';
            echo '<b>Phone No.</b> ' . $q['phoneno'] . '<br>';

            echo '<h3>Statistics</h3>';
            echo 'Has booked a total of ' . $number_of_bookings[0] . " appointments.";
        echo '</div>';
        echo '<div class="right">';
            if ($q['banned'] == 0){
                echo '<button value="' . $q['id'] . '" name="id_ban">Ban</button>';
            } else {
                echo '<button value="' . $q['id'] . '" name="id_unban">Unban</button>';
            }
        echo '</div>';
        echo '</div>';
        echo '</form>';

    } else {
        //list alphabet
        echo '<ul id="pagination">';
    }
}

```

```

        foreach (range('A', 'Z') as $character) {
            if ($char == $character){
                echo '<li id="active"><a href="users.php?char=' . $character . '">' .
$character . "</a></li>\n";
            } else {
                echo '<li><a href="users.php?char=' . $character . '">' . $character . "</a></li>
\n";
            }
        }
        echo '</ul>';

        echo '<ul id="pagination">';
        //Show page links
        for ($i = 1; $i <= $pages; $i++){
            if ($page == $i){
                echo '<li id="active"><a href="users.php?char=' . $char . '&page=' .
$i . '">' . $i . "</a></li>' ;
            } else {
                echo '<li><a href="users.php?char=' . $char . '&page=' . $i . '">' . $i . "</a></
li>' ;
            }
        }
        echo '</ul>';

        if (!count($num) < 1){
            foreach ($num as $row) {
                echo '<p><a href="users.php?usertype=customers&user_id=' . $row['id'] . '">' ;
                echo $row['id'] . " - " . $row['forename'] . " " . $row['surname'] . " (" .
$row['username'] . ")" . '</a></p>' ;
            }
        } else {
            echo '<h1>There\'s no customers here..</h1>';
            echo 'Try the other letters?';
        }
    }
}
elseif ($usertype == 'staff') {

    echo "<h1>Staff</h1>";
    array_shift($num);
    foreach ($num as $row) {
        list_staff($row);
    }
    echo '<h2>Register New Staff</h2>';
    list_staff();
}
?>

<script>
$( 'html' ).bind( 'keypress' , function(e)
{
    if(e.keyCode == 13)
    {
        return false;
    }
});
</script>

```

assets/recaptcha_values.php

```
<?php
// ReCaptcha Properties
// if ($_SERVER['HTTP_HOST'] != "localhost") {
$publickey = "6LcquPwAAAAAHDtQdsJgDyjVAo_eNkNHO0R1UvV";
```

```

    $privatekey = "6LcquPwAAAAAFW168bbf835aADGzK_If5wctI-y";
// } else {
// $publickey = "6Lf_ufwAAAAAH12NOzKjIBZsEiMhIhG4q6B-_Re";
// $privatekey = "6Lf_ufwAAAAALj2xh6s2SxMFu_16xG1MEkojGLL";
// }
// # the response from reCAPTCHA
$resp = null;
# the error code from reCAPTCHA, if any
$error = null;
?>

```

classes/booker.php

```

<?php
class booker {
    public $day, $month, $year, $selected_date, $first_day, $back, $forward;
    function database() {
        $this->db = new database();
        $this->db->initiate();
    }
    function QuickFetch($query) {
        $this->db->DoQuery($query);
        $fetch_value = $this->db->fetch();
        return $fetch_value[0];
    }
    function minutes($time) {
        $time = explode(':', $time);
        return ((($time[0] * 3600) + ($time[1] * 60)) / 60;
    }
    function make_calendar($selected_date, $first_day, $back, $forward, $day, $month, $year) {
        $this->database();
        $this->day = $day;
        $this->month = $month;
        $this->year = $year;
        $this->selected_date = $selected_date;
        $this->first_day = $first_day;
        $this->back = $back;
        $this->forward = $forward;
        $this->booking_start_time = $this-
>QuickFetch("SELECT value FROM metadata WHERE id = '1'");
        $this->booking_end_time = $this-
>QuickFetch("SELECT value FROM metadata WHERE id = '2'");
        $this->booking_frequency = $this-
>QuickFetch("SELECT value FROM metadata WHERE id = '3'");
        $this->start_booking($year, $month);
    }
    function post($month, $day, $year) {
        //error lists
        $errors = array();
        include('assets/recaptcha_values.php');
        include_once('assets/recaptcha.php');

        if (isset($_POST['booking_time']) && $_POST['booking_time'] == 'selectvalue') {
            array_push($errors, "Please select a booking time.");
        }
        if (strlen($_POST['comments']) >= 140) { // In the event the html limit is bypassed
            array_push($errors, "You have exceeded the character limit for the comments.
Please shorten it!");
        }
        if (isset($_POST['booking_service']) && $_POST['booking_service'] == 'selectvalue')
        {
            array_push($errors, "Please select a service.");
        }
        if (strlen($_POST["recaptcha_response_field"]) == 0) {

```

```

        array_push($errors, "Please type in the captcha.");
    }
    if ($_POST["recaptcha_response_field"]) {
        $resp = recaptcha_check_answer($privatekey, $_SERVER["REMOTE_ADDR"], $_POST["recaptcha_challenge_field"], $_POST["recaptcha_response_field"]);
        if (!$resp->is_valid) {
            array_push($errors, "The captcha is incorrect. Please try again!");
        }
    }
    if (!empty($errors)) {
        display_errors($errors); //no errors, continue with saving into the database.
    } else {
        $this->database(); //
declared again as post will go to this function instead of loading the calendar.
        $booking_date = date("Y-m-d", mktime(0, 0, 0, $month, $day, $year));
        $booking_time = $_POST['booking_time'];
        $booking_service = $_POST['booking_service'];
        $query = "INSERT INTO booking (date, time, user_id, comments, confirmedbystaff, service_id, staff_id) VALUES (:booking_date, :booking_time, :user_id, :comments, :confirmed, :service_id, :staff_id)";
        $query_params = array(
            ':booking_date' => $booking_date,
            ':booking_time' => $booking_time,
            ':service_id' => $booking_service,
            ':user_id' => $_COOKIE['userdata']['user_id'],
            ':comments' => $_POST['comments'],
            ':confirmed' => 0,
            ':staff_id' => 1
        );
        $this->db->DoQuery($query, $query_params);
        $this-
>confirmation($booking_date, $booking_time, $booking_service, $_COOKIE['userdata']
['user_id']);
    }
}
function confirmation($date, $time, $serviceid, $user_id) {
    include('functions/email.php');
    $query = "SELECT type FROM service WHERE id = '$serviceid'";
    $this->db->DoQuery($query);
    $service = $this->db->fetch();
    $query = "SELECT email FROM users WHERE id = '$user_id'";
    $this->db->DoQuery($query);
    $email = $this->db->fetch();
    $extra = array(
        ':bookingday' => $date,
        ':bookendtime' => $time,
        ':bookingservice' => $service[0]
    );
    email($email['email'], $user_id, $_COOKIE['userdata']
['forename'], "appointment", $extra);
    echo "<meta http-equiv='refresh' content='0; url=confirmation.php?
type=appointment' />";
}
function start_booking($year, $month) {
    $period = $year . '-' . $month . '%';
    $query = "SELECT * FROM booking WHERE date LIKE '$period'";
    $this->db->DoQuery($query);
    $this->count = $this->db->RowCount();
    while ($rows = $this->db->fetch()) {
        $this->prior_bookings[] = array(
            "date" => $rows['date'],
            "start" => $rows['time']
        );
    }
    $this->create_days_arr($year, $month);
}

```

```

function create_days_arr($year, $month) { // Creates array of days in the month.

    $num_days_month = cal_days_in_month(CAL_GREGORIAN, $month, $year); // Make array
called $day with the correct number of days
    for ($i = 1; $i <= $num_days_month; $i++) {
        $d           = mktime(0, 0, 0, $month, $i, $year);
        $this->days[] = array(
            "daynumber" => $i,
            "dayname"   => date("l", $d)
        );
    }
    for ($j = 1; $j <= $this->first_day; $j+
+) { // Add blank elements to start of array if the first day of the month is not a Monday.
        array_unshift($this->days, '0');
    }
    $padding_end = 7 - (count($this-
>days) % 7); // Add blank elements to end of array if required.
    if ($padding_end < 7) {
        for ($j = 1; $j <= $padding_end; $j++) {
            array_push($this->days, NULL);
        }
    }
    $this->start_calendar();
}

function create_days_table() {
    $day_capacity = ((minutes($this->booking_end_time) + 30) - minutes($this-
>booking_start_time)) / $this->booking_frequency;
    $fdm = $this->year . "-" . $this->month . "-" . "01"; //day 01 of the month
    $closed_days_query = "SELECT date FROM closed_days WHERE YEAR(date) = YEAR('$fdm'
) AND MONTH(date) = MONTH('$fdm')";
    $this->db->DoQuery($closed_days_query);
    $closed_days = $this->db->fetchAll();
    $i = 0;
    foreach ($this->days as $row) {
        $box = '';
        if ($i % 7 == 0) {
            echo "</tr><tr>";
        } // Modulus used to give <tr> after every seven <td> cells
        if (isset($row['daynumber']) && $row['daynumber'] !=
= 0) { // Padded days at the start of the month will have a 0 at the beginning
            echo "<td class='days'>";
            if ($this->count > 0) {
                $bookings_on_day = 0;
                foreach ($this->prior_bookings as $booking_date) {
                    if ($booking_date['date'] == $this->year . '-' . $this-
>month . '-' . sprintf("%02s", $row['daynumber'])) {
                        $bookings_on_day++;
                    }
                }
            }
            $this_day = $this->year . "-" . $this->month . "-" . $row['daynumber'];
            foreach ($closed_days as $item) { // It's a closed day, set from the data
base.
                if (strtotime($item['date']) == strtotime($this_day)) {
                    $box = 2;
                }
            }
            if (mktime(0, 0, 0, $this-
>month, sprintf("%02s", $row['daynumber']) + 1, $this->year) < strtotime("now")) {
                $box = 2; // Past Day / Unavailable
            }
            if ($row['dayname'] == 'Sunday') {
                $box = 2; // It's a Sunday
            }
            if ($bookings_on_day >= $day_capacity && $box == '') {

```

```

        $box = 3; // Fully Booked
    }
    if ($bookings_on_day > 0 && $box == '') {
        $box = 1; // Part booked day.
    }
    echo $this->day_switch($box, $row['daynumber']) . "</td>";
} else { // Show NULL day.
    echo "<td class='days'><div class='box' id='key_null'></div></td>";
}
$i++;
}
echo "</tr></table>";
$this->make_booking_slots();
}

function day_switch($box, $daynumber) {
    $date_number = "<p>" . str_replace(NULL, ' ', $daynumber) . "</p>";
    $link = "<a href='calendar.php?month=" . $this->month . "&year=" . $this->year . "&day=" . sprintf("%02s", $daynumber) . '#selected_date' . "'>";
    switch ($box) {
        case (1):
            $text = $link . "<div class='box' id='key_partbooked'>" . $date_number . "</div></a>";
            break;
        case (3):
            $text = "<div class='box' id='key_fullybooked'>" . $date_number . "</div>";
            break;
        case (2):
            $text = "<div class='box' id='key_unavailable'>" . $date_number . "</div>";
            break;
        default:
            $text = $link . "<div class='box' id='key_available'>" . $date_number . "</div></a>";
            break;
    }
    return $text;
}

function make_booking_slots() {
    if ($this->day == 0) { // default day = 0; done so to show that date is not chosen.
        echo "<form id='calendar_form' method='post' action='>";
        echo "<div class='status' id='selected_date'>Please select a day.</div>";
    } else {
        $this->create_form();
    }
}

function start_calendar() {
    echo "<table cellpadding='0' id='calendar'>
        <div id='week'>";
    echo "<div class='buttons' align='left'>";
    if ($this->selected_date > strtotime('-6 months')) {
        echo "<a href='?month=" . date("m", $this->back) . "&year=" . date("Y", $this->back) . "'>";
    } else {
        echo "<a class='invisible'>";
    };
    echo "&#8592;</a></div>";
    echo "<div id='center_date'>" . date("F, Y", $this->selected_date) . "<b class='keyguide'>
        <i>?</i>
        <ul class='keylist'>
            <h3>Key</h3>
                <li id='key_fullybooked'>Fully Booked</li>
                <li id='key_partbooked'>Free (Partially)</li>

```

```

        <li id='key_available'>Free</li>
        <li id='key_unavailable'>Not Available</li>
    </ul>
</div></b>";
echo "<div class='buttons' align='right'>";
if ($this->selected_date < strtotime('+6 months')) {
    echo "<a href=?month=" . date("m", $this-
>forward) . "&year=" . date("Y", $this->forward) . "'>";
} else {
    echo "<a class='invisible'>";
};
echo "&#8594;</a></div>";
echo "</div>
<tr>";
$days = array("Mo", "Tu", "We", "Th", "Fr", "Sa", "Su");
for ($i = 0; $i < 7; $i++) {
    echo '<th>' . $days[$i] . '</th>';
}
if (empty($this->prior_bookings)) {
    $this->create_days_table($this->days, $this->month, $this->year);
} else {
    $this->create_days_table($this->days, $this->prior_bookings, $this-
>month, $this->year);
}
}

function create_form() {
    $this->db->DoQuery("SELECT * FROM service");
    $services = $this->db->fetchAll();
    for ($i = strtotime($this->booking_start_time); $i <= strtotime($this-
>booking_end_time); $i = $i + $this->booking_frequency * 60) {
        $booking_times[] = date("H:i:s", $i);
    }
    echo "<form id='calendar_form' method='post' action='>";
    echo "<div id='left'>";
    echo "<div class='status' id='selected_date'>Selected Date is: " . date("D, d F Y
", mktime(0, 0, 0, $this->month, $this->day, $this->year)) . "</div>";
    $option = "<select id='select' name='booking_time'><option value='selectvalue'>Please select a booking time</option>";
    if ($this->count >= 1) {
        foreach ($this-
>prior_bookings as $row) { // Check for bookings and remove any previously booked slots
            foreach ($booking_times as $i => $r) {
                if ($row['start'] == $r && $row['date'] == $this->year . '-' . $this-
>month . '-' . $this->day) {
                    unset($booking_times[$i]);
                }
            }
        }
    }
    foreach ($booking_times as $booking_time) {
        $finish_time = strtotime($booking_time) + $this-
>booking_frequency * 60; // Calculate finish time
        $option .= "<option value='" . $booking_time . "'>" . $booking_time . " -
" . date("H:i:s", $finish_time) . "</option>";
    }
    echo $option . "</select>";
    echo "<select id='select' name='booking_service'>";
    echo "<option value='selectvalue'>Please select a Service</option>";
    foreach ($services as $row) {
        $text = $row[1] . " - &pound;" . $row[2];
        echo '<option value=' . $row[0] . "'>" . $text . '</option>';
    }
    echo '</select>';
    echo "<textarea rows='3' cols='30' maxlength='140' name='comments' placeholder='A
ny comments? (140 Character Limit!)></textarea>";
}

```

```

        echo "</div><div id='right'>";
        include('assets/recaptcha_values.php');
        include_once('assets/recaptcha.php');
        echo recaptcha_get_html($publickey, $error);
        echo "<button type='submit'>Submit</button></div></form>";
    }

}
?>

```

classes/database.php

```

<?php
class database {
    public function initiate() {
        include($_SERVER['DOCUMENT_ROOT'].'/includes/db.php');
        try {
            $this-
        } catch(PDOException $e) {
            die($e->getMessage());
        }
        $this->database->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    }
    public function DoQuery($query, $arguments = array()) {
        try {
            $this->result = $this->database->prepare($query);
            $this->result->execute($arguments);
        } catch(PDOException $e) {
            die($e->getMessage());
        }
    }
    public function fetch() { // Shows single row
        return $this->result->fetch();
    }
    public function fetchAll() { // Shows all associated rows (in array)
        return $this->result->fetchAll();
    }
    public function RowCount() { // Count rows
        return $this->result->RowCount();
    }
}
?>

```

functions/check_username.php

```

<?php
require( "../classes/database.php");
$db = new database();
$db->initiate();

if (isset($_POST['username'])) {
$value = $_POST['username'];
};

$query = ("SELECT username FROM users WHERE username = :username");
$query_params = array(
    ':username' => $value
);
$db->DoQuery($query, $query_params);
$rows = $db->fetch();

if ($value == NULL)
    echo '';
else if (strlen($value) <= 3)

```

```

    echo 'This username is too short!';
else {
    if ($rows >= 1)
        echo "This username isn't available.";
else if ($rows == 0)
    if (ctype_alnum($value)) {
        $username_available == TRUE;
        echo "This username is available!";
    }
else
    echo "This username is invalid.";
}
?>

```

functions/email.php

```

<?php
function email($email, $user_id, $forename, $type, $extra = array()) {
$headers = "From: robot@tnguyen.ch";
if ($type == "confirm_registration") {
    if(isset($extra[:pin])) {
        $pin = $extra[:pin];
        $subject = 'Time to confirm your email';
        $message = "Hi $forename, \n
\n You need to activate your account. Click the link below.\n\n";
        $message .= "http://projects.tnguyen.ch/confirmation.php?
type=registration&value=$pin";
    } else {
        echo "No pin is inserted.";
    }
}
if ($type == "forgotten_password") {
    if(isset($extra[:code])) {
        $code = $extra[:code];
        $subject = 'Forgotten Password request?';
        $message = "Hi $forename, \n
\n You have requested to reset your password. Click the link below. \n\n";
        $message .= "http://projects.tnguyen.ch/confirmation.php?type=forgot&value=
$code";
    }
}
if ($type == "password_reset_confirmed") {
    if(isset($extra[:newpassword])) {
        $newpassword = $extra[:newpassword];
        $subject = 'Your New Password.';
        $message = "Hi $forename, \n\n Your new password is: \n\n";
        $message .= "$newpassword";
    }
}
if ($type == "booking") {
    $subject = 'Time to confirm your booking';
    $message = "Hi $forename, \n
\n You need to activate your account. Click the link below. \n\n";
}
if ($type == "appointment") {
    if(isset($extra[:bookingday]) & isset($extra[:bookingtime]) & isset($extra[:bookingservice])) {
        $booking_date = $extra[:bookingday];
        $booking_time = $extra[:bookingtime];
        $booking_service = $extra[:bookingservice];
        $subject = 'Your Appointment';
        $message = "Hi $forename,\n
\n You have an appointment at $booking_date, $booking_time for a $booking_service.";
    }
}

```

```

if ($type == "new_pin") {
    if(isset($extra[":pin"])) {
        $pin = $extra[":pin"];
        $subject = 'Your New Pin';
        $message = "Hi $forename, \n\n Your new pin is $pin.";
    }
}
if ($type == "new_staff") {
    if(isset($extra[":pin"])) {
        $pin = $extra[":pin"];
        $subject = 'Welcome!';
        $message = "Hi $forename, \n
        \n Welcome to our company. Your new pin is $pin.";
    }
}
$message .= "\n\n\n";
$message .= "Nails Club\n";
$message .= "128 Barking Road\n";
$message .= "London\n";
$message .= "E16 1EN\n";
// Send
mail($email, $subject, $message, $headers);
}
?>

```

functions/encryption.php

```

<?php
function encrypt($value) {
    $salt = '}B8>){9#3*4L3t98d9QF3)&#9j?tKf';
    $salt = md5($salt);
    $value = md5($value);
    $value = hash('ripemd160', $salt . $value);
    return $value;
}
?>

```

functions/list_appointment_results.php

```

<?php
function list_appointments($num = array()) {
if(!empty($num)) {
    echo "<form method='post' action='appointments.php'>";
    echo "<ul class='accordion'>";
    foreach ($num as $row) {
        echo '<li>';
        echo '<div id="topbox" class="base">';
        if ($row['confirmedbystaff'] == 1) {
            echo '<div id="indicator" class="checkedin"></div>';
        } elseif ($row['confirmedbystaff'] == 0) {
            if (strtotime($row['date'] . " " . $row['time']) <= time() - 300) {
                echo '<div id="indicator" class="missing"></div>';
            } else {
                echo '<div id="indicator" class="notcheckedin"></div>';
            }
        }
        echo '<p>' . date("g:i A", strtotime($row['time'])) . " - " . $row['forename'] . " " .
        $row['surname'] . '</p>';
        echo '</div>';
        echo '<ul id="group">';
        echo '<div id="left">';
        echo $row['forename'] . " " . $row['surname'] . '<br>';
        echo $row['time'] . '<br>';
        echo $row['type'] . '<br>';
        echo '</div>';
        echo '<div id="right">';
        echo 'Total Price: &pound;' . $row['price'] . '<br>';
    }
}
?>

```

```

        if ($row['confirmedbystaff'] == 1){
            echo ' Checked in by '.$row['s_forename'].' '.$row['s_surname'] . "<br>";
        }

        if (isset($_COOKIE['staff'])){
            if ($row['confirmedbystaff'] == 0) {
                echo '<button type="submit" name="checkin_customer_id" value="'.$
$row['id'].'">Checkin</button>';
            } else {
                echo '<button type="submit" name="uncheck_customer_id" value="'.$
$row['id'].'">Uncheck</button>';
                // echo '<button value="'.$row['id'].'">Bill</button><br>';
            }
        }

        echo '</div>';
        if (!empty($row['comments'])){
            echo '<h2>Customers Comment</h2>';
            echo '<p id="checkin_comments">' . $row['comments'] . '</p><br>';
        }
        echo '</ul>';
        echo'</li>';
    }
    echo "</ul>";
} elseif (empty($num)) {
    echo "<h1>Nothing was found here...</h1>";
    echo "Try searching for their forename or surname.";
}
echo "</ul></form>";
}
?>

```

functions/search_appointments.php

```

<?php
require( "../classes/database.php");
$db = new database();
$db->initiate();

setcookie('staff[id]', $_POST['id'], $staff_expiry, '', '', '', TRUE);

if (isset($_POST['username'])) {
$value = $_POST['username'];
};

$query = "SELECT booking.id, booking.date, users.forename, users.surname,
booking.time, booking.comments, booking.confirmedbystaff, booking.staff_id,
staff.s_forename, staff.s_surname, service.type, service.price
FROM booking
INNER JOIN users ON booking.user_id = users.id
INNER JOIN staff ON booking.staff_id = staff.id
INNER JOIN service ON booking.service_id = service.id
WHERE booking.date = CURDATE() AND users.forename like :user
OR booking.date = CURDATE() AND users.surname like :user
ORDER BY booking.time ASC";
$query_params = array(
    ':user' => $value
);
$db->DoQuery($query, $query_params);
$num = $db->fetchAll();

include("../functions/list_appointment_results.php");
list_appointments($num);

```

?>

includes/core.php

<?php

```

$timeout = time() + 600;

//Automatic Logout Sessions

if ($require_user == true) {

    if (!$_COOKIE['userdata']['loggedin'] == 1) {

        header('Location: login.php?timeout=true');

    } else {

        //rewrite cookie with new time.

        setcookie('userdata[loggedin]', TRUE, $timeout, '', '', '', TRUE);

        setcookie('userdata[user_id]', $_COOKIE['userdata']
['user_id'], $timeout, '', '', '', TRUE);

        setcookie('userdata[forename]', $_COOKIE['userdata']
['forename'], $timeout, '', '', '', TRUE);

        setcookie('userdata[surname]', $_COOKIE['userdata']
['surname'], $timeout, '', '', '', TRUE);

    };

};

if ($require_admin == true) {

    // echo "this page requires admin priv";

    // print_r($_COOKIE);

    if (!($_COOKIE['admin']['loggedin']) == 1){ //checks if cookie is expired.

        header('Location: login.php?timeout=true');

    } else {

        //rewrite cookie with new time.

        setcookie('admin[loggedin]', $_COOKIE['admin']
['loggedin'], $timeout, '', '', '', TRUE);

    }

};

    if ((strpos($_SERVER['SCRIPT_NAME'], 'admin') !=
== false) OR strpos($_SERVER['SCRIPT_NAME'], 'staff') !=
== false OR strpos($_SERVER['SCRIPT_NAME'], '/includes/search.php') != false) {

        $directory = "../";

```

```

    } else {

        $directory = "";

    }

include_once($directory . 'assets/recaptcha_values.php');

mb_internal_encoding("UTF-8");

date_default_timezone_set("GMT");

$errors = array();

$update = array();

function display_errors($errors = array()) {

    if (!empty($errors)) {

        foreach ($errors as $msg) {

            echo '<div class="error">';

            echo $msg;

            echo "</div>";

        }

    }

}

function display_updates($message = array()) {

    if (!empty($message)) {

        foreach ($message as $msg) {

            echo "<div class='update'>";

            echo $msg;

            echo "</div>";

        }

    }

}

include($directory . "classes/database.php");

if (file_exists($directory . "setup.php") && (strpos($_SERVER['SCRIPT_NAME'], 'setup') == false)) {

    header("Location: setup.php");

}

$db = new database;

```

```
$db->initiate();
```

?>

includes/db.php

```
<?php
$hostname      = "localhost";
$database_name = "thien_projects";
$username      = "root";
$password      = "root";
?>
```

includes/footer.php

```
</div>
<!-- <div id="footer">
COPYRIGHT POTATO
</div> -->
</body>
</html>
```

includes/header.php

```
<?php

if (file_exists($directory . "setup.php")) {
    $company_name = "Booker";
} else {
    $query = "SELECT value FROM metadata WHERE id = 5";
    $db->DoQuery($query);
    $value = $db->fetch();
    $company_name = $value[0];
    $query = "SELECT value FROM metadata WHERE id = 6";
    $db->DoQuery($query);
    $value = $db->fetch();
    $slogan = $value[0];
}

?>

<html>
<!DOCTYPE HTML>
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
<head>
<title><?php echo $company_name;?> - <?php echo $title; ?></title>
<link rel="stylesheet" href="<?php echo $directory . "assets/style.css";?>">
<meta name="apple-mobile-web-app-capable" content="yes">
<meta name="format-detection" content="telephone=yes">
<meta name="apple-mobile-web-app-status-bar-style" content="black-translucent">
<meta name="viewport" content="width=device-width, initial-scale=1">

<script src="../assets/jquery.js"></script>
<script src="../assets/jqueryui.js"></script>
<script type="text/javascript">
    // Listen for ALL links at the top level of the document. For
    // testing purposes, we're not going to worry about LOCAL vs.
    // EXTERNAL links - we'll just demonstrate the feature.
    $( document ).on(
        "click",
        "a",
        if(window.location.contains("?month=")){
            function( event ){
                // Stop the default behavior of the browser, which
            }
        }
    )
</script>
```

```

        // is to change the URL of the page.
        event.preventDefault();

        // Manually change the location of the page to stay in
        // "Standalone" mode and change the URL at the same time.
        location.href = $( event.target ).attr( "href" );
    }
}

);
</script>
</head>
<body>
    <div id="heading">
        <div id="branding">

            <a href="index.php"><?php echo $company_name;?></a>
            <div id="slogan"><?php if (isset($slogan)){echo $slogan;}?></div>
        </div>
        <?php
        if (strpos($_SERVER['SCRIPT_NAME'], 'admin') !== false){
            echo 'A';
        } elseif (strpos($_SERVER['SCRIPT_NAME'], 'staff') !== false){
            echo 'S';
        }
        ?>
        <div id="headingright">
            <?php
            // print_r($_COOKIE);

            if (isset($_COOKIE['userdata'])) {
                if (!strpos($_SERVER['SCRIPT_NAME'], 'staff')){
                    if ($_COOKIE['userdata']['loggedin'] == 1) {
                        echo 'Hello, <a href="profile.php">' . $_COOKIE['userdata']
                            ['forename'] . '</a>';
                        echo ' (<a href="logout.php">Logout</a>)';
                    };
                }
            } elseif (isset($_COOKIE['admin']['loggedin'])){
                if ($_COOKIE['admin']['loggedin'] == 1) {
                    echo 'Hello, <b>Administrator</b>';
                    echo ' (<a href="logout.php">Logout</a>)';
                }
            }
            ?>
            </div>
        </div>
    <div id="container">

        <?php
        display_errors($errors);
        display_updates($update);

        if (isset($menutype)) {
        if ($menutype == "user_dashboard") { ?>

            <header>
                <ul>
                    <li><a href="calendar.php" <?php if ($title == 'Book Appointment')
{echo "class='active'";}?>>Book Appointment</a></li>
                    <li><a href="manage_appointments.php" <?php if ($title == 'Manage Appointments')
{echo "class='active'";}?>>Manage Appointments</a></li>
                </ul>
            </header>

            <?php
        };
        if ($menutype == "admin_dashboard"){?>

```

```

<header>
    <ul>
        <li><a href="index.php" <?php if ($title == 'Dashboard') {echo "class='active'";}?>Dashboard</a></li>
        <li><a href="appointments.php" <?php if ($title == 'Appointments') {echo "class='active'";}?>Appointments</a></li>
        <li><a href="calendar.php" <?php if ($title == 'Calendar') {echo "class='active'";}?>Calendar</a></li>
        <li><a href="services.php" <?php if ($title == 'Services') {echo "class='active'";}?>Services</a></li>
        <li><a href="users.php" <?php if ($title == 'Users') {echo "class='active'";}?>Users</a></li>
        <li><a href="settings.php" <?php if ($title == 'Settings') {echo "class='active'";}?>Settings</a></li>
    </ul>
</header>
<?php
}};

?>

```

staff/appointments.php

```

<?php
$title = "Checkins";
include_once("../includes/core.php");
if (!isset($_COOKIE['staff'])) {
    header('Location: index.php?timeout=true');
} else {
    $staff_expiry = time() + 60;
    setcookie('staff[loggedin]', $_COOKIE['staff']['loggedin'], $staff_expiry, '', '', TRUE);
    setcookie('staff[id]', $_COOKIE['staff']['id'], $staff_expiry, '', '', TRUE);
    setcookie('staff[forename]', $_COOKIE['staff']['forename'], $staff_expiry, '', '', TRUE);
    setcookie('staff[surname]', $_COOKIE['staff']['surname'], $staff_expiry, '', '', TRUE);
}
if (isset($_POST['checkin_customer_id'])) {
    $query = "UPDATE booking SET confirmedbystaff=1, staff_id=:staff_id WHERE id = :id";
    $query_params = array(
        ':staff_id' => $_COOKIE['staff']['id'],
        ':id' => $_POST['checkin_customer_id']
    );
    $db->DoQuery($query, $query_params);
    header("Location: appointments.php");
}

if (isset($_POST['uncheck_customer_id'])) {
    $query = "UPDATE booking SET confirmedbystaff=0, staff_id=:staff_id WHERE id = :id";
    $query_params = array(
        ':staff_id' => 1,
        ':id' => $_POST['uncheck_customer_id']
    );
    $db->DoQuery($query, $query_params);
    header("Location: appointments.php");
}

$query = "SELECT booking.id, booking.date, users.forename,
users.surname, booking.time, booking.comments, booking.confirmedbystaff,
booking.staff_id, service.type, service.price, staff.s_forename, staff.s_surname
FROM booking
INNER JOIN users ON booking.user_id = users.id
INNER JOIN staff ON booking.staff_id = staff.id

```

Scheduling & Booking System

```
INNER JOIN service ON booking.service_id = service.id
WHERE booking.date = CURDATE()
ORDER BY booking.time ASC";
$db->DoQuery($query);
$num = $db->fetchAll();

?>

</script>
<html>
<head>
<title>Concierge - <?php echo $title; ?></title>
<meta name="apple-mobile-web-app-capable" content="yes">
<link rel="stylesheet" href="<?php echo $directory."assets/style.css";?>" />
<meta name="viewport" content="width=device-width, initial-scale=1, user-scalable=no">
<script src="../assets/jquery.js"></script>
<script src="../assets/jqueryui.js"></script>
<script>
$( document ).on(
    "click",
    "a",
    if(window.location.contains("?month=")){
        function( event ){
            // Stop the default behavior of the browser, which
            // is to change the URL of the page.
            event.preventDefault();

            // Manually change the location of the page to stay in
            // "Standalone" mode and change the URL at the same time.
            location.href = $( event.target ).attr( "href" );
        }
    };
</script>
</head>
<body>
<div id="checkin_topbar">
    <div class="group">
        <div class="left">
            <b><?php echo date("l, js F");?><br></b>
            <?php echo $_COOKIE['staff']['forename']." ".$_COOKIE['staff']['surname'];?>
        </div>
        <div class="right">
            <h1>Checkins</h1>
        </div>
    </div>
    <form method="post" action="<?PHP echo htmlspecialchars($_SERVER['PHP_SELF']); ?>" autocomplete="on" id="forms">
        <input id="username_" name="username" type="text" placeholder="Search"/>
    </form>
</div>
<div class="blank"></div>
<div id="container">

<?php

include("../functions/list_appointment_results.php");
if (!empty($num)) {
    list_appointments($num);
}
elseif (empty($num)) {
    echo "<h1>There's no appointments today.</h1>";
    echo "Time to relax!";
}

Cand. Name: Thien Nguyen (7461) Centre No. 12290 271
```

```

include '../includes/footer.php';
?>
</div>
<script>
$(".accordion").accordion({
    animate: {
        duration: 250
    }
});
$(".pinToggles").click(function(event){
    event.stopPropagation();
});
</script>

<script type="text/javascript">
$(document).ready(function() {
    var originalState = $("#container").html();
    $('#username_').keyup(function() {
        var value = $("#username_").val();
        if (value.length > 1) {
            $.post('/functions/search_appointments.php', { username: forms.username.value, id: <?
php echo $_COOKIE['staff']['id'];?> },
                function(result) {
                    $('#container').html(result).show();
                });
        } else {
            $("#container").html(originalState);
        }
    });
});
</script>

```

staff/index.php

```

<?php
$title = "Staff Login";
include("../includes/core.php");
include("../functions/encryption.php");
if (isset($_COOKIE['staff']['logged_in'])) {
    //if true, show admin index
    header('Location: appointments.php');
} else {
    if (isset($_GET['timeout'])){
        array_push($errors, 'You were logged out automatically for being inactive.');
    }
    //show login
    if (isset($_POST['pin'])){
        $pin = $_POST['pin'];
        if (strlen($pin) < 2){
            $error = 'You need to type in your PIN!';
        } else {
            $pin = encrypt($_POST['pin']);
            $query = "SELECT * FROM staff WHERE pin = :pin";
            $query_params = array(':pin' => $pin);
            $db->DoQuery($query, $query_params);
            $num = $db->fetchAll();
            if ($num) {
                //user entered correct details
                // echo "<pre>";
                // print_r($num);
                // echo "</pre>";
                $staff_expiry = time() + 10;
                setcookie('staff[loggedin]', TRUE, $staff_expiry, ' ', ' ', TRUE);
                setcookie('staff[id]', $num[0]['id'], $staff_expiry, ' ', ' ', TRUE);
                setcookie('staff[forename]', $num[0]
['s_forename'], $staff_expiry, ' ', ' ', TRUE);
            }
        }
    }
}

```

Scheduling & Booking System

```
        setcookie('staff[surname]', $num[0]
['s_surname'], $staff_expiry, '', '', TRUE);

        header('Location: appointments.php');
        exit();
    } else {
        //user entered incorrect details
        array_push($errors, 'Details were incorrect, try again!');
    }
}

include("../includes/header.php");
?>

<html>
    <head>
        <title>
            Login
        </title>
    <link rel="stylesheet" href="../assets/login.css"/>
    <link rel="apple-touch-icon" href="/icon.png"/>
    </head>
    <body>
        <div class="login-container">
            <div class="login">
                <h1>Enter Your Pin</h1>
                <form action="index.php" method="post" autocomplete="off">
                    <input type="tel" min="5" max="5" name="pin" placeholder="pin" />
                    <button type="submit">Enter</button>
                </form>
            </div>
        </div>
        </body>
    </html>

<?php
}
?>
```

404.php

```
<?php
$title = 'Error 404, Something\'s Missing!';
$require_user = false;
include 'includes/header.php';
?>
<h1>Uh oh, Something's missing here.</h1>
Not to worry. You can either head <a onclick="window.history.go(-1); return false;">back</
/a> to where you was before, or sit there and listen to a goat scream like a human.<br><br>
<iframe src="http://www.youtube.com/embed/SIaFtAKnqBU?
vq=hd720&rel=0&showinfo=0&controls=0&iv_load_policy=3&loop=1&playlist=SIaFtAKnqBU&modestb
randing=1&autoplay=1" width="100%" height="inherit" frameborder="0" webkitAllowFullScreen
allowFullScreen></iframe>
<?php
include 'includes/footer.php';
?>
```

calendar.php

```
<?php

$title = 'Book Appointment';

$menu_type = "user_dashboard";
Cand. Name: Thien Nguyen (7461)           Centre No. 12290           273
```

```

$require_user = true;

include_once('includes/core.php');

include('classes/booker.php');

$calendar = new booker();

if (isset($_GET['month'])) {
    $month = $_GET['month'];
} else {
    $month = date("m");
}

if (isset($_GET['year'])) {
    $year = $_GET['year'];
} else {
    $year = date("Y");
}

if (isset($_GET['day'])) {
    $day = $_GET['day'];
} else {
    $day = 0;
}

$selected_date = $year . "-" . $month . "-" . $day;

$selected_date_timestamp = mktime(0, 0, 0, $month, 01, $year); // Make a timestamp based
on the GET values

$first_day = date("N", $selected_date_timestamp) - 1;

$back = strtotime("-1 month", $selected_date_timestamp);

$forward = strtotime("+1 month", $selected_date_timestamp);

include('includes/header.php');

if($_SERVER['REQUEST_METHOD'] == 'POST') {

    $calendar->post($month, $day, $year);
}

```

```

$closed_days_query = "SELECT date FROM closed_days WHERE YEAR(date) = YEAR('$selected_date') AND MONTH(date) = MONTH('$selected_date')";

$db->DoQuery($closed_days_query);

$closed_days = $db->fetchAll();

$start = "SELECT value FROM metadata WHERE id = 1";

$db->DoQuery($start);

$start = $db->fetch();

$end = "SELECT value FROM metadata WHERE id = 2";

$db->DoQuery($end);

$end = $db->fetch();

$freq = "SELECT value FROM metadata WHERE id = 3";

$db->DoQuery($freq);

$frequency = $db->fetch();

function minutes($time){

$time = explode(':', $time);

return ((($time[0]*3600) + ($time[1]*60))/60;

}

$fullybooked = ((minutes($end[0])+30) - minutes($start[0]))/$frequency[0];



$numberofbookingsonday = "SELECT COUNT(*) FROM booking WHERE YEAR(date) = YEAR('$selected_date') AND MONTH(date) = MONTH('$selected_date') AND DAY(date) = DAY('$selected_date')";

$db->DoQuery($numberofbookingsonday);

$day_count = $db->fetch();

foreach ($closed_days as $item) { // It's a closed day, set from the database.

```

Scheduling & Booking System

```
if (strtotime($item['date']) == strtotime($selected_date)){
    array_push($errors, "We would be closed on the date chosen.");
}

if (checkdate($month,$day,$year) == FALSE && $day !== 0){
    array_push($errors, "This date is invalid.");
}

if (strtotime($selected_date) > strtotime('+6 months') OR strtotime($selected_date) < strtotime('-6 months')){
    array_push($errors, "This date is currently beyond our booking threshold for now.");
}

if(date('w', strtotime($selected_date)) == 0 AND $day !== 0) {
    array_push($errors, "We don't open on sundays.");
}

if ($day_count[0] >= $fullybooked){
    array_push($errors, "Today is fully booked.");
}

if (empty($errors)) {
    $calendar-
    >make_calendar($selected_date_timestamp, $first_day, $back, $forward, $day, $month, $year );
} else {
    echo "<h1>This date is unavailable.</h1>";
    foreach ($errors as $error) {
        echo $error . "<br>";
    }
    echo "You can click <a href='calendar.php'>here</a> to go back on the calendar.";
}
echo "</div>";
?>

<script type="text/javascript">
$( 'b.keyguide' ).on( "touchstart", function ( e ) {
```

```

var link = $(this);

if (link.hasClass('hover')) {
    link.removeClass("hover");
    $('ul.keylist').hide();
    $('a.taphover').removeClass("hover");
} else {
    link.addClass("hover");
    $('ul.keylist').show();
    $('a.taphover').not(this).removeClass("hover");
}

});

</script>

<?php

include('includes/footer.php');

?>

```

confirmation.php

```

<?php
$title = 'Confirmation';
$require_user = false;
include_once('includes/core.php');
include_once('functions/encryption.php');
include('functions/email.php');
include('includes/header.php');

if (isset($_GET['value'])){
$value = $_GET['value'];
} else {
$value = null;
}
if (isset($_GET['type'])){
$type = $_GET['type'];
}

if (isset($type)){
    if ($type == "registration"){
        if($value !== null){
            $query = "SELECT activation_code FROM users WHERE activation_code = :value";
            $query_params = array(':value' => $value);
            $db->DoQuery($query, $query_params);
            $rows = $db->fetch();
            if ($rows) {
                $query = "UPDATE users SET activated = :key, activation_code = :newcode
WHERE activation_code = :value";
                $query_params = array(
                    ':key' => '1',
                    ':newcode' => NULL,
                    ':value' => $value
                );
                $db->DoQuery($query, $query_params);
            }
        }
    }
}

```

```

        echo '<h1>Okay, It\'s been confirmed.</h1>';
        echo '<div>Your account is now active. You may now <a href="login.php">Log in</a>.</div>';
    } else{
        echo "Sorry, We can't activate your account right now. You may want to try again.";
    }
} else {
    echo "<h1>Okay, It's time to confirm.</h1>";
    echo "A email has been sent to confirm your registration.";
}
}elseif ($type == "confirmed"){
echo "<h1>Okay, It's been confirmed.</h1>";
echo '<A HREF="javascript:javascript:history.go(-1)">Click here to go back to previous page</A>';
}elseif ($type == "forgot"){
if (isset($value)){
$query = ("SELECT forgot_code FROM users WHERE forgot_code = :value");
$query_params = array(':value' => $value);
$db->DoQuery($query, $query_params);
$rows = $db->fetch();
if ($rows) {
    $query = ("SELECT username, email, forename FROM users WHERE forgot_code = :value");
    $query_params = array(':value' => $value);
    $db->DoQuery($query, $query_params);
    $userdetails = $db->fetch();
    $newpassword = rand(100000, 999999);

    $extra = array(':newpassword' => $newpassword);
    email($userdetails['email'], $userdetails['username'], $userdetails['forename'], "password_reset_confirmed", $extra);

    $query = "UPDATE users SET password = :password, forgot_code = :blank WHERE forgot_code = :value";
    $query_params = array(
        ':password' => encrypt($newpassword),
        ':blank' => NULL,
        ':value' => $value
    );
    $db->DoQuery($query, $query_params);

    echo '<h1>Okay, Its been sorted.</h1>';
    echo '<div>We\'ve sent you a email which contains your new password. Please change the password as soon as you <a href="login.php">Log in</a>.</div>';
} else{
    echo "We can't reset your password right now, Please try again.";
}
} else {
    echo '<h1>Okay, We\'ve sent you an email.</h1>';
    echo '<div>We\'ve sent you a email containing further instructions.</div>';
}
}elseif ($type == "appointment") {
echo "<h1>Okay, Your appointment is confirmed.</h1>";

echo 'Your appointment details are sent to your email address.';
}elseif ($type == "updated") {
echo "<h1>Okay, Your information is updated.</h1>";
echo "Your changes will take effect the next time you log in. <br>";
echo 'You can now continue with your day. :)';
}
}

// header('Location: 404.php');
?>

```

edit.php

```

<?php
$title = 'Edit';
require_user = true;
include_once("includes/core.php");
include("functions/encryption.php");

$user_id = $_COOKIE['userdata']['user_id'];
$query = "SELECT * FROM users WHERE id = :id";
$query_params = array(
    ':id' => $_COOKIE['userdata']['user_id']
);
$db->DoQuery($query, $query_params);
$prevalue = $db->fetch();

if($_POST) {
    // $username = $_COOKIE['userdata']['username'];
    $user_id = $_COOKIE['userdata']['user_id'];
    $password = trim($_POST['new_password']);
    $current_password = encrypt(trim($_POST['current_password']));
    $password_confirm = trim($_POST['new_password_confirm']);
    $forename = trim(ucfirst($_POST['forename']));
    $surname = trim(ucfirst($_POST['surname']));
    $email = trim($_POST['email']);
    $phoneno = trim($_POST['phoneno']);
    $errors = array();

    // check if password is available
    $query = ("SELECT id FROM users WHERE password = :currentpassword");
    $query_params = array(
        ':currentpassword' => $current_password
    );
    $db->DoQuery($query, $query_params);
    $rows = $db->fetch();
    if ($rows[0] !== $_COOKIE['userdata']['user_id']) {
        array_push($errors, "Your current password is incorrect, Please try again.");
    }
    // Validate the input
    if (!filter_var($email, FILTER_VALIDATE_EMAIL)){
        array_push($errors, "Please specify a valid email address.");
    }

    $query = ("SELECT email FROM users WHERE email = :email");
    $query_params = array(':email' => $email);
    $db->DoQuery($query, $query_params);
    $rows = $db->fetch();
    if ($rows) {
        if (!$rows['email'] = $email){
            array_push($errors, "This email is already associated with an account. Please choose another email.");
        }
    }

    if (!preg_match("/^(\+44\s?7\d{3}|\(?07\d{3}\)\?)\s?\d{3}\s?\d{3}$/, $phoneno)){
        array_push($errors, "Please specify a valid phone number.");
    }

    if (isset($password) & $password !== ""){
        if (strlen($password) < 5)
            array_push($errors, "Please enter a password. Passwords must contain at least 5 characters.");
    }
}

```

```

if (count($errors) == 0) {

    $password = encrypt($password);
    if ($password == "532fa54f4ce9af28bc9eb777cc8b70b28d361f9f"){
        $password = NULL;
    }

    if ($forename != $prevalue['forename']){
        // $newdetails["0"][] = $forename;
        $query = "UPDATE users SET forename = '$forename' WHERE id = '$user_id'";
        $db->DoQuery($query);
        // echo "updated forename";
    }
    if ($surname != $prevalue['surname']){
        // $newdetails["1"][] = $surname;
        $query = "UPDATE users SET surname = '$surname' WHERE id = '$user_id'";
        $db->DoQuery($query);
        // echo "updated surname";
    }
    if ($password !== NULL AND $password !== $prevalue['password']){
        // $newdetails["2"][] = $password;
        $query = "UPDATE users SET password = '$password' WHERE id = '$user_id'";
        $db->DoQuery($query);
        // echo "updated password";
    }
    if ($email != $prevalue['email']){
        // $newdetails["3"][] = $email;
        $query = "UPDATE users SET email = '$email' WHERE id = '$user_id'";
        $db->DoQuery($query);
        // echo "updated email";
    }
    if ($phoneno != $prevalue['phoneno']){
        // $newdetails["4"][] = $phoneno;
        $query = "UPDATE users SET phoneno = '$phoneno' WHERE id = '$user_id'";
        $db->DoQuery($query);
        // echo "updated phoneno";
    }

    header("Location: confirmation.php?type=updated");

}

include('includes/header.php');
?>

<script type="text/javascript" src="assets/jquery.js"></script>
<script src="//ajax.aspnetcdn.com/ajax/jquery.validate/1.9/jquery.validate.min.js"></script>
<script type="text/javascript">
$(document).ready(function() {
$('#username_availability').load('functions/check_username.php').show();
$('#username_').keyup(function() {
$.post('functions/check_username.php', { username: forms.username.value },
function(result) {
$('#username_availability').html(result).show();
});
});
});
```

```

// When the browser is ready...
$(function() {

    // Setup form validation on the #register-form element
    $("#forms").validate({

        // Specify the validation rules
        rules: {
            forename: "required",
            forename: "required",
            phoneno: {
                required: true,
                minlength: 11,
                maxlength: 11
            },
            email: {
                required: true,
                email: true
            },
            username: "required",
            current_password: "required",
            password: {
                minlength: 6
            },
            email_confirm: {
                required: true,
                email: true,
                equalTo: "#email"
            },
            password_confirm: {
                equalTo: "#password"
            }
        },

        // Specify the validation error messages
        messages: {
            forename: "Please enter your forename.",
            surname: "Please enter your surname.",
            email: "Please enter a valid email address.",
            username: "Please enter a valid username.",
            current_password: {
                required: "Please provide a password."
            },
            password_confirm: {
                equalTo: "Please provide a password."
            },
            phoneno: {
                required: "Please provide a phone number.",
                minlength: "This phone number is invalid.",
                maxlength: "This phone number is invalid."
            }
        },

        submitHandler: function(form) {
            form.submit();
        }
    });

    });

</script>

<script type="text/javascript" src="assets/password_meter.js"></script>

```

```

<h1>Your Details.</h1><form method="post" action="<?
PHP echo htmlspecialchars($_SERVER['PHP_SELF']); ?>" autocomplete="on" id="forms">
<div id="group">
<div id="left">
    <label>Forename:</label><br>
    <input name="forename" required="required" type="text" placeholder="First" value="<?
php echo $prevalue['forename'];?>" />
</div>

<div id="right">
    <label>Surname:</label><br>
    <input name="surname" required="required" type="text" placeholder="Last" value="<?
php echo $prevalue['surname'];?>" />
</div>
</div>
<hr>

<div id="group">
<div id="left">
    <label>Password:</label><br>
    <input name="new_password" type="password" id="password" placeholder="Password"/><br>
    <label>Confirm Password:</label><br>
    <input name="new_password_confirm" type="password" placeholder="Password"/>
</div>
<div id="right">
    <label>Password Strength:</label><br>
    <span id="password_bar">
        <span id="password_str">No Data</span>
    </span>
</div>
</div>
<hr>
<div id="group">
<div id="left">
    <label>Email:</label><br>
    <input name="email" required="required" type="email" id="email" placeholder="example@domain.com" value="<?php echo $prevalue['email'];?>" /><br>
    <label for="phoneno" class="phone" data-icon="n" >Phone Number:</label><br>
    <input name="phoneno" required="required" maxlength="10" type="number" value="<?
php echo $prevalue['phoneno'];?>" placeholder="" />
</div>
<div id="right">
</div>
</div>
<div>
<hr>
<div id="group">
<div id="left">
    <label>Enter your current password to confirm changes:</label><br>
    <input name="current_password" required="required" type="password" id="current_password" placeholder="Password"/>
</div>
</div>
<hr>
<p class="signin button">
<input type="submit" id="next" value="Update"/>
</p>
</div>

</div>

<?php include 'includes/footer.php';
?>

```

forgot.php

```
<?php
$title = 'Forgot Password?';
session_start();
include_once("includes/core.php");
include_once("functions/encryption.php");
include("functions/email.php");

//show login
if (isset($_POST['email'])){
$query = "SELECT * FROM users WHERE email = :email";
$query_params = array(
    ':email' => $_POST['email']
);
$db->DoQuery($query, $query_params);
$userdetails = $db->fetchAll();
if ($userdetails) {
    $code = encrypt(rand(99341, 1102400));
    $query = "UPDATE users SET forgot_code = :code WHERE email = :email";
    $query_params = array(
        ':code' => $code,
        ':email' => $_POST['email']
    );
    $db->DoQuery($query, $query_params);
    $extra = array('code' => $code);
    email($_POST['email'], $userdetails[0]['username'], $userdetails[0]
['forename'], "forgotten_password", $extra);
}
header('Location: confirmation.php?type=forgot');
exit();
}
include('includes/header.php');
?>
```

<h1>Forgot your password?</h1>
You can reset your password by typing in the email associated with your account.

```
<?php if (isset($error)) { ?>
<div class="error"><?php echo $error; ?></div>
<?php } ?>
<form action="forgot.php" method="post" autocomplete="off">
<input type="text" name="email" placeholder="email" /><br>
<input type="submit" value="Next" id="submit"/>
</form>
```

</div>

index.php

```
<?php
$title = 'Home';
$require_user = true;
include_once('includes/core.php');
include 'includes/header.php';
?>
<div id="group">
<div id="left"><div class="bigbox"><a href="calendar.php">Book an Appointment</a></div></div>
<div id="right"><div class="bigbox"><a href="manage_appointments.php">Manage Appointments
</a></div></div>
</div>
<?php
include 'includes/footer.php';
?>
```

login.php

```
<?php
$title = 'Login';
Cand. Name: Thien Nguyen (7461)
```

```

$menutype = NULL;
include_once("includes/core.php");
include("functions/encryption.php");
if (isset($_GET['timeout'])) {
    array_push($errors, 'You were logged out automatically for being inactive.');
}
if (isset($_SESSION['logged_in'])) {
    header('Location: index.php');
} else {
if (isset($_POST['username']) && (isset($_POST['password']))) {
    $username = trim($_POST['username']);
    $password = encrypt(trim($_POST['password']));
    $query = "SELECT * FROM users WHERE username = :username AND password = :password";
    $query_params = array(
        ':username' => $username,
        ':password' => $password
    );
    $db->DoQuery($query, $query_params);
    $num = $db->fetch();
    if ($num) {
        if ($num['activated'] == 1) {
            if ($num['banned'] == 0) {
                $user_id = $num['id'];
                $forename = $num['forename'];
                $surname = $num['surname'];
                // //user entered correct details
                setcookie('userdata[loggedin]', TRUE, $timeout, '', '', TRUE);
                setcookie('userdata[user_id]', $user_id, $timeout, '', '', TRUE);
                setcookie('userdata[forename]', $forename, $timeout, '', '', TRUE);
                setcookie('userdata[surname]', $surname, $timeout, '', '', TRUE);
                header('Location: welcome.php');
                exit();
            }
            else {
                array_push($errors, "You have been banned.");
            }
        }
        else {
            array_push($errors, "You didn't activate your account!");
        }
    } else {
        array_push($errors, 'The username and password combination is not recognised,
try again!');
    }
}
include('includes/header.php');
?>

<div id="left">
    <h1>New here?</h1>
    <p>You'll need an account to book an appointment. Registration will open in a new window. You can come when you have registered!</p>
    <a href="register.php"><button type="register">Register</button></a>
</div>

<div id="right">
    <h1>Have an account?</h1>

    <form action="login.php" method="post" autocomplete="off">
        <input type="text" name="username" placeholder="Username" /><br>
        <input type="password" name="password" placeholder="Password" /><br>
        <input type="submit" value="Login" id="submit"/>
        <a href="forgot.php" class="login-link">Forgot your password?</a>
    </form>

```

</div>

</div>

<?php

}

?>

logout.php

<?php

//unset cookies

```
if (isset($_SERVER['HTTP_COOKIE'])) {
    $cookies = explode(';', $_SERVER['HTTP_COOKIE']);
    foreach($cookies as $cookie) {
        $parts = explode('=', $cookie);
        $name = trim($parts[0]);
        setcookie($name, '', time()-1000);
        setcookie($name, '', time()-1000, '/admin');
        setcookie($name, '', time()-1000, '/staff');
    }
}
```

header('Location: login.php');

?>

manage_appointments.php

<?php

```
$title = 'Manage Appointments';
$menutype = "user_dashboard";
$require_user = true;
include_once("includes/core.php");
$date = date("Y-m-d");
$query = "SELECT booking.id, booking.date, booking.time, booking.comments,
    service.type, service.price FROM booking
    INNER JOIN service ON booking.service_id=service.id
    WHERE booking.user_id = :user_id
    ORDER BY date ASC ";
$query_params = array(
    ':user_id' => $_COOKIE['userdata']['user_id']
);
```

if (isset(\$_GET['option'])) {

\$option = \$_GET['option'];

} else {

\$option = 'upcoming';

}

\$db->DoQuery(\$query, \$query_params);

\$results = \$db->fetchAll();

if (isset(\$_POST['id_delete'])) {

\$query = "DELETE FROM booking WHERE id = :id";

\$query_params = array(':id' => \$_POST['id_delete']);

\$db->DoQuery(\$query, \$query_params);

array_push(\$update, "Your appointment for is now cancelled.");

}

include 'includes/header.php';

?>

```
<select id="navigator" autofocus onchange="location = this.options[this.selectedIndex].value;">
    <option value="?option=upcoming" <?
        php if($option == "upcoming") {echo 'selected="selected"';}?>>Upcoming</option>
    <option value="?option=past" <?php if($option == "past") {echo 'selected="selected"';}?
        >>Past</option>
```

Scheduling & Booking System

```
<option value=?option=all" <?php if($option == "all") {echo 'selected="selected"';}?>
>>All</option>
</select>

<?php
if (isset($option)){
if ($option == "past"){
echo '<h1>Past Appointments.</h1>';
} elseif ($option == "upcoming"){
echo '<h1>Upcoming Appointments.</h1>';
} elseif ($option == "all"){
echo '<h1>All Appointments</h1>';
}}?>
<form method='post' action=''>
<div class="appointments">
<?php
if (isset($option)){
if ($option == "past"){
foreach ($results as $row) {
    $dtA = strtotime($row['date'] . " " . $row['time']);
    if ($dtA<time()) {
        echo "<div id='group'>";
        echo "<div class='left'>";
        echo '<b>' . date("D, d M Y", strtotime($row['date'])) . '</b><br>';
        echo date("g:i A", strtotime($row['time'])) . '<br>';
        echo $row['type'] . "<br><br>";
        echo $row['comments'] . "<br>";
        echo "</div><div class='right'>";
        echo '&pound;' . $row['price'] . '<br>';
        echo '</div></div>';
    }
}
}
elseif($option == "upcoming"){
foreach ($results as $row) {
    $dtA = strtotime($row['date'] . " " . $row['time']);
    if ($dtA>time()) {
        echo "<div id='group'>";
        echo "<div class='left'>";
        echo '<b>' . date("D, d M Y", strtotime($row['date'])) . '</b><br>';
        echo date("g:i A", strtotime($row['time'])) . '<br>';
        echo $row['type'] . "<br><br>";
        if (strlen($row['comments']) >= 1){
            echo "<h2>Your Comments</h2>";
            echo $row['comments'] . "<br>";
        }
        echo "</div><div class='right'>";
        echo '&pound;' . $row['price'] . '<br>';
        if ($dtA>time()){
            echo "<button type='submit' class='buttons' name='id_delete' value=
'" . $row['id'] . "'>Cancel</button>";
        }
        echo '</div></div>';
    }
}
}
elseif($option == "all"){
foreach ($results as $row) {
    $dtA = strtotime($row['date'] . " " . $row['time']);
    echo "<div id='group'>";
    echo "<div class='left'>";
    echo '<b>' . date("D, d M Y", strtotime($row['date'])) . '</b><br>';
    echo date("g:i A", strtotime($row['time'])) . '<br>';
    echo $row['type'] . "<br><br>";
    echo $row['comments'] . "<br>";
    echo "</div><div class='right'>";
```

Scheduling & Booking System

```
        echo '&pound;'.$row['price'].'<br>';
        if ($dtA>time()){
            echo "<button type='submit' class='buttons' name='id_delete' value=
'".$row['id']."'>Cancel</button>";
        }
        echo '</div></div>';
    }
}
?>
</div>
</form>

<div class="appointments">

<?php
//foreach ($results as $row) {
//    $dtA = strtotime($row['date'] . " " . $row['time']);
//    if ($dtA>time()) {
//        echo "<div id='booking'>";
//        echo '<tr>';
//        echo '<td>' . $row['date'] . '</td>';
//        echo '<td>' . $row['time'] . '</td>';
//        echo '<td>' . $row['type'] . '</td>';
//        echo '<td>' . $row['comments'] . '</td>';
//        echo '<td>&pound;' . $row['price'] . '</td>';
//        echo '</tr></div>';
//    }
?>
</div>

<br>
<a href="calendar.php"><button class="buttons" value="Make New Reservation">Make New Rese
rvation</button></a>

<?php
include 'includes/footer.php';
?>
```

profile.php

```
<?php
$title = 'Profile';
$require_user = true;
include_once("includes/core.php");
include("functions/encryption.php");

$date = date("Y-m-d");
$query = "SELECT * FROM users WHERE id = :id";
$query_params = array(
    ':id' => $_COOKIE['userdata']['user_id']
);
$db->DoQuery($query, $query_params);
$q = $db->fetch();

include 'includes/header.php';
?>

<h1>Your Information.</h1>

<div id="left">
<?php
    echo '<b>Username</b> ' . $q['username'] . '<br>';
    echo '<b>Name</b> ' . $q['forename'] . ' ' . $q['surname'] . '<br>';
```

```

echo '<b>Email</b>  '.$q['email'].'<br>';
echo '<b>Phone No.</b>  '.$q['phoneno'].'<br>';
?>
</div>
<div id="right">
<p align="right">
<a href="edit.php">Edit</a>
</p>
</div>
<?php
include 'includes/footer.php';
?>

```

register.php

```

<?php

$title = 'Register';

include_once("includes/core.php");

include_once("functions/encryption.php");

include_once("functions/email.php");

require_once('assets/recaptcha.php');

if($_POST) {

//sanitize variables

$username = strtolower(trim($_POST["username"]));

$username = filter_var($username, FILTER_SANITIZE_STRING, FILTER_FLAG_STRIP_LOW|
FILTER_FLAG_STRIP_HIGH);

$password = trim($_POST['password']);

$password_confirm = trim($_POST['password_confirm']);

$forename = trim(ucfirst($_POST['forename']));

$surname = trim(ucfirst($_POST['surname']));

$email = strtolower(trim($_POST['email']));

$email_confirm = strtolower(trim($_POST['email_confirm']));

$phoneno = trim($_POST['phoneno']);

$errors = array();

// check if username is available

$query = ("SELECT username FROM users WHERE username = :username");

$query_params = array(':username' => $username);

$db->DoQuery($query, $query_params);

```

Scheduling & Booking System

```
$rows = $db->fetch();

if ($rows) {
    array_push($errors, "This username is already chosen. Please choose another username.");
}

if (!filter_var($email, FILTER_VALIDATE_EMAIL))
array_push($errors, "Please specify a valid email address");

if (filter_var($email, FILTER_VALIDATE_EMAIL) != filter_var($email, FILTER_VALIDATE_EMAIL))
array_push($errors, "The email addresses do not match");

// check if email is available

$query = ("SELECT email FROM users WHERE email = :email");
$query_params = array(':email' => $email);
$db->DoQuery($query, $query_params);

$rows = $db->fetch();

if ($rows) {
    array_push($errors, "This email is already associated with an account. Please choose another email.");
}

if (!preg_match("/^(\+44\s?7\d{3}|\(07\d{3}\)\s?)\s?\d{3}\s?\d{3}$/, $phoneno)){
    array_push($errors, "Please specify a valid phone number.");
}

if (strlen($username) == 0){
    array_push($errors, "Please enter a username.");
} else {
    if (ctype_alnum($username) !== true){
        array_push($errors, "Please enter a valid username.");
    }
}

if (!preg_match("/^[a-zA-Z ]*$/", $forename)) {
    array_push($errors, "Your forename has invalid characters.");
}
```

```

if (!preg_match("/^[\a-zA-Z ]*$/", $surname)) {
    array_push($errors, "Your surname has invalid characters.");
}

if (strlen($password) < 5){
    array_push($errors, "Please enter a password. Passwords must contain at least 5 characters.");
}

if ($_POST["recaptcha_response_field"]) {
    $resp = recaptcha_check_answer ($privatekey, $_SERVER["REMOTE_ADDR"], $_POST["recaptcha_challenge_field"], $_POST["recaptcha_response_field"]);
    if (!$resp->is_valid) {
        array_push($errors, "The captcha is incorrect.");
    }
}

// If no errors were found, proceed with storing the user input
if (count($errors) == 0) {
    $password = encrypt($password);

    $email_parameters = array(
        ':pin' => encrypt($username)
    );
    email($email, $username, $forename, "confirm_registration", $email_parameters);

    $query = "INSERT INTO users (username, password, forename, surname, email, phoneno, activated, activation_code) VALUES (:username, :password, :forename, :surname, :email, :phoneno, :activated, :activation_code)";
    $query_params = array(
        ':username' => $username,
        ':password' => $password,
        ':forename' => $forename,
        ':surname' => $surname,
        ':email' => $email,
        ':phoneno' => $phoneno,
        ':activated' => '0',
    );
}

```

```

':activation_code' => encrypt($username)

);

$db->DoQuery($query, $query_params);

header("Location: confirmation.php?type=registration");

}

}

include('includes/header.php');

?>

<script type="text/javascript" src="assets/jquery.js"></script>

<script type="text/javascript" src="assets/jquery.validate.min.js"></script>

<script type="text/javascript" src="assets/password_meter.js"></script>

<script type="text/javascript">

$(document).ready(function() {

$('#username_availability').load('functions/check_username.php').show();

$('#username_').keyup(function() {

$.post('functions/check_username.php', { username: forms.username.value },

function(result) {

$('#username_availability').html(result).show();

});

});

});

// When the browser is ready...

$(function() {

jQuery.validator.addMethod("lettersonly", function(value, element) {

return this.optional(element) || /^[a-zA-Z0-9]+$/i.test(value);

}, "Letters only please");



// Setup form validation on the #register-form element

$("#forms").validate({

```

Scheduling & Booking System

```
// Specify the validation rules
rules: {

    forename: "required",
    forename: "required",
    phoneno: {
        required: true,
        minlength: 11,
        maxlength: 11
    },
    email: {
        required: true,
        email: true
    },
    username: {
        required: true,
        lettersonly: "/([A-Za-z0-9])\w+/"
    },
    password: {
        required: true,
        minlength: 6
    },
    email_confirm: {
        required: true,
        email: true,
        equalTo: "#email"
    },
    password_confirm: {
        required: true,
        equalTo: "#password"
    }
},
// Specify the validation error messages
```

```

messages: {

    forename: "Please enter your forename.",

    surname: "Please enter your surname.",

    email: "Please enter a valid email address.",

    username: {

        required: "Please enter a username.",

        lettersonly: "Usernames can only have letters or numbers."
    },

    password: {

        required: "Please provide a password.",

        minlength: "Your password must be at least six characters long."
    },

    phoneno: {

        required: "Please provide a phone number.",

        minlength: "This phone number is too short.",

        maxlength: "This phone number is too long."
    }
}

submitHandler: function(form) {

    form.submit();
}
});

});

</script>

```

```

<h1>Register</h1>

<form method="post" action="<?PHP echo htmlspecialchars($_SERVER['PHP_SELF']); ?>" autocomplete="on" id="forms">

<div id="group">

<div id="left">

```

Scheduling & Booking System

```
<label>Forename:</label><br>

<input name="forename" required="required" type="text" placeholder="First" />

</div>

<div id="right">

<label>Surname:</label><br>

<input name="surname" required="required" type="text" placeholder="Last" />

</div>

</div>

<hr>

<div id="group">

<div id="left">

<label>Username:</label><br>

<input id="username_" name="username" required="required" type="text" placeholder="User name"/>

</div>

<div id="right">

<div id="username_availability"></div>

</div>

</div>

<div id="hr_invisible"></div>

<div id="group">

<div id="left">

<label>Password:</label><br>

<input name="password" required="required" type="password" id="password" placeholder="Password"/><br>

<label>Confirm Password:</label><br>

<input name="password_confirm" required="required" type="password" placeholder="Password"/>

</div>

<div id="right">

<label>Password Strength:</label><br>

<span id="password_bar">

<span id="password_str">No Data</span>

Cand. Name: Thien Nguyen (7461)           Centre No. 12290
```

```

        </span>
    </div>
</div>
<hr>
<div id="group">
    <div id="left">
        <label>Email:</label><br>
        <input name="email" required="required" type="email" id="email" placeholder="example@domain.com"/><br>
        <label>Confirm Email:</label><br>
        <input name="email_confirm" required="required" type="email" placeholder="example@domain.com"/> <br>
        <label for="phoneno" class="phone" data-icon="n" >Phone Number:</label><br>
        <input name="phoneno" required="required" maxlength="10" type="number" placeholder="" />
    </div>
    <div id="right">
    </div>
</div>
<div>
<hr>
<label>Captcha</label>
<div class="captcha">
    <?php echo recaptcha_get_html($publickey, $error); ?>
</div>
<hr>
<?php if (isset($error)) { ?>
    <div class="error"><?php echo $error; ?></div>
<?php } ?>
<?php if($_POST) {
    #echo $output;
}

```

```
?>

<p class="signin button">
<input type="submit" id="next" value="Next"/>
</p>
</form>

</div>

<?php include 'includes/footer.php';?>
```

setup.php

```
<?php
$title = "Setup";
require("includes/core.php");
require("functions/encryption.php");

if (isset($_POST['username'])) {

    $username = trim($_POST['username']);
    $password = trim($_POST['password']);
    $password_again = trim($_POST['password_again']);
    $email = trim($_POST['email']);

    if (!filter_var($email, FILTER_VALIDATE_EMAIL))
        array_push($errors, "Please specify a valid email address");

    if (strlen($username) == 0){
        array_push($errors, "Please enter a username.");
    } else {
        if (ctype_alnum($username) !== true){
            array_push($errors, "Please enter a valid username.");
        }
    }

    if (strlen($password) < 5)
        array_push($errors, "Please enter a password. Passwords must contain at least 5 characters.");

    if (isset($password_again)) {
        if ($password !== $password_again)
            array_push($errors, "The passwords do not match.");
    } else {
        array_push($errors, "Please repeat your password.");
    }

// If no errors were found, proceed with storing the user input
if (count($errors) == 0) {
    $password = encrypt($password);
    $create_table = array(
        "CREATE TABLE `admin` (
        `id` int(11) unsigned NOT NULL AUTO_INCREMENT,
        `username` varchar(255) DEFAULT NULL,
        `password` varchar(255) DEFAULT NULL,
        `email` varchar(255) DEFAULT NULL,
        PRIMARY KEY (`id`)
    ) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=latin1;",
}
```

```

        "CREATE TABLE `closed_days` (
`id` int(11) unsigned NOT NULL AUTO_INCREMENT,
`date` date DEFAULT NULL,
PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=13 DEFAULT CHARSET=latin1;",
"CREATE TABLE `metadata` (
`id` int(11) unsigned NOT NULL AUTO_INCREMENT,
`option` varchar(255) DEFAULT '',
`value` varchar(255) DEFAULT NULL,
`description` text,
`rule` varchar(255) DEFAULT NULL,
PRIMARY KEY (`id`)
) ENGINE=MyISAM AUTO_INCREMENT=9 DEFAULT CHARSET=utf8;",
        "CREATE TABLE `service` (
`id` int(11) unsigned NOT NULL AUTO_INCREMENT,
`type` varchar(25) DEFAULT NULL,
`price` int(5) DEFAULT NULL,
`description` text,
PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=8 DEFAULT CHARSET=latin1;",
        "CREATE TABLE `staff` (
`id` int(11) unsigned NOT NULL AUTO_INCREMENT,
`pin` varchar(128) DEFAULT NULL,
`s_forename` varchar(25) DEFAULT NULL,
`s_surname` varchar(25) DEFAULT NULL,
`s_email` varchar(100) DEFAULT NULL,
`banned` tinyint(1) DEFAULT NULL,
PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=utf8;",
        "CREATE TABLE `users` (
`id` int(11) unsigned NOT NULL AUTO_INCREMENT,
`username` varchar(55) DEFAULT NULL,
`password` varchar(255) DEFAULT NULL,
`forename` char(55) DEFAULT NULL,
`surname` char(55) DEFAULT NULL,
`email` varchar(255) DEFAULT NULL,
`phoneno` varchar(11) DEFAULT NULL,
`activated` tinyint(1) DEFAULT NULL,
`banned` tinyint(1) DEFAULT NULL,
`activation_code` varchar(128) DEFAULT NULL,
`forgot_code` varchar(128) DEFAULT NULL,
PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=6 DEFAULT CHARSET=latin1",

"CREATE TABLE `booking` (
`id` int(100) NOT NULL AUTO_INCREMENT,
`date` date NOT NULL,
`time` time NOT NULL,
`user_id` int(11) unsigned NOT NULL,
`comments` text COLLATE utf8_unicode_ci NOT NULL,
`confirmedbystaff` tinyint(1) DEFAULT NULL,
`service_id` int(11) unsigned DEFAULT NULL,
`staff_id` int(11) unsigned DEFAULT NULL,
PRIMARY KEY (`id`),
KEY `service_id` (`service_id`),
KEY `booking_ibfk_2` (`staff_id`),
KEY `idx_user_id` (`user_id`) USING BTREE,
CONSTRAINT `booking_ibfk_3` FOREIGN KEY (`user_id`) REFERENCES `users` (`id`),
CONSTRAINT `booking_ibfk_1` FOREIGN KEY (`service_id`) REFERENCES `service` (`id`),
CONSTRAINT `booking_ibfk_2` FOREIGN KEY (`staff_id`) REFERENCES `staff` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=15 DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;",
);

for ($x = 0; $x <= 6; $x++) {
    switch ($x) {

```

```

        case 0:
            $table_name = "admin";
            break;
        case 1:
            $table_name = "closed_days";
            break;
        case 2:
            $table_name = "metadata";
            break;
        case 3:
            $table_name = "service";
            break;
        case 4:
            $table_name = "staff";
            break;
        case 5:
            $table_name = "users";
            break;
        case 6:
            $table_name = "booking";
            break;
    }
    $check_if_exists = "SHOW TABLES LIKE '$table_name'";
    $db->doQuery($check_if_exists);
    $fetch = $db->fetch();
    if (!$fetch) {
        $db->doQuery($create_table[$x]);
    }
}

$insert_base_queries = array(
    "INSERT INTO `metadata` (`id`, `option`, `value`, `description`, `rule`) VALUES
    (1, 'opening_time', '10:00', 'The time of the first slot', 'Opening Time'),
    (2, 'closing_time', '19:30', 'The time of the last slot', 'Closing Time'),
    (3, 'booking_frequency', '30', 'The amount of slots per hour, expressed in minutes.
    ', 'Booking Frequency'),
    (5, 'business_name', 'Nails Club', 'The name of the business.', 'Business Name'),
    (6, 'business_slogan', 'is it a mad ting', 'The slogan of the business.', 'Business
    Slogan');
    ",
    "INSERT INTO `admin` (`username`, `password`, `email`) VALUES ('$username', '$passw
ord', '$email');",
    "INSERT INTO `service` (`id`, `type`, `price`, `description`)
VALUES
    (1, 'Manicure', 25, 'A luxurious beauty treatment for the fingernails and hands which i
ncludes perfectly-polished nails.'),
    (2, 'Infil', 26, 'A infill.'),
    (3, 'Pedicure', 29, 'A pedicure.'),
    (4, 'Full Set', 48, 'Manicure & Pedicure.'),
    (5, 'File & Polish', 15, 'A File & Polish'),
    (6, 'Nail Repair', 10, 'A Nail Repair.')
",
    "INSERT INTO `staff` (`id`, `pin`, `s_forename`, `s_surname`, `s_email`)
VALUES
    (1, NULL, NULL, NULL, NULL);
"
);

foreach ($insert_base_queries as $query) {
    $db->doQuery($query);
}
header('Location: setup.php?done');
}
}

array_push($update, "<h1>Warning</
h1>Please delete this file (setup.php) when you are finished with this page. This is nece
Cand. Name: Thien Nguyen (7461)                               Centre No. 12290
");

```

Scheduling & Booking System

```
ssary to improve the security of this program. Also, the program will not work until the
setup is completed.");
include_once("includes/header.php");
?>

<?php
if(isset($_GET['done'])){
    echo "<h1>All Ready to go!</h1>";
    echo "<h2>Setup is now complete!</h2>";
    echo "Customers will be able to access your booking system at <a href='http://".
$_SERVER['SERVER_NAME']."'>http://" . $_SERVER['SERVER_NAME'] . "</a>.<br>";
    echo "Staff will be able to access checkins at <a href='http://".
$_SERVER['SERVER_NAME'].'/staff'>http://" . $_SERVER['SERVER_NAME'] . "/staff" . "</
a>.<br>";
    echo "You, the Administrator, will be able to access settings at <a href='http://".
$_SERVER['SERVER_NAME'].'/admin'>http://" . $_SERVER['SERVER_NAME'] . "/admin" . "</
a>.<br>";
    echo "It is now safe to delete this (setup.php) file.";
} else {
?>
<h1>Setup Administrator's Account</h1>
```

You'll need to create a account to get into the Administrators Dashboard.

```
<form action="" method="post" autocomplete="off">
<input type="text" name="username" placeholder="Username"/>
<input type="password" name="password" placeholder="Password"/>
<input type="password" name="password_again" placeholder="Password Again"/>
<input type="email" name="email" placeholder="Email">
<button type="submit">Submit</button>
</form>
<?php } ;?>
```

welcome.php

```
<?php
$title = 'Welcome';
if (!$COOKIE['userdata']['loggedin'] == 1) {
    header('Location: login.php');
} else {
    header("refresh: 100; url=index.php");
};
include_once('includes/core.php');
echo '<link rel="stylesheet" href="'.$directory.'assets/style.css'.'><br><br><br><br>';
echo '<div id="container"><center><p>';
echo "Welcome <b>" . $COOKIE['userdata']['forename'] . "</b>." ;
echo "<br>Please wait while we redirect you to the menu." ;
echo "<br>Alternatively you can click <a href='index.php'>here</
a> if you're not being redirected." ;
echo '</p></center></div>';
?>
```

assets/style.css

```
/* @override
   http://localhost/Applications/MAMP/htdocs/booker/real_system/assets/style.css
   http://localhost/assets/style.css
   http://projects.tnguyen.ch/comp4/assets/style.css
   http://projects.tnguyen.ch/assets/style.css */
@import url(http://fonts.googleapis.com/css?family=Montserrat:400,700|Open+Sans:
400italic,400,600);

body {
    font-family: "Helvetica Neue", Helvetica, Arial, Geneva, sans-serif;
    background-color: #d3d5dd;
    overflow-y: scroll;
    overflow-x: hidden;
}
```

```
#container {
    max-width: 800px;
    margin-right: auto;
    margin-left: auto;
    background-color: #ffffff;
    display: block;
    overflow: hidden;
    padding: 50px 2em;
    margin-bottom: 50px;
}

@media (min-width: 800px) {

    #container {
        width: 768px;
        margin-left: auto;
        margin-right: auto;
    }

    .heading {
        font-size: 2em;
    }
}

/* @group nav */

header {
    background: #feffff;
    font-family: 'Open Sans';
}

header ul {
    list-style: none;
    margin: 0 auto;
    padding-right: 0;
    padding-bottom: 0;
    padding-left: 0;
    overflow: hidden;
    text-align: center;
}

header ul li {
    display: inline-block;
    margin-right: 10px;
}

header ul a {
    text-transform: uppercase;
    color: #000;
    font-size: 14px;
    padding: 10px 15px;
    line-height: 18px;
    text-decoration: none;
    border-radius: 5px;
    display: block;
}

header ul a:hover {
    background: gray;
    color: #fff;
}

header ul a:active, ul a.active {
    color: #fff;
    background-color: #434443;
```

```
}

/* @end */

#heading, #footer {
    width: 90%;
    max-width: 800px;
    margin-right: auto;
    margin-left: auto;
    height: 100px;
    padding-top: 40px;
}
/* .img-replace {
    /* replace text with an image */
    display: inline-block;
    overflow: hidden;
    text-indent: 100%;
    color: transparent;
    white-space: nowrap;
} */

#branding {
    font-size: 2em;
    float: left;
    font-weight: bold;
}

#slogan{
    font-size: 12px;
    font-style: normal;
    width: 160px;
    font-weight: normal;
    white-space: normal;
}

#headingright {
    text-align: right;
    overflow: hidden;
    font-size: 0.9em;
    margin-top: 15px;
}

h1, h2, h3, h4, h5, h6 {
    font-family: Montserrat, Helvetica, Arial, sans-serif;
}

#left {
    float: left;
    width: 45%;
    display: inline-block;
    height: auto;
}

#right {
    float: right;
    height: auto;
    display: inline-block;
    width: 45%;
}

#group {
    background-color: #f0f0f2;
    width: 100%;
    display: inline-block;
    margin-bottom: 1em;
    padding: 0.0000012em;
```

```
}

#forms #group {
    background-color: #ffffff;
}

.accordion h2 {
    display: inline-block;
}

#group .left, .accordion #group #left {
    float: left;
    width: 40%;
    height: auto;
    padding: 1px;
}

#group .right, .accordion #group #right {
    float: right;
    height: auto;
    width: 40%;
    padding: 1px;
    text-align: right;
}

a {
    color: #444;
    text-decoration: none;
}

a:hover {
    transition: 0.25s;
    color: #3585bb;
}

.vertical-line {
    border-left: thick solid #000;
    height: 100px;
    width: 1px;
    right: auto;
    left: auto;
}

canvas {
    background-color: #fcfcfc;
    padding-top: 1em;
    padding-bottom: 1em;
    border: 1px solid #d2d3d2;
}

p a{
    color: #0c60a5;
}

hr {
    border-top-style: solid;
    border-top-color: #9b9b9b;
    border-bottom-style: hidden;
    border-right-style: hidden;
    border-left-style: hidden;
    margin-top: 2em;
    margin-bottom: 2em;
}

#hr_invisible {
    border-bottom-style: hidden;
    border-right-style: hidden;
    border-left-style: hidden;
    border-style: solid;
```

Scheduling & Booking System

```
border-color: #ffffff;
margin-top: 1em;
}

label {
    display: block;
}

.bigbox a {
    background-color: #b1b4bf;
    display: block;
    width: 100%;
    height: 150px;
    text-align: center;
    padding-top: 5em;
    margin-top: 3em;
    margin-bottom: 3px;
}

.bigbox a:hover {
    background-color: #7b8093;
    color: #ffffff;
}

table#statistics{
    width: 100%;
    font-size: 0.9em;
}

#statistics tr:nth-child(even) {background: #f4f4f4}
}
#statistics tr:nth-child(odd) {background: #FFF}
table#statistics, #statistics th, #statistics td {
    border: 1px solid #d3d3d3;
    border-collapse: collapse;
}
#statistics th,#statistics td {
    padding: 10px;
}
/* @group Accordion */

.accordion {
    list-style: none;
    width: 100%;
}

.accordion li {
    position: relative;
    right: 2.5em;
}

.accordion #topbox {
    margin: 1px;
    cursor: pointer;
}

.accordion .base {
    background-color: #424342;
    height: 50px;
}

.accordion .base p {
    padding-top: 1em;
    padding-bottom: 1em;
    margin-top: 0.2em;
    padding-left: 35px;
```

Scheduling & Booking System

```
color: #ffffff;
font-weight: bold;
}

.accordion #indicator {
    width: 20px;
    height: 50px;
    float: left;
    display: inline-block;
}

.accordion #indicator.checkedin {
    background-color: #b6efa7;
    display: inline-block;
}

.accordion #indicator.notcheckedin {
    background-color: #626262;
    display: inline-block;
}

.accordion #indicator.missing {
    background-color: #db8a8a;
    display: inline-block;
}

.accordion :focus, .accordion :focus p {
    background-color: #fdefb0;
    color: #393939;
}

.accordion ul li {
    font-weight: normal;
    cursor: auto;
    background-color: #f3f3f3;
    padding: 0 0 0 7px;
    display: table-cell;
}
/* @end */

/* @group Login */

input[type=text], input[type=password], textarea {
    background-color: #ECF0F1;
    transition: border .5s;
}

input[type=text]:focus, input[type=password]:focus, textarea:focus {
    border: 1px solid #335ea6;
}

#submit {
    background-color: #303c49;
    color: #ffffff;
}

#submit:hover {
    background-color: #335ea6;
}

.login-container {
    text-align: center;
}
/* @end */

/* @group Inputs */
```

Scheduling & Booking System

```
button, input, .credentials, textarea, #select {
    border: 1px solid transparent;
    line-height: 25px;
    transition: 0.25s;
    font-size: 1em;
    padding: 5px;
    -webkit-appearance: none;
}

input, textarea {
    border: 1px solid rgba(0, 0, 0, 0.09);
    background-color: #f4f4f4;
    color: #000000;
}

#forms input {
    width: 220px;
}

select#navigator {
    margin-top: 0.5em;
    margin-bottom: 0.5em;
}

form#date.select {
    text-align: center;
}

#date.select select {
    margin-top: 0.5em;
    margin-bottom: 0.5em;
    display: inline-block;
    width: 27%;
}

input#price {
    width: 90px;
}

#select, select {
    width: 100%;
    -webkit-appearance: button;
    -webkit-border-radius: 1px;
    color: #8d8d8d;
    font-size: inherit;
    padding: 10px;
    background-color: #f3f4f3;
    border: 1px solid #d4d4d4;
}

button:hover {
    background-color: #335ea6;
    color: #ffffff;
}

textarea#description {
    height: 37px;
    width: 250px;
}

textarea#description:focus {
    height: 7em;
}

#forms input#next, input#next_disabled {
```

Scheduling & Booking System

```
border: 1px solid transparent;
background-color: #dac3c3;
float: right;
width: 100px;
height: 50px;
}

#forms #verify {
background-color: #d33232;
float: right;
height: 100px;
width: 400px;
}

#forms input#next:hover {
background-color: #9b2828;
cursor: pointer;
}

#forms input#next_disabled {
background-color: #dfdfdf;
}

@media screen and (max-width: 480px) {
#forms .left,
#left,
#group .left, #group .right,
#forms .right, #right {
float: none;
width: 100%;
}
}

#forms .group:after {
content: "";
display: table;
clear: both;
}

#forms img {
max-width: 100%;
height: auto;
}

/* @group Register */
#password_bar {
background-color: #d6d6d6;
display: block;
width: 276px;
padding: 2px;
}

#password_str {
border: 1px solid transparent;
line-height: 25px;
transition: 0.25s;
/*pseudo strength bar*/
color: #000000;
font-size: 1em;
background-color: #d5d6d5;
display: block;
border-top-style: none;
text-align: center;
padding-top: 6px;
padding-bottom: 6px;
height: 24px;
```

Scheduling & Booking System

```
width: 80px;
}

#password_str.short {
    color: #ffffff;
    background-color: #fe7c7c;
    width: 30%;
}

#password_str.weak {
    color: #383838;
    background-color: #ffd1d1;
    width: 40%;
}

#password_str.good {
    color: #ffffff;
    background-color: #79d300;
    width: 70%;
}

#password_str.strong {
    color: #ffffff;
    background-color: #5dab1e;
    width: 90%;
}

/* @end */

/* @group Captcha */
#recaptcha_area, #recaptcha_table {
    line-height: 0 !important;
}

/* @end */

/* @end */

/* @group Checkin */

#checkin_topbar {
    width: 100%;
    position: fixed;
    top: 0;
    right: 0;
    background-color: #383838;
    height: 150px;
    color: #ffffff;
    z-index: 25;
}

#checkin_topbar .group {
    padding: 20px 10px;
    height: 100px;
}

#checkin_topbar .group .left {
    float: left;
    line-height: 19px;
}

#checkin_topbar .group .right {
    float: right;
    position: relative;
    top: -20px;
}
```

Scheduling & Booking System

```
#checkin_topbar form {
    margin-left: auto;
    margin-right: auto;
    text-align: center;
}

#checkin_topbar input#username_ {
    text-align: center;
    margin-top: -50px;
    width: 80%;
}

.blank {
    height: 160px;
    width: 100%;
}

#checkin_comments {
    background-color: #c9c9c9;
    padding: 10px;
}

.accordion button {
    background-color: #d5d5d5;
}
/* @end */

/* @group Calendar */

#calendar {
    margin-bottom: 1em;
    margin-top: 1em;
    width: 100%;
    border: 0px;
    padding: 0px;
    border-spacing: 0px;
}
#calendar th {
    background-color: #313231;
    font-size: 12px;
    color: #FFFFFF;
    text-align: center;
    width: 14.285%;
    height: 30px;
}

#calendar #week a {
    font-size: 32px;
    color: #323232;
    padding: 0 14px 0 14px;
}
#week .buttons a{
    font-size: 32px;
    width: 100px;
}
#week .buttons{
    font-size: 32px;
    width: 10%;
    display: inline-block;
    float: left;
}
#week a.invisible{
    visibility: hidden;
}
#center_date {
```

```
margin: 10px auto;
text-align: center;
font: bold 16px Montserrat, Helvetica, Serif;
float: left;
width: 80%;

}

.days {
/*    text-decoration: none;
color: #FFFFFF;*/
font-size: 0;
border: 1px solid #313131;
margin: 0;
position: relative;
}

.box {
height: 60px;
}

.days p {
font-size: 12px;
text-align: right;
padding-top: 32px;
margin-right: 10px;
margin-top: 0;
}

.keylist ul {

}
.keylist{
display: none;
position: absolute;
z-index: 100;
background-color: #ffffff;
padding-top: 40px;
padding-bottom: 40px;
right: 20%;
left: 20%;
-webkit-box-shadow: -2px 3px 30px 0px rgba(0,0,0,0.5);
-moz-box-shadow: -2px 3px 30px 0px rgba(0,0,0,0.5);
box-shadow: -2px 3px 30px 0px rgba(0,0,0,0.5);
}
.keylist h3{
position: relative;
left: -20px;
}
.keylist li{

position: relative;
right: 20px;
display: inline-block;
font-weight: normal;
list-style-type: none;
padding: 10px;
}
.keyguide i{
display: inline-block;
height: 25px;
width: 25px;
line-height: 25px;
text-align: center;
font-weight: normal;
font-style: normal;
}
```

Scheduling & Booking System

```
background-color: #c7c5c5;
-moz-border-radius: 30px;
border-radius: 30px;

}

.keyguide i:hover {
    background-color: #c86c6c;
}
.keyguide:hover .keylist{
display : block;

}

#key {
    width: 100%;
    background-color: #f2f2f2;
    font-size: 0.8em;
    margin-top: 2em;
    margin-bottom: 2em;
    padding: 0;
}
#key ul, #key li {
    list-style-type: none;
}
#key li{
    width: 100%;
}
#key_fullybooked {
    background-color: #d34747;
    color: #ffffff;
}

#key_partbooked {
    background-color: #ffd364;
}
#key_partbooked:hover {
    background-color: #ffe53d;
    color: #000000;
}

#key_available {
    background-color: #a7d18d;
    color: #000000;
}
#key_available:hover {
    background-color: #d9fbcc;
    color: #000000;
}

#key_unavailable {
    background-color: #555555;
    color: #ffffff;
}

#key_null {
    background-color: #313131;
    color: #ffffff;
}

/* @end */
```

Scheduling & Booking System

```
#calendar_form textarea {
    width: 100%;
}

.error, .update, .status {
    line-height: 25px;
    transition: 0.25s;
    text-align: center;
    margin-top: 5px;
    margin-bottom: 5px;
    padding: 1em;
}

.error {
    background: #d44848;
    color: #ffffff;
}

.update {
    background: #ffbf24;
    color: #000000;
}

.status {
    background-color: #3a6da4;
    color: #ffffff;
}

/* @group Pagination */
ul#pagination {
    list-style: none;
    text-align: center;
    padding-top: 1em;
    padding-right: 0;
    padding-left: 0;
}

#pagination li {
    display: inline-block;
    height: 25px;
    width: 25px;
    line-height: 25px;
    text-align: center;
}

#pagination #active, #pagination li:hover {
    -moz-border-radius: 30px;
    /* or 50% */
    border-radius: 30px;
    /* or 50% */
    text-align: center;
    position: relative;
}

#pagination li:hover {
    background-color: #fce7b3;
}

#pagination li#active {
    background-color: #f6bf3e;
}

#pagination li#active:hover, #pagination li#active a:hover {
    background-color: #cfccc4;
    color: #ffffff;
}

/* @end */
```

assets/password_meter.js

```

$(document).ready(function()
{
    $('#password').keyup(function()
    {
        $('#password_str').html(password_validator($('#password').val()))
    })

    function password_validator(password){

        var strength = 0

        if (password.length < 5) {
            $('#password_str').removeClass()
            $('#password_str').addClass('short')
            return 'Too short!'
        }

        if (password.length > 7) strength += 1

        if (password.match(/([a-z].*[A-Z])|([A-Z].*[a-z])/)) strength += 1

        if (password.match(/([a-zA-Z])/) && password.match(/([0-9])/)) strength += 1

        if (password.match(/([!,%,&,@,#,$,^,*_,~,~])/)) strength += 1

        if (password.match(/(.*[!,%,&,@,#,$,^,*_,~,~].*[!,%,&,@,#,$,^,*_,~,~])/)) strength += 1

        if (strength < 2){
            $('#password_str').removeClass()
            $('#password_str').addClass('weak')
            return 'Weak'
        } else if (strength == 2){
            $('#password_str').removeClass()
            $('#password_str').addClass('good')
            return 'Good'
        } else
        {
            $('#password_str').removeClass()
            $('#password_str').addClass('strong')
            return 'Strong'
        }
    });
}
);

```

References

Encryption

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

Regular Expressions

<http://www.regular-expressions.info>

PHP Manual

<http://php.net/manual/en/>

Calendar

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

JQuery Validation Library

<http://jqueryvalidation.org/documentation/>

JQuery Library

<https://jquery.com/>

Google Recaptcha

<https://www.google.com/recaptcha/admin#mailhide>