



**University
of Dundee**

Development of an Online Reservation System and Food Delivery Service for a restaurant chain

Author: Ricky Shek

Supervisor: Dr Keith Edwards

MSc Applied Computing
School of Science and Engineering
University of Dundee

September 2017



Computing
at the University of Dundee

Executive Summary

It can now be considered standard for a company to have an online presence as a method of communicating and engaging with customers.

The main aim of this project was to create an online website for a fictional catering establishment named Restaurant X which included a booking system and online food ordering service. The website would allow customers to make a table reservation at one of four locations in Scotland – Aberdeen, Dundee, Edinburgh and Glasgow. The online food delivery service will offer the customer the ability to order a range of dishes and beverages offered by Restaurant X and have them delivered to their homes.

Programming languages used included HTML, CSS, Javascript and Python. The Django framework was adopted due to its numerous built in features including administration system and security.

Prototypes were created – both low fidelity using hand drawings and high fidelity using Axure. Evaluations and feedback were sought using the high fidelity prototype and changes were made to the design accordingly.

Testing was largely conducted using the NASA Task Load Index (NASA TLX) to determine the workload index score of conducting various tasks of the website, once it was fully functional.

Final evaluation concluded that the majority of initial goals were met. The website and its associated database system was successfully implemented, however there is room for future developments.

Declaration

"I declare that the special study described in this dissertation has been carried out and the dissertation composed by me, and that the dissertation has not been accepted in fulfilment of the requirements of any other degree or professional qualification."

Signed:

Ricky Shek

Certificate

"I certify that (your full name) has satisfied the conditions of the Ordinance and Regulations and is qualified to submit this dissertation in application for the degree of Master of Science."

Signed:

Dr Keith Edwards

Acknowledgements

I would like to thank Dr Keith Edwards for his support, guidance, and encouragement throughout the course of this MSc Project.

I would also like to thank the participants who agreed to take part during testing and providing valuable feedback.

And finally, I would like to thank my colleagues on the MSc Applied Computing course, friends and family, all of whom have been supportive throughout the whole process.

Contents

Chapter One - Introduction	1
Chapter Two - Background.....	2
2.1 Online Food Order and Delivery	2
2.2 Online Table Reservation	3
Chapter Three – Requirements Specification	4
3.1 Requirements.....	4
3.2 Use Cases	5
3.3 Use Case Specifications.....	5
3.4 Project Management	5
Chapter Four – Design (Early Stages).....	6
4.1 Prototyping	6
4.2 Low Fidelity – Sketches	6
4.3 High Fidelity – Axure	8
4.4 Testing of High Fidelity Prototype	9
4.4.1 Participants	9
4.4.2 Results.....	9
4.4.2.1 Colour Scheme	9
4.4.2.2 Font Styles.....	9
4.4.2.3 Font Colour	9
4.4.2.4 Adverts.....	10
4.4.2.5 Highlighting the current page.....	10
4.4.2.6 Separate Login Button	10
Chapter Five – Development and Implementation	11
5.1 Software.....	11
5.1.1 Visual Studio 2015	11
5.1.2 Django Framework.....	11
5.2 Security	13
5.2.1 SQL Injections.....	13
5.2.2 Cross-Site Scripting (XSS)	13
5.2.3 Cross-Site Request Forgery (CSRF).....	13
5.2.4 Password Hashing	13
5.3 Built-in Features.....	14
5.3.1 SQLite3 Database	14
5.3.2 Forms	14
5.3.3 Admin Site.....	15
5.4 Mobile First Approach.....	17
5.5 Functionality and Interfaces	19
5.5.1 HTML Base Template	19
5.5.2 Authentication Required Areas.....	19
5.5.2.1 User Registration	20
5.5.2.2 User Login/Logout.....	20
5.5.2.3 Online Ordering System.....	21
5.5.2.4 Checkout	23

5.5.3 Non-Authentication Required Areas.....	24
5.5.3.1 Contact Form	24
5.5.3.2 Menus	25
5.5.3.3 Book a Table.....	26
5.5.4 Administrator Access	27
Chapter Six – Evaluation	29
6.1 Tasks.....	29
6.2 NASA TLX.....	30
6.3 Additional Feedback	30
Chapter Seven – Appraisal	31
7.1 Design Decisions	31
7.1.1 Authentication of User.....	31
7.1.2 User Profile	31
7.1.3 Google Maps API.....	32
7.1.4 Calendar Date Picker.....	32
7.2 Changes to initial designs and requirements.....	33
7.2.1 Use Case Specifications.....	33
7.2.2 Online Order System.....	33
7.2.3 Highlighting the current page	33
7.3 Wider Context	34
7.4 Self-Appraisal	34
Chapter Eight – Conclusion	35
Chapter Nine – Recommendations for Future Work	36
References	37
Appendices	38

List of Figures

Figure 1 - Project Schedule	5
Figure 2 - Sketch of Home Page	6
Figure 3 - Sketch of “Book a Table” page	7
Figure 4 - Sketch of the “Order Online” page	7
Figure 5 - Axure RP 8 – Home Page Prototype	8
Figure 6 - Axure Prototype in a browser	8
Figure 7 - Visual Studio 2015 Interface	11
Figure 8 - Django’s MVC Structure	12
Figure 9 - Django’s Form with CSRF protection	13
Figure 10 - Django’s Form Class	14
Figure 11 - Django’s ModelForm Class: Python Code to create form	14
Figure 12 - Python & HTML Code to display the form on website	15
Figure 13 - User’s view of result of coding from Figures 11 & 12	15
Figure 14 - Admin Site Dashboard	15
Figure 15 - Syntax to display certain fields and filter criteria	16
Figure 16 - Admin Site view of displaying fields and filter criteria	16
Figure 17 - CSS @media rule code excerpt.....	17
Figure 18 - Desktop version of Restaurant X website.....	18
Figure 19 - Example of Mobile version of Restaurant X website.....	18
Figure 20 - Login Portal & link to Registration page	20
Figure 21 - Registration form with error message.....	20
Figure 22 - User Log out option	20
Figure 23 - Authentication Status	21
Figure 24 - User Login or Registration required message.....	21
Figure 25 - List of products and “Add to cart” button	22
Figure 26 - Cart with items.....	22
Figure 27 - Checkout	23
Figure 28 - Admin Site displaying order from Figure 27	23
Figure 29 - Function to receive completed Contact form and email it to administrator	24
Figure 30 - Contact Form	24
Figure 31 - Example of Contact Form sent to email	24
Figure 32 - Menus page Python code	25
Figure 33 - ModelForm class for the Booking Form.....	26
Figure 34 - Function to display & save Booking Form.....	26
Figure 35 - Booking confirmation message & details of booking	26
Figure 36 - Admin Site login portal	27
Figure 37 - Example code excerpt for Admin CRUD operations	28

Figure 38 - Result of figure 37's code excerpt	28
Figure 39 - Graph of average scores of each task for all participants.....	30
Figure 40 - Non-Authenticated User Navigation Bar View	31
Figure 41 - Authenticated User Navigation Bar View	31
Figure 42 - Example of Profile view	31
Figure 43 - JavaScript code to implement Google Maps	32
Figure 44 - Result of JavaScript in Figure 43	32
Figure 45 - JQuery Calendar Date Picker.....	32
Figure 46 - Navigation bar menu code to highlight current page.....	33
Figure 47 - Result of code in Figure 46	33

Chapter One – Introduction

Restaurant X is a fictional company in the food and beverages industry. With four locations in Scotland (Aberdeen, Dundee, Glasgow, and Edinburgh), the company wishes to kick-start their online presence in order to communicate and engage with their customers through an online medium. This will come in the form of a website.

The company wishes to offer two main services to its customers on the website:

1. Offer an online table reservation system
2. Order food from an extensive menu online and deliver this to the customer's location

This project aims to implement a fully functional website and back-end database which incorporates the two features mentioned above. The website will also include standard features including a locations page to display each restaurant's location with the use of Google Maps API, and a contact page for customers to contact the company.

Chapter Two – Background

According to the Office for National Statistics, in the first quarter of 2017, 89% of adults in the UK had used the internet. With the increase in people having access to the internet, this gives people the option of using online services such as booking tables and ordering food for home delivery, rather than going out to do these tasks, as it is more convenient.

2.1 Online Food Order and Delivery

A main feature of the website for Restaurant X is the ability for a customer to order food online and have it delivered to their chosen destination. Having researched the UK's market for similar online services, two companies stood out from the rest – Just Eat and Hungry House.

A Danish group saw an opportunity to capitalise on the convenience of an online food ordering and delivery service and formed the company Just Eat. The concept was simple – act as the middle man between eatery establishments and their customers. The restaurant or takeaway is considered a partner with Just Eat while the customer can easily place an order online using the Just Eat website. In Just Eat's 2016 annual report, they reported a 31% increase in active users in 2016 (17.6 million) from 2015 (13.4 million) and a reported 11% increase in the number of restaurant partners in 2016 (68,500) compared to 2015 (61,500). This shows the ever increasing popularity of the online service of food ordering.

In 2006, another online platform for ordering of takeaway food was launched – Hungryhouse, which uses the same concept as Just Eat. Having initially started with approximately 250 participating restaurants, as of 2017 it boasts a staggering 11,000 restaurant partners – a 4300% increase from when it started. Though the platform is significantly smaller than Just Eat, Hungryhouse still offers an alternative platform to its competitors.

This steady growth of Just Eat and Hungryhouse indicates that there is an increasing demand for such a service online.

Why not use Just Eat or Hungryhouse?

With the success of companies such as Just Eat and Hungryhouse offering such a service, the question would be – why not use them? There are associated costs with joining such companies. White (2015) wrote that Just Eat charge restaurant partner's an average 11.7% commission per order processed and an initial joining fee which vary from restaurant to restaurant. Hungryhouse, while not charging a joining fee, also charge their restaurant partners a commission per order. While this may appear to be a small fee to pay for joining an established online delivery company, it can be argued that if a company were to implement their own online food ordering system then there would not be any joining fee or commission to pay. While there may be initial costs to implement the system, the long-term benefits would easily outweigh the short-term costs. For this reason, it would be highly beneficial to Restaurant X to implement their own online food ordering system.

2.2 Online Table Reservation

In a traditional restaurant setting, all table bookings would be manually written and kept in a book. While this method has been useful and suitable for small-scale establishments, it is less convenient for larger businesses, and thus it would be easier if the process could be fully automated. This would save time for staff taking table bookings manually via telephone or in person. An automated system would also be easier for management and staff to keep track of any changes to any booking including cancellations.

From the customer's perspective, they can make a booking at any time of the day if it were an online system. Therefore, the customer does not have to rely on the restaurant being open. This gives the customer a faster, more flexible experience when making a booking.

Having researched restaurant chains similar to Restaurant X, an online booking system was found on each of their websites. These chains included: Wagamama, Zizzi, Harvester, TableTable, Cosmo, and Eating Inn. All of these chains have multiple locations and offer the online user the ability to reserve a table for a set number of people on a date and time of their choice. Each system is easy to understand, and simple to use.

With the above mentioned benefits of having an automated online booking system, it is for these reasons that such a system will be implemented into the website of Restaurant X.

Chapter Three – Requirements Specification

Before any development can commence, it is important for any project to consider the requirements of the end users. This is to ensure the finished project will function the way users expect it to. Therefore, requirement specifications have been drawn up for the Restaurant X website.

3.1 Requirements

Given that the company for this project is fictional, requirement gathering could not be conducted via conventional means by asking a real client. Instead, requirement gathering was done by researching similar websites and what they offer their users, which was used as the basis for the requirements of this project. Emphasis was put on the booking system, online ordering system, users, and admin requirements as these were deemed high priority.

The requirements are separated into two types – Functional and Non-Functional:

1. Functional requirements can be seen as tasks that the system must perform. A summary of Functional requirements can be seen in the table below.

Requirement No.	Requirement
1	Navigate through website
2	Booking System
3	Order Online
3.1	Connect to Database
3.2	Adding items to cart
3.3	Customer Cart
3.3.1	Display details of order
3.4	Checkout
3.4.1	Cancel Order
3.4.2	Pay Online
4	Submit Contact Form
5	Register New User
6	User Login
6.1	Edit Personal Information
7	Admin Login

2. Non-Functional requirements can be seen as the constraints upon the system and/or the end user. A summary of Non-Functional requirements can be seen in the table below.

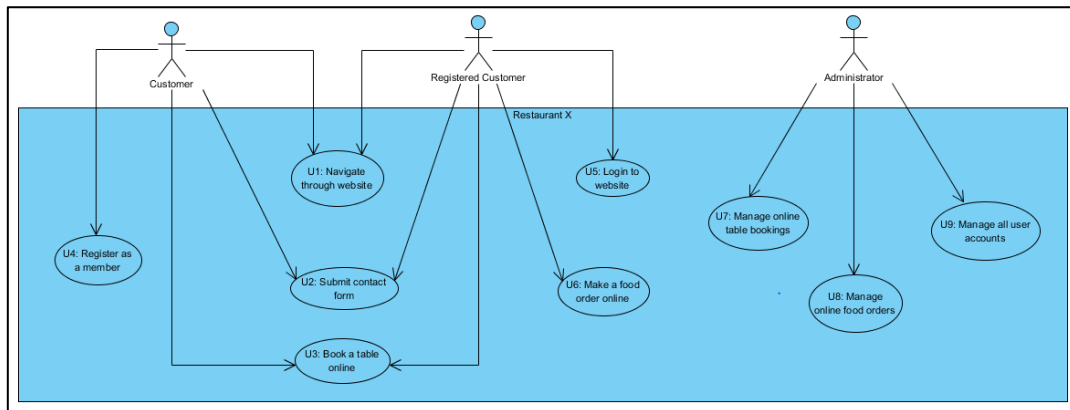
Requirement No.	Requirement
8	Access to an internet connection
8.1	Users
8.2	Admin
9	Security
9.1	Incorrect login credentials
9.2	Restricted Admin Access

A full detailed Requirements Specification can be found in Appendix 1.

3.2 Use Cases

To further help develop and gain a deeper understanding of the requirement specifications, a use case diagram and use case specifications were produced. The diagram gave a visual representation of what the user and administrator require the website to do and how they would use it.

There are three actors associated with the use cases – Customer, Registered Customer and Administrator. It should be noted that a Registered Customer has authenticated access to certain features of the website for example making an online order. A full explanation of each Actor can be found in Appendix 2.



3.3 Use Case Specifications

Once Use Cases have been established, Use Case Specifications can be formulated. These specifications are an expansion on the use cases and give a full explanation of what each Use Case should accomplish along with the basic flow of events. The advantage of producing a Use Case Specification is that alternative flows can be identified which may not have been thought of before. For example, an alternative flow in Use Case 6 “Make a food order online” could be that the Registered Customer cancels their order (Appendix 2 – Use Case Specification 2, A2). A full list of Use Case Specifications can be found in Appendix 2.

3.4 Project Management

Due to the nature of the development of this project, a schedule was created at the beginning outlining the main tasks, timelines, and milestones, using a Gantt Chart. This was important to ensure the project would progress at a reasonable rate and the deadline could be met. Each task had a set duration which helped with keeping track where the project should be at any given point. It was also important to include slack time into the schedule for any unforeseeable situations that could delay the project which the slack time could make up for. An example of the schedule can be seen in Figure 1 while the full timeline can be viewed in Appendix 3.

		Name	Duration	Start	Finish	Predecessors	F	S
1		Scope	10 days	01/05/17 08:00	12/05/17 17:00			
2		Determine Project Scope	1 day	01/05/17 08:00	01/05/17 17:00			
3		Learn HTML, CSS, JavaScript, PHP, MySQL	7 days	02/05/17 08:00	10/05/17 17:00	2		
4		Research other websites	2 days	11/05/17 08:00	12/05/17 17:00	3		
5		Scope Complete	0 days	12/05/17 17:00	12/05/17 17:00	4		
6		Requirements Gathering	10 days	15/05/17 08:00	26/05/17 17:00	5		
7		Conduct Requirements Analysis	1 day	15/05/17 08:00	15/05/17 17:00	5		
8		Draft Requirement Specifications	4 days	16/05/17 08:00	19/05/17 17:00	7		
9		Draft out Use Case Diagrams	3 days	22/05/17 08:00	24/05/17 17:00	8		
10		Ethics Form	2 days	25/05/17 08:00	26/05/17 17:00	9		
11		Requirements Gathering Complete	0 days	26/05/17 17:00	26/05/17 17:00	10		
12		Prototyping	5 days	29/05/17 08:00	02/06/17 17:00			
13		Sketches of website	3 days	29/05/17 08:00	31/05/17 17:00	11		
14		Feedback	1 day	01/06/17 08:00	01/06/17 17:00	13		
15		Amendments to sketches	1 day	02/06/17 08:00	02/06/17 17:00	14		
16		Prototyping Complete	0 days	02/06/17 17:00	02/06/17 17:00	15		

Figure 1 – Project Schedule

Chapter Four – Design (Early Stages)

Research was conducted into websites similar to Restaurant X where companies had an online booking system to reserve a table and those who had an online food ordering system as mentioned in Chapter 2. With ideas of similar features that would be beneficial to the Restaurant X website, initial designs and prototypes were created and then evaluated with participants.

4.1 Prototyping

Prototyping can be defined as the “externalising and making concrete a design idea for the purpose of evaluation” (Munoz 1992) which Arnowitz et al (2007) agree with as they argue that “...prototypes are tangible software representations, which permit the software team to experience a design without needing to program the software”. This was important for the design of the website for this project as early decisions were required to be made regarding the stylings, layouts and content. This was done using low and high fidelity prototyping.

4.2 Low Fidelity – Sketches

In order to bring a concept to life, low fidelity prototyping can be useful as a starting point. Sketches can be considered as low fidelity prototyping and therefore some sketches were drawn initially to get a feel for the design and layout of the website. Figures 2, 3 and 4 are initial sketches of the Restaurant X website. It details a basic layout and content for the website’s Home Page, Bookings page and Online Orders page.

Figure 2 is a sketch of the initial home page design. This design was used as a template for the majority of the website sketches. The design includes standard website features such as the logo and navigation bar. The design includes a main content container for information and also two “advert” features located in the left-hand side of the main content container. The purpose of the two adverts is to promote the availability of online table booking and online food ordering services – two of the important use cases for the project as mentioned in Chapter 3.

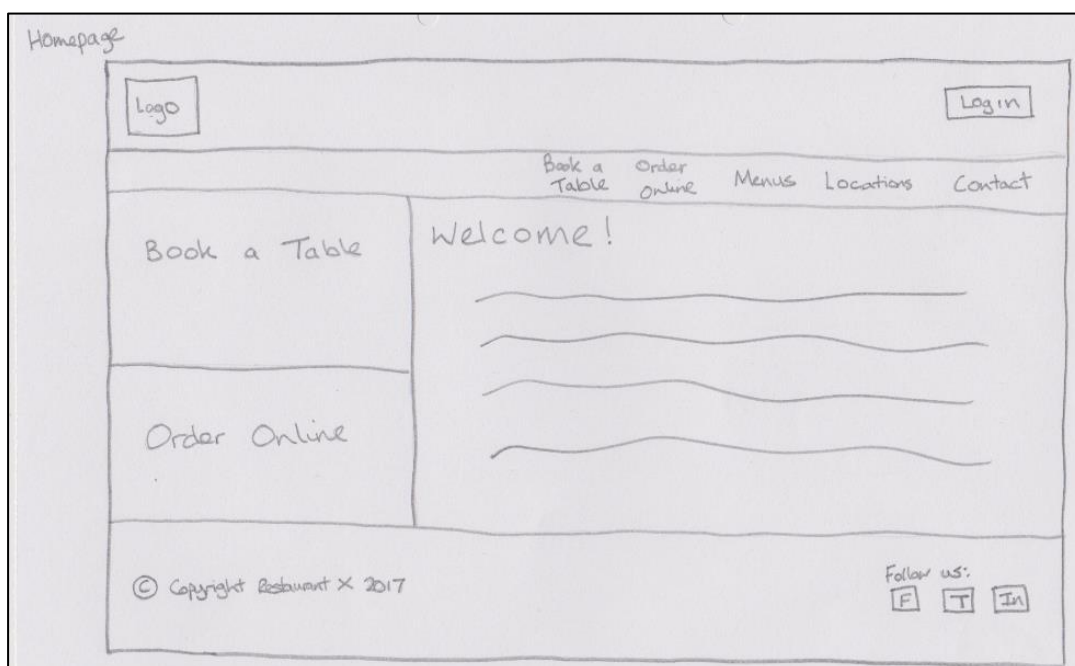


Figure 2 – Sketch of Home Page

Figure 3 shows a sketch of the initial design of the “Book a Table” web page. The user will first be presented with the option to select a location from a drop-down menu. After the location selection the user will then be taken to the next page to select the number of people in their party, date and time.

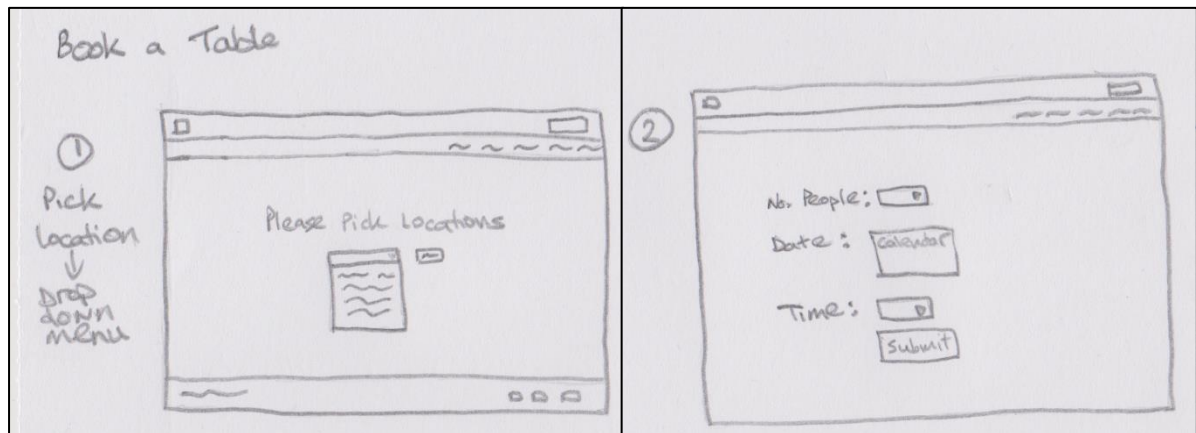


Figure 3 – Sketch of “Book a Table” page

Figure 4 demonstrates the initial design of the online food ordering web page. This included three sections – a category selector on the left, main menu items in the middle and a “cart” on the right which lists all added items from the user. The cart will total up their order dependent on the user’s choices and allow the user to checkout.

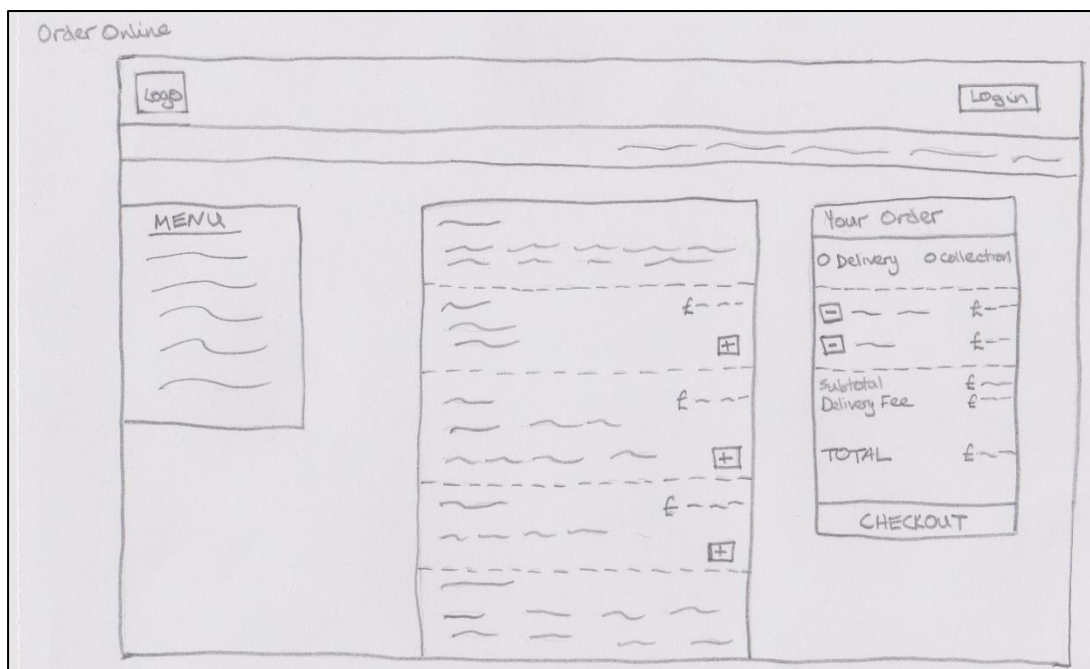


Figure 4 – Sketch of the “Order Online” page

4.3 High Fidelity – Axure

The next logical step was to use a form of high fidelity prototyping in order to give a hi-tech visual representation of the website. Axure RP 8, a prototyping software, was adopted as it is capable of producing a fully functional prototype of the final website including interaction and dynamic features like hyperlinks and buttons. Figure 5 shows the Axure RP 8 software interface with an example of the prototype of the home page.

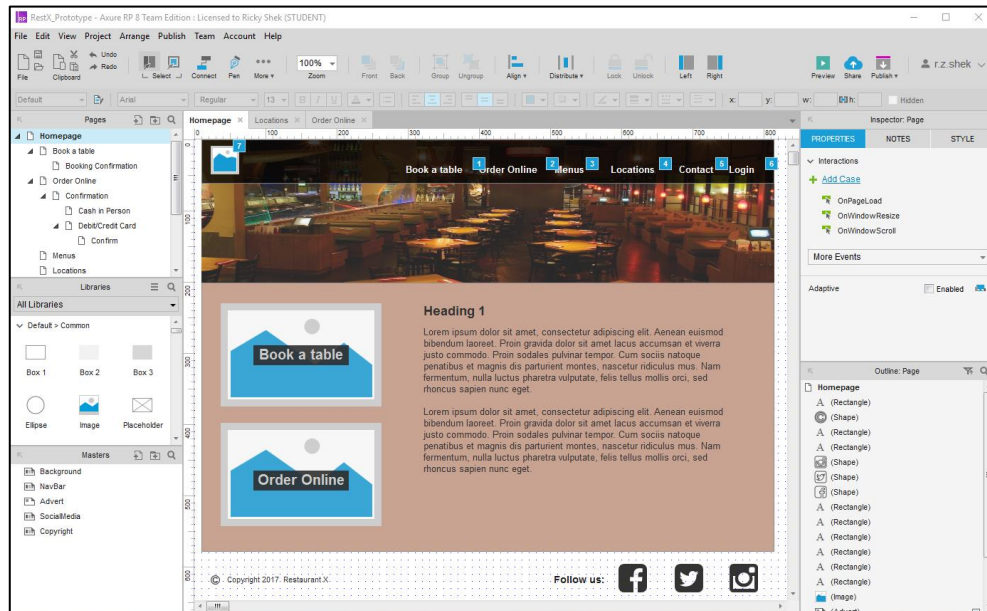


Figure 5 – Axure RP 8 – Home Page Prototype

Axure has the ability to preview the prototype on a browser. This can be seen in Figure 6 which is the browser view of the home page prototype in Figure 5. This is useful for the developer and the user to get a feel of how the final website will look and function. More importantly, this prototype can be shown to clients to gather their feedback and to establish any major changes required to be made – both aesthetically and functionally. Changes at the design stage are more easily implemented than changes in the development stage which can be more costly and could lead to delays.

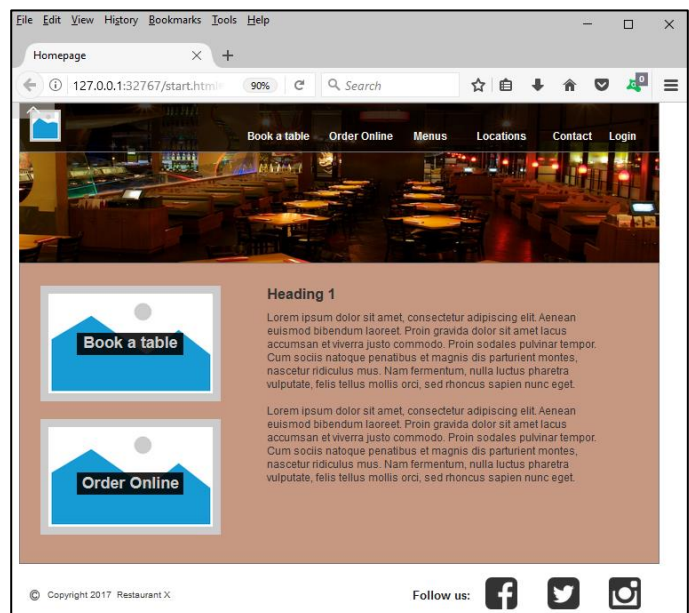


Figure 6 – Axure Prototype in a browser

4.4 Testing of High Fidelity Prototype

4.4.1 Participants

A total of five participants were invited to review the Axure high fidelity prototype and give their feedback and opinions. The process asked each participant to read an information sheet (Appendix 4) detailing the nature of the task which they will conduct and then asked to sign a consent form (Appendix 5) agreeing they will participate in the study.

Each participant was asked to complete a survey (Appendix 6) based on the browser view of the prototype as seen in Figure 6. The survey includes a list of questions and statements to which each participant uses a five-point Likert Scale (1 for strongly agree to 5 for strongly disagree) or multiple-choice questions to find out their opinions on each question or statement.

The survey first asks their opinion on the aesthetics of the website. They were then asked to go through each of the main web pages of the prototype and were asked how much they agreed with each statement using the Likert Scale. Participants were given the option to provide any additional feedback at the end of the survey which was not covered anywhere else.

4.4.2 Results

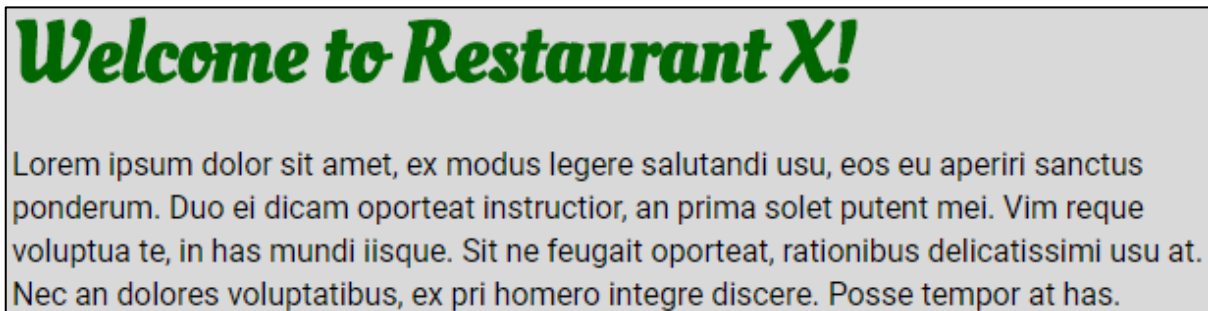
As a result of the testing with participants, some design changes were taken into consideration and implemented into the final design of the website.

4.4.2.1 Colour Scheme

This included a change in colour scheme of the website. A participant, with a background in design, provided feedback that the design lacked a colour scheme and adopting one would greatly improve the look and design on the website. Different shades of grey were suggested for the background and content containers.

4.4.2.2 Font Styles

Two different font styles were suggested for the website – one for headers and one for the main pieces of text. Google provides an open source library of alternative font styles to the conventional ones which can be used in website development. Ultimately the font style “Oleo Script” was used for the main headers while “Roboto” was selected for the main text. Both can be seen in the example below.



4.4.2.3 Font Colour

As a further recommendation for the colour scheme and font styles, headers were coloured in either green or gold to match the grey backgrounds while main text remained black. An example of the green used for the header to match the grey background can be seen in the image above.

4.4.2.4 Adverts

Advertisements for online table booking and online food ordering were initially located on the left-hand side of the main content container and feedback indicated that this would be best located to the right-hand side of the main content container. It was found that each participant would automatically be searching for the main content first, which in many websites, are located underneath the navigation bar to the left. Therefore, both adverts would ultimately be relocated to the right of the main content container instead.

4.4.2.5 Highlighting the current page

To aid the user in determining their location of the website, it was suggested by a participant that the navigation menu item could be highlighted a different colour to the navigation bar. For purposes of accessibility, especially for users who may have issues differentiating between colours, each highlighted menu item would also be underlined to aid those with such difficulties. These were eventually adopted into the main development of the website.

4.4.2.6 Separate Login Button

In initial sketches, a Login feature, in the form of a button, was located in the top right of the website to take the user to the login webpage. The Axure prototype changed this design and included the Login feature as a menu item in the navigation bar. The survey asked each participant their thoughts on whether the Login feature should be made into a separate button as opposed to a navigation menu item. The majority of participants (4 out of 5) responded that they did not feel it was necessary for the login feature to be made into a separate button as they felt most people would know where to locate it. Therefore, the Login feature was eventually merged into the navigation bar as a menu item.

Chapter Five – Development and Implementation

5.1 Software

5.1.1 Visual Studio 2015

The main development of the project was developed using Visual Studio 2015 edition. From an organisation and planning perspective, this system was useful for development as it allows for easy organisation of folders and files. It allows for different file types to be accessible within cascading folders as the project included many HTML, CSS, JavaScript, Python and miscellaneous files. This can be seen in Figure 7.

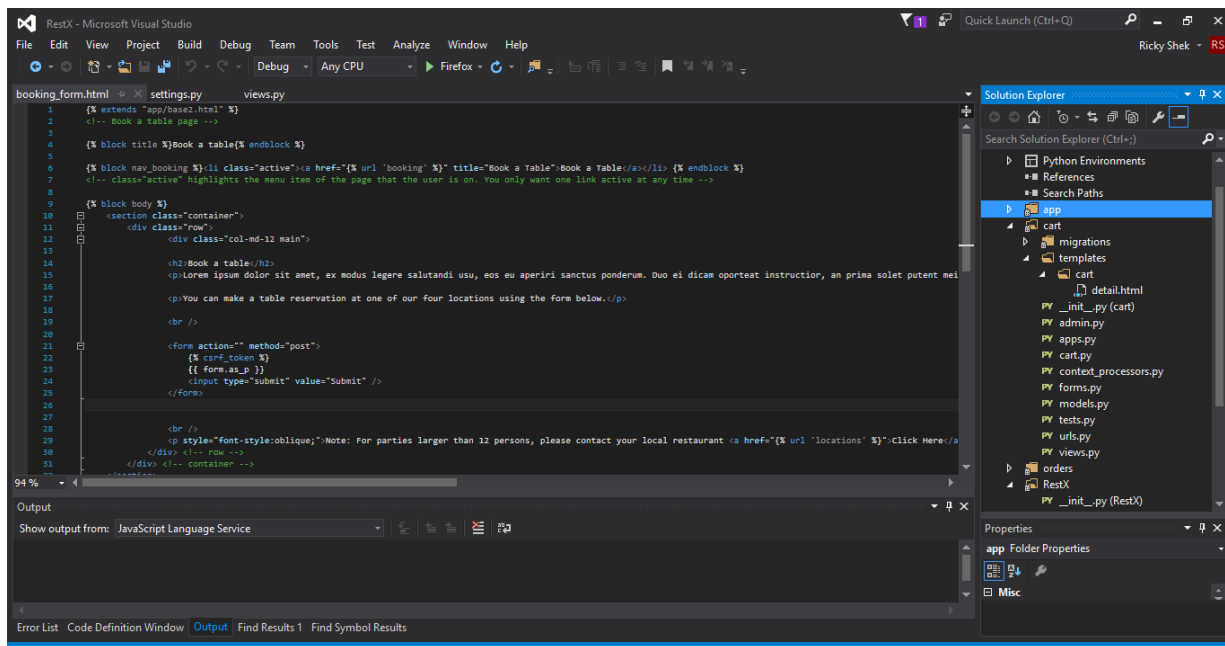


Figure 7 – Visual Studio 2015 Interface

5.1.2 Django Framework

A web framework is a software which is a useful tool for the development of web applications. These frameworks offer a more standardised method of creating and deploying web applications through the use of libraries for database access, standardised templates for frameworks, and session management. The web frameworks often promote efficient code re-usage.

There are many web frameworks available to web developers, including Ruby on Rails, ASP.Net and CakePHP, however the framework chosen for this project was Django. Django is an open source high level web framework using the programming language Python that allows developers to create “apps” within an overall project. Each app can run remotely or in conjunction with other apps in the same project. It adopts a similar design pattern to the Model View Controller (MVC) pattern of software architecture which Holovaty and Kaplan-Moss (2009) describes as having three components: “data access logic, business logic and presentation logic”. The authors expand on these components: “...in this pattern, “Model” refers to the data access layer, “View” refers to the part of the system that selects what to display and how to display it, and “Controller” refers to the part of the system that decides which view to use, depending on user input, accessing the model as needed.” The MVC pattern is of particular use to database-driven web applications as it allows for

“loose coupling” and “strict separation” between different sections of an application. This allows for modification of different sections or “apps” without affecting the overall application or other apps.

The process of Django’s MVC pattern is demonstrated in Figure 8. The user will request a web page from the system however the request for the file is not direct. The system reads through a URL Python file and searches for the associated URL ending. For example, if the user wishes to load the index or home page they could add “index” to the end of the URL without the need for the extension .html or .htm as Django uses regular expressions like the one below for the home page or index page.

```
url(r'^index$', app.views.index, name='index'),
```

The system will in turn contact a Views Python file which lists classes and functions. Depending on the URL requested, the system will read through the Views file and call the appropriate Class or Function in order to perform an action for example loading an HTML file. In the example of requesting the home page, the views would be coded as below:

```
def index(request):  
    assert isinstance(request, HttpRequest)  
    return render(request, 'app/index5.html')
```

This function takes in the request to display the home page and calls the appropriate HTML file and displays it to the user.

Django uses an Object Relational Mapping (ORM) system which allows for transferring of data stored in a relational database. The ORM is useful for a framework like Django as it allows the developer to use Python code instead of SQL statements for CRUD (create, read, update and delete) operations on data and schema.

For example, to retrieve a user from a table with the name “John” the SQL statement would be:

```
SELECT * FROM USERS WHERE name=John;
```

While in Django, the Python syntax would be:

```
users = Users.objects.get(name='John')
```

This is extremely useful for a Python programmer as it takes away the need for them to learn SQL Statements when using Django.

If the user has requested data from the database, the system would go from URLs to the Views and access data from the database via the ORM. The database would send the requested data via the ORM again, back to the Views, load the appropriate template and display the results to the user.

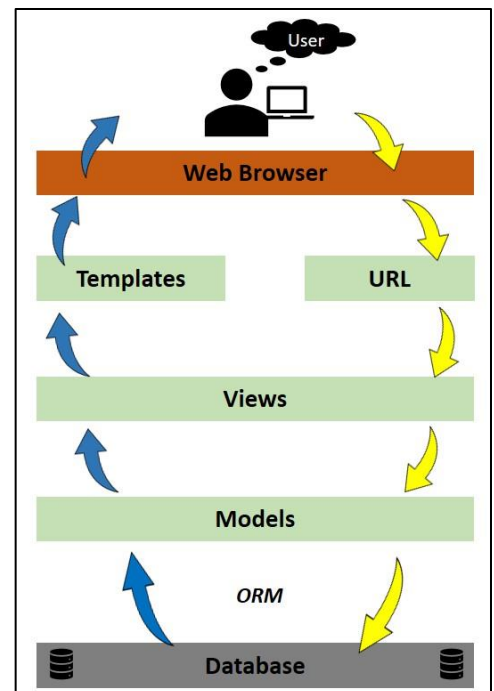


Figure 8 – Django’s MVC Structure

5.2 Security

Security is an integral part of any project especially those involving databases which could contain sensitive information. Django provides features to protect websites from security issues such as the following:

5.2.1 SQL Injections

The w3school.com website describes an SQL Injection as “a code injection technique that might destroy your database” and “...is one of the most common web hacking techniques”. This usually occurs when a hacker inserts SQL Statements into an entry field in a form, for example username or email, instead of data which will execute in a database. Django combats this through its built-in form validation framework, under the assumption the developer has used the standard Form class of Django, to ensure that only valid data has been inputted by the user. For example, a form has a date input field and if a user attempts to insert an SQL statement instead of a valid date, Django will reject it automatically.

5.2.2 Cross-Site Scripting (XSS)

Cross-Site Scripting occurs when a website fails to escape content submitted by the user before the content is rendered into HTML. Such an attack can lead to users providing sensitive information to the hacker without their knowledge. This is prevented by Django as all variable values are automatically escaped using Django’s template system.

5.2.3 Cross-Site Request Forgery (CSRF)

This type of hacking occurs when a hacker tricks a user into loading a webpage from a site on which they are already authenticated. Therefore, the hacker takes advantage of the user’s authenticated status. Django has an automatic enabled CSRF feature when processing any post requests from a user. Developers are encouraged to use Django’s `csrf_token` syntax when developing forms as a way to prevent CSRF attacks by including the syntax in forms like the one in Figure 9.

```
<form action="" method="post">
  {% csrf_token %}
  {{ form.as_p }}
  <input type="submit" value="Submit" />
</form>
```

Figure 9 – Django’s Form with CSRF protection

5.2.4 Password Hashing

When a user is created in Django, the password submitted by the user is hashed meaning no raw passwords (i.e. readable passwords by humans) are stored in the Django database. The hashed password is stored in random strings of characters which has been mathematically transformed so that the passwords cannot be misused. This can be seen in Django’s built in Administration system below:

Username:	<input type="text" value="admin"/>
	<small>Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.</small>
Password:	algorithm: pbkdf2_sha256 iterations: 36000 salt: VkrA5***** hash: w1RpBS*****
	<small>Raw passwords are not stored, so there is no way to see this user's password, but you can change the password using this form.</small>

5.3 Built-in Features

Django is designed to make development of web applications easy for web developers. It has some pre-built features that come with the creation of every new project. For the purposes of this project the following built in features were of notable interest:

5.3.1 SQLite3 Database

When the project was created in Visual Studio, SQLite3 database was set up as default and was used as the database for the project. While this may be the default database for Django, developers have the option to adopt a different database system such as MySQL and PostgreSQL. Primary keys are important to any database and Teorey (2011) describes them as “an attribute of an entity or an equivalent SQL, which may be either an identifier or a descriptor”. This is to ensure each record is unique and can be identified by a value unique to it. Django assigns a primary key automatically to each record without the developer having to do so, though this can be overwritten if desired.

5.3.2 Forms

Forms allow websites to accept data input from users to make the website more interactive and dynamic to users. Django takes away the complexity of having to build them from scratch for developers by:

- preparing and restructuring data to make it ready for rendering
- creating HTML forms for the data
- receiving and processing submitted forms and data from the client

This project used a combination of Django’s ModelForm class and Form class. ModelForm classes were used to directly use the same fields and attributes as those created for the entities (i.e. tables) within the database while the Form class was used to make brand new forms that did not have an existing table in the database.

An example of the Form class being used can be found in the “Contact Us” webpage using the Python code in Figure 10.

```
class ContactForm(forms.Form):
    full_name = forms.CharField(required=True)
    email = forms.EmailField(required=True)
    location = forms.ChoiceField(choices = city_choices, label="City", initial='', widget=forms.Select(), required=True)
    message = forms.CharField(widget=forms.Textarea, required=True)
```

Figure 10 – Django’s Form Class

An example of the ModelForm class can be seen in Figures 11, 12 and 13 which is for the Booking system. These figures show the progression from code to the final result of what the user will see.

```
class BookingForm(ModelForm):
    date = forms.DateField(widget = forms.DateInput(format='%d/%m/%Y'), attrs = {'class': 'datepicker', 'placeholder': 'Select a date'})
    class Meta:
        model = Booking
        fields = ['booking_name', 'email', 'phone_number', 'number_of_people', 'location', 'time_slot', 'date']
```

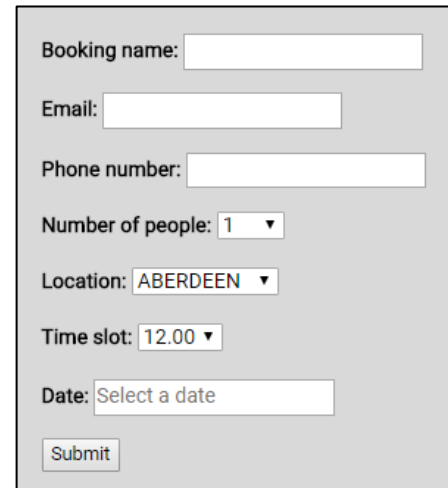
Figure 11 – Django’s ModelForm Class: Python Code to create form


```

<form action="" method="post">
    {% csrf_token %}
    {{ form.as_p }}
    <input type="submit" value="Submit" />
</form>

```

Figure 12 – Python & HTML Code to display the form on website



A screenshot of a web form for booking. It contains the following fields: 'Booking name:' with a text input, 'Email:' with a text input, 'Phone number:' with a text input, 'Number of people:' with a dropdown menu showing '1', 'Location:' with a dropdown menu showing 'ABERDEEN', 'Time slot:' with a dropdown menu showing '12.00', and 'Date:' with a date picker showing 'Select a date'. At the bottom is a 'Submit' button.

Figure 13 – User's view of result of coding from Figures 11 & 12

5.3.3 Admin Site

With Django, a built-in admin site is provided with each new project. The developer has the option to use this or develop their own. The admin site “reads metadata from your models to provide a quick, model-centric interface where trusted users can manage content on your site.” The built-in admin site was used for the purposes of this project with particular attention to managing users, online bookings and online food orders which are all major features of the website. The dashboard for the Admin Site can be seen in Figure 14.

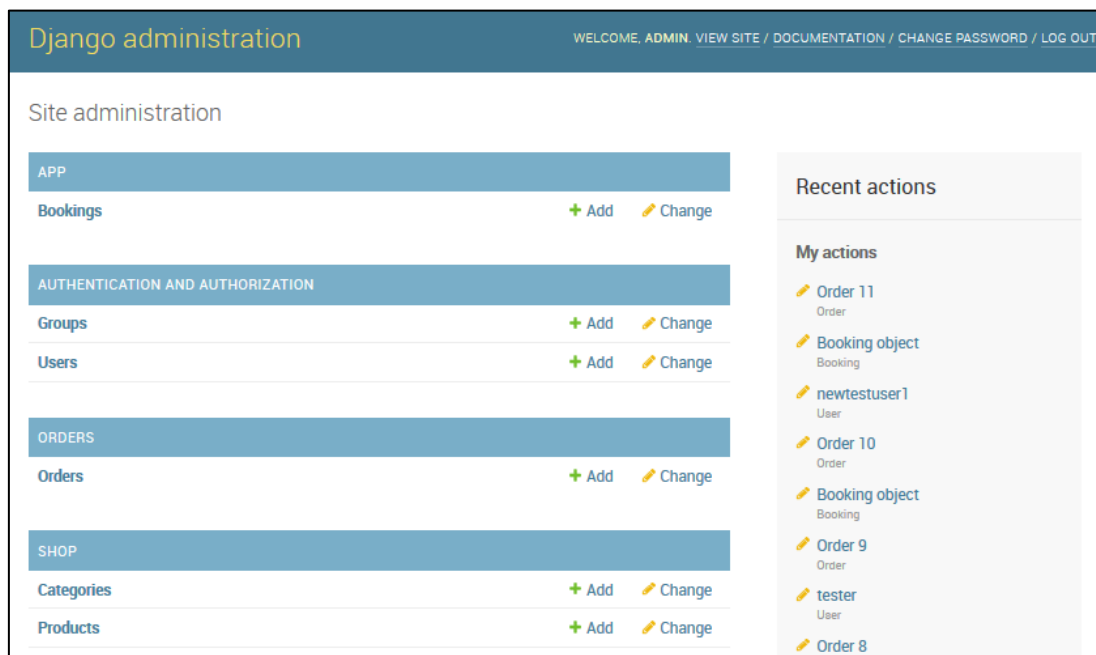


Figure 14 – Admin Site Dashboard

Though the admin site is pre-built, it does allow for some customisation. This includes what and how data is displayed, filtering to only display certain records, sorting records by field name, and editing multiple records simultaneously. This can be seen in Figures 15 and 16 which shows the Python code and the live admin site view.


```

from django.contrib import admin
from .models import *

class BookingAdmin(admin.ModelAdmin):
    list_display = ['id', 'booking_name', 'number_of_people', 'location', 'time_slot', 'date']
    list_filter = ['number_of_people', 'location', 'time_slot', 'date']

admin.site.register(Booking, BookingAdmin)

```

Figure 15 – Syntax to display certain fields and filter criteria

Django administration

WELCOME, ADMIN [VIEW SITE](#) / [DOCUMENTATION](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

Home > App > Bookings

Select booking to change ADD BOOKING +

Action: 0 of 26 selected

<input type="checkbox"/>	ID	BOOKING NAME	NUMBER OF PEOPLE	LOCATION	TIME SLOT	DATE
<input type="checkbox"/>	38	michal	6	ABERDEEN	20.00	Aug. 10, 2017
<input type="checkbox"/>	37	Shek	5	EDINBURGH	19.00	Aug. 21, 2017
<input type="checkbox"/>	36	Bronze	5	DUNDEE	20.00	Aug. 18, 2017
<input type="checkbox"/>	35	Shek	6	EDINBURGH	15.00	Aug. 18, 2017
<input type="checkbox"/>	34	Shek	6	EDINBURGH	15.00	Aug. 18, 2017
<input type="checkbox"/>	33	Shek	3	EDINBURGH	15.00	Aug. 12, 2017
<input type="checkbox"/>	31	Shek	3	GLASGOW	16.00	Aug. 31, 2017
<input type="checkbox"/>	30	Shek	8	GLASGOW	15.30	Aug. 30, 2017
<input type="checkbox"/>	29	Shek	2	DUNDEE	14.00	Aug. 5, 2017
<input type="checkbox"/>	28	Glass	12	DUNDEE	21.30	Aug. 2, 2017
<input type="checkbox"/>	27	robert	1	ABERDEEN	12.00	Aug. 9, 2017
<input type="checkbox"/>	26	bob	1	ABERDEEN	12.00	Aug. 2, 2017
<input type="checkbox"/>	25	Shek	5	EDINBURGH	18.00	July 31, 2017
<input type="checkbox"/>	24	Shek	3	EDINBURGH	20.30	July 31, 2017
<input type="checkbox"/>	23	Shek	3	DUNDEE	19.00	July 31, 2017
<input type="checkbox"/>	22	Shek	2	DUNDEE	18.00	July 31, 2017
<input type="checkbox"/>	21	Lime	11	ABERDEEN	16.00	Aug. 4, 2017
<input type="checkbox"/>	20	Green	2	DUNDEE	17.30	July 29, 2017
<input type="checkbox"/>	19	Gold	5	EDINBURGH	19.30	July 30, 2017
<input type="checkbox"/>	18	Bronze	4	EDINBURGH	18.00	July 27, 2017
<input type="checkbox"/>	17	Goldie	10	EDINBURGH	12.00	July 25, 2017

FILTER

By number of people

- All
- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 12+

By location

- All
- ABERDEEN
- DUNDEE
- EDINBURGH
- GLASGOW

By time slot

- All
- 12.00
- 12.30
- 13.00
- 13.30
- 14.00
- 14.30
- 15.00
- 15.30

Figure 16 – Admin Site view of displaying fields and filter criteria

5.4 Mobile First Approach

Wroblewski (2011) stated that “...smartphones were boldly predicted to out-ship the combined global market of laptop, desktop and notebook computers in 2012”. The smartphone market had surpassed them by the end of 2010. This had meant that users were increasingly using their mobile smartphones to access the internet and using their desktop or laptop computers as a secondary option. With the rise in mobile access to the web came the need for mobile design of web applications and therefore the need for the Mobile First Approach. This concept is based on initially designing for the mobile with the option of expanding further into bigger devices for example tablets, laptops and desktops. Given the significant difference in screen sizes of the smartphone compared to its counterparts, prioritisation should be given to the most important features of the website. For the website of Restaurant X, Bootstrap was implemented to help with a mobile first approach as it has an extensive library of HTML, CSS and JavaScript material to quickly and easily build webpages that would also facilitate smartphones and larger devices.

The website also makes use of CSS’s @media rule which aids a developer in defining different styles for different devices and screen sizes. Figure 17 is an example of the @media rule applied to the Restaurant X website. This excerpt of code shows the decreasing sizes of headers, paragraphs, and text to fit in line with the smaller screen sizes of mobiles. This can be seen in a comparison between Figures 18 and 19. Figure 18 is the desktop version while Figure 19 is the mobile version.

```
@media screen and (min-device-width: 320px)
and (max-device-width: 568px) and (orientation: portrait) {

    section.container div.main h2 {font-size: 2em;}
    section.container div.advert h3 {font-size: 0.9em;}
    .container p {font-size: 1em;}

    section div.main {margin-top: 30px;}

    section div.advert .booknow {position:absolute; top: 120px; left: 65px; /* positioning of the text */}

    section div.advert .ordernow {position:absolute; top: 340px; left: 55px; /* positioning of the text */}

    section div.advert h3 a {font-size: .7em;}

}
```

Figure 17 – CSS @media rule code excerpt



Figure 18 – Desktop version of Restaurant X website

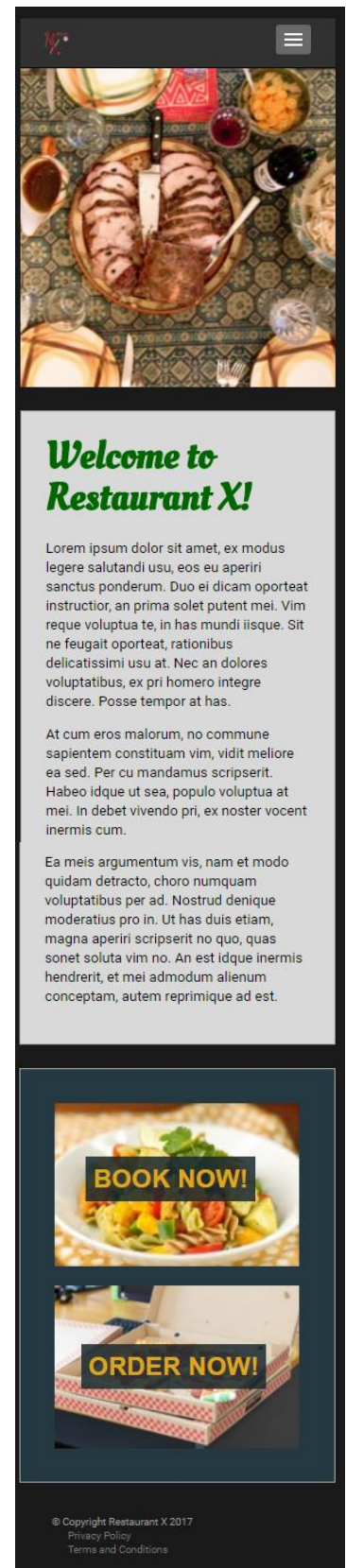


Figure 19 – Example of Mobile version of Restaurant X website

5.5 Functionality and Interfaces

5.5.1 HTML Base Template

Django allows for the creation of a base template on which all HTML files can be based upon. This is useful as it prevents repetition of code in multiple HTML files and it is also useful when changes to the layout of HTML files are required. The change only needs to be done in the base template and the other HTML files will adopt the change.

The base template is set up like a normal HTML file which also includes CSS and JavaScript references. In order to use the base template in other HTML files, the developer is required to use the following syntax in the base template HTML file:

```
{% block body %}{% endblock %}
```

The developer places the syntax within the <body> tags and before the <footer> tags. The variable “body” above can be changed to any variable name.

The following syntax should be included at the top of each HTML file that should adopt the base template:

```
{% extends "app/base2.html" %}
```

The HTML files will adopt all layouts and stylings from the base template including CSS and JavaScript.

5.5.2 Authentication Required Areas

The Restaurant X website is widely accessible to most users however there are certain areas which are only available through authentication which the user can access by successfully registering on the website. The “Registered Customer” Actor, mentioned in section 3.2, would be known as the authenticated user.

Two methods were adopted to ensure only authenticated users could access restricted areas:

1. Django’s built-in function `@login_required()` which was used in the Views Python files of the “cart”, “orders” and “shop” apps of the project included above each required function and class. This method was adopted to ensure users could not access certain areas of the online ordering system by typing in the associated URL links for example <http://127.0.0.1:8000/cart/>. If the non-registered user attempted to request this link they will be presented with Figure 24.
2. Make use of the if else statement – `if user.is_authenticated`
This can be seen in Figure 23.

The following features were implemented with authentication:

5.5.2.1 User Registration

Before a user can access authenticated areas, they are required to successfully register on the website. The link to the registration page can be found in the login page underneath the login portal for users. This can be seen in Figure 20. The registration page presents the user with a form to fill in and if successfully completed they can submit the form where upon all their details are saved into the database. The registration form can be seen in Figure 21. As an extra feature, an additional password checker field (“Confirm Password” in the form) was added to ensure the user types in the same password twice. If the passwords do not match then an error message will appear. This can also be seen in Figure 21.

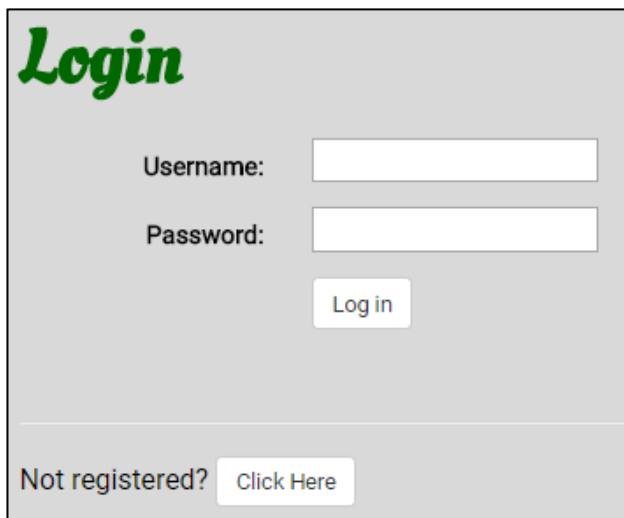
The image shows a login portal with a green 'Login' title. It contains two input fields: 'Username:' and 'Password:'. Below the password field is a 'Log in' button. At the bottom, there is a link 'Not registered? Click Here'.

Figure 20 – Login Portal & link to Registration page

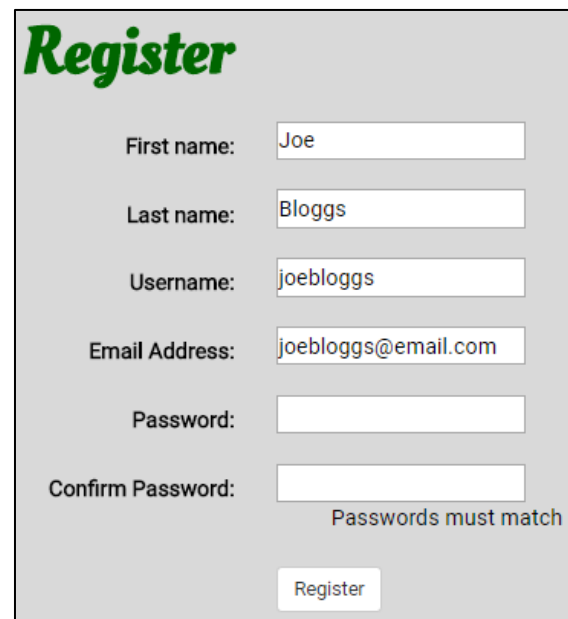
The image shows a registration form with a green 'Register' title. It contains several input fields: 'First name:' (filled with 'Joe'), 'Last name:' (filled with 'Bloggs'), 'Username:' (filled with 'joebloggs'), 'Email Address:' (filled with 'joebloggs@email.com'), 'Password:', and 'Confirm Password:'. Below the 'Confirm Password' field, there is a red error message 'Passwords must match'. At the bottom right, there is a 'Register' button.

Figure 21 – Registration form with error message

5.5.2.2 User Login/Logout

After registration, the user is automatically logged into the website. If the user decides to log in at another time, they can simply click the “Log in” menu item from the navigation bar along the top. They will be presented with the login portal like the one in Figure 20. Once the user has logged in they are redirected to the home page. The user can log out at any time by clicking on the dropdown menu (located in the navigation bar with their name) and click “Log Off”. This will redirect the user back to the home page and the navigation bar will display the Login menu item again.

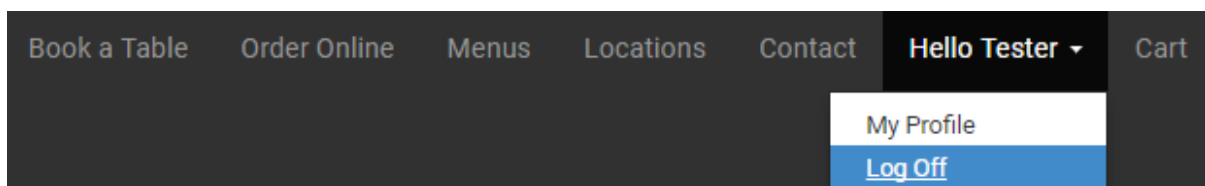
The image shows a navigation bar with links: 'Book a Table', 'Order Online', 'Menus', 'Locations', 'Contact', 'Hello Tester', and 'Cart'. The 'Hello Tester' link has a dropdown menu open, showing 'My Profile' and 'Log Off'.

Figure 22 – User Log out option

5.5.2.3 Online Ordering System

This section is only available to users who have successfully registered on the website. The authentication status is handled with the code in Figure 23 using an if else statement.

```
order_form_partial.html  -b X
1
2   {% if user.is_authenticated %}
3
4   <li><a href="{% url 'shop:product_list' %}" title="Order Online">Order Online</a></li>
5
6   {% else %}
7
8
9   <li><a href="{% url 'order_login' %}" title="Order Online">Order Online</a></li>
10
11
12  {% endif %}
```

Figure 23 – Authentication Status

The following code is included in the base template with the other navigation bar menu items. By including this piece of code, the system knows to refer to the HTML file “order_form_partial.html” and, depending on the authentication status of the user, will select which option in Figure 23 to display to the user.

```
{% block nav_order %}{% include 'app/order_form_partial.html' %}{% endblock %}
```

If a non-registered user clicks on the “Order Online” menu item in the navigation bar to access the online ordering system, they will be presented with the message in Figure 24.

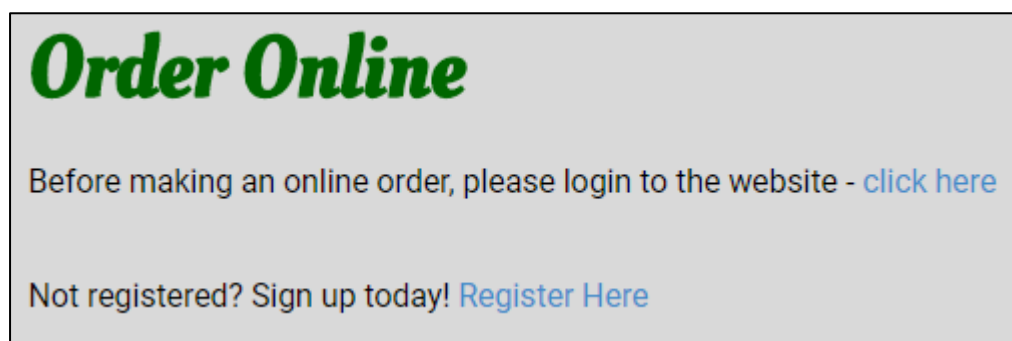


Figure 24 – User Login or Registration required message

If the user is authenticated, they will have access to the online ordering system. They will be presented with a list of products which they can add to their cart by clicking on the “Add to cart” button as demonstrated in Figure 25.

When the user has added the desired products, they can visit their cart by clicking on the navigation bar menu item “Cart”. This will display a list of all products the user has added and will provide information such as the unit price of each, the quantity of each, the total price of each item added (if the quantity is more than one) and the total price of all items. The user also has update and deletion capabilities in the cart as they can update the quantity of items and remove items. This can be seen in Figure 26.

Starters			
Item	Price	Quantity	
Pate	£4.50	1 ▼	Add to cart
Salad	£3.50	1 ▼	Add to cart
Soup	£3.25	1 ▼	Add to cart

Figure 25 – List of products and “Add to cart” button

Your shopping cart				
Product	Quantity	Remove	Unit price	Price
Chicken Fajitas	1 ▼ Update	Remove	£8.75	£8.75
Pate	1 ▼ Update	Remove	£4.50	£4.50
			Total	£13.25
				Continue shopping
				Checkout

Figure 26 – Cart with items

5.5.2.4 Checkout

Once the user is satisfied with their order, they can checkout their cart by clicking on the “Checkout” button. This will present the user with the same list of items that they have added to their cart and the total price of all items. They will then be required to fill in a form (Figure 27) in order for the appropriate restaurant to know to whom and where they are delivering. After successfully filling in the form and placing the order, the user will receive a confirmation message including their order number. The order and information the user provides will be saved in the database. This can be seen in Figure 28 which shows the order and customer information on the Admin Site.

Checkout

Your order

1x Chicken Fajitas £8.75
1x Pate £4.50

Total: £13.25

First name:

Last name:

Email:

Address:

Postal code:

City:

Telephone number:

Figure 27 – Checkout

Django administration WELCOME, ADMIN. [VIEW SITE](#) / [DOCUMENTATION](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

Home > Orders > Orders > Order 12

Change order HISTORY

First name:

Last name:

Email:

Address:

Postal code:

City:

Telephone number:

☐ Paid

ORDER ITEMS

PRODUCT	PRICE	QUANTITY	DELETE?
19 <input type="text" value="11"/> Q Chicken Fajitas	<input type="text" value="8.75"/>	<input type="text" value="1"/>	<input type="checkbox"/>
20 <input type="text" value="6"/> Q Pate	<input type="text" value="4.5"/>	<input type="text" value="1"/>	<input type="checkbox"/>

Figure 28 – Admin Site displaying order from Figure 27

5.5.3 Non-Authentication Required Areas

5.5.3.1 Contact Form

When users click on the Contact menu item from the navigation bar, they are presented with a form that they can fill in to contact the staff at Restaurant X. Figure 29 shows the code to display the webpage and form. Figure 29 also shows the ability for the message to be submitted and sent to an email address. This is a function built into Django `send_mail()` which has the following parameters: subject, from_email, and message. Figure 30 shows the form that the user completes and Figure 31 shows an example of a submitted contact form received by email.

```
def contact(request):
    if request.method == 'GET':
        form = ContactForm()
    else:
        form = ContactForm(request.POST)
    if form.is_valid():
        subject = 'Restaurant X - Website Contact'
        from_email = form.cleaned_data['email'] # if website had own email then change this to website email address e.g admin@restx.com
        message = 'From: ' + form.cleaned_data['full_name'] + '---- City: ' + form.cleaned_data['location'] + '---- Message: ' + form.cleaned_data['message']
        try:
            send_mail(subject, message, from_email, ['example@email.com'])
        except BadHeaderError:
            return HttpResponse('Invalid header found.')
        return redirect('thanks')
    return render(request, "app/contact5.html", {'form': form})

def thanks(request):
    return render(request, "app/contact5_thankyou.html")
```

Figure 29 – Function to receive completed Contact form and email it to administrator

click here'. The form fields are: 'Full name:' with a text input, 'Email:' with a text input, 'City:' with a dropdown menu showing 'ABERDEEN', and 'Message:' with a large text area. A 'Submit' button is at the bottom." data-bbox="117 436 494 743"/>

Figure 30 – Contact Form



Figure 31 – Example of Contact Form sent to email

5.5.3.2 Menus

The Menus web page offers the user an opportunity to view the products offered at the physical locations of the Restaurant X chain.

```
def menu(request):
    assert isinstance(request, HttpRequest)
    return render(request, 'app/menus.html',)

def menulunch(request):
    try:
        return FileResponse(open('app/static/app/_assets/LunchMenu.pdf', 'rb'), content_type='application/pdf')
    except FileNotFoundError:
        raise Http404()

def mensdinner(request):
    try:
        return FileResponse(open('app/static/app/_assets/DinnerMenu.pdf', 'rb'), content_type='application/pdf')
    except FileNotFoundError:
        raise Http404()
```

Figure 32 – Menus page Python code

There are three functions associated with the Menus web page (Figure 32):

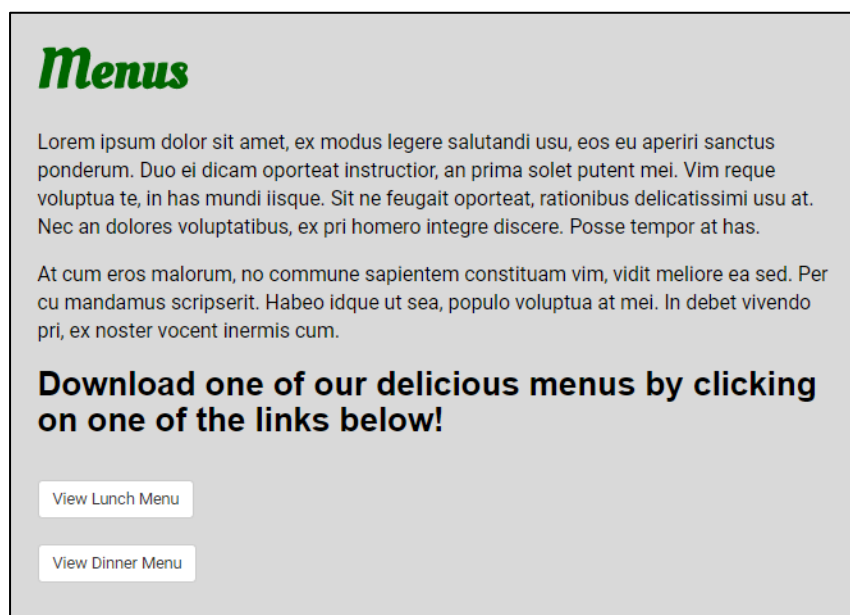
1. To render the HTML file for the Menus web page
2. To open the Lunch Menu PDF file
3. To open the Dinner Menu PDF file

The two functions to open up the PDF files have a code to display a HTTP 404, also known as a “File not found”, error message in the event that the PDFs cannot be located.

The following code is used in the HTML file of the Menus page to create two buttons to open the Lunch Menu PDF and Dinner Menu PDF.

```
<a href="{% url 'menulunch' %}" target="_blank" class="btn btn-default">View Lunch Menu</a><br /><br />
<a href="{% url 'mensdinner' %}" target="_blank" class="btn btn-default">View Dinner Menu</a><br /><br />
```

The image below is what the user will see when they visit the Menus page.



5.5.3.3 Book a Table

As one of the main features of the website, the reservation system is built using a form. For this particular feature, Django's ModelForm class was used. This can be seen in Figure 33.

```
class BookingForm(ModelForm):
    date = forms.DateField(widget = forms.DateInput(format='%d/%m/%Y'), attrs = {'class': 'datepicker', 'placeholder': 'Select a date'})
    class Meta:
        model = Booking
        fields = ['booking_name', 'email', 'phone_number', 'number_of_people', 'location', 'time_slot', 'date']
```

Figure 33 – ModelForm class for the Booking Form

This ModelForm is then used in a function (Figure 34). The form is then used in the HTML file for the booking web page. If the user is performing a “GET” request then it will display the web page with the form. If it is a “POST” request then the completed form will be saved in the database. After successful submission of the Booking form, the user will be redirected to a confirmation page with details of their booking (Figure 35).

```
def booking(request):
    if request.method == 'GET':
        form = BookingForm()
    else:
        form = BookingForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('booking_confirm')
    return render(request, "app/booking_form.html", {'form': form})
```

Figure 34 – Function to display & save Booking Form

Book a table

Thank you for booking a table at one our restaurants!

Your booking details are as follows:

Name:	michal	Number of People:	6
Location:	Aberdeen	Time Slot:	20.00
Date:	Aug. 10, 2017		

Figure 35 – Booking confirmation message & details of booking

5.5.4 Administrator Access

As mentioned in section 5.3.3, Django comes with a built in Admin Site. This was adopted for the Restaurant X website rather than building a new Admin Site from scratch. The Admin Site is accessed via the URL ending `/admin`. The Administrator is presented with an Admin login portal which can be seen in Figure 36.

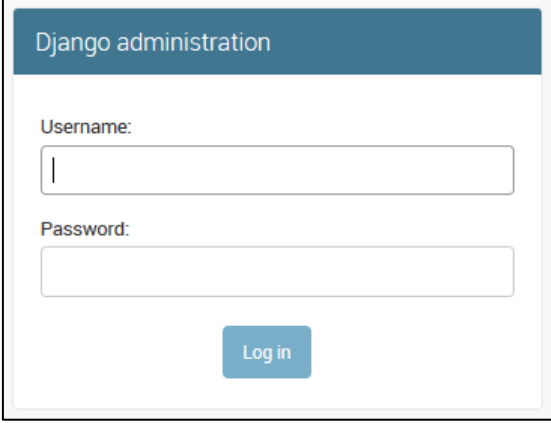
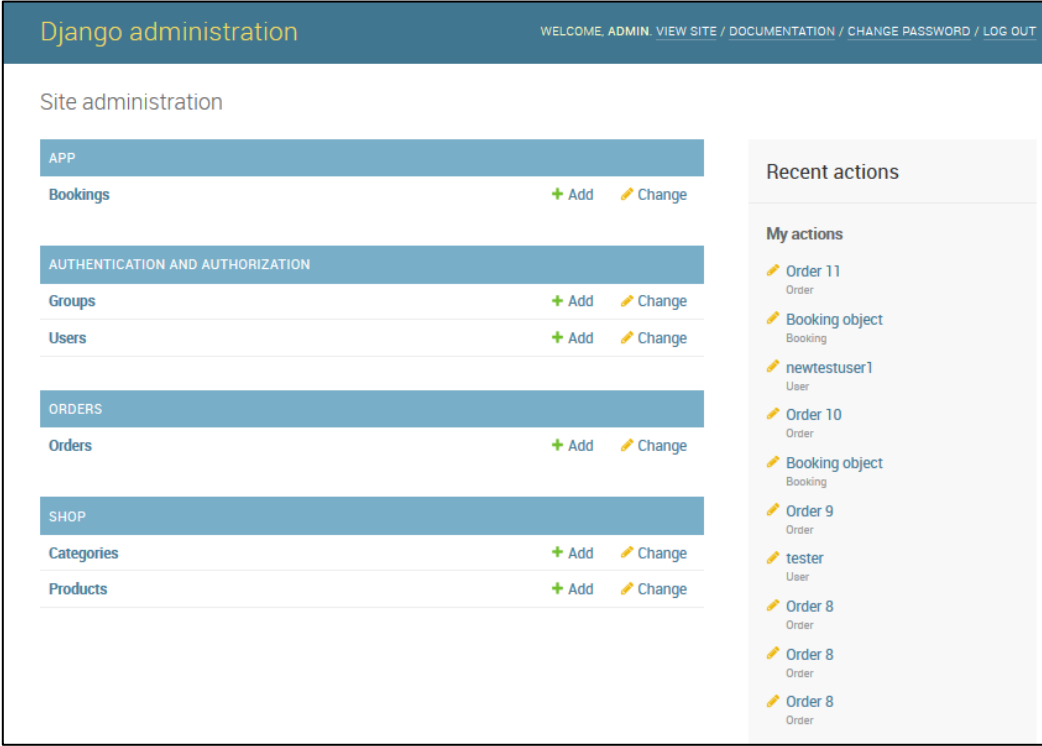
The image shows the Django administration login portal. It has a dark blue header with the text "Django administration". Below the header, there are two input fields: "Username:" and "Password:". The "Username:" field has a cursor inside. Below the "Password:" field, there is a blue "Log in" button.

Figure 36 – Admin Site login portal

The administrator will then be presented with a dashboard (as shown in the image below) where they have full CRUD (create, read, update, and delete) capabilities. The dashboard can be seen in the image below. Any CRUD operations that occur in the Admin Site will be saved into the SQLite database. For the purposes of the Restaurant X website, the areas entitled Users, Bookings, Orders, and Product would be of most interest. An instructional manual has been created for administrators to learn how to use the Admin Site which can be found in Appendix 7.

The image shows the Django administration dashboard. At the top, there is a dark blue header with the text "Django administration" on the left and a navigation bar on the right with links: "WELCOME, ADMIN", "VIEW SITE", "DOCUMENTATION", "CHANGE PASSWORD", and "LOG OUT". Below the header, the main content area is titled "Site administration". It contains several sections: "APP" with a "Bookings" link and "+ Add" and "Change" buttons; "AUTHENTICATION AND AUTHORIZATION" with "Groups" and "Users" links and "+ Add" and "Change" buttons; "ORDERS" with an "Orders" link and "+ Add" and "Change" buttons; and "SHOP" with "Categories" and "Products" links and "+ Add" and "Change" buttons. On the right side, there is a "Recent actions" section titled "My actions" which lists a series of actions: "Order 11", "Booking object", "newtestuser1", "Order 10", "Booking object", "Order 9", "tester", "Order 8", "Order 8", and "Order 8". Each action is preceded by a yellow pencil icon.

The Admin Site can display a list of all records that have been saved in the database. Figure 37 shows excerpt of code used to give the Administrator CRUD operations on the Product table. The file imports the Product table from the Models Python file. A class is made called ProductAdmin and within it are operations to display which fields from the table, types of filter the Administrator has access to and which fields are editable simultaneously. In order to make the table and list appear on the Admin site, the code `admin.site.register(Product, ProductAdmin)` is required. Figure 38 shows the result of Figure 37's code excerpt.

```
from django.contrib import admin
from .models import Category, Product

class ProductAdmin(admin.ModelAdmin):
    list_display = ['name', 'slug', 'category', 'price', 'stock', 'available', 'created', 'updated']
    list_filter = ['available', 'created', 'updated', 'category']
    list_editable = ['price', 'stock', 'available']
    prepopulated_fields = {'slug': ('name',)}
    admin.site.register(Product, ProductAdmin)
```

Figure 37 – Example code excerpt for Admin CRUD operations

Django administration

WELCOME, ADMIN VIEW SITE / DOCUMENTATION / CHANGE PASSWORD / LOG OUT

Home > Shop > Products

Select product to change

Action: Go 0 of 10 selected

<input type="checkbox"/>	NAME	SLUG	CATEGORY	PRICE	STOCK	AVAILABLE	CREATED	UPDATED
<input type="checkbox"/>	Vegetable Stew	vegetable-stew	Mains	6.5	1	<input checked="" type="checkbox"/>	Aug. 15, 2017, 7:42 p.m.	Aug. 15, 2017, 7:42 p.m.
<input type="checkbox"/>	Lemonade	lemonade	Drinks	1.45	1	<input checked="" type="checkbox"/>	Aug. 15, 2017, 7:41 p.m.	Aug. 15, 2017, 7:41 p.m.
<input type="checkbox"/>	Coca Cola	coca-cola	Drinks	1.45	1	<input checked="" type="checkbox"/>	Aug. 15, 2017, 7:40 p.m.	Aug. 15, 2017, 7:40 p.m.
<input type="checkbox"/>	Chicken Fajitas	chicken-fajitas	Mains	8.75	1	<input checked="" type="checkbox"/>	Aug. 15, 2017, 7:38 p.m.	Aug. 15, 2017, 7:38 p.m.
<input type="checkbox"/>	Chocolate Cake	chocolate-cake	Desserts	2.5	1	<input checked="" type="checkbox"/>	Aug. 15, 2017, 7:37 p.m.	Aug. 15, 2017, 7:37 p.m.
<input type="checkbox"/>	Vanilla Cheesecake	vanilla-cheesecake	Desserts	4.5	1	<input checked="" type="checkbox"/>	Aug. 15, 2017, 7:36 p.m.	Aug. 15, 2017, 7:36 p.m.
<input type="checkbox"/>	Steak and Chips	steak-and-chips	Mains	14.95	1	<input checked="" type="checkbox"/>	Aug. 15, 2017, 7:34 p.m.	Aug. 15, 2017, 7:34 p.m.
<input type="checkbox"/>	Soup	soup	Starters	3.25	1	<input checked="" type="checkbox"/>	Aug. 15, 2017, 7:33 p.m.	Aug. 15, 2017, 7:33 p.m.
<input type="checkbox"/>	Pate	pate	Starters	4.5	1	<input checked="" type="checkbox"/>	Aug. 15, 2017, 7:30 p.m.	Aug. 15, 2017, 7:30 p.m.
<input type="checkbox"/>	Salad	salad	Starters	3.5	1	<input checked="" type="checkbox"/>	Aug. 15, 2017, 6:23 p.m.	Aug. 15, 2017, 8:35 p.m.

10 products

Save

ADD PRODUCT +

FILTER

By available

All

Yes

No

By created

Any date

Today

Past 7 days

This month

This year

By updated

Any date

Today

Past 7 days

This month

This year

By category

All

Desserts

Drinks

Mains

Starters

Figure 38 – Result of figure 37's code excerpt

Chapter Six – Evaluation

“NASA-TLX is a multi-dimensional scale designed to obtain workload estimates from one or more operators while they are performing a task or immediately afterwards” (Hart and Field, 2006). As a need for a quantifiable method to measure “workload”, Hart et al created the NASA TLX in 1988 and consists of six subscales to measure workload: Mental, Physical, and Temporal Demands, Frustration, Effort, and Performance. The task involves two parts – weighting and rating.

1. Weighting section – participants are shown 15 subscale comparison cards. The participant is asked to compare 2 subscales on each card and tell the researcher which one was most important to their experience during the task. As a way to reduce paper usage, the 15 subscale comparisons were combined into a 3 by 5 table and participants were asked to circle the subscale that was most significant to them in each box. This can be seen in Appendix 8.
2. Rating section – this section shows each participant the 6 subscales and asks them to mark an X on the appropriate point that they feel matches their experience. Each line has two points from low on the left to high on the right. Subscale Performance has a scale of good on the left and bad on the right. The scale increments by 5 on each line. Scale is 0-100. This can be seen in Appendix 9.

The weighting and rating are multiplied for each subscale to form an Adjusted Rating for that subscale. Then all Adjusted Ratings are totalled up and the total is then divided by 15 to gain the Task Load Index score or rating for that task. The template used to calculate the scores can be seen in Appendix 10.

6.1 Tasks

To evaluate the website for Restaurant X, five participants were recruited to conduct a series of eight tasks. These tasks were as follows:

Task	Instruction
User – Register as a member	To register themselves on the website
User – Login to website	After registering themselves, the participants were logged out and were then required to log back in to the website
User – Submit a contact form	Participants were required to fill out the contact form and submit it
User – Booking a table	The participants were required to visit the “Book a Table” page and fill in the form and submit their request to book a table
User – Make an order online	The participants were required to visit the “Order Online” page and add items into their cart then proceed to visit their cart and check out
Admin – Users	The participants were asked to act as an administrator and visit the Admin site. They were then asked to find the list of current registered users and edit the username and email address of one of the users
Admin – Bookings	Acting as an administrator, the participants were asked to visit the list of bookings. They were asked to change the number of people and the time slot on one of the bookings
Admin – Orders	Acting as an administrator, the participants were asked to visit the list of orders. They were asked to change the address and tick the “paid” option on one of the orders

6.2 NASA TLX

Once participants had completed each task, they were asked to complete the two parts of the NASA TLX. Figure 39 gives a visual representation of the average score of each task of all 5 participants.

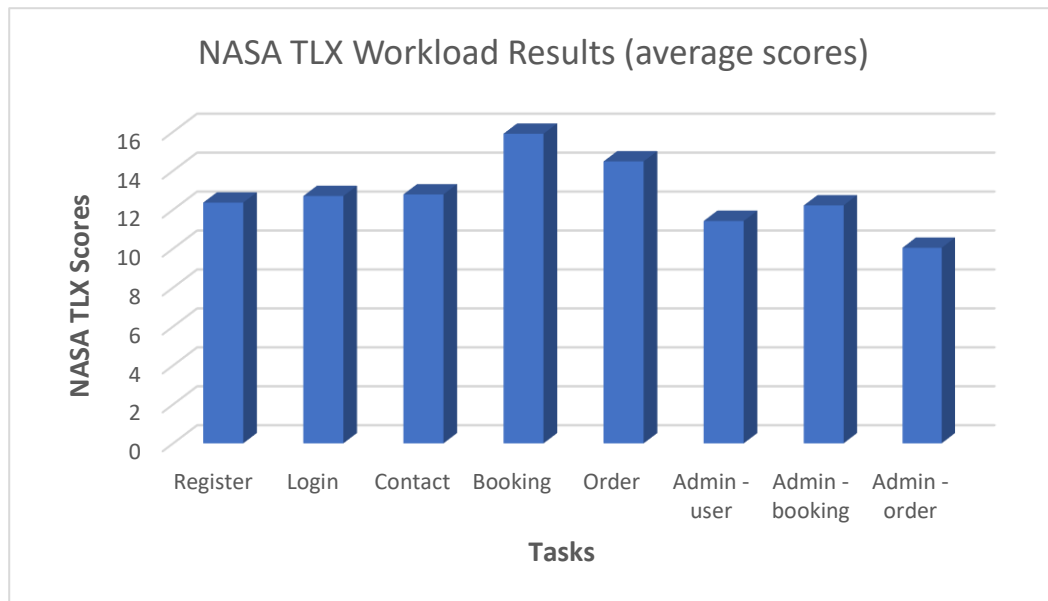


Figure 39 – Graph of average scores of each task for all participants

For the purposes of this project, the researcher has deemed any task with a score below 20 to be as acceptable. It can be seen from the graph that the majority of tasks scored around 12. The highest score was 15.9 (Booking) while the lowest score was Admin – order with a score of 10.04. As all average scores are below 20, the tasks given to participants were relatively easy to complete. This shows a positive sign as users were able to fulfil the requirements specifications and use cases set out in Chapter 3 and it is an indication that the website is intuitive and easy to use.

6.3 Additional Feedback

As well as completing the NASA TLX tasks, participants were also given the option to provide comments and feedback on any of the features of the website they had come across. Two participants made a comment that there did not appear to be a confirmation after a booking was made therefore giving no indication if the booking had been processed and accepted. This was an oversight whilst developing the booking system section and therefore a confirmation page was implemented to provide an indication to the user that their booking had been successfully processed.

One participant had mentioned that during the “make an order online” task that they would prefer if the user was redirected back to the list of products rather than directed straight to the cart after each item was added. This could be seen as frustration for the user as it did mean extra steps to get back to the product list just to add another product to their cart. Therefore, this was changed so that users would be redirected back to the list of products page after an item was added to the cart.

Chapter Seven – Appraisal

7.1 Design Decisions

7.1.1 Authentication of User

Many websites offer features and restricted access areas to users who are authenticated and therefore require users to be registered on their website. This was adopted for Restaurant X with the restricted areas mentioned in Chapter Five. With the user logged in, i.e. authenticated, the navigation bar menu items change. The “Log in” option changes to a “Hello {username}” and acts as a dropdown menu for the “User Profile” webpage and “Log Off” option. The “Cart” menu item also appears to the authenticated user which they can access to see any items that they have added to their cart. This can be seen in Figures 40 and 41.

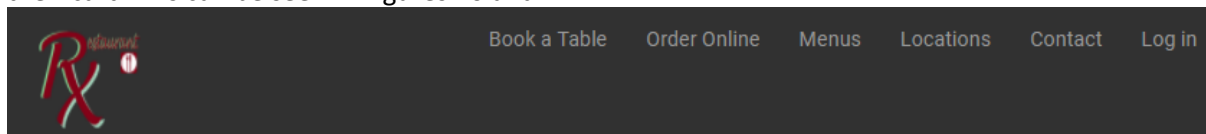


Figure 40 – Non-Authenticated User Navigation Bar View

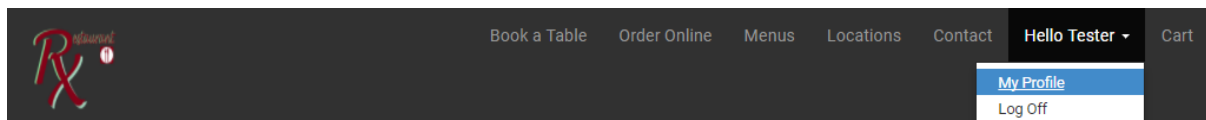


Figure 41 – Authenticated User Navigation Bar View

7.1.2 User Profile

As with most websites, if a user has been required to register on a website they should be able to view their profile which displays their personal information. Therefore an option was implemented for the user to view their profile. This was built into a dropdown menu in order to save space in the main navigation bar. This can be seen in Figure 42.

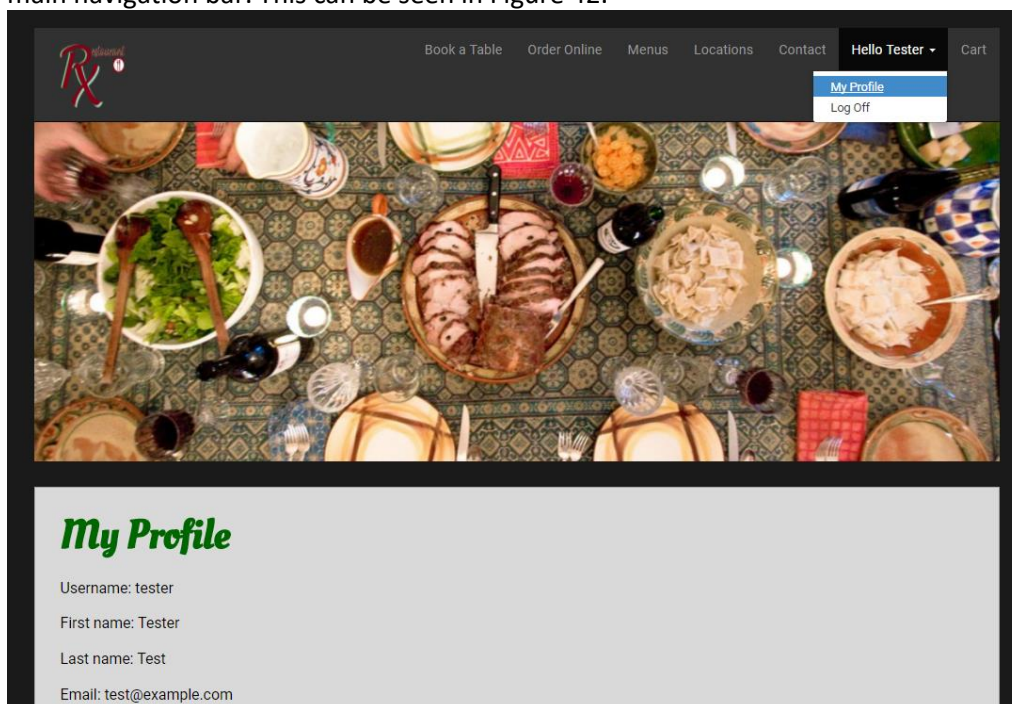


Figure 42 – Example of Profile view

7.1.3 Google Maps API

To aid the user in locating each branch of Restaurant X, a Google Maps API was implemented in the Locations webpage. Each branch's address was selected to be located inside a shopping centre and the longitude and latitude coordinates of these shopping centres were found. These coordinates were used for the JavaScript code in order to display an interactive Google Map for each branch including a marker to pinpoint the exact location. An example of the JavaScript can be seen in Figure 43 while the results of the code can be seen in Figure 44 for the Aberdeen branch.

```
var mapaber = { lat: 57.142762, lng: -2.09616 };  
var map1 = new google.maps.Map(document.getElementById("mapab"), { zoom: 12, center: mapaber });  
var marker1 = new google.maps.Marker({ position: mapaber, map: map1 });
```

Figure 43 – JavaScript code to implement Google Maps

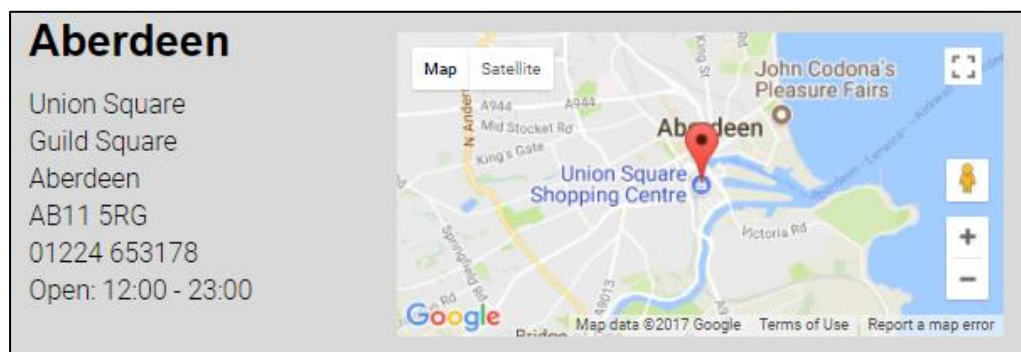


Figure 44 – Result of JavaScript in Figure 43

7.1.4 Calendar Date Picker

As the online Booking system requires the user to select a date as part of the Booking form, in order to make the process of picking a date more dynamic, a Calendar Date Picker feature was implemented using JQuery. The calendar view is more user friendly and displays days of the week as well as the date. The user simply selects a date and the Date field will automatically populate in date format for the user. This can be seen in Figure 45 below.

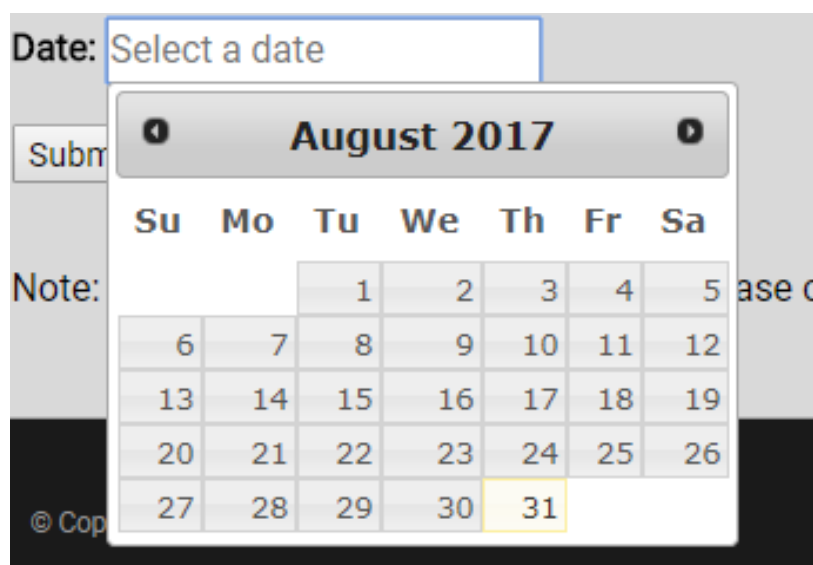


Figure 45– JQuery Calendar Date Picker

7.2 Changes to initial designs and requirements

7.2.1 Use Case Specifications

Use Case Specification 3 (Book a table online) initially asked the user to fill in three fields – number of people, time, and date. During development, this did not appear to be sufficient information from the user therefore additional fields were added including – booking name, email, phone number, and location.

Initially in Use Case Specification 7, 8 and 9, the administrator was to use the same login portal as the customer to the main website to gain access to the back end of the website in order to perform CRUD operations. With the built-in feature of Django, as mentioned in Chapter Five, the administrator has their own login portal to manage all users, bookings and online orders.

7.2.2 Online Order System

The cart for the online ordering system had initially been designed to be displayed on the same page as the list of products so that users could keep track of what they have ordered (Figure 4). However due to time constraints this could not be implemented and therefore a separate web page was built instead to list all added items.

7.2.3 Highlighting the current page

As mentioned in Chapter 4, a recommendation was made to highlight the navigation menu item in order to let the user know where they were in the website. This was a simple yet useful feature to implement using Bootstrap's `class="active"` syntax on an unordered list item. The code and result of the feature can be seen in Figures 46 and 47 below.

```
<li class="active"><a href="{% url 'booking' %}" title="Book a Table">Book a Table</a></li>
```

Figure 46 – Navigation bar menu code to highlight current page

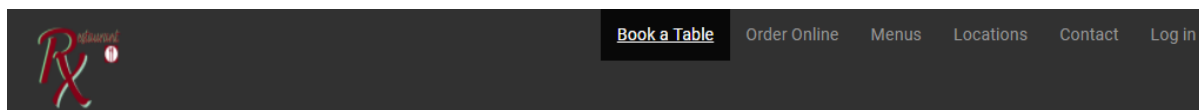


Figure 47 – Result of code in Figure 46

7.3 Wider Context

The aim of the Restaurant X website was to create a website with an online reservation system and online food ordering system. This project and its website can be used in a wider context in future for the following reasons:

- The website contains generic web pages such as Contact and Locations which are easily adaptable to any project. Additional webpages could be added as part of the navigation bar.
- The booking system could be changed and adapted for an appointments or meetings booking system.
- The online shop lists all products available sorted by categories. The types of products and categories can all be changed to reflect the type of products on offer by any company.
- Each product's name is hyperlinked to a more detailed page where a description can be found. There is also the option to add images to each detailed page to give the user a visual representation of the product.

7.4 Self-Appraisal

While a schedule was produced at the start of this project, as mentioned in Chapter Three, during the course of this project, the schedule proved difficult to follow and milestones were not always met on time. This was, for the most part, due to unforeseeable difficulties while developing the website. This included importing pre-existing HTML files with finalised stylings applied to the Django Project. Once imported, some stylings of the HTML were incorrect due to the path files to where Django stores its CSS files.

It was also decided that the Bootstrap library would be adopted into the project during mid development in order to make the website mobile friendly. This proved to be a difficult adoption due to the pre-existing stylings within the project meaning restyling was necessary for the second time in the project. It would have been advantageous to have adopted Bootstrap from the beginning to avoid restyling again.

While these difficulties were overcome, this did lead to delays in developing and implementing other features of the website. Having gone through the whole process of this project, with hindsight, it would have been beneficial to have implemented more slack time to certain parts of the project to accommodate unforeseeable situations that caused a delay and also set out longer durations for the bigger features such as the online ordering system as these took longer to develop.

Despite all the issues encountered, this project gave me the opportunity to learn a new skill of developing a fully operational, dynamic website and database, to learn new programming languages, and to experience working with a Framework like Django. With these skills, I was able to successfully implement an online booking and food ordering system which test participants were able to successfully use.

Chapter Eight – Conclusion

The aim of the project was to create an online medium in the form of a website for a fictional restaurant chain named Restaurant X. The website was to include two main features - an online booking system to reserve a table and an online food ordering system for home delivery to users.

A website was produced using Django – a framework which uses Python as its programming language. The website has a fully operational database system using SQLite.

As well as the two main features mentioned above, the website also allows administrators their own login portal to an Admin Site which they can perform CRUD operations to user accounts, bookings and online orders. The Admin Site also allows the administrator to add categories and products which are dynamically added to the website.

Testing was conducted using the NASA TLX with 5 participants who were all asked to perform 8 tasks. Each participant was then required to fill in a weighting and rating scale for each task. All participants found the tasks to be easy as demonstrated by their overall average TLX scores which is a good indication the website is intuitive to use.

The final website provides users with a range of easy to use features, for both customers and administrators, but also has room for future improvements.

Chapter Nine – Recommendations for Future Work

The project was part of the MSc Applied Computing course and therefore had a time restriction of approximately 4 months to learn, research, develop and implement the project. If the time frame had been longer than the 4 month period, the following features would have improved the website and heightened the user experience:

1. Making use of a “Custom User Model” rather than Django’s default user model to generate own fields including setting email address as username and including the capability for users to register other personal information, for example an address, into the database.
2. Booking System
 - Have a dynamic function to display available time slots for the day.
 - Automatically send an email confirmation of their booking to the customer.
 - The date format of the Calendar Date Picker could be changed to region specific for example in UK the date format could be changed to DD/MM/YY.
3. Online Ordering System
 - Implement a confirmation message to appear after each item has been added to the cart.
 - Auto populate customer details in the checkout phase since they will have already logged in. This could be achieved with the implementation of custom user model mentioned above.
 - Implement an online payment system as this would fully satisfy Use Case Specification 6 (Make a food order online) in Appendix 2.
 - Implement a confirmation page to include details of the user’s order, total price and personal details after payment has been made.
4. Login Page
 - Adding a “Forgotten password” option to the Login page.
 - Add error messages to login fields if they type in their username and/or password incorrectly – for example “incorrect username and/or password”.
5. The Contact Form email to administrators could use an HTML template in order to display the Contact Form message in a more coherent method than the one currently used.

With more time, a comparative testing could have been conducted against other similar websites with online booking systems and online food ordering systems. This would gain an understanding of which existing features of Restaurant X could be improved and any new features that were not previously considered that could be added or replace existing features.

References

A. Holovaty & J. Kaplan-Moss (2009). *The Definitive Guide to Django: Web Development Done Right*. USA: Apress.

Django Software Foundation. (2005-2017). *Django's role in forms*. Available: <https://docs.djangoproject.com/en/1.11/topics/forms/>
Last accessed Aug 2017.

Django Software Foundation. (2005-2017). *Django's security policies*. Available: <https://docs.djangoproject.com/en/1.11/topics/security/>
Last accessed Aug 2017.

Django Software Foundation. (2005-2017). *The Django admin site*. Available: <https://docs.djangoproject.com/en/1.11/ref/contrib/admin/>
Last accessed Aug 2017.

Hungryhouse (2017). *About*. Available: <https://hungryhouse.co.uk/blog/about/>
Last accessed Aug 2017.

J. Arnowitz, M. Arent, N. Berger (2007). *Effective Prototyping for Software Makers*. USA: Elsevier. 3-4.

Just Eat (2017). *About us*. Available: <https://www.just-eat.co.uk/about>
Last accessed Aug 2017.

Just Eat (2016). *Annual Report*. Available: <https://www.justeatplc.com/investors/results-reports>
Last accessed Aug 2017.

L. Wroblewski (2011). *Mobile First*. USA: Ingram.

Office for National Statistics (2017). *Internet users in the UK: 2017*. Available: <https://www.ons.gov.uk/businessindustryandtrade/itandinternetindustry/bulletins/internetusers/2017>
Last accessed Aug 2017.

O. White (2015). *Is Just Eat Really the Best Option For Your Takeaway?* Available: <https://www.preoday.com/blog/just-eat-really-best-option-takeaway/>
Last accessed Aug 2017.

S. Hart & M. Field (2006). *NASA-TASK LOAD INDEX (NASA-TLX); 20 YEARS LATER*. USA.

T. J. Teorey (2011). *Database modeling and design: logical design*. 5th ed. Amsterdam: Burlington. 32.

w3 Schools. (1999-2017). *SQL Injection*. Available: https://www.w3schools.com/sql/sql_injection.asp
Last accessed Aug 2017.

Appendices

Appendix 1 – Restaurant X – Requirements Specification

Appendix 2 – Restaurant X – Use Case Diagram & Actors

Appendix 3 – Restaurant X Timeline

Appendix 4 – Information Sheet

Appendix 5 – Consent Form

Appendix 6 – Evaluation Survey of Axure Prototype

Appendix 7 – Administrator Site User Manual

Appendix 8 – NASA TLX Weighting Section

Appendix 9 – NASA TLX Rating Section

Appendix 10 – NASA TLX Score Results Template