

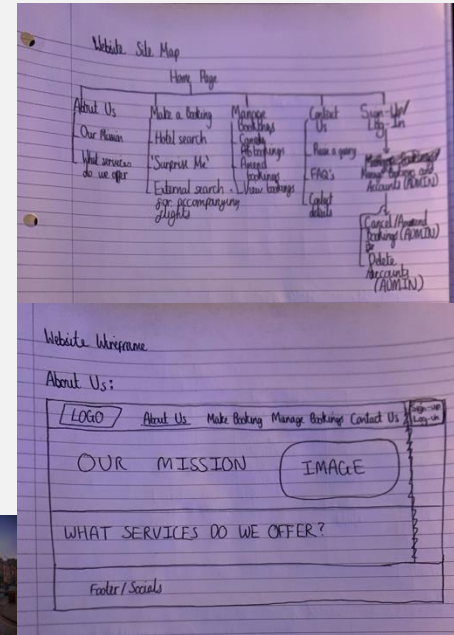
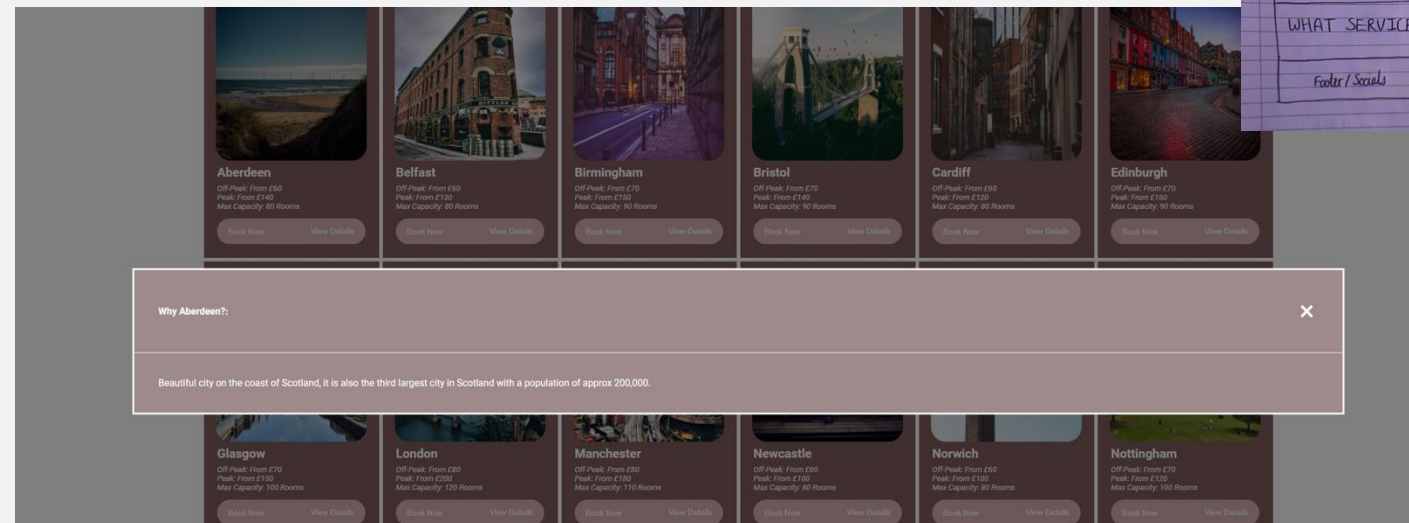
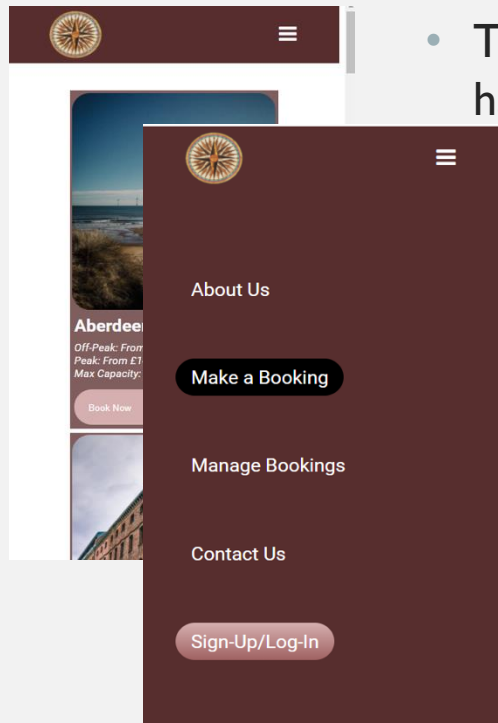


HORIZON HOTELS WEBSITE PROJECT

By Arthur Milner 21035478


WEBSITE DESIGN: HTML, CSS, JAVASCRIPT

- The website design began with site maps and wireframes before I eventually took the development to the stage of actually coding in some HTML.
- Once the base HTML had been finished I researched colour palettes and website styles in an attempt to create a visually appealing and interactive website using CSS and JavaScript.
- The majority of styling is done in CSS and JavaScript was mainly used for the hamburger menu and the modals present on the make a booking page.



DATABASE DESIGN: REQUIREMENTS

- To start designing my database I first made sure to list the requirements of the database and from these requirements identify fields and entities.
- The main requirement is for the database to efficiently store user accounts and the bookings those users make, along with all the details of those bookings.
- It is also important for it to be able to calculate the information on these bookings such as booking price using other information stored within the database.
- Things such as hotel capacity must also be calculated from information within the database.



DATABASE DESIGN: NORMALIZING THE DATABASE

- Unnormalized Form: Firstly I listed all my fields in a single table, I also started to identify functional dependencies to assist me in moving towards 3rd normal form.
- 1st Normal Form: For first normal form I identified the repeating tables and removed them into a separate relation, leaving a foreign key, and also ensuring all fields were atomic.
- 2nd Normal Form: For second normal form I identified fields which were partially dependant, meaning they were dependent on only one primary key of a composite key, once these fields were identified I removed them into separate relations, using foreign keys to maintain links. I also made sure the non-primary key attributes were fully functionally dependent on the primary key.
- 3rd Normal Form: Here I identified transitive dependencies, this is to make sure fields can be determined by their primary/composite key only, these are again removed and placed into a separate relation.

Why normalize this data?:

- It is important for this data to be normalized as, whilst creating more tables can perhaps make the database look more complex, it in fact has the opposite result as normalization will minimise many potential errors. These errors include high amounts of data redundancy and update/insertion/deletion errors as the way unnormalized data is stored in a database can make modifications a very lengthy process. Finally normalized data means greater data integrity, which is particularly important in large databases.

DATABASE DESIGN: DATA DICTIONARY

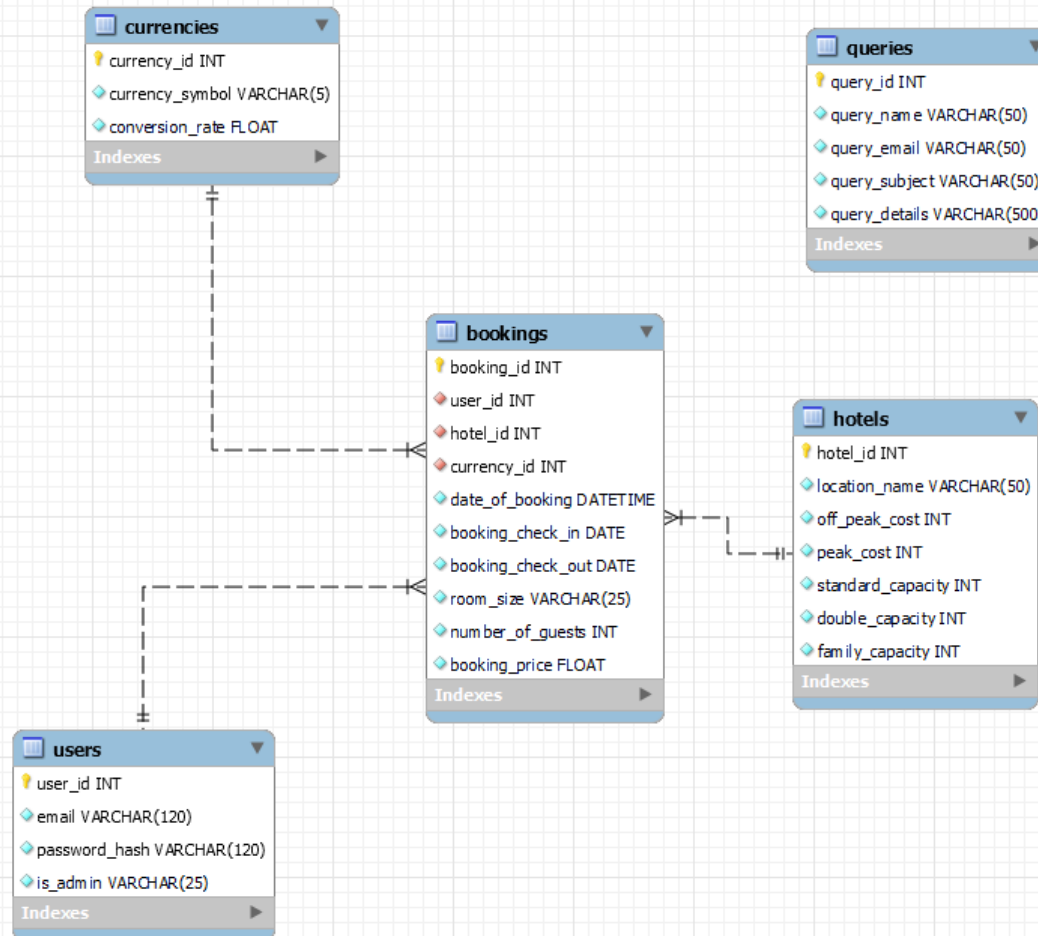
- Within the database dictionary the domains are given for each of the fields within my database, the data dictionary was my reference point when creating the actual database/website as the constraints are nicely listed ready to be coded in.

Entity	Field Name	Description	Domain Definition
Users	<u>userID</u>	Set of all user ID	Character: size 10, range 1000 onwards, not null, PK, auto increment
	<u>userEmail</u>	Email attached to a user account	Character: size 50, not null
	<u>userPassword</u>	Password attached to a user account	Character: size 20, not null
	<u>isAdmin</u>	Check box to declare whether the user is admin	Yes/No
Hotels	<u>hotelID</u>	Set of all hotel ID	Character: size 10, range 100 onwards, not null, PK, auto increment
	<u>locationName</u>	Name of the location which the hotel is based	Character: size 30, not null
	<u>standardCapacity</u>	Remaining capacity of double rooms of the hotel in this location	Integer: size 3, not null
	<u>doubleCapacity</u>	Remaining capacity of double rooms of the hotel in this location	Integer: size 3, not null
Rooms	<u>roomID</u>	Set of all room ID	Character: size 10, range 100 onwards, not null, PK, auto increment
	<u>roomSize</u>	"Standard", "Double", "Family"	Character: size 10, not null
	<u>roomPrice</u>	Lowest price of the type of room across all the hotels	Integer: size 3, not null
	<u>bookingID</u>	Set of all booking ID	Character: size 10, range 1000 onwards, not null, PK, auto increment
Bookings	<u>dateOfBooking</u>	Date on which the booking was made	Date: size 11, not null, dd/mm/yyyy
	<u>bookingCheckIn</u>	Date of check-in for the booking	Date: size 11, not null, dd/mm/yyyy
	<u>bookingCheckOut</u>	Date of check-out for the booking	Date: size 11, not null, dd/mm/yyyy
	numberOfGuests	Number of guests on the booking	Integer: size 1, not null, value 1, 2, 3, 4, 5, 6

Currency	<u>roomSize</u>	"Standard", "Double", "Family"	Character: size 10, not null
	<u>roomPrice</u>	Lowest price of the type of room across all the hotels	Integer: size 3, not null
	<u>bookingPrice</u>	Price of the booking, will be calculated from room size, number of guests, if a night is in peak or off-peak season, how far in advanced the booking was made and the number of nights stay	Integer: size 10, not null
	<u>currencyID</u>	Set of all currency ID	Character: size 10, range 1 onwards, not null, PK, auto increment
Currency	<u>currencySymbol</u>	Symbol for the currency	Character: size 1, not null, value "\$", "£", "€"
	<u>conversionRate</u>	Conversion rate from the currency into GBP, GBP conversion rate will be set to 1	Integer: size 3, not null

DATABASE DESIGN: ER DIAGRAM

- Here the relationships between the relations are clearly shown, as well as the domains placed upon the fields.



127.0.0.1:5000 says

Warning: By using this site you agree to the use of sessions in order to store data such as account details and user bookings. Please view our privacy statement for more information.

OK

Warning message on the use of sessions/cookies

LESP: LEGAL, ETHICAL, SOCIAL & PROFESSIONAL

- **Licensing Concerns:** All photos within my website were obtained from free to use sources, they are all available to use without giving credit to the author or the source of the image, the websites I used to find these images includes unsplash.com and pexels.com. This is to prevent any legal issues which could leave me in violation of copyright laws, resulting often in hefty fines or worse. The text of the website is also free to use to again prevent those same legal issues.
- **Ethical Data Storage:** The website ensures ethical data storage as upon a users first visit they are greeted with an alert informing them of the websites use of Flask session and how in using the website they consent to this usage, the message is also transparent towards how any stored information will be used, this is in compliance with both internet standards and the law. The amount of data stored on a user is also ethical as it is essentially the bare minimum required in order to carry out the functional side of the website, data on a user is not stored past the point it is no longer needed.
- **Security Measures:** I have considered security issues and standard constantly in the creation/design of my website, one example of this is the format of my SQL statements, they have been crafted in such a way that prevents SQL injections and any valid SQL that might be passed through by a user to the database will be sanitised. Further examples of security measures within my website includes hashed password, meaning that should a hacker gain access to my database the passwords would still be encrypted and consequently very hard to make human readable. I have also utilized the escape feature in Flask just to further ensure any user inputs remain sanitised and prevent issues such as an XSS attack. For one final example you can look at the fact a users payment card information is not stored anywhere on the website/database, again making any unauthorized access to the websites contents somewhat useless.
- **Accessibility Concerns:** The text on my website is easy to read, it uses a very widely used font type, is of a decent size and is generally either white text on a dark background for black text on a light background. The website itself is also very easy to understand as it just utilises a simple menu at the top of the screen and should a user encounter any errors they are always informed in a descriptive manner, through this the site is hopefully much more accessible for the cognitively impaired. It is also worth noting the website maintains its accessibility across multiple device sizes, upon encountering a small screen size the website automatically increases font size, decreases image sizes and changes the layout of the website to a more vertical setup. The website can also be accessed using only the keyboard by using the tab key to cycle through the buttons/text fields of the website, this is again to improve accessibility for those who are cognitively impaired. All images on the website also have an attached descriptive alt text for the visually impaired, the website also ensures all forms have easy-to-read labels above each input field. The contact us page of the website also helps improve accessibility as it helps those who might be struggling to understand the website with an FAQ section and a form in which they can submit a custom query to be reviewed by an admin. The colours of the website have also been carefully selected in order to prevent them from colliding with each other to improve readability. I have also implemented a dark mode into my website which can help those with sensitive eyes who prefer a darker background.
- **Environmental Ethical Issues:** I have helped reduce the environmental footprint of my website by using compressed JPEG images and designing my website in a way which does not rely on high quality/long videos which can cost the planet a large amount of pollution in the long term. Should I host my website on the internet I would also heavily consider a more eco-friendly/green web hosting to further reduce the websites footprint.

```
userAndBookingDataset = (session.get("CurrentUserID"), bookingID)

getBookingPriceStatement = "SELECT booking_price FROM bookings \
                             WHERE booking_id = %s;" #Gets price of the booking being cancelled
```

Format of SQL statements to prevent SQL injection

```
password_hash = generate_password_hash(request.form["new-password"], method="sha256")
```

4	4	hello@gmail.com	sha256\$Pe56n69aXz6Wl	User
---	---	-----------------	-----------------------	------

Example of a stored hashed password



LESP (PROFESSIONAL): FOLLOWING A TYPICAL SDLC

- **Requirements Definition and Analysis:** Here I studied the assignment description as best as I could in order to achieve the greatest understanding of the skills required to create the website, the functionality required within the website and begin planning how I will go about the creation of the website with rough plans such as site maps/wireframes.
- **System Architecture and Design:** Here I began make more precise plans for my website and even drawing up the methods I will use in order to bring the site to life. By going to my lectures/practical's and through online resources such as w3c I had all the content I required to learn the skills I had identified previously. I then began working on the rough HTML/CSS in order to create a draft of a non-functioning GUI. The design of a database to store all required information in the 3rd normal form was also achieved within this step.
- **Implementation and Unit Testing:** Here I finished up my HTML/CSS by referring to my previous designs and made sure it all looked clean and there were no visual or accessibility issues, I also added some basic JavaScript to create some extra visual features. Similarly, I finished up my database and again ensured it all functioned correctly, for example guaranteeing the SQL queries run successfully and features such as check statements only allowing valid entries into the database. Finally, I put all the contents together using Flask, leaving me with the full website ready to be intensely tested, before testing the full website however, I made sure the functions within my Flask app all worked individually in separate python files before implementing them into the full web app. Of course, if I encountered any errors with any features in the website along the way I would retrace my steps to a previous stage of the SDLC and make the appropriate fixes.
- **System Testing:** Here I tested every function/feature of the full website on many different screen sizes to ensure to the best of my ability the website functions correctly, this means testing all of the HTML/CSS/Flask/MYSQL/JS in tandem.
- **Operations and Maintenance:** This is the current stage of my website, here I am just making sure the website operates smoothly and monitoring any issues/user queries that might come up. I could also identify potential improvements and begin to implement them by going back to previous stages of the SDLC.

EVALUATING MY WEBSITE: TESTING THE WEBSITE

- I believe the website meets the criteria required within the assignment specification. To ensure this as well as I could I have undergone testing for each feature of the website such as the booking price being accurate, etc. Beyond this testing I have also scattered print statements throughout my flask code to confirm the data being passed through my website is storing correctly and this has also allowed for easier debugging of the website during creation. Below are a few of my test cases:

Test Case	Expected Result	Actual Result
Users are stored with hashed password in database	User password is stored as hash	As expected
Admin account redirects to admin page	Redirected to admin page once logged in	As expected
User/guest is unable to access the admin page	User/guest is unable to access the admin page	As expected
Page layout changes depending on screen size	Page layout changes depending on screen size	As expected

EVALUATING MY WEBSITE: POSSIBLE IMPROVEMENTS

- If I were to start a project of this size again I would certainly make sure to use some form of version control such as GitHub, it would make debugging/changing the project much less stressful as I can find comfort knowing all previous versions have been stored somewhere should something go wrong.
- Further improvements could be to use Flask extensions such as Flask-login, Flask-admin or WTForms as they are generally considered more elegant solutions to the issues they solve.
- Security improvements could include a captcha upon sign-up or submission of a query to prevent bots.