Task 1 – Data Requirements and Test Strategy

Data Requirements

There will be a number of elements that the website will need to store as data in order to function correctly. Some of this data will be tied to the users, such as in the form of creating an account, whilst others will be used to update certain web-pages based on how users interact with it. Currently, two pages will use data in these ways:

- Account Page – accept user information including:
  - Email address
  - Password
  - Date of Birth
  - Location (from set options)
- Location Page – set data for each available location:
  - Location Name
  - Air Quality Index
  - Pollen Count
  - Temperature

Below are examples of each table, as well as Data Dictionaries for each of them.

Account Table

AccountID	LocationID	Email	Password	DOB	Location_Name
1	2	john@gmail.com	PinkMountain19!	05/09/1998	London
2	3	Lsmith19@outlook.com	ThunderHut55*	23/08/2001	Basildon

Data Dictionary

Field Name	Data Type	Format	Field Size	Description	Example
AccountID	Text (Primary Key)	#####	6	Unique identifier for each account	12345
LocationID	Text (Foreign Key)	#####	6	Unique identifier for each location	12345
Email_Address	Text		20	User's email	john@gmail.com
Password	Text		15	User's password	PinkMountain19!
Date_of_Birth	Date/Time	DD/MM/YYYY	10	Date user was born	05/09/1998
Location_Name	Text		15	Location from set options	London

For the account table, all information must be not-null, except for location which is optional. There will also be validation rules for the email address to check for a correct domain and for appropriate symbols (e.g. presence of '@'), and a number of rules to check the strength of the password – length check, check for uppercase and lowercase, symbols etc.

Location Table

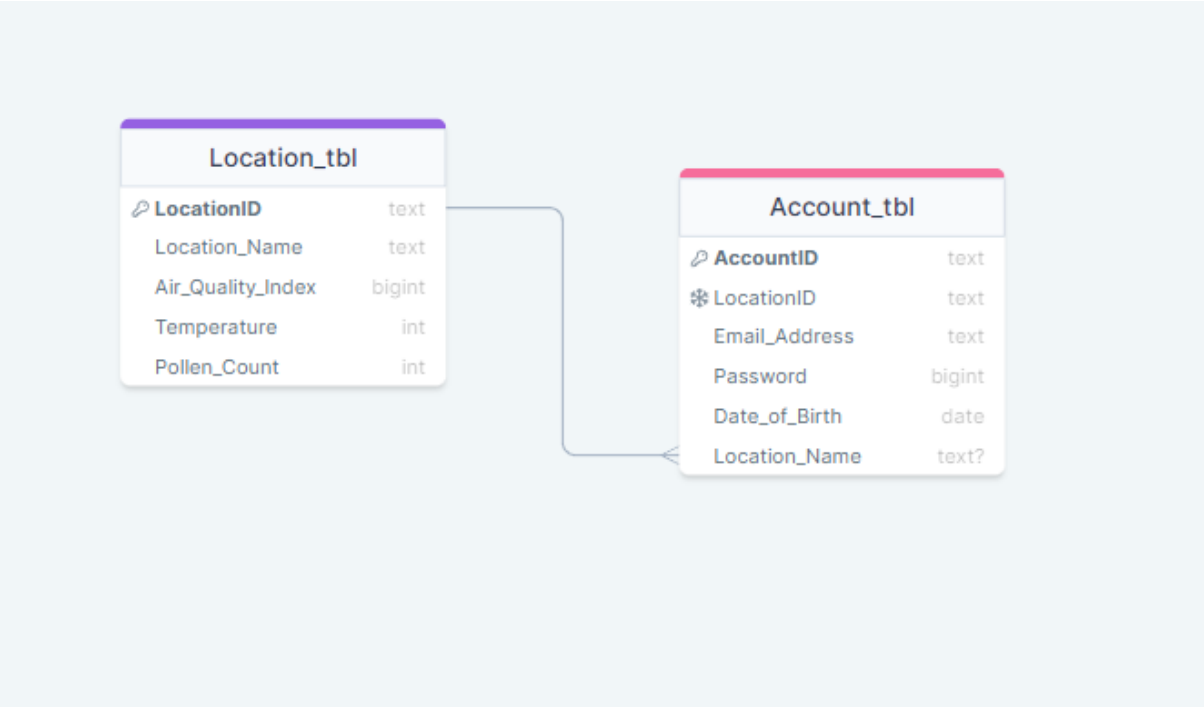
LocationID	Location_Name	Air_Quality_Index	Temperature	Pollen_Count
2	London	8	13	50
3	Basildon	4	12	65

Data Dictionary

Field Name	Data Type	Format	Field Size	Description	Example
LocationID	Text (Primary Key)	#####	6	Unique identifier for location name	12345
Location_Name	Text		20	Name of a given town/city	London
Air_Quality_Index	int		5	Value between 1-10, refers to pollution level	10
Temperature	int		2	Temperature of location in Celsius	15
Pollen_Count	int		4	Grains of pollen per cubic metre	50

For the Location table, it is used as both a foreign key for the account database, as well as in order to update the information on the location page about the weather of each town or city. In more efficient versions of this website, this database could instead be replaced by an API that auto-updates weather information.

Entity Relationship Diagram



This ERD shows that between these two tables, the information of Location_Name will be transferred. It has a one-to-many relationship as one location can feature across many accounts.

Test Strategy

During development, tests will be segmented into the following categories:

- Unit Testing – testing each feature of development (end of each main sprint)
- Integration Testing – testing how multiple features/versions work once combined
- Acceptance Testing – testing to see how well the solution meets the user acceptance criteria

These three categories concern particular stages of development – unit and integration testing will be staggered throughout development as they focus on when a particular unit is completed and when multiple units are integrated together. Acceptance testing will take place at the end stages of development, as the focus now changes to getting user feedback and making adjustments. The types of tests carried out in these stages may also differ, as well as a variety of potential test strategies including:

1. White Box Testing

Def – Testing strategy that involves having the source code to hand, where structural testing is the goal. The aim is to test all manner of inputs, including erroneous data and inputs that breach validation for certain inputs.

This type of testing would be used particularly during unit testing to verify that key features work effectively – e.g. testing validation for email addresses, entering emails that shouldn't work such as with multiple '@' symbols, false domains etc. It may also be used during integration testing to make sure that features continue to work when brought together.

By testing all data that is expected to be invalid, if any inputs incorrectly pass through then the validation rules can be adjusted to make sure they are more accurately written and properly deny

invalid inputs. This testing may also showcase any problems with the way the code is structured and any semantic errors, as well as identifying where the code can be shortened and made more efficient.

2. Black Box Testing

Def – Testing strategy similar to white box, but instead involves testing from an outside view – no access to the source code, and only entering the expected inputs from a user perspective.

This type of testing would occur towards the end of unit testing and integration testing, as well as a part of the run up to acceptance testing. This would help verify that expected inputs are not being wrongly denied, as well as verify that prior to entering user hands, the necessary features are working as intended.

This type of testing is likely to uncover any areas where validation rules are too broad, particularly if they use incorrect logic to keep invalid inputs out, but by proxy keep valid inputs from being entered too. For example, a validation rule for a phone number that limited the input to 11 characters would work to keep incorrect values out, but some users may enter spaces between their numbers and could have them invalidated. Also acknowledging what the user experience is like, such as what happens if an input is wrong or not correct? Is the feedback for an invalid password informative? Are the pages responsive enough so a user knows what is taking place?

3. Alpha and Beta Testing

Def – stages of user acceptance testing, where different groups are established to take part in the final stages of bug catching and feedback gathering.

This type of testing would take place exclusively during Acceptance testing. Alpha testing would involve a select group of individuals who would have access to a non-live version of the solution with the intent on providing feedback, whilst Beta testing provides it to a wider group of actual users for the same purpose. Alpha testing helps to gather important information about the solution before it reaches the wider public, with beta testing acting as the last stages of all testing, making sure that the product works for a sample of the target demographic.

These testing strategies should be able to catch any bugs before the website goes online to all intended users, and gives time for final adjustments to be made not just to the functionality of the website, but also to the experience – do users like the colour scheme, find the website easy to navigate, think there should be improvements etc.

Testing Table

All tests will be recorded in a testing table similar to the following design:

Test description	Test Data (if required)	Expected Outcome	Actual Outcome	Action to take

This testing table will allow me to describe what test is being carried out, the data that will be input, what is anticipated to happen versus what actually happens, and the steps to take in order to address this.