**Task 2**

Steps:

1. Open asset log
2. Open project configuration
3. Create versioning folder
4. Create documentation
5. Start designing UI/UX

**Functionality:** Use two different languages. Use logic and programming structure

* Modularisation
  + Use more advanced techniques, avoid simple loops and conditionals.
* Programming Structures
* Complex data models
* Web service APIs

**Efficiency**

* Avoid linear structure
* OOP design features
* Recursive algorithms
* Loading data when necessary

**Code Organisation**

* Avoid multiple pages of nested if clauses and unnecessary repeated code
* Indentation
* Logic, classes or objects with structure
* Comments
* Local variables and minimal use of global variables
* Use of constants
* Consistent style throughout

**User Experience**

* Inputs must be handled appropriately
  + Invalidation and sanitisation
  + Ease of user input
  + Well designed interface
    - Useful?
    - Useable?
    - Desirable?
    - Findable?
    - Accessible?
    - Creditable?
* User guidance and error messages
  + Short and meaningful error messages
  + Provides information about solution/corrective action that can be taken
  + Help/instructions or meaningful labels to reduce errors
* Robust solution including
  + Good exception handling that can deal with incorrect outputs
  + Fall back code to deal with different systems w/o crashing
  + Appropriate use of validation

**Legal and Regulatory**

* Accessibility
  + Adjustable fonts, colours
* Compatibility
* Legal and ethical
  + Cookie consent notice, Privacy policy
  + GDPR
  + IP consideration
  + T&Cs

**Testing**

**Data testing types:** *Normal, Erroneous, Boundary, Absent*

* ***Regression Testing:*** *Whenever changes are made, test previously working features to ensure no issues arise.*
* ***Test-specific data****: Influence the system behaviour and reveal the case specifics under the test*
* ***Test-reference data****: have little influence on the test performance*
* ***Application reference data****: irrelevant to the behaviour under test, and are needed to start the application*
* ***Valid test data****: does the system functions follow the requirements, does the system process and store data as intended*
* ***Invalid test data****: check to see if the software correctly processes invalid values, shows relevant messages and notifies the user*
* ***Boundary test data****: help to reveal defects connected with processing boundary values*
* ***Wrong data****: entering the data of inappropriate format, whether it shows the correct error messages thus showing use of validation*
* ***Absent data****: should check that the solution handles entering a blank field*

**Use of testing to inform the iterative development process**

* Runtime errors that occurred when implementing code
* Fixing of syntax errors
* How logical errors were detected through the use of appropriate test data
* Descriptions of how issues were fixed
* Refinements made in the code to mitigate issues
* Details of regression testing
* Comparison of manual calculations against program calculations to ensure logic and outputs are correct
* Testing around boundaries to ensure logic of validation is correct

**Documentation**

* **Clearly provide information regarding development process including:**
  + A commentary on the changes they have made, and features included
  + A change log detailing significant changes at each stage of development
  + Information relating to how testing outcomes impacted on changes made
* **Should provide justification as to why changes were made including:**
  + Outcomes of testing in relation to the needs
  + More efficient solutions that came out a result of development and or further research
  + Order in which requirements were dealt with e.g why version 1 focused on specific functional requirement and other requirements were implemented later
* **Demonstrating appropriate use of versioning**
  + Sensible and clear naming conventions for each version
  + Suitable folder structures that make each version easy to find
  + Each version shows a notable change in the product (e.g additional functionality, error correction)

• **Requirements phase**, in which the requirements for the software are gathered and analysed, eventually result in a completed and final specification of requirements.

• **Design phase** that the software solution meets the requirements is designed. This may be a new design or an extension of an earlier design.

• **Implementation** and Test phase, when the software is coded, integrated and tested. Using the correct and appropriate test strategy

• **Review phase**, in which the software is evaluated, the current requirements are reviewed, and changes and additions to requirements proposed.

For each iteration, documentation should demonstrate a decision has been made as to whether the software produced in this phase will be discarded or kept as a starting point for the next iteration. Should be rigorous validation of requirements, and verification (including testing) of each version of the software against those requirements within each iteration.

**Overview of Project**

**Summary of Purpose and Intention to Users**

**Specific requirements, constraints etc**

# Contents

# General layout

* Comment on the layout and how close it was to the design
* Comment on noticeable changes
* Colour theme

# Homepage

* Place screenshot of design
* Place screenshot of prototype
* Include HTML snippets and description

# About Us Page

* Place screenshot of design
* Place screenshot of prototype
* Include HTML snippets and description

# Page

* Place screenshot of design
* Place screenshot of prototype
* Include HTML snippets and description

# Page

* Place screenshot of design
* Place screenshot of prototype
* Include HTML snippets and description

# Accessibility

* Place screenshot of design
* Place screenshot of prototype
* Include HTML snippets and description

+ show SQL databases involved

+ also include types of error messages and success messages

+ show python code

# Technical notes

* Frontend
  + HTML
  + CSS
  + Bootstrap
  + JS
* Backend
  + Python
  + SQL

# Task 2: Assets log

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Content** | **Source** | **Type of content** | **Purpose** | **Date gathered** |
| Boostrap template |  |  |  |  |
| ICON |  |  |  |  |
| APIs |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |