Module Learning Outcome Assessment Matrix

|  |  |
| --- | --- |
| **Learning Outcomes** | **Coursework** |
| Build a simple application from a given high-level requirement. | X |
| Describe the programming concepts and syntax used to build a predefined application. | X |
| Apply a basic systems development approach to the building of a predefined application. | X |
| Explain the key skills, tools and techniques, roles and responsibilities that support the systems development process. | X |

Coursework Brief

Total Marks: 100

Word Count: 1,500 words

Note, that the word count only includes words within the main body of the assignment from the start of the first task to the end of the last task. Words in diagrams are ignored unless deliberately wordy. The words in the Cover Sheet, Table of Contents, Bibliography and Appendices are not counted.

**All submissions must have a completed Cover Sheet (see Appendix A) attached to your submission.**

Summative Assessment Brief

**Instructions**

Please carefully read the full Assessment Brief including the Marking Rubric before starting your assignment and to check with your tutor, if necessary, to ensure that you have fully understood what is required.

**Requirements**

You are required to design, develop and test a simple console-based application in Python using a basic systems development approach with a supporting description of the programming concepts and syntax, and the systems development processes used.

The application should be based on something of relevance to your job role and/or organisation. For example, this could be to manage IT Assets, Help Desk Tickets or Customer Details. Note, you may need to simplify your real data to fit the brief. This is fine but If you are unsure seek guidance from your tutor.

The application should include a Data Store that can be implemented as either a simple Comma Separated Value (CSV) file or a single table in an SQLite database. The finished Data Store should contain 10 sample records (rows) and each record should be made up of at least 8 fields (columns) that include a range of appropriate data types. A suitable field should be included to act as the Primary Key.

The application should start with a menu that allows the user to browse all the records in the Data Store, add a new record, or amend, delete or display the full details of a selected record.

Validation should be included so that the user cannot perform an invalid action or enter non-conformant data for a field. Appropriate error messages should be displayed when validation rules are breached.

**Guidelines**

* Your solution should adhere to the basic design guidelines of modularisation by dividing the functionality up into appropriate elements.
* Ensure the quality and readability of your code by following basic programming guidelines such as naming conventions, indentation, comments and refactoring to avoid duplication of logic.
* Usability should be considered. For example, after completing an option, the user should be shown an appropriate message that indicates the success or failure of their action. Also, to confirm with the user before quitting the application or deleting a record.
* The user should be able to return to the Main Menu no matter where they are in the application.
* When the Data Store is read into the application, there should be checks to ensure no data is corrupt. For example, all Primary Keys are unique and all data conforms to the validation rules. If the data in a returned row is found to be corrupt, an error message should be written to a log file, the row should be skipped and the rest of the file/dataset treated normally.

Task 1: Design Document (15 Marks)

Create and document a high-level design for the application that incorporates:

* A Use Case Diagram describing the actors and basic functionality of the application.
* The Data Store specification including data types and validation rules.
* The application/program structure showing the navigation, simple screen layouts and a brief description of the functionality of each element.

Note, that it is suggested you create diagrams using a drawing package such as [draw.io](https://www.draw.io/). The package provides all the standard UML templates and is easy to use. Export your finished diagram and insert it as an image. Please ensure the diagram and any included text is readable.

**Evidence required:**

* **A high-level Design Document for the application.**

Task 2: Develop Application (50 Marks)

Use either the PyCharm or online Repl-It IDE to develop a Python application that satisfies the requirements and guidelines.

**Evidence required:**

* **A zip file of all coding artefacts, including automated unit tests if applicable. This should enable the application to be run and tested by your tutor.**
* **Also, include the source code for all coding artefacts as text (copy/paste) in the appendices.**

Task 3: Test Plan (15 Marks)

Create and document a Unit Test Plan for the application including test data, actions and expected results. Test the actual application, checking the actual results with the expected results keeping a log for each test that identifies any discrepancies and records amendments made to correct errors.

This can be completed using a manual Unit Test Plan and/or make use of unittest (Python’s automated testing tool).

**Evidence required:**

* **A completed Unit Test Plan for the application.**

Task 4 – Review and Reflection (10%)

Review and reflect on both the System Development process and the Python language used to design, develop and test the application:

* What were the strengths and weaknesses of your approach?
* What were the key lessons you learned?
* What would you do differently next time?

**Evidence required:**

* **A completed Review and Reflection.**

Task 5 – Presentation and Referencing (10%)

Your application code adheres to basic programming guidelines such as naming conventions, indentation, comments and refactoring to avoid duplication of logic.

Screen displays and messages should be well presented, using consistent formatting, with correct spelling, punctuation and grammar where appropriate.

The supporting documentation likewise should be well presented, using consistent formatting, diagrams and images where appropriate, with correct spelling, punctuation and grammar. At a minimum, it should include:

* A fully completed Cover Sheet including correct word count - See Appendix A
* A Table of Contents
* A Bibliography, listing where appropriate any references that were cited in your report. Relevant references should be cited and formatted using the [Roehampton Harvard style](https://libguides.roehampton.ac.uk/c.php?g=604242&p=4247622).

**Evidence to be uploaded:**

**Two** **separate submissions are required via Canvas:**

* **A zip file of all coding artefacts, including automated unit tests if applicable. This should enable the application to be executed and tested by your tutor.**
* **A single Word document (.docx file) that contains:**
  + **Task 1 Design Document**
  + **Task 3 Unit Test Plan**
  + **Task 4 Review and Reflection**
  + **Appendix A Source Code of all coding artefacts - Copy/Paste as text.**

Marking Rubric

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Criteria** | **Outstanding**  **80% -100%** | **Excellent**  **70% - 79%** | **Very Good**  **60% - 69%** | **Good**  **50% - 59%** | **Pass**  **40% - 49%** | **Poor**  **0% - 39%** |
| **Task 1**  **Design Document**  **[15 Marks]** | **Outstanding**  **[12 to 15 Marks]**  Outstanding Design Document for the application that incorporates all required artefacts. | **Excellent**  **[11 Marks]**  Excellent Design Document for the application that incorporates almost all required artefacts. | **Very Good**  **[9 to 10 Marks]**  Very Good Design Document for the application that incorporates many required artefacts. | **Good**  **[8 Marks]**  Good Design Document for the application that incorporates key required artefacts. | **Basic**  **[6 to 7 Marks]**  Basic Design Document for the application that incorporates some required artefacts. | **Poor**  **[0 to 5 Marks]**  Poor design Document for the application that incorporates a limited number of the required artefacts. |
| **Task 2**  **Develop Application**  **[50 Marks]** | **Outstanding**  **[16 to 20 Marks]**  Outstanding Application that executes and satisfies all requirements and guidelines. The code is outstandingly modularised and utilises both files and functions in an exemplary manner. Usability has been outstandingly considered with extreme attention to detail. Code is outstandingly refactored with no duplication. Validation of all data is exemplary and utilises a model approach to error logging. The storage and retrieval of data have been implemented outstandingly. | **Excellent**  **[14 to 15 Marks]**  Excellent Application that executes and satisfies almost all requirements and guidelines. The code is excellently modularised and utilises both files and functions in an almost flawless manner. Usability has been excellently considered with thorough attention to detail. Code is excellently refactored with little duplication. Validation of all data is almost flawless with appropriate error logging being carried out. The storage and retrieval of data have been implemented excellently. | **Very Good**  **[12 to 13 Marks]**  Very Good Application that executes and satisfies many requirements and guidelines. The code has been very well modularised and utilises both files and functions in a very good way. Usability has been very well thought through with very good attention to detail. Code has been well refactored with moderate duplication. Validation of most data has been very well carried out as has the logging of errors. The storage and retrieval of data have been very well implemented. | **Good**  **[10 to 11 Marks]**  Good Application that executes and satisfies key requirements and guidelines. The code has been well modularised and utilises functions in a well thought out way. Usability has been well thought through with good attention to detail. There is some evidence of refactoring but there are areas of logic that need attention. Validation of data and error logging has generally been well carried out but there are some gaps. The storage and retrieval of data have been well implemented with few problems. | **Basic**  **[8 to 9 Marks]**  Basic Application that executes and satisfies some requirements and guidelines. There has been some attempt at modularisation but more could be done. Usability is lacking with some limited messaging. There is little evidence of refactoring Some validation of data and error logging has been done but there are some significant gaps. An attempt has been made at the storage and retrieval of data but there are significant issues. | **Poor**  **[0 to 7 Marks]**  Poor Application that may not execute and satisfies only a few requirements and guidelines. There has been little or no attempt at modularisation. Usability is lacking with little or no messaging. There is no evidence of refactoring Little or no attempt has been made at the validation of data or error logging. Little or no attempt has been made at the storage and retrieval of data. |
| **Task 3**  **Test Plan**  **[15 Marks]** | **Outstanding**  **[12 to 15 Marks]**  Outstanding Test Plan that includes all required elements and updated with actual results. | **Excellent**  **[11 Marks]**  Excellent Test Plan that includes almost all required elements and updated with actual results. | **Very Good**  **[9 to 10 Marks]**  Very Good Test Plan that includes many required elements and updated with actual results. | **Good**  **[8 Marks]**  Good Test Plan that includes key required elements and updated with actual results. | **Basic**  **[6 to 7 Marks]**  Basic Test Plan that includes some required elements and updated with actual results. | **Poor**  **[0 to 5 Marks]**  Little or no Test Plan. At best, the plan includes a limited number of the required elements, which may or may not be updated with actual results. |
| **Task 4**  **Review and Reflection**  **[10 Marks]** | **Outstanding**  **[8 to 10 Marks]**  Outstanding Review and Reflection that covers all required elements. | **Excellent**  **[7 Marks]**  Excellent Review and Reflection that covers almost all required elements. | **Very Good**  **[6 Marks]**  Very Good Review and Reflection that covers many required elements. | **Good**  **[5 Marks]**  Good Review and Reflection that covers key required elements. | **Basic**  **[4 Marks]**  Basic Review and Reflection that covers some required elements. | **Poor**  **[0 to 3 Marks]**  Poor Review and Reflection that covers a few required elements. |
| **Task 5**  **Presentation and Referencing**  **[10 marks]** | **Outstanding**  **[8 to 10 Marks]**  Outstanding Presentation of documentation, code and program output that meets all required elements. Referencing where relevant is outstandingly cited and formatted. | **Excellent**  **[7 Marks]**  Excellent Presentation of documentation, code and program output that meets almost all required elements. Referencing where relevant is excellently cited and formatted | **Very Good**  **[6 Marks]**  Very Good Presentation of documentation, code and program output that meets most required elements. Referencing where relevant is mostly cited and correctly formatted | **Good**  **[5 Marks]**  Good Presentation of documentation, code and program output that meets key required elements. Referencing where relevant is cited but there are gaps and mostly correctly formatted | **Basic**  **[4 Marks]**  Basic Presentation of documentation, code and program output that meets some required elements. Referencing where relevant is partial and incorrectly formatted | **Poor**  **[0 to 3 Marks]**  Poor Presentation of documentation, code and program output meets a few required elements. No or poor attempt at referencing, |