# Design Document

(See SAGs Appendix G – 12.4.16)

## 2.1 User Interface Design

(See SAGs Appendix G – 10.2.5, 10.4.12, 11.4.12)  
ALL screens for the user, must be completely specified in this section.  
All required data on screens, must be in relevant GUI components.  
All action elements on screens (with input devices), must be listed and clearly described.   
Screen capture from a design tool (including IDE), sketches and mock-ups are acceptable.

|  |  |
| --- | --- |
| **[5-6]** | Sufficient level of user interface design present with consideration given to good design principles for an effective user interface. Correct components used for required data, on each screen.  Correct action elements, with input devices, on each screen, listed and described in detail. |
| **[3-4]** | Sufficient level of user interface design present with some consideration given to good design principles for an effective user interface. Some incorrect components used for required data, on screens.  Some incorrect action elements or incorrect input devices or action elements not described in detail, on screens. |
| **[0-2]** | No user interface design present or no consideration has been given to good design principles for an effective user interface. Incorrect components used for required data on screens.  No action elements and input devices or not described in detail, on screens. |

## 2.2 Program Flow Diagram

Use a flow diagram or any other form of illustration to present a global overview of how the program is used.

|  |  |
| --- | --- |
| **[4-5]** | Flow is clear, well represented and easy to understand. No logical gaps are evident. |
| **[2-3]** | Flow is substantial but still has some logical gaps. |
| **[0-1]** | No flow and/or large logical gaps. |

## 2.3 Class Design and OOP Principles

(see SAGs Appendix G – 11.4.3, 12.4.3, 11.4.5, 12.4.5)  
The candidates must provide their class design represented as a UML class diagram with class name, fields, and methods demonstrating the application of OOP principles.  
Only provide backend classes NOT user interface classes.

### Class Design

|  |  |
| --- | --- |
| **[3]** | Class design is thorough – all fields and methods are present. Fields and methods clearly relate back to the Specifications Document. |
| **[1-2]** | Class design is substantial but shows obvious gaps in missing fields/methods or has minor errors. |
| **[0]** | No class design or class design is incorrect or is rudimentary with little detail. Fields are incomplete, methods are minimal/not well thought out. |

### OOP Principles

|  |  |
| --- | --- |
| **[4-5]** | Fields and methods are separated logically, into classes. Fields and methods are protected sensibly. Good use of OOP principles where necessary. |
| **[2-3]** | Fields and methods are separated logically, into classes. Some instances of incorrect or inappropriate use of OOP principles. |
| **[0-1]** | No attempt to separate into classes. Some attempt at a class diagram with little to no organisation. |

2.4 Secondary Storage Design **(see SAGs Appendix G – 11.4.7, 12.4.7, 10.4.10, 12.4.10, 10.4.11, 11.4.11, 12.4.11)**Candidate must show how data structure in primary memory described in section 2.3, will be permanently stored.  
Storage design should be done using tables in a database, text files, JSON files or a combination thereof. Storage can be local, remote or cloud based.  
For a database, screenshots of tables with record structure and field types from database software are acceptable along with sample data for each table.  
For text files, an explanation of the structure of the file must be explained together with sample data.

|  |  |
| --- | --- |
| **[5]** | Storage design is well described – fields are listed with data types and description. Storage design is appropriate to purpose and matches the Specification Document. There are no missing aspects. |
| **[3-4]** | Storage design is well described but with a few missing aspects. |
| **[1-2]** | Storage design is evident, but description is superficial/vague/incomplete or with errors. |
| **[0-1]** | No storage design evident or storage design is rudimentary. |

## 2.5 Explanation of Secondary Storage Design

(See SAGs Appendix G – 12.4.7, 12.4.10)  
The candidate must provide an explanation of their secondary storage design.  
The explanation must demonstrate a justification of the secondary storage design and an understanding of the implications of the chosen design as opposed to other storage designs.

|  |  |
| --- | --- |
| **[3]** | Explanation shows in-depth understanding of the implications of the secondary storage design and is completely justified. |
| **[2]** | Explanation is substantial, but it is not completely justified. There are some areas of confusion or lack of understanding of the implications of the storage design. |
| **[0-1]** | No explanation of secondary storage design is provided or no evidence of understanding of the storage design |

## 2.6 Explanation of how Primary Data Structures relate to Secondary Storage

(See SAGs Appendix G – 10.4.3, 11.4.3, 12.4.3, 11.4.7, 12.4.7)  
Description of how the primary data structures described in class diagrams (assessed in section 2.3), will represent the secondary storage design (assessed in section 2.4).  
There should be a description for each backend class listed in section 2.3, that will translate to how data is sent to and from secondary storage.

|  |  |
| --- | --- |
| **[3]** | A clear and detailed representation of which class relates to which secondary storage data and how the data will be represented, when the data is read from or written to secondary storage. |
| **[1-2]** | Some form of representation of which class relates to which secondary storage data and how the data will be represented, when the data is read from or written to secondary storage, however there are missing details. |
| **[0]** | No representation of which class relates to which secondary storage data. |