**cs3307a – Object oriented analysis and design**

**Design Inspection Instrument**

**Instructions:**

* The purpose of this document is to assist in the inspection of object-oriented design.
* Under each question is a choice of answers; please choose one (either replace the box with a checkmark or highlight it)

🞏 yes 🞏 no 🞏 partly, could be improved

* Two types of comments are required under each question. One is your analysis. The other is your finding (in the form of a comment). The analysis would typically show how you arrived at the finding.
* Add new lines as necessary for your analysis or findings.

**Scope of the system to be considered for inspection:**

* With reference to Appendix B – Dashboard Screens, take Demo 1 feature, focusing on that part of the code that produces one Dashboard summary.
* Visualisation code is out of scope of this inspection.

+++++++++++++++++++

**Structural correspondence between Design and Code:**

Are all the classes and interrelationships programmed in the application explicitly represented in the class diagram of the system?

🞏 Yes 🞏No 🞏Partly (Can be improved)

Comment on your analysis: Analyzed by comparing UML design to summary.h

Comment on your findings: Additional relations to other classes not in UML exist, and some additional UI classes are not in UML

**Functionality:**

Do all the programmed classes perform their intended operations as per the requirements?

🞏 Yes 🞏No 🞏Partly (Can be improved)

Comment on your analysis: Compared functions implemented in .cpp files to design

Comment on your findings: Each class needed for a summary has an intended purpose and each fulfills it. There are no classes that lack functionality specified in system design

**Cohesion:**

Do the methods encapsulated in each programmed class, together perform a single, well defined, task of the class? (High-Cohesion: the functionalities embedded in a class, accessed through its methods, have much in common, e.g., access common data)

🞏 Yes 🞏No 🞏Partly (Can be increased)

Comment on your analysis: Analyzing code for summary.cpp and examining the inputs or function for each method for common data and purpose

Comment on your findings: The purpose of the class is very clear, and while it deals with a lot of data, it is specific about which data it works with. However, there are several functions that handle some processing for other features such as graphs.

**Coupling:**

Do the programmed classes have excessive inter-dependency? (High Coupling: In this case a class shares a common variable with another, or relies on, or controls the execution of, another class.)

🞏 Yes 🞏No 🞏Partly (Can be reduced)

Comment on your analysis: Analyzed what data or classes functions rely on in summary.cpp

Comment on your findings: Many classes are used and relied upon for functionality. While this is an unfair class to analyze for this (as it serves as a hub for analysis), it depends a lot on data, references and other classes to function

**Separation of concerns:**

Is the scoped problem decomposed into separate concerns where each concern is encapsulated in a construct such as a class with well-defined interface and cohesive functions with minimal of connections with other concerns?

🞏 Yes 🞏No 🞏Partly (Can be improved)

Comment on your analysis: Analyzing main purpose of functions in summary.cpp

Comment on your findings: While Summary serves a purpose of summarizing the input data and serves mostly just visualization purposes, it does also handle some functionality for graphing.

Do the classes contain proper access specifications (e.g.: public and private methods)?

🞏 Yes 🞏No 🞏Partly (Can be improved)

Comment on your analysis: Analyzed summary.h declarations

Comment on your findings: The only public methods are to do with loading and savings settings, as well as simply calling up the class. All other functionality is private.

**Reusability:**

Are the programmed classes reusable in other applications or situations?

🞏 Yes, most of the classes 🞏No, none of the classes 🞏Partly, some of the classes 🞏Don’t know

Comment on your analysis: Examined dependencies of summary.h

Comment on your findings: If creating a similar program to analyze csv files, then the summary would useful to reuse. However, it is very dependent on QT, so attempting to develop outside of QT would not be possible.

**Simplicity:**

Are the functionalities carried out by the classes easily identifiable and understandable?

🞏 Yes 🞏No 🞏Partly (Can be improved)

Comment on your analysis: Looked at commenting and followed logic of methods

Comment on your findings: At first glance it seems like a mess of variables, methods, and conditional statements but if you read the code names are easily understood and the logic is easy to follow.

Do the complicated portions of the code have /\*comments\*/ for ease of understanding?

🞏 Yes 🞏No 🞏Partly (Can be improved)

Comment on your analysis: Read through summary.cpp for comments

Comment on your findings: There are comments for methods and sections of code

**Maintainability:**

Does the application provide scope for easy enhancement or updates? (e.g., enhancement in the code is not anticipated to require too many changes in the original code)

🞏 Yes 🞏No 🞏Partly (Can be improved) 🞏Don’t know

Comment on your analysis: Analyzed functions to do with processing and storing data

Comment on your findings: Additional types of data can easily be added, provided they are processed and stored in similar processed record objects.

**Efficiency:**

Does the design introduce inefficiency in code (e.g., causes too many nested loops or delays in concurrent processing)?

🞏 Yes 🞏No 🞏Partly (Can be improved) 🞏Don’t know

Comment on your analysis: Looked for complex code or nested loops

Comment on your findings: As we separated backend from frontend, there are very few loops or processing methods in the class. Loops are mostly from looping through a vector of data. However, due to time constraints, we did have to add some processing into summary, which has hurt the efficiency. There are also some redundant queries or sorting of data

**Depth of inheritance:**

Do the inheritance relationships between the ancestor/descendent classes go too deep in the hierarchy? (The deeper a class in the hierarchy, the greater the number of methods it will probably inherit from its ancestors, making it harder to predict its behaviour).

🞏 Yes 🞏No 🞏Partly (Can be improved)

Comment on your analysis: Looked through classes for inheritance from or to summary class

Comment on your findings: There are no inheritance relations with summary but for implementing some QT classes for the GUI

**Children:**

Does a parent class have too many children classes? (This could possible suggest an abstraction problem.)

🞏 Yes 🞏No 🞏Partly (Can be improved)

Comment on your analysis: Looked through classes for inheritance from or to summary class

Comment on your findings: There are no inheritance relations with summary but for implementing some QT classes for the GUI

**Behavioural analysis:**

From the system’s requirements, **create several scenarios** starting from the **user’s** point of view: consider identifying one or more **typical** scenarios (e.g., those expected to be used with high frequency) and one or more **low-frequency** scenarios .

Each scenario is described as follows:

1. Title of scenario
2. Anticipated frequency of use (high, normal, low)
3. End-user trigger (starting point) for the scenario.
4. Expected type of outputs.
5. List of bullet points linking end-user inputs and identifying all the key features of the system expected to be “touched” by the scenario and producing the anticipated outputs.

Follow the code (structured walkthrough) to ascertain whether this scenario is properly implemented both in terms of logic and design.

Comment on your findings, with specific references to the design/code elements/file names/etc.:

**Scenario 1: Opening a CSV**

Frequency of use: High

Starting point: Dashboard screen (opens on launch)

Expected output: Processed CSV that can be summarized and graphed

Workflow:

* Upon launch the program, user clicks ‘Open CSV’
* User selects their desired CSV file
* The processed CSV appears in the dashboard box, and available for summarizing and graphing.

**Scenario 2: Edit a CSV**

Frequency of use: Medium

Starting point: Dashboard screen, with the CSV in question already opened

Expected output: Error-free CSV, ready for summarizing and graphing

Workflow:

* User highlights the CSV that they want to edit in the dashboard box
* User clicks ‘Edit’
* A new dialog opens showing the CSV file contents as tabular data
* The user makes the appropriate changes within the dialog box
* The user saves the changes

**Scenario 3: Viewing a Summary and Graph**

Frequency of use: High

Starting point: Dashboard screen

Expected output: Summary and Graph of data

Workflow:

* From the Dashboard, the user clicks ‘Summary’. Note that there must already be at least one CSV opened
* The Summary screen will be opened, displaying summary data on the left, and the graph on the right
* Opened CSV files can be switched by choosing the appropriate tab (ex ‘Grants and Clinical’)
* The graph shows the data of the currently opened summary tab

**Scenario 4: Filtering Data**

Frequency of use: Medium

Starting point: Summary screen

Expected output: Summary and graph displaying filtered data

Workflow:

* As an example, imagine the user wants to filter data between the years 1990-2000
* At the top of the Summary screen, the user locates the ‘Date Range Selected’
* The user enters 1990 in the first input and 2000 in the second input
* The user clicks ‘Filter’, and the summary table and graph are automatically changed
* The user may click ‘Clear’ to clear the input boxes and remove the filter

**Scenario 5: Exporting the Summary and Graph**

Frequency of use: normal

Starting point: Summary screen

Expected output: A PDF of the summary table or graph

Workflow:

* To export the summary table, the user clicks ‘Export Summary’ at the bottom right of the screen
* To export the graph, the user clicks ‘Export Graph’ above the graph, top the right
* In either case, a save dialog will open, allowing the user to select the name and location of the exported PDF
* The user clicks ‘Save’, and the PDF is generated

**Scenario 6: Changing the type of Graph**

Frequency of use: high

Starting point: Summary screen

Expected output: A specific type of graph, based on the summary table data

Workflow:

* Once the appropriate data has been chose in the summary table and filtered appropriately, the user may want to change how the data is graphed
* Above the graph, there are 3 select boxes
* In the first select box, the user may switch between a Bar Graph and a Scatter Plot
* In the second, the use may toggle between available graphs for the current data set, for example, for Grants and Clinical data:
  + Amount of Graph Funding by Year
  + Number of Grants per Year
  + Roles taken in grants for single year
* In the third select box, the user may choose to include all researchers in the data, or choose a single one by name