**Software Design Document**

**AJJ BNB**

**Student Names:**

**Ashley Pergoliti (s5311775)**

**Jonas Sajonas (s5284977)**

**Jodie Thomson (s118338)**

Table of Contents

[1.0 System Vision 3](#_Toc142068931)

[1.1 Problem Background 3](#_Toc142068932)

[1.2 System Overview 3](#_Toc142068933)

[1.3 Potential Benefits 3](#_Toc142068934)

[2.0 Requirements 4](#_Toc142068935)

[2.1 User Requirements 4](#_Toc142068936)

[2.2 Software Requirements 4](#_Toc142068937)

[2.3 Use Cases & Use Case Diagrams 5](#_Toc142068938)

[3.0 Software Design and System Components 6](#_Toc142068939)

[3.1 Software Design 6](#_Toc142068940)

[3.2 System Components 6](#_Toc142068941)

[3.2.1 Functions 6](#_Toc142068942)

[3.2.2 Data Structures / Data Sources 6](#_Toc142068943)

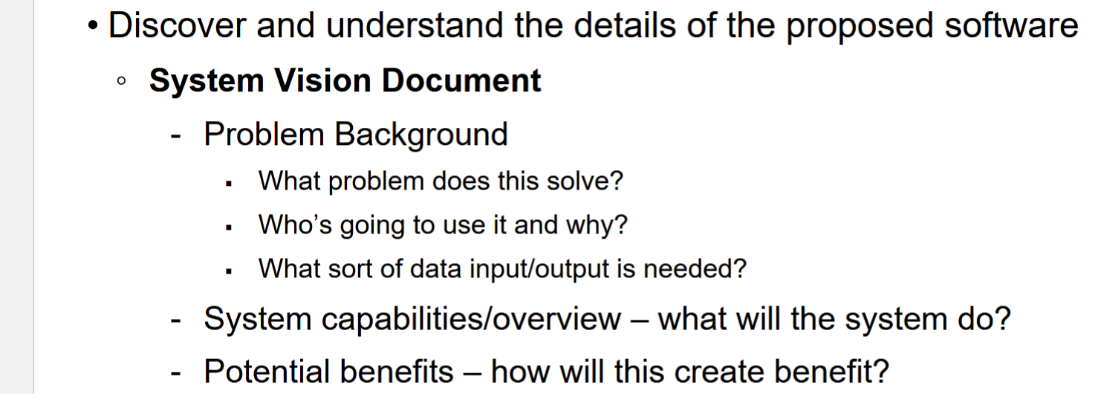
[3.2.3 Detailed Design 6](#_Toc142068944)

[4.0 User Interface Design 7](#_Toc142068945)

[4.1 Structural Design 7](#_Toc142068946)

[4.2 Visual Design 7](#_Toc142068947)

# System Vision



## Problem Background

Excel sheets are widely used to store large sets of text data. Users face problems when trying to view these large sets of data. This can be due to there being many unnecessary sections of data. This makes it hard for a user to sort through the data for what they are looking for.

This applies to users trying to view data relating to Airbnb. Users want to pick a specific date range and see information about different properties. This information includes the distribution of prices, listings by a specific suburb, records including user chosen keywords, customers that commented on the cleanliness of the property, and ratings based on a suburb.

Users can also find issues in trying to analyse data by just text on a screen. This is where graphs and charts become useful. A chart can be used to plot a set of data, which helps users analyse trends. This will be especially useful in the analysis of the prices of properties. This paired with the ability to pick the start and end period, means business owners can utilise this to make smart business decisions.

It is recommended that a system is created to help these users view the data quickly and effectively. This system will input two csv files provided by Kaggle.com, which will be converted to a dataframe for manipulation. The data will then be used to output on the screen for the user to analyse. This will be displayed as a record of data that the user can read through, and in chart form.

## System Overview

This system should be able to complete the following:

* Create a user interface that has interactable buttons and input fields that change what is displayed on the interface.
* Read different CSV files.
* Display different data based on different inputs added by the user, such as:
  + Start date and end date, picked in two different calendars
  + Property string
  + Suburb string
  + Array of keywords

## Potential Benefits

This system will provide the following benefits for the users:

* View large sets of data through a smooth user-friendly interface.
* View this data in text form and graph form.
* Filter through the data for their specific chosen criteria.
* Speed up the process of analysing the data.
* Financial benefits for making decisions based on the given data.

# Requirements

2. Requirements (10 marks) There are 2 types of requirements to consider

## User Requirements

• User Requirements: How a user will interact with the program. What do they need to do ?

In this section you detail how a user is supposed to interact with or use your program. What do they ***need*** to be able to do? This should all be from the end users perspective. Can be a combination of narrative text and listing of needs.

**Assignment note: You have not been given a client/user, so you can make one up. Who do you think would be using your software?**

* The user shall be able to start / open the GUI script
* The user shall be able to interact with the functionalities of the system
* The user shall be able to type for keywords and click options
* The user shall be able to see the data from Airbnb in a user-friendly format
* The user shall be able to easily recover from mistakes using the system
* The user shall be able to navigate easily in the system
* The user shall be able to easily learn how to use the system
* The user shall be able to read and understand the text used
* The user shall be able to recognise the charts and lists used by the system

## Software Requirements

Software Requirements: What functionality will the software provide (think functional requirements)

In this section you detail what the requirements for the software are. What functionality will it provide? This is usually a formal listing, with requirements often using the word ‘Shall’. IE:

R1.1 The program shall accept multiple file names as arguments from the command line.

R1.2 Each file name can be a simple file name or include the full path of the file with one or more levels.

etc …

Can be primarily functional requirements, though you may include other types if you think of them.

* The program shall be run as a single file in the command line.?
* The script shall open to a GUI interface.?
* The GUI interface shall have several buttons and other input fields that the user can interact with.
* These buttons shall trigger further actions and interfaces- a graph.

Functional Requirements: <<Please Check>>

* The system shall be able to read Graphical User Interface script.
* The system shall be able to display the Graphical User Interface.
* The system shall be able to accept User Inputs.
* The system shall be able to display result of the user interaction.
* The system shall be able to read multiple excel files.
* The system shall be able to Filter Keywords from the search query
* The system shall be able to return none when search value has no matches
* The system shall be able to return an error when the search parameters are incomplete
* The system shall be able to Filter cleanliness keyword from comments
* The system shall be able to Get ratings of listings
* The system shall be able to Get Lists of Suburbs
* The system shall be able to Get Dates of Listings
* The system shall be able to Verify Search queries
* The system shall be able to Produce charts
* The system shall be able to Display Records of listings

(Check

Required Features:

For a user-selected period, report the information of all listings in a specified suburb

For a user-selected period, produce a chart to show the distribution of prices of properties

For a user-selected period, retrieve all records that contain a keyword (user entered), e.g. pool, pet.

Analysing how many customers commented on factors related to cleanliness (multiple key words may be associated with cleanliness – justify your selection).

## Use Cases & Use Case Diagrams – need 2-3 • Use Cases: These Use Cases should show the blending of user and software requirements by identifying use cases and how the user will interact with the product. Any diagramming format is acceptable, but the diagrams should clearly display the sequence of events and interactions between the user and the software. Expecting about ~5 use cases (1 for each of the functions) and a few accompanying Use Case Diagrams.

2.3.1 Jonas use case diagram to return

**2.3.2 Use Cases**

|  |  |
| --- | --- |
| **Use case ID** | 1.0 |
| **Use Case Name** | Display Listing for Suburb |
| **Actors** | User |
| **Description** | For a user selected period and suburb, display all listings. |
| **Flow of Events** | 1. User navigates to Suburb Listing Page 2. User selects the time frame from date pickers 3. User types the suburb 4. User clicks Display 5. System Displays listings for the selected suburb and dates. |
| **Alternative flow** | 1. System returns no listings if no matches 2. User option to enter new dates and/or suburb 3. System displays an error message if there was an error. |

|  |  |
| --- | --- |
| **Use case ID** | 2.0 |
| **Use Case Name** | Display Pricing Chart |
| **Actors** | User |
| **Description** | Display Chart for pricing distribution for specified period |
| **Flow of Events** | 1. User navigates to Price Chart Page 2. User selects time frame 3. User enters the suburb 4. System displays the Price Chart for the selected suburb and dates. |
| **Alternative Flow** | 1. System returns no chart when it doesn’t have any matches 2. User option to enter new dates and/or keywords 3. System displays an error message if there was an error. |

|  |  |
| --- | --- |
| **Use case ID** | 3.0 |
| **Use Case Name** | Display Search Records |
| **Actors** | User |
| **Description** | Display all records that contain user selected keywords in a selected period |
| **Flow of Events** | 1. User Selects Keywords option 2. User Enters Keywords 3. User selects time period from date picker 4. System Displays all records that contain the specified keywords in that period. |
| **Alternative Flow** | 1. System returns no listings 2. User option to enter new dates and/or keywords 3. System displays an error message if there was an error. |

|  |  |
| --- | --- |
| **Use case ID** | 4.0 |
| **Use Case Name** | Display chart of suburbs according to cleanliness |
| **Actors** | User |
| **Description** | Display chart of suburbs with keyworded comments about cleanliness. |
| **Flow of Events** | 1. User selects suburb 2. User select chart option 3. User selects cleanliness option 4. System returns a chart of suburbs with comments related to cleanliness |
| **Alternative Flow** | 1. System returns a message stating no search results. 2. System returns an error. 3. System displays an instruction to retry. |

|  |  |
| --- | --- |
| **Use case ID** | 5.0 |
| **Use Case Name** | Display Listings in a suburb by Ratings |
| **Actors** | User |
| **Description** | Display a list of Airbnb listings by suburb sorted according to ratings in descending order. |
| **Flow of Events** | 1. User enters suburb location 2. User select Ratings option 3. System displays Records of listings with highest or lowest ratings on top to bottom. |
| **Alternative Flow** | 1. System returns a message stating no search results. 2. System returns an error. 3. System displays an instruction to retry. |

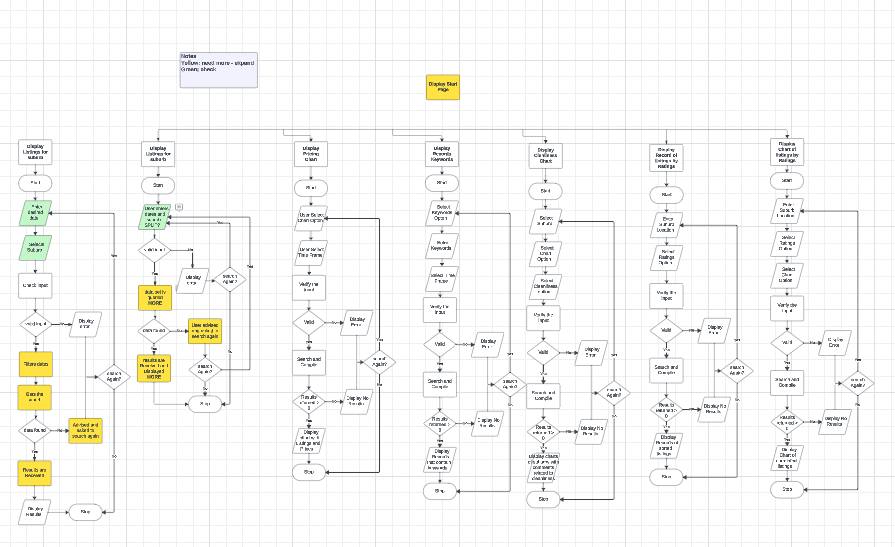
|  |  |
| --- | --- |
| **Use case ID** | 6.0 |
| **Use Case Name** | Display Chart of Listings in a suburb sorted by Ratings |
| **Actors** | User |
| **Description** | Display chart of listings in a suburb by ratings |
| **Flow of Events** | 1. User enters suburb location 2. User select Ratings option 3. User select Chart option 4. System displays a Chart of listings correlated to ratings |
| **Alternative Flow** | 1. System returns a message stating no search results 2. System returns an error 3. System displays an instruction to retry |

# Software Design and System Components

## Software Design

A block diagram/flowchart of how your software might work ? IN PROGREE LUCID

<https://lucid.app/lucidchart/ceaee37b-dc96-4511-a486-ecab53bc4d8e/edit?viewport_loc=-552%2C-250%2C3840%2C1803%2C0_0&invitationId=inv_8b81398c-d081-4c1b-a3e9-c768bbc6af0a>



## System Components

<<This is a placeholder text to remind that we need something here>>.

### Functions

Preliminary list of all functions in the software. For each function in the list the following information is provided:

* a brief description of what it does (1 or 2 sentences);
* a list of the input parameters, and their data types, and what they are used for;
* **a list of any side effects caused by the function (ie change global or member variables, changes data passed by reference from calling function etc)**
* a description of the function’s return value

**Do we want separate functions for queries in each function?**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Function Name** | **Description** | **Parameters** | **Side Effects** | **Return Value** | **Notes** |
| **def readExcel(fileName)** | A function that will be used to read the CSV files to grab the data. | The fileName variable will be a string, that will specify the name of the file. | There will be a global variable that will change to hold the dataframe of the Excel data. | Returns the dataframe of the data (of type DataFrame). This is a data representation of the CSV file. |  |
| **def selectDate(startDate,endDate)** | A function that will grab the date from the user-selected period. | The startDate parameter will hold the start date picked by the user in the start calendar.  The endDate parameter will hold the end date picked by the user in the end calendar. | This will set a global variable that holds the date range which will be used for other functions. | Return the date range. |  |
| **def startSession()** | A function that will initialise the application. It will create the graphical interface of the application, and it will call readExcel() to grab the required data from the Excel files. | No input parameters required. | Global variables will be changed from the readExcel() function. | No return value required. |  |
| **def displayPriceChart()** | This function will display a price chart for a user-selected period, to show the distribution of the prices of properties. It will be the command of a button. | No input parameters required. | This will create a matplotlib graph showing the data. | No return value required. | This function will use the data returned by getPriceChartData() |
| **def getPriceChartData(from, to, property, dataframe)** | Using the date parameters, a query is created, and a list of property prices is returned. | From and to, will be the global variables that are holding the start and end dates the user selected respectively. Property will be a string variable holding an input from the user. Dataframe will be the dataframe used in the data. | This will populate a variable with data, which will be used for displayPriceChart(). | The return value is a list of property prices. |  |
| **def displaySuburbListings()** | This function will display a chart with all listings for a specified suburb for a user-selected period. | No input parameters required. | This function will make the suburb filtered records appear on screen. | No return value required. | This function will use the data returned by getSuburbListings(). |
| **def getSuburbListings(from, to, suburb, dataframe)** | Using the 4 input parameters, a query is created, and the data received is all listings for the properties in that range in the suburb. | From and to, will be the global variables holding the start and end dates the user selected respectively. Suburb will be a string variable holding an input from the user. Dataframe will be the dataframe used in the data. | This function will populate a variable with data, that will be used for displaying the suburb records. | This function returns the listings based on the input variables. |  |
| **def displayKeywordResults()** | This function will display the records with the user input keywords. | No input parameters required. | The keyword records will be displayed on screen. | No return value required. |  |
| **def getKeywordResults(keyWords,from,to,dataframe)** | Using the 3 input parameters, a query is created, and the data received is all the listings/records that are based on the parameters. | The keyWords variable will be an array of words input by the user. From and to, will be the global variables holding the start and end dates the user selected respectively. Dataframe will be the dataframe used in the data. | This function will return data that will be held in a variable, that will be used by displayKeywordResults() | This function will return the data with the list of records that contain the keywords in that period. |  |
| **def displaySuburbRatingsChart()** | This function will display a chart by using the getSuburbRatings() function. | No input parameters required. | This function will make a chart display on the screen. | No return value required. |  |
| **def displaySuburbRatingsRecords()** | This function will call getSuburbRatings(), and display the rows that are returned by the filter. | This function requires no input variables. | This function will display the suburb rating records on screen. | No return value required. |  |
| **def getSuburbRatings(suburb, rating, dataframe)** | This function will return the data based on the user input variables. | The suburb variable will hold the suburb the user wishes to search for. The rating will hold the rating the user has selected. The dataframe will hold the data to filter. | This function will return data which will be assigned to a variable. | This function will return the filtered dataframe data with the suburb records. |  |
| **def displayCleanliness()** | This function will display the cleanliness data by calling getCleanlinessData(). | No input parameters required. | This function will display the cleanliness data on screen. Which means a variable will be altered to display the data. | No return value required. |  |
| **def getCleanlinessData(keywords, suburb, dataframe)** | This function will get the cleanliness data by using the input variables. | The input parameter keywords, will be an array of keywords. The suburb will be user selected. The dataframe will be the data to filter the cleanliness by. | This function will return data which will be assigned to a variable. | This function will return the filtered dataframe data. |  |
| **def cleanUserInput(input)** | This function will take in the input of a user and clean the input. Remove special characters, split the input on the spaces. | The ‘input’ variable will be the input entered by the user. | This function will alter the user input. | This function will return the array of the cleaned user input. |  |
| **def cleanExcelData(dataframe)** | This function will take in the dataframe, and will remove any records that are missing data in certain columns. | The input variable will be the dataframe of each data set. | This function will alter the dataframe variables. | This function will return the cleaned dataframe. |  |
| **def displayErrorMessage(errorMessage)** | This function will display an error message on screen for the user, on an error being triggered. | The input variable is errorMessage, which will be a string holding the error message that will be displayed on screen. | This function will cause error messages to display. | This function will return the error message. |  |
| **def removeExtraColumns(dataframe, wantedColumns)** | This function will remove columns from the input dataframe, that are not going to be used for displaying data. | Dataframe is the input dataframe that will have extra columns removed from it. WantedColumns will be an array of the desired columns. | This function will alter the dataframe variables. | This function will return the dataframe with the removed columns. |  |
| **def clearSearchQuery()** | This function will clear the search box, it will be tied to a button. | The query is the input variable. | When the search button is clicked, it will not return anything. The search field will be cleared. | Returns the search query as null. |  |

### Data Structures / Data Sources

List of all data structures in the software (eg linked lists, trees, arrays etc) or eternal data sources. For each data structure in the list the following information is provided:

* Type of structure (tree, list etc),
* Description of where and how it is used
* List of data members, and what each one is for do
* List of functions that use it

https://realpython.com/python-data-structures/

<https://www.geeksforgeeks.org/data-structures/>

These are the Data Structures that we used for the software:

* Arrays Data Structure: We used lists to compare variables for sorting and searching algorithms.
* Data Frame Structure??

### Detailed Design

Function to read in the data from the csv files:

def readExcel(fileName):  
 dataframe = pandas.readFile(fileName)  
 return dataframe

dataframeListings = readExcel(listings.csv)  
dataframeReviews = readExcel(reviews.csv)

Function for removing rows with empty columns:

def cleanExcelData(dataframe):  
 cleanedDataframe  
 for row in dataframe:  
 if column is not empty:  
 cleanedDataframe.push(row)

dataframeListings = cleanExcelData(dataframeListings  
dataframeReviews = cleanExcelData(dataframeReviews)

Function for keeping only the desired column names:

def removeExtraColumns(dataframe, wantedColumns)  
 removedUselessColumns  
 for row in dataframe:  
 if column is in wantedColumns:  
 removedUselessColumns.push(row)  
 return removedUselessColumns  
  
desiredColumnNamesListings = [names]  
  
dataframeListings = removeExtraColumns(desiredColumnNamesListings, desiredColumnNames)  
  
desiredColumnNamesReviews = [names]  
  
dataframeReviews = removeExtraColumns(desiredColumnNamesReviews, desiredColumnNamesReviews)

Function to get the price data from a dataframe:

def getPriceChartData(from, to, property, dataframe):  
 dataFrameNewData  
 for row in dataframe:  
 if column > from and column < to and column equals property:  
 dataFrameNewData.push(row)  
 return dataFrameNewData

dataForPriceChart = getPriceChartData(fromDate, toDate, property, dataframeListings)

Function to get the suburb:

def getSuburbListings(from, to, suburb, dataframe):  
 dataFrameNewData  
 for row in dataframe:  
 if column > from and column < to and column equals property:  
 dataFrameNewData.push(row)

return dataFrameNewData

dataForSuburbListings = getSuburbListings(fromDate, toDate, suburb, dataframeListings)

Function to get data based on keywords, from and to:

getKeywordResults(keyWords,from,to,dataframe):  
 keywordData  
  
 for row in dataframe:  
 if column > from and column < to and keyWords in column:  
 keywordData.push(row)  
 return keywordData  
  
userInputKeywords = [keywords]

dataKeywordResults = getKeywordResults(userInputKeywords, fromDate, toDate, dataframeListings)

Function to get records related to cleanliness:

getCleanlinessData(keywords,suburb,dataframe):  
 cleanlinessData  
  
 for row in dataframe:  
 if keywords in column and suburb in column:  
 cleanlinessData.push(row)  
 return cleanlinessData  
  
cleanlinessKeywords = [keywords]  
  
dataCleanliness = getCleanlinessData(cleanlinessKeywords,suburb,dataframeReviews)

Function to get suburb ratings:

getSuburbRatings(suburb,rating,dataframe):  
 suburbRatingData

for row in dataframe:  
 if suburb in column and column > rating:  
 suburbRatingData.push(row)  
  
 suburbRatingData.sort(descending)  
 return suburbRatingData  
  
dataSuburbRatings = getSuburbRatings(suburb,rating, dataframeListings)

Function to clean the user input:

def cleanUserInput(input):  
 newUserInput  
  
 newUserInput = regex.remove(‘unwanted\_symbols’)  
 return newUserInput  
  
  
cleanedUserInput = cleanUserInput(input)

# User Interface Design

This is your initial interface design. Describe the tools you used for this design stage and any key findings that informed your design. **This introduction is descriptive and should explain what you have completed for the actual design work you will present in the sub-sections below.**

The system hierarchy was designed to emulate website navigation, aligning with principles of user-centred interaction for enhanced usability and a smooth transition between pages. This decision was made to ensure an intuitive and familiar experience for users. The Lucid app from Lucidchart.com was employed as the tool of choice to implement this structure effectively.

The Modern design ethos inspires the visual design, focusing on a contemporary and visually appealing look and feel. Using enlarged texts and buttons aims to improve accessibility and readability for all users. The choice of fonts, [WHAT FONT??], further contributes to clear and legible content. The Moqups App was used for the design process from Moqups.com.

## Structural Design

A diagram of a system

Description automatically generated

The System Hierarchy structure begins with the landing page, where users are greeted with a welcome message and instructions. This interface establishes the tone for user interaction. Navigation buttons are positioned on the left side across all system pages, facilitating user familiarity and usability. These buttons establish connections to other pages. Each page is linked to others, creating a cohesive user experience. The system's Rating Page presents a choice between two display formats: a list or a chart. This option enhances data presentation and caters to different user preferences and needs.

Structural design refers to the navigational and information structure of your product – the structure that supports the interface layout. How will you structure your product? How will you group your information? How will you navigate through your product? Why? **This can take the form of a diagram showing structure and hierarchy,** supported by a **discussion and justification** of your choices. Why have you made these design choices? Describe and outline the structure of your interface and of your information.

## Visual Design

Detail your visual design: Layout, visual elements, icons, graphics, style, colour, fonts general screen designs. This can be sketches, wireframes, mockups etc, supported by a discussion, explanation, and justification of your choices.

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**Question do we want all options to appear ready to use on landing area?**

NEED:

* Choose colour palette
* Choose Fonts – used standard – may need to re think or just justify
* Screen design – wireframes + story board?
* Discussion



Marking Guide  
Part A – Project Management (worth 25% of total course mark)  
There are 3 key components to this:  
1. Project Plan and Gantt Chart (40 marks)  
2. Software Design Document (50 marks)  
3. Software Version Control (10 marks)  
Marked out of 100, then times by 0.25 to get total mark. Unless otherwise specified, all  
group members should receive same mark.  
Project Plan and Gantt Chart (40 marks)  
In the Project Plan.docx  
1. Introduction (5 marks)  
Should contain an overview of the project (from a project management/component perspective) and mention the scope and outline of the project management document  
2. Work Breakdown Structure (WBS) (10 marks)  
Should be a breakdown of all the different activities involved in completing the project. For Part A  
this should contain all of the work involved including preparing the project plan and software design document, as well as all related preparatory/organisation work. For Part B this should include all of the required implementation, testing and reporting activities. This can be somewhat high level for Part B, but should still contain some reasonable assumptions. This should be presented as some form of diagram/hierarchy.  
3. Activity Definition & Estimation (10 marks)  
For each item in the WBS, the item should be explained in detail and include a time estimate that is reasonable.  
4. Gantt chart (15 marks)  
All of the items in the Activity definition should be listed in the Gantt chart with the relevant  
estimates and scheduling. The students should have also tracked the actual start time and time taken. Also need to submit the Gantt chart.xlsx

Software Design Document (50 marks)  
In the Software Design Document.docx  
1. System Vision (10 marks)  
Should include a background on the dataset, software overview and potential benefits of the software.  
2. Requirements (10 marks)  
There are 2 types of requirements to consider:  
• User Requirements: How a user will interact with the program. What do they need to do?  
• Software Requirements: What functionality will the software provide (think functional  
requirements)  
• Use Cases: These Use Cases should show the blending of user and software requirements by identifying use cases and how the user will interact with the product. Any diagramming format is acceptable, but the diagrams should clearly display the sequence of events and interactions between the user and the software. Expecting about ~5 use cases (1 for each of the functions) and a few accompanying Use Case Diagrams.  
3. Software Components and Software Design (15 marks)  
• Software Design: Flow chart / block diagram (5 marks)  
• Software Components: Functions, Classes/Data Structures, Algorithms (10 marks)  
There should be a listing of the main functions (I would expect at least a loadData function and some kind of display function), the main classes/data structures used and potentially a description of any algorithms that might be used for data analysis. Each component should have the relevant information (name, type, details).  
4. User Interface Design (15 marks)  
Structural design should almost be like a flowchart/hierarchy chart showing the structure of  
the interface. What screens/components and how do they interact.  
Wireframes/mock ups of the interface. Should have clearly labelled interface components. No hand drawing (must be digital design). Different screens/menus/options should have their own wireframes. No colour/graphics required – just position/size of components and component layout.  
Additional design information can be included here too.  
Software Version Control (10 marks)  
Git\_log.txt : A Git repository is correctly used for all contributions to the project. A Git log is  
attached and shows regular commits and pushes from all group members.