****

**School of Information and Communication Technology**

**Griffith University**

**3821ICT – WIL Single Project**

**Improving the efficiency and effectiveness of large-scale extraction of electricity demand data published by AEMO as part of the 2022 Integrated System Plan (ISP).**

**Sprint 1**

**Unisoft**

*[29/08/2022 (Trimester:- 2)]*

**Industry Partner:** The Centre for Applied Energy Economics and Policy Research (CAEEPR)

**Client:** Nancy Spencer

**Team members:**

Nathan Cowan -s5143344

Akshay Devnani -s5268458

Joshua Martin -s5220620

Naman Sharma -s5155752

**1. Functional Requirements**

1. All files from the AEMO can be loaded into the database.
2. Data from each row in the csv files will be converted to columns representing date for a single as separate column and each candidate year will be represented as 11 separate columns in the output file.
3. The files can be searched through using a GUI with a range of searching options.
4. The selected files are sorted by categories and can be printed to a csv, excel and space formatted text file.
5. Simple or frequent commands and requests should be automated down to the minimum number of user-actions.
6. Filtering will be based on selecting regional/subregional, component, year either a group such as all of a component, or down to a selection of exact years and timeframes.
7. File Types other than .csv, and .xlsx such as .txt or other types of delimited files should still be given to the parser before being rejected.
8. Internal functions and processes should be accessible from a terminal.
9. GUI will include dropdown, text boxes and radio buttons where appropriate for the filters.
10. The search results will be organised in table and can also be sorted using excel functions in csv/excel files.

**2. Non-Functional Requirements**

1. Third party libraries should have appropriate licences.
2. GUI should be user friendly while also providing complex searches.
3. Software should be reliable to allow for a smooth user experience.
4. The software should use programming techniques that ensure that the system is secure.
5. The system should allow for scalability by ensuring that any amount of data provided can be processed.
6. Code should be thoroughly commented to ensure that the system is maintainable.
7. The system should be robust to ensure that any future data from the AEMO can be processed
8. All inputs should get a basic sanity check, such as dates being a real date, and in the data range. Integer inputs should not contain letters, etc., considering not only what a component **should** do, but what it **shouldn’t** do, and how an implementation might be broken or intentionally manipulated.
9. The program should meet the performance requirement as the GUI should be responsive to multiple queries from the end user.

## **3. Product Backlog**

Key:

|  |  |
| --- | --- |
| Joshua | Nathan |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| User Story | Technical Requirement | Priority (sprint) | Difficulty | Deadline |
| I am able to load files into the database using the GUI. | Write a function which is able to open, read, and parse the given datafile and connect it to the TKinter GUI file dialog.  Must be able to handle:   * Missing data cells * Different data type (string instead of int) * Missing column label * Empty rows | ~ | ~ |  |
| * AEMO .csv files | 1 | 1 | 05/09/22 |
| * Excel files in .xls, .xlsx, .xlsm, etc. | 2 | 1 | 12/09/22 |
| * Excel generated space delimited files | 2 | 1 | 12/09/22 |
| I am able to enter flawed data without the program crashing or completely rejecting the file. | Some errors should have the option for the user to resolve if the program cannot, such as a popup for the user to specify what data type a column should contain or what to do with incomplete entries (delete or include as-is) | 3 | 2 | 10/10/22 |
| I am able to easily get the data I want out of the database. | Write a function that can find and display the data that is queried efficiently and output it to a file.  The user should have the option to choose the output file  Must output to:   * Csv file * Excel file * Space formatted text file | 1 | 2 | 12/09/22 |
| Modify the GUI so that the user can utilise this function effectively | 2 | 1 | 16/09/22 |
| * The output is in the correct format. | The format should be consistent no matter the input file or file type. | 1 | 1 | 05/09/22 |
| I am able to request data even from different components | Write a function that can read through all queries given and find the data from all selected components.  Must be able to:   * Find all instances of a search even though if are in different files. * Be able to display the data correctly despite the different categories. | 2 | 1 | 19/09/22 |
| I am able to specify what date/time range of the data to view | Write a function that can filter the data down from a full year to only entries between a given datetime or only AT a given datetime  Must:   * Give a detailed error if the date/time range specified does not exist. | 2 | 2 | 26/09/22 |
| I am able to select exactly what data to display arbitrarily (any collection of data from any region,component, or year) | Add a section to the gui that allows individual years to be selected separately rather than just working down the hierarchy.  This means instead of selecting a region to display the data from, the user could select a region, component, and year, and then add data from a completely separate region to be added to the output.  This may mean making multiple queries or constructing a very complicated query. | 3 | 1 | 12/10/22 |
| I am able to enter a custom SQL query to the database through the GUI | Write a function that gives the user a text box to input the query.  The program should check this query to ensure that it is a valid request. If it is not an error should be given.  The custom query should allow the user to insert any query that is not possible through the simpler drop down menu system.  The system should output correctly despite the custom query. | 4 | 1 | 12/10/22 |
| If the program cannot understand an input or detects an error I can tell the program what to do to correct or interpret it. | Modify GUI to give functional error popups instead of just messages so that the user can handle them where possible.  E.G. if columns of a data file are missing labels ask the user what the labels should be. | 4 | 3 | 12/10/22 |

## **4. Sprint 1 Test Log**

|  |  |  |  |
| --- | --- | --- | --- |
| Feature | Requirement | Test case | Test result |
| Load: | Each filetype should be accepted and mistakes such as unexpected datatypes, missing entries, and missing column labels should be detected and either handled or an exception raised. |  |  |
| -.csv files | .csv files should be parsed accepting any size of file, any number of columns/ rows, including rows with empty entries, or unexpected data types such as strings instead of an integer | The raw AEMO files were tested as the project requires, along with other, smaller .csv files designed to stress the error handling by having arbitrary data types, missing entries, incorrect or missing separators, and unreadable data. | In every case the program either succeeded in parsing the input file OR raised an exception which would allow a future component of the program to handle the error without crashing. |
| -Alternative delimited files | Ascii based files of delimited formats other than .csv such as tab separated values or space delimited files should be accepted by the program regardless of what character is used to separate their values even if they are chosen arbitrarily by the user. | The .csv files used for previous testing were modified to use other delimitation characters and used as test inputs for the program. |
| -Excel files | The excel file type is commonly used by our client and should be accepted by the program for our database. | The processed AEMO data sample sent by the client was used to test the program’s acceptance of excel files. Similar modifications were made as above to test the program's handling of bad inputs: missing data, missing column names, empty rows, and unexpected datatypes. | Xlsxwriter package did prove insufficient for all of our needs and was mostly replaced with openpyxl instead, although some functions were retained for their utility.  Ultimately the program was able to accept excel files of any type without issue. |

## **5. Sprint 1 Review**

5.1 Functional goals

5.2 Completed tasks

5.3 Quality assessment

5.4 Client needs

* Data from files in .csv, .xls, .xlsx, .xlsm, excel generated space delimited files could be loaded into database using a graphical user interface.
* Design database that includes region, component and year tables.
* User will input component, region, year using input features implemented within graphical user interface.
* Each time user requests data, each row is read from the from the input file(.csv), each row is converted to column format and is displayed as output.
* The data extracted from input files(.csv files) is stored in database.
* The user can select to view the output data in various formats such as csv file, excel, space formatted text file.
* The program does not crash while handling large data.
* When the user enters incorrect data for the year, component, region input column within the graphical user interface, appropriate error message is displayed.

5.5 Key Metrics

**Criteria :**

**Task :** The features implemented for the project.

**Quality :** Meeting the standard required for the project.

**Ease-of-use:** Easiness for client while using the features implemented for each task.

**Learnability:** Features implemented within the software are easy to adapt.

**Result accuracy:** The output achieved for each feature meets the required outcome.

**Task Ratings :**

1. 0 – 10 : Score to rate each criteria that is fully completed.
2. N/A :

\*1 The tasks that we are still working on.

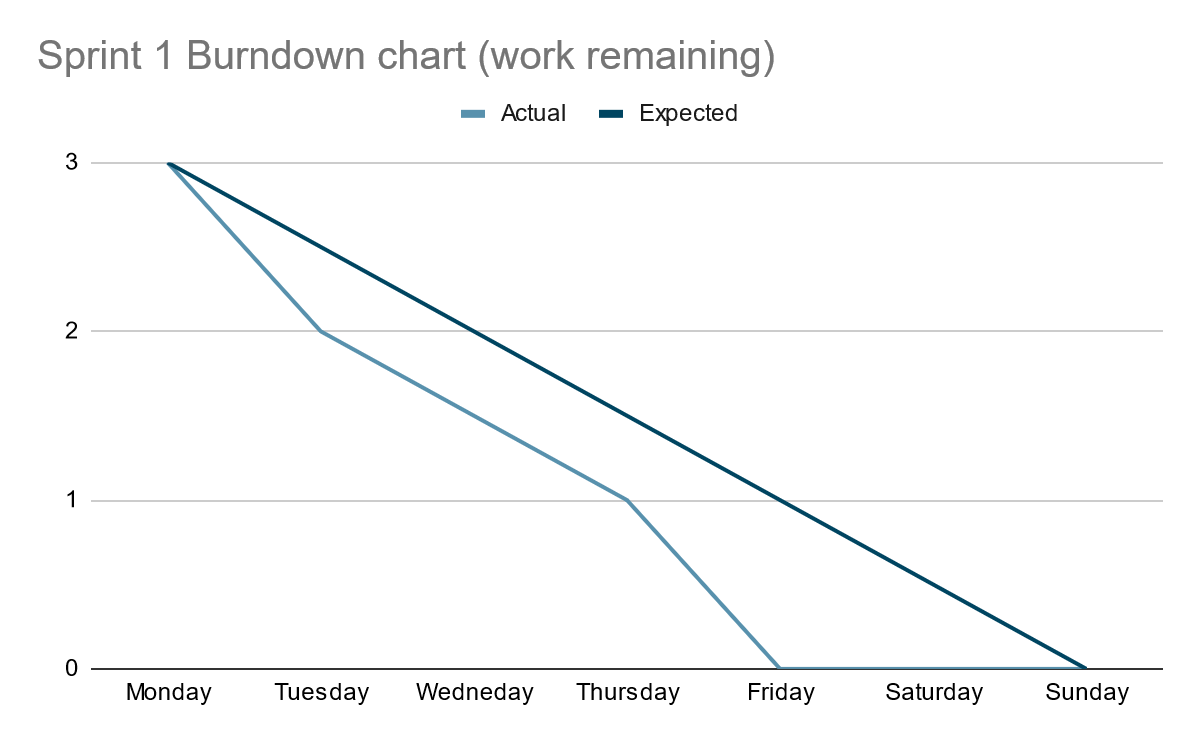
\*2 Features to be implemented in the future.

\*3 Features that are being tested .

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Task | Quality | Ease-of-use | Learnability | Result accuracy |
| Load files(read, parse, open into database using GUI. | 9 | N/A\*1 | 9 | 10 |
| Extracting data for specific region | 7 | N/A\*1 | N/A\*3 | N/A\*1 |
| Extracting data for specific component | 8 | N/A\*1 | N/A\*3 | N/A\*1 |
| Extracting data for specific year | 8 | N/A\*1 | N/A\*1 | N/A\*1 |
| Integration of region, component, year to work together | N/A\*2 | N/A\*2 | N/A\*2 | N/A\*2 |
| Entering flawed data without program crashing. | N/A\*2 | N/A\*2 | N/A\*2 | N/A\*2 |

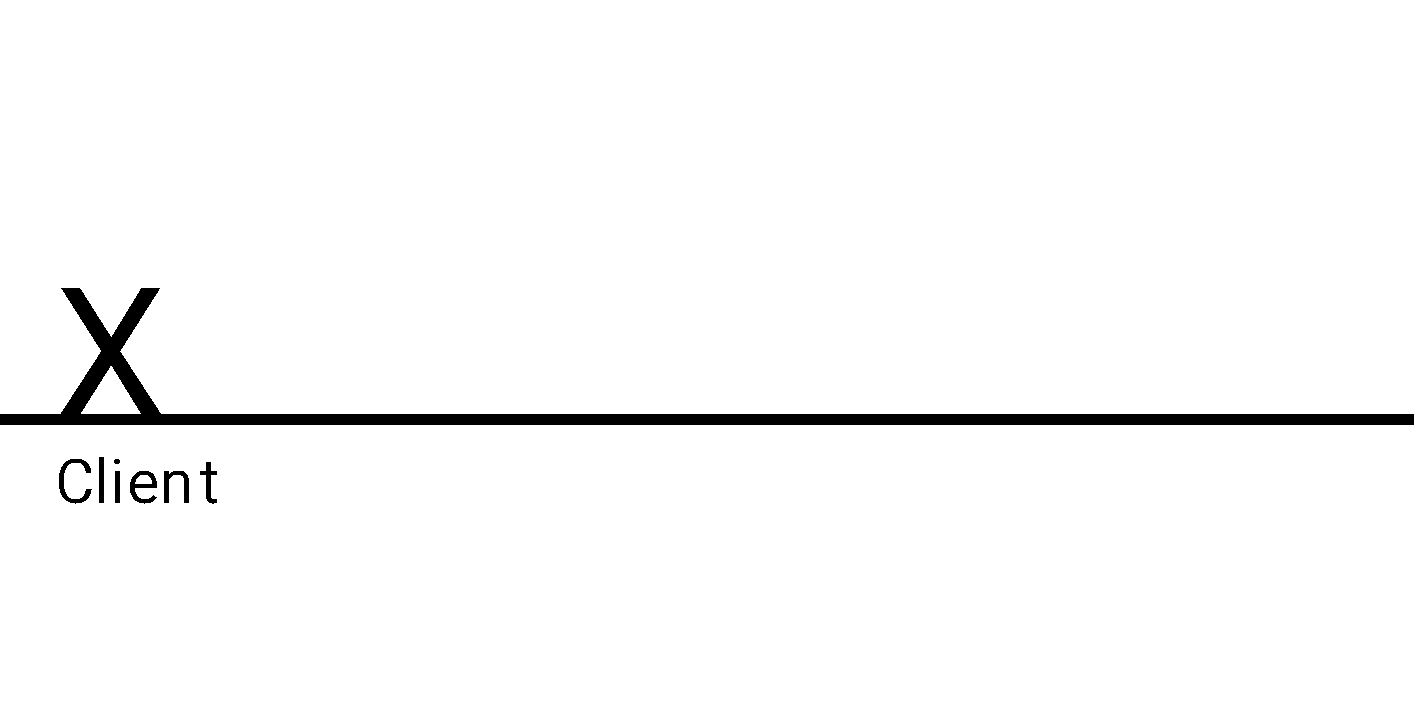
## **6. Sprint 1 Burndown**

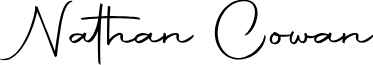
6.1 Nathan Cowan



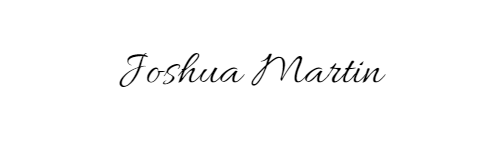
6.2 Joshua Martin

# **4. AGREEMENTS**

****

Nathan Cowan 

Akshay Devnani Akshay Devnani

Joshua Martin 

Naman Sharma