**3 Solution Description and System Requirements**

**3.1 Solution Description**

The aim of the project is to create a software solution that provides to a user an interactive data visualisation suite to support research in the area of short-term electricity load forecasting. Fundamentally, the manual processes of analysis detailed in **chapter 2** will be automated in the solution for the user to perform investigate analysis on the load dataset. Users will be able to discover patterns in an historical load dataset through dynamic visualisations of system load and related variables. Load forecasting models that are known to be produce accurate load forecasts through analysis in **chapter 2** will be incorporated into the project. Users will be able to visualise the model forecasts to perform a systematic comparison of each model’s load forecasting accuracy with accompanying error statistics over the historical data. For a day ahead forecast, the solution will automatically choose a single performant model or average of performant multiple models to produce an accurate load forecast.

This investigative analysis that will be facilitated by the solution would only otherwise be available by using a specialist computing environment, for example MATLAB. These environments commonly require commercial licences and an assured programming background to be able to achieve basic data analysis tasks. The solution will be a free open source alternative that allows for complex tasks to be performed through a functional user interface. The solution will be implementation agnostic and only require an elementary knowledge of load forecasting.

**2.1 Target Users**

Tailoring the solution to a specific user allows to realise the experience of a user using the software to be realised in development, and hence the software requirements can be prioritised to what value they provide to a user. Following discussions with the stakeholder the following users were identified as being the two target users of the solution:

**2.1.1 Academic User**

*Motivation for using the solution:*

Academic users can use the solution as an introductory tool to develop a foundational understanding of the challenges of forecasting load for a specific dataset and identify the load forecasting models that have perform well for a range of the dataset.

*What would they do if this solution didn’t exist:*

Academic users can use data analysis and visualisation environments such as MATLAB and Jupyter Notebook to:

* arrange and clean the load dataset
* create load forecasting model functions
* evaluate model performance visually and statistically.

*Value gained by using the solution:*

The solution enables the academic user to automate the complex mathematical functionality required for load forecasting that they would have otherwise had to create manually in a specialist data analysis environments.

**2.1.2 Industrial User**

As the dataset for the solution is from SONI, employees of SONI whose role is to produce accurate load forecasts a day ahead are a ‘real’ industrial user that the solution targets.

*Motivation for using the solution:*

Industrial users can use the solution to generate a day ahead load forecast for a chosen day using a forecast through prior analysis to perform optimally for that type of day. For a system operator they can use this forecast to buy electricity from a day ahead market. The greater the accuracy in the load forecast to the real system demand the more valuable the solution is to them.

*What would they do if this solution didn’t exist:*

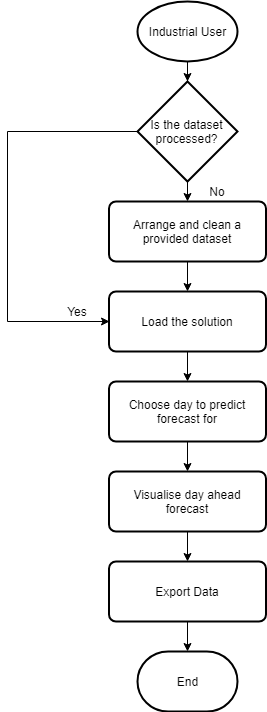
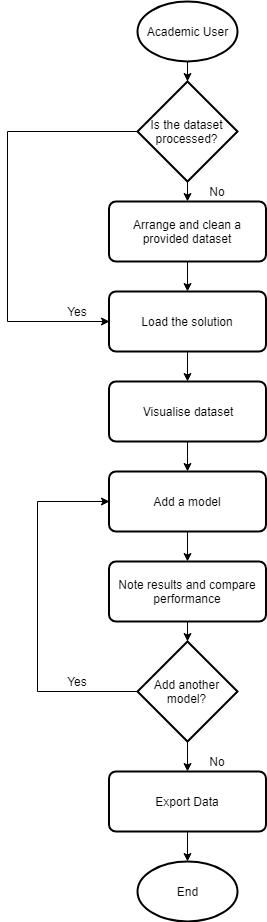
Industrial users can use data analysis and visualisation environments to perform the same operations as an academic user. Crucially, they would have to manually choose a model identified through their own individual analysis to have the best performance for the type of day they are forecasting as the day ahead model forecast.

*Value gained by using the solution:*

The solution enables the industrial user to generate an accurate day ahead load forecast.

**2.2 User Flows**

The expected flows of the target users through solution are visualised below:



**Figure 2.3 (A)** Flow chart visualising the expected flow of an academic user using the solution. **(B)** The expected flow an industrial user using the solution.

**(A)**

**(B)**

**2.3 Use Cases**

A specification was derived from the use cases of the prospective users of the solution identified in **2.3.** The specification is written from the viewpoint of the customer and captures the value provided to the user by the solution and not the developer [3].

Academic User Use Cases:

* Arrange and clean a provided dataset through an automated step-by-step process.
* Dynamically visualise a dataset with load values to learn about the dataset’s historical trends and characteristics.
* Use known good performing load forecasting models to compare their performance over historical date ranges.
* Augment the list of known well performing load forecasting models with experimental models identified to be perform optimally in their own analysis.
* Export visualisations and statistical values to include in an analysis paper

Industrial User Use Cases:

* Arrange and clean a provided dataset through an automated step-by-step process.
* Visualise the day ahead load forecast for a chosen given day.
* Export the forecasted load values as their guide forecast.

The two identified target users can be considered ‘actor’s with behaviours associated with them. A ‘generic user’ that has the shared behaviour of industrial users and academic user is identified to prioritise these for development as implementing them provides value to all targeted users.



<<extends>>

<<extends>>

Industrial User

Academic User

Generic User

**Figure 2.2** A UML representation of the actors and the behaviours associated with the actor. The industrial user and academic user inherit the behaviour the generic user has.

A use case grid was created to augment with attributes on the use cases identified in the visualisation and assign an ID that can be used to link with requirements:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case ID** | **Use Case Name** | **Primary Actor** | **Complexity** | **Priority** |
| 1 | Arrange and clean a provided dataset | Generic User | Low | High |
| 2 | Export data | Generic User | Medium | Medium |
| 3 | Visualise dataset | Academic User | Medium | High |
| 4 | Add models | Academic User | High | High |
| 5 | Compare model performance | Academic User | High | High |
| 6 | Visualise day ahead forecast | Industrial User | Medium | Medium |

Developing the use cases using this approach facilitated an iterative process that is accommodating to proposed feature additions and changes by the stakeholder. This approach is derivative of the approach in [7].

**Table 2.1** Use case grid. Complexity: the perception of the difficulty in implementing the feature into the solution. Priority: priority of delivering the feature to the primary actor.

**2.4 Functional Requirements**

The functional requirements of the system are written as minimum viable features (MVFs) derived from the use cases in **Table 2.1.**  MVFs are ‘minimally viable’ as they provide value to a stage in the user flow. The analysis work done on the SONI dataset in the prior chapter determined which proposed features can leverage the dataset and which cannot be implemented because of the limitations of the dataset. MVFs include user stories that fulfil an atomic part of the feature and relate directly to implementation work. User stories are prioritised to ensure development time is focused on high priority items to the stakeholder. A user story identified can be consider a “nice to have” to the target user and hence not necessary to the deployment of the MVF and being parked. A single feature is developed at one time and the progress of it and its associated user stories is visualised using a virtual Kanban board. The IDs of both are included in the virtual Kanban board to relate to this documentation.

The format the features are listed in is as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Feature ID** | **Feature Name** | | | |
| User Story ID | User Story Complexity | User Story Priority | A high-level user action | Status – done, backlog or parked |
| * Acceptance criteria: user functionality listed that must be tested for the user story to be ‘done’ | | | | |

Feature List

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **FTR00** | | | **Visualise and interact with the historical load data** | | | | | | | | |
| US00 | | | High | | | | Open the application in a web browser | | Done | | |
| * View the application in a web browser by navigating to an URL. | | | | | | | | | | | |
| US01 | | | High | | | | View a graph visualising historical load data | | Done | | |
| * The full date range of historical load data must be plotted. * Date values on the x axis labelled. * Load values on the y axis labelled. | | | | | | | | | | | |
| US02 | | | High | | | | Interact with the date range visualised. | | Done | | |
| * The range of load data visualised must change when changing the start and/or end date. * The range of load data visualised must change when increasing or decreasing the start or end date by a date-time unit. * Midnight of the end date must be the final point plotted. | | | | | | | | | | | |
|  | | | | | | | | | | | |
| **FTR01** | | **Evaluate the forecasting performance of a displacement model** | | | | | | | | | |
| US03 | | High | | Visualise displacement model’s load forecasting performance | | | | | | | Done |
| * The user must be able to choose what unit of time and magnitude for the displacement model construction. * When a model has been added the displaced plots must superimpose the visualised load data. * The new model’s name should appear along with the ‘Actual Load’ legend with matching colours to points on the graph. The name of the model should be in the format: *displacement | unit* e.g. -7 Days * The date the load forecast is derived from must be visible hovering over a model forecasted point. * The forecasting model forecast must change when changing the visualised date range. | | | | | | | | | | | |
| US04 | | High | | View forecasting performance metrics of the visualised displaced model | | | | | | | Done |
| * When a model is added to the load visualisation graph a table containing statistical data relevant to judging the model’s predictive performance must be visible with the identifying column named the model name in the visualised graph. * When the visualised load date range changes the statistical data in the table updates dynamically. * The table is not visible when no models are added. | | | | | | | | | | | |
|  | | | | | | | | | | | |
| **FTR02** | | **Compare multiple forecasting models’ performance** | | | | | | | | | |
| US05 | | High | | Compare two displaced model’s forecasting performance | | | | | | | Done |
| * When two models have been added two forecast model plots with different styles should be superimposed onto the visualised load data graph. * Two displaced model’s forecasting performance statistically presented through two distinct columns in the metrics table with the model’s name as the identifying column name. * If test data is selected a sub column labelled ‘Test Data’ must be added to the table beside the model’s visualised data metrics. * When a model is removed the columns should be removed from the metrics table and not superimposed on the graph. | | | | | | | | | | | |
| US06 | | High | | View two graphs with different y axis over the same time range to compare characteristics contributing to differences in model performance | | | | | | | Done |
| * The user should be able to add or remove a characteristics graph. * Y axis options are the dataset’s columns. * Changing the visualised date range of the Date-Load graph changes the characteristics graph’s visualised date range. * If a model is plotted, the displacement models’ plots of the chosen y-axis are superimposed onto the characteristics graph. * The date x axis of the visualised load graph must not be visible. * The date x axis of the characteristics graph must be visible. * The grid lines of the characteristics graph should be seamlessly aligned with the gridlines of the visualised load graph. | | | | | | | | | | | |
|  | | | | | | | | | | | |
| **FTR03** | **Advanced Visualisation and Interaction** | | | | | | | | | | |
| US07 | Medium | | | | | Highlight different types of days | | | | Done | |
| * When choosing to highlight a specific holiday day, the full list of holidays in the dataset must be listed. * When choosing to highlight a specific year, the full list of years in the dataset must be listed. * The selected chosen type of day must be plotted as red points superimposing the load points if matching the day characteristic chosen to highlight. * When a highlighted day type is chosen, the legend must include ‘Highlighted Load’. | | | | | | | | | | | |
| US08 | High | | | | | Visualise the error distribution of a forecasting model | | | | Done | |
| * An error distribution graph must be visible in a separate tab if model(s) are added. * APE values on the x axis labelled. * Cumulative percentage values on the y axis labelled. * The forecasting model points should be the same colour as in the load visualisation graph. * When a model is added/removed it should be added/removed from the error distribution visualisation graph. * If there is test data range selected the test data model forecasting error points must be visible on the graph. * Visualised data forecasting error points must have the model name and (Visualised) appended to it in the legend. * Test data forecasting error points must have the model name and (Test) appended to it in the legend. | | | | | | | | | | | |
| US09 | Low | | | | | Choose how the graph is presented | | | | Parked | |
| * When the marker dots properties (e.g. colour, width, line type) are modified the forecast model and load points visualised must have the properties specified. | | | | | | | | | | | |
|  | | | | | | | | | | | |
| **FTR04** | **Assess the forecasting performance of linear regression models** | | | | | | | | | | |
| US11 | High | | | | Visualise a linear regression forecasting model to evaluate its performance | | | Done | | | |
| * The list of models the user can choose must be the models included in the dataset configuration file. * The user must be able to choose a training data start date and end date. * The user must be able to view the variables used in the linear regression model and a description of the linear regression model. * The user must be able to change the model name from default. * The linear regression model plots must be superimposed onto the visualised load data. * The name of linear regression model must be visible in the legend of the load and error visualisation graphs. * If the regression model has a default training range, it must by default change the start date and end date to it when changing the model chosen. | | | | | | | | | | | |
|  | | | | | | | | | | | |
| **FTR05** | **Export the visualisations and statistical metrics** | | | | | | | | | | |
| US12 | High | | | | | Export the visualised graphs | | | Backlog | | |
| * When the user clicks export a zip file should be downloaded to their local machine that includes: * If a range of data is selected, the load visualisation graph with, if model(s) are added, superimposed model forecasts named ‘load\_graph’ * If a characteristic is selected, the characteristics graph named ‘characteristics\_graph’ * If model(s) are added, the model error visualisation graph ‘model\_error\_visualisation\_graph’ * The format of the visualisations graph images must be SVG. * The zip file should be named ‘lftool\_export’ | | | | | | | | | | | |
| US13 | High | | | | | Export the metrics table statistical data | | | Backlog | | |
| * When the user clicks export a zip file should be downloaded to their local machine that includes the statistical data in the format:  |  |  |  | | --- | --- | --- | | **Model Name** | **Data\_Type** | **Metrics…** | | *Name of the model in the visualised graph* | *Visualised or Test Data* | *The metric value* |  * The format of the data must be CSV. * The name of the data must be ‘metrics’ * The metric values must be added to their calculated raw value, not the rounded values in the metrics table. * If no models are added the file will not be included. | | | | | | | | | | | |

**2.5 Non-Functional Requirements**

From the use cases the following non-functional requirement that describe the system behaviour and characteristics were identified as being important to the target users:

Usability

* Functionality is prioritised over accessibility. Users must be able achieve their desired functionality with a minimal number of interactions.
* Each user interface control must have a descriptive label or tool tip describing its functional purpose.
* Constructing a model must be understandable as to what the structure of the model will be.

Responsiveness

* User should be able to change what is visualized without encountering an extensive processing phase or a page refresh.
* A model added should not require an excessive amount of time before being visualised and its accompanying statistics listed.

Quality

* Users should consider the functionality offered by the solution comparable to commercial offerings.

Extensibility

* Users should be able to format an electronic dataset following a contract to be able to perform the same operations on the dataset as the SONI dataset.
* Users should be able to engage with other forecasting models implemented in the future without any noticeable change in method of interaction.

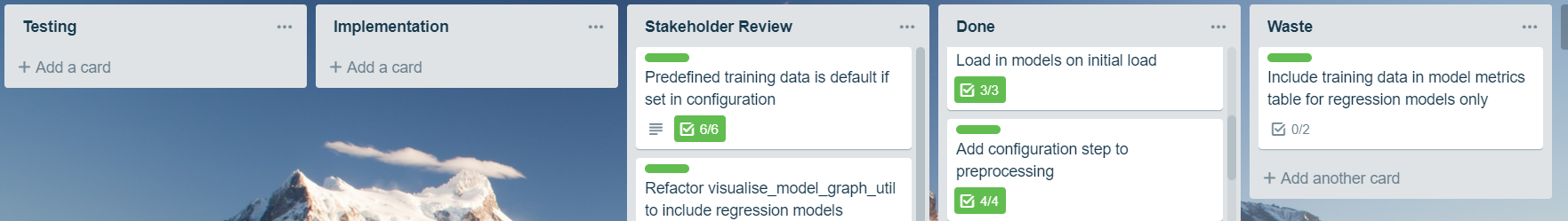
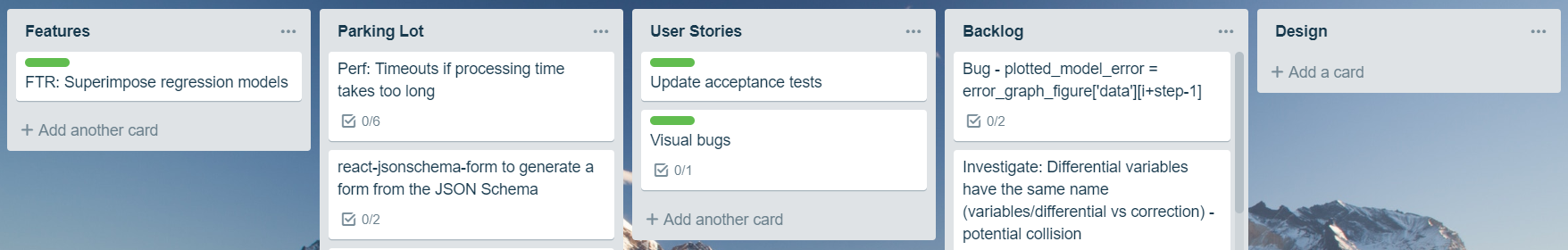
Robustness

* The solution should not unexpected behaviour with a relevant error message visible to the user.
* The system should not crash unexpectedly.

**2.6 AGILE Development Process**

The solution was developed using an agile approach [5]. The specification of the project was gathered from the dissertation’s supervisor, referred to in this chapter as the stakeholder, through face to face initial meetings. Continual weekly meetings with the stakeholder generated continual feedback of developed functionality that ensured it met stakeholder expectations. Guidance on the complex mathematical processes used in the analysis initially was verified by the stakeholder before the implementation was integrated into the solution. The requirements and their prioritisation evolved as both the developer’s and the stakeholder’s appreciation of the cadence and the implementation difficulty of desired features developed. The non-traditional approach to gathering requirements in an agile reactive manner facilitated this evolution.

Trello [4] was used as the main process tool as a virtual Kanban board, **see Figure 2.1.** This tracked the progression of MVFs and associated tasks. They were added to the board and their progression from initial analysis and completion (done) was tracked. The limit (WIP – work in progress) of tasks/stories allowed to be in the design, testing and implementation stages was one, which ensured that work was delivered as fast as possible with stories and tasks’ taking longer to complete being flagged. There were no time boxed sprints, instead rolling development of MVFs was on a development (feature) code branch and once a feature was completed it was promoted to the deployable master code branch after stakeholder verification and testing.



**Figure 2.1** An example of the virtual Kanban board on Trello

Meaning of each stage:

* Features: List of MVFs currently being developed with a colour code (green) that links stories with the feature.
* Parking Lot: User stories that are blocked by planned user stories – unable to be picked up for development as they
* User Stories: Stories that are planned to be developed on to fulfil the feature
* Backlog: Stories/Tasks not associated with a feature that are improvements and minor bug fixes.
* Design: Stage in which the user story is, if sensible, broken into smaller tasks and the acceptance criteria is defined for acceptance testing. The implementation details are added to the tasks after an investigation of the solution.
* Testing: The testing defined in the acceptance criteria occurs and new test cases are added to the manual acceptance tests documentation.
* Implementation: The stories and tasks currently being worked on
* Stakeholder Review: The stories and associated tasks that have yet to be demonstrated to the stakeholder for verification they meet their expectations.
* Done: Stories and associated tasks that are fully tested and passes the stakeholder’s evaluation.
* Waste: Stories and tasks previously analysed that are now not required to fulfil the MVF.

**2.7 Work Plan**

**Figure 2.2** A high-level visualisation of planned functional features (blue) and project deadlines (red):

**2.8 System Constraints**

There are constraints identified that influence the design of the solution:

* There is a requirement by the providers of the dataset SONI to treat the load dataset in a confidential manner as it not publicly available and has commercial value for their competitors. The design should incorporate a prevention mechanism to mitigate the scraping of data.
* Data manipulation functions that can be done before processing should be pre-processed. With multiple potential clients of the solution this would cause unnecessary processing on the server end.
* Parallel processing in data manipulation and streaming leveraging multithreaded processors should be a consideration.

**Citations**

**[1] https://www.devbridge.com/articles/lean-requirements-in-practice/**

**[2] Non Functional Requirement in Agile Software**

**Development**

**[3] Requirements Management in an**

**Agile and Lean Environment**

**[4]** [**https://trello.com/en**](https://trello.com/en)

**[5]** <https://agilemanifesto.org/principles.html>

**[6] Se.rit.edu, 2019. [Online]. Available: http://www.se.rit.edu/~se555/Use%20Case%20Overview.pdf. [Accessed: 25- Mar- 2019].**

**[7]** <https://www.gatherspace.com/use-case-examples/>

**[8]** <https://www.scaledagileframework.com/nonfunctional-requirements/>