Technological University Dublin
Blanchardstown Campus

SCHOOL OF INFORMATICS AND ENGINEERING

SINGLE ELECTRICTY MARKET DATA

by

Dean Ross

A report submitted in partial fulfilment of the requirements for the degree

BACHELOR OF ENGINEERING (HONOURS)
IN
COMPUTER ENGINEERING
IN
MOBILE SYSTEMS

SUPERVISOR: MARIE ARMSTRONG
SUBMISSION DATE: 16/05/2021

ABSTRACT

This project was composed of designing and implementing a hosted website which displayed live REMIT Updates from an SQL server database. The project is titled "Single Electricity Market Data". The student sought out to display the Single Electricity Market Data from the Single Electricity Market Operator, SEMO, on the hosted website through using an SQL server database.

This data is sent to email subscribers of SEMO. The purpose of the hosted website is to allow for users/subscribers to view parsed email data from SEMO and view it all in one place.

The REMIT Update data includes information on electricity outages across Ireland and Northern Ireland. The outage information consists of which registered unit is affected, the type of outage that occurred, and the time the outage began and stopped. This information was stored within a SQL server database, using a cloud-based service, Microsoft Azure.

The SQL server database contained parsed email information using Azure's logic app feature. This feature converts the HTML email into a text-based format, creates variables of type integer and string, then stores the variables in a table, "Remit2", within the created database, "semo".

All the REMIT Update data is available on a hosted website, which was created using Visual Studio. The hosted website displayed information regarding the REMIT Update data, project information, and the Registered units with SEMO as of March 2nd, 2021.

This report and project will establish the process of creating a logic app, SQL server database, hosted website, and combining all aspects with Microsoft Azure, to display the importance of both front-end and back-end cloud-based development.

ACKNOWLEDGEMENTS

I would like to acknowledge everybody who played a serious role in my academic life and accomplishments.

First, my parents, Naoimh and Robbie, who have supported me throughout my years in TU Dublin Blanchardstown with both love and encouragement.

Second, my girlfriend, Orlagh, who is my rock, she has always been by my side and has kept me sane throughout my years in college.

Third, my amazing grandparents, and my late grandad Joe who have always motivated me through tough times and gave me a sense of guidance.

Fourth, my friends throughout my time at TU Dublin, specifically, my best friend, Alex, who provided me with plenty of great laughs and fun moments in TU Dublin.

Finally, a special thank you to Marie Armstrong, my project supervisor, for being a phenomenal mentor in the practice of engineering, and a massive thank you to Benjamin Toland for being an amazing year head in a difficult year, and for providing me with plenty of advice throughout my time at TU Dublin.

DECLARATION

The work submitted in this report is the results of the candidate's own investigations and has not been submitted for any other award. Where use has been made of the work of other people it has been fully acknowledged and referenced.

Student Name

Dean Ross

Table of Contents

LIST OF ABBREVIATIONS

AAOU	Affected Asset or Unit
AWS	Amazon Web Services
CER	Commission for Energy Regulation
CSS	Cascading Style Sheet
ECC	European Commodity Clearing
ESB	Electricity Supply Board
GBP	Great British Pound
HTML	Hyper Text Markup Language
IP	Internet Protocol
I-SEM	Integrated Single Electricity Market
MSQL	Microsoft Structured Query Language
MVC	Model-View-Controller
MW	Mega Watts
PC	Personal Computer
REMIT	Regulation on Energy Market Integrity and Transparency
RN	Resource Name
SEM	Single Electricity Market
SEMO	Single Electricity Market Operator
SONI	System Operator for Northern Ireland
SQL	Structured Query Language
TSO	Transmission System Operator
UK	United Kingdom
UR	Utility Regulator

LIST OF FIGURES

LIST OF TABLES

Chapter 1 - Introduction

Integrated Single Electricity Market Data (I-SEM): The student was to research the Electricity Market data within Ireland and Northern Ireland. The student was required to build a website/product that can display collated data. This data was then monitored by the student through a variation of emails sent to the student's email address "B00094969@mytudublin.ie". The student will then monitor the market data emailed to "B00094969@mytudublin.ie". The data that the student will monitor relates to REMIT power outage data. A logic app was used for the project. Microsoft Azure was used to collate and parse the emails into one specific area using a built-in logic app on Microsoft Azure.

The student polled live xml or csv updates, then collate and parse the data on Microsoft Azure's logic app. Once the student has gathered the REMIT power outage data, the parsed data from the logic app will then be displayed on a HTML designed website from the SQL database.

Research was required on the following topics for the understanding of the project:

- I-SEM.
- SEMO.
- Other European Markets.
- Logic apps.
- Implementation of HTML design.
- SQL Databases.
- Power Grid in Ireland/Northern Ireland and European countries.

Research was carried out on SEMO, a joint venture between EirGrid of Ireland and SONI of Northern Ireland. Research was conducted on SEMO. This was mainly done for getting a better understanding of their goals for Ireland and Northern Ireland. This research also consisted of learning the ways of the Integrated Single Electricity Market. This included the standards for the Market, the day-to-day trading, operating hours, and order types. [1] [2]

Research was also carried out on other countries across Europe. The two markets other than I-SEM that were selected were the German Electricity Market and the UK's Electricity Market. These markets were selected to determine what their electricity market looks like and how it operates compared to I-SEM's market. Germany was chosen as it is still within the EU, and the UK was chosen as they do not operate within the European market anymore due to Brexit.

Research was key to the student as it gave a better understanding of a new engineering field within Ireland/Northern Ireland.

This project was chosen as it gave the student a new insight into an upcoming market, not just in Ireland/Northern Ireland, but in Europe and even the United States of America. This project gave the student an understanding of engineering business models and enabling the student to be knowledgeable in a new engineering practice, which could lead to the student being employable to an electricity supplier such as SEMO.

The skills that were learned throughout this project would be applicable to any published xml data, which can then be collated, analysed, and displayed on various platforms.

Problem

In 2018, a new wholesale market was founded. This market is known as the Integrated Single Electricity Market, also abbreviated as I-SEM. I-SEM is operated by a Single Electricity Market Operator, SEMO. SEMO is a joint venture between Ireland’s EirGrid power grid supplier and Northern Ireland’s SONI, System Operator of Northern Ireland. They have a shared goal of qualifying with Europe’s standards for a broader electricity market.

Ireland hopes to make itself more self-sustainable by 2030, with 70% of its electricity coming from renewable energy, such as wind, solar and hydro. However, to do this, Ireland must improve its power grid across the country.

This project will discuss the aims to make the student more knowledgeable to the I-SEM project and how the student can create an app/website that analyses collated and parsed data.

I-SEM is a new wholesale market for both Ireland/Northern Ireland. The market is integrating both Ireland/Northern Ireland’s electricity market with Europe’s electricity markets. With the I-SEM project, it enables the free flow of energy without borders.

The student is to design, build and update a website/product that displays data collated from their student email “B00094969@mytudublin.ie”, this email enables the student to be sent new emails frequently of any power outages across Ireland/Northern Ireland, and changes in the market data. The student created a logic app, database, and website that collects REMIT Updates across the island and displayed the information on a designed website for analysing the data.

Background

In 2007, a wholesale Single Electricity Market was founded known as (SEM). It is regulated by the SEM Committee. However, on October 1st, 2018, I-SEM replaced the SEM. The new market has been designed and overseen by the SEM committee. This committee consists of the Commission for Energy Regulation (CER), Utility Regulator (UR) and an independent member/deputy independent member.

The single electricity market is new within both Ireland/Northern Ireland. There was a demand for electricity to be more affordable, cleaner, and available to consumers in the two countries. Specific regions within Ireland are constantly losing power, such as the West of Ireland. EirGrid have planned to increase the size of Ireland's power grid, thus generating more electricity for affected regions within Ireland.

Benefits from the inception of I-SEM:

- Access to cheaper forms of electricity.
- Open and efficient European electricity markets providing benefits to its consumers.
- A new form of market with the introduction to future markets.

Scope and Objectives

The goal of this project is to focus on learning the new electricity market in Ireland & Northern Ireland, and to implement the data from SEMO into a cloud-based database, these techniques will be shown later within the report.

The aim of the project is to apply methods for creating a cloud-based database using the created Microsoft Azure account, then displaying the stored data on a user created website using HTML, ASP.NET, C#, and SQL coding languages and techniques. This will allow the user to check the resulting outages sent from the REMIT officer to the student's email account.

Document Overview

Chapter 1 – Introduction: This chapter covers information for the project. It is within this chapter that the student will display a guided tour for the remainder of this report. It is more than just a list of the following chapters to come.

Chapter 2 – Literature Review: This chapter covers the materials that were researched for the development of the project. It also displays the ideas that were thought of and further explored.

Chapter 3 – Implementation and Design: In this chapter the student discusses the techniques and coding languages that were required for the completion of the project, this chapter also includes a step-by-step guide on how to access the SQL database, and how the logic app was designed to parse the REMIT Update data, project testing that was required for each step, and how each of the webpages within the website looks.

Chapter 4 – Results and Discussions: This chapter primarily covers a discussion on the student’s thoughts and opinions on the project, such as areas in which they could have improved, and if there were any other projects similar.

Chapter 5 – Conclusions and Recommendations: This is the final chapter in the report that conveys the student’s conclusion to the project, it also includes areas in which the student felt he could have made the project better and more user friendly.

References.

Appendices (A, B, C, D): The Appendices include project planning, salient extracts from the students learning journal throughout the project, the software code required for the project, and any drawings that were used for the project.

Chapter 2 - Literature Review

The electricity market in Ireland and Northern Ireland is a relatively new wholesale market to consumers in Ireland and Northern Ireland. The Integrated Single Electricity Market, I-SEM, is to integrate both Ireland and Northern Ireland, for the purpose of integrating both the all-island and European electricity markets. This concept will allow for the free flow of energy between borders. SEMO is a joint venture between Ireland's EirGrid and Northern Ireland's SONI. Both companies came together to found SEMO. SEMO offers a subscription service via email sign up. There are multiple options to choose from when signing up. For the purpose and intention of this project, the student solely signed up to receive REMIT Update emails to their student email. Emails are sent to subscribers regarding electricity outage information. The email will contain the following information: Message ID, Affected Asset or Unit, Resource Name, Type of Unavailability, Event Start and Stop Times, Available Capacity Megawatts, and Remarks. These emails contain information that allows the subscriber to see the outage information for the registered units with SEMO.

The student set out to create a website which will display live updates on the market, by displaying the email information shown above. This will allow for SEMO Remit Update subscribers to access all the information in one place, instead of going through emails to find outage information.

The Literature Review chapter will cover what was needed by the student for the success of the project, such as knowledge of the European markets, project timelines, studying of logic apps, and knowledge on cloud-based services.

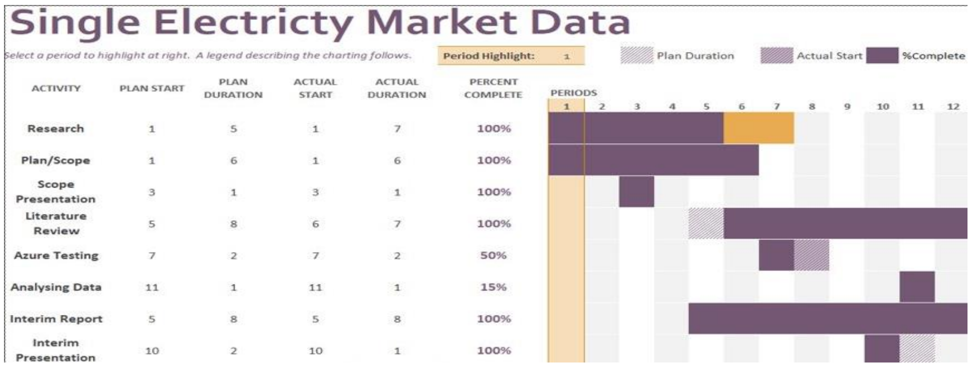
Project timeline for Semester 1.

Below in tabular form is a guideline that was created and followed by the student for the project timeline in Semester 1.

Table 1: Timeline for Semester 1.

Project Timeline in weeks			
Activity	Plan Start	Plan Duration	Percent Complete
Research	1	7	100%
Scope	1	6	100%
Scope Presentation	3	1	100%
Literature Review	6	7	100%
Azure Testing	7	2	100%
Data Analysis	11	1	100%
Interim Report	5	8	100%
Interim Presentation	10	1	100%

The table was created to aid the student in keeping track of the deliverables that were needed for the completion of the project. A Gantt chart was created to give the student a visualisation of the project timeline.



Single Electricty Market Data

Select a period to highlight at right. A legend describing the charting follows.

Period Highlight: 1 ▨ Plan Duration ▨ Actual Start ■ %Complete

ACTIVITY	PLAN START	PLAN DURATION	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE
Research	1	5	1	7	100%
Plan/Scope	1	6	1	6	100%
Scope Presentation	3	1	3	1	100%
Literature Review	5	8	6	7	100%
Azure Testing	7	2	7	2	50%
Analysing Data	11	1	11	1	15%
Interim Report	5	8	5	8	100%
Interim Presentation	10	2	10	1	100%

Figure 1: Gantt chart for Semester 1.

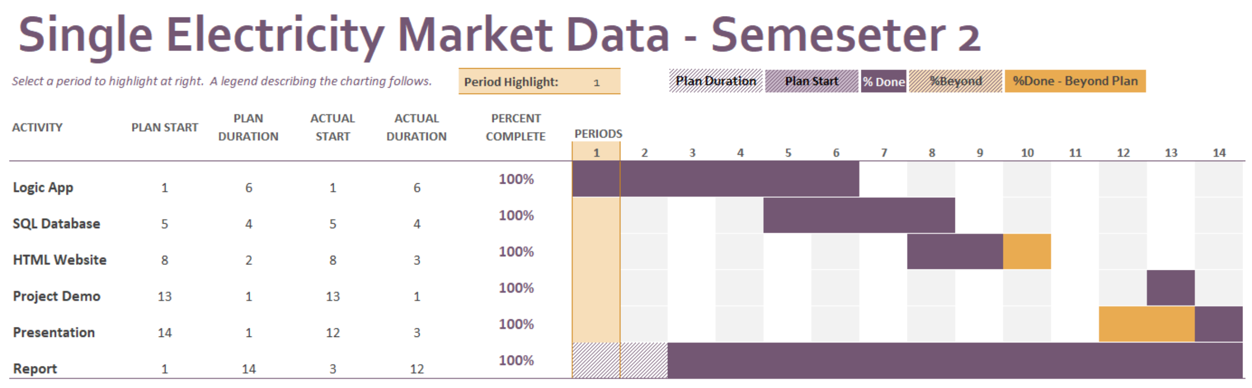
Project Timeline for Semester 2.

As seen below within a tabular format is a guideline which was followed by the student for their project timeline for Semester 2.

Table 2: Timeline for Semester 2.

Project Timeline in weeks			
Activity	Plan Start	Plan Duration	Percent complete
Logic App	1	6	100%
SQL Database	5	4	100%
HTML Website	8	2	100%
Project Demo	13	1	100%
Presentation	14	1	100%
Report	1	14	100%

The table was useful as it kept the student on track for meeting the project deliverables in May 2021. A Gantt chart was also replicated, to give the student and the project supervisor a visual image of the project's timeline.



ACTIVITY	PLAN START	PLAN DURATION	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE
Logic App	1	6	1	6	100%
SQL Database	5	4	5	4	100%
HTML Website	8	2	8	3	100%
Project Demo	13	1	13	1	100%
Presentation	14	1	12	3	100%
Report	1	14	3	12	100%

Figure 2: Gantt chart for Semester 2.

The Gantt chart offered the student a chance to visually see the deadlines for the project deliverables. This enabled the student to stay on track. However, there were a couple of scenarios when the student started a deliverable later or earlier than expected.

What is Microsoft Azure?

For this project, Microsoft's Azure was used. Other cloud-based services were considered, such as AWS and Google cloud. Microsoft Azure is a cloud-based computing software that enables users to build, test and deploy created software models by the user. Azure was designed and created by Microsoft. It is primarily used by large scale companies and often hobbyists/students, for when a cloud-based platform is required. Azure offers multiple applications for use upon signing up for the service. Such as Virtual Machines, Virtual Networking, Firewall Access, Storage such as an SQL database, App services for creating a user designed HTML website, and much more. Azure offers limitless opportunities for any user.

There were a few applications needed for the completion of this project. The Azure applications that were required by the student: logic app, sql database, and a web service. The web service was created separately from Azure, but the student's account was linked to the cloud-based service. Both the logic app and sql database were created on the Azure interface.

The logic app that was created will retrieve emails sent to the student from info@sem-o.com, these emails will contain REMIT Updates regarding recent changes to electrical power being provided, the updates will include information for if a generator is not producing any electrical Megawatt capacity. The logic app will convert the email into text, and store the text fields within a database, using variables. The SQL database will be displayed on the Web service.

The app will enable the user of the system to view incoming data and analyse in real time by utilising a display data button.

Figure 3: Microsoft's Azure logo. [3]

Microsoft Azure Vs. AWS & Google Cloud.

When choosing the design of this project, there were three available cloud-based web service providers that were available to choose from: Microsoft Azure, AWS, and Google Cloud. Below are some advantages for each web service provider:

Microsoft Azure:

- This cloud-based service was used within a module in semester 1. The student has a basic understanding to the concepts that are used within the platform.
- There is a 12-month free trial with $200 credit on the account.
- It is both Windows and Linux compatible.
- There are large service choices available.
- It is very fast and responsive.
- Very user friendly.

AWS:

- Consistent and Reliable. Known for backups and recovery.
- It can be flexible and scalable to individual users. Whether a small company or large company, AWS meets the needs of each person on a personal or professional level.
- Pay as you go pricing plan.
- Highly customizable to each user.

Google Cloud:

- User friendly.
- Allows for big data to be stored, such as machine learning (AI).
- Its global.

For the reasons shown above, Microsoft Azure was chosen for the remainder of this project. [4]

What is a Logic App?

A logic app is a container that is used for workflow. It can be designed using trigger words created by the user of the app, such words can include “Power” or “Outage”. The trigger word that was used for the project was a parameter for the subject of the emails from info@sem-o.com, “REMIT Update”. A trigger word can initialise a workflow. An example of a workflow is to pull the trigger words from an email into the logic app, allowing for the user to view. This will enable the user to read the emails from a connected email account and further analyse the data that is required. [5]

Below are some steps on how to create a logic app using Microsoft Azure:

First Log in using your outlook account on Azure. A variety of options are available for the user to choose from, such as SQL databases, app services, and logic apps. All the options just mentioned will be used to finalise the project build. The logic app displayed in the red square should be selected.

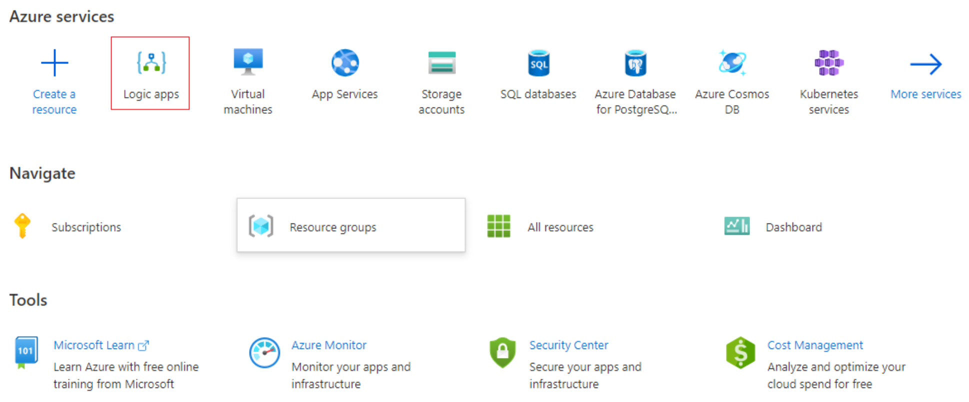


Figure 4: Choosing the logic app.

Once the user clicks the logic app, they are then capable of creating a logic app. They should then choose the option “Create logic app”. This is shown in a blue button in the middle of the screen.

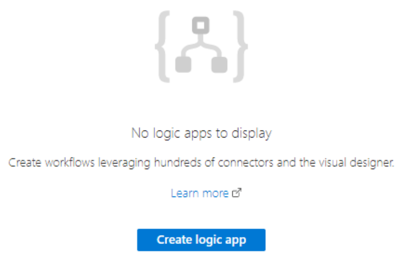
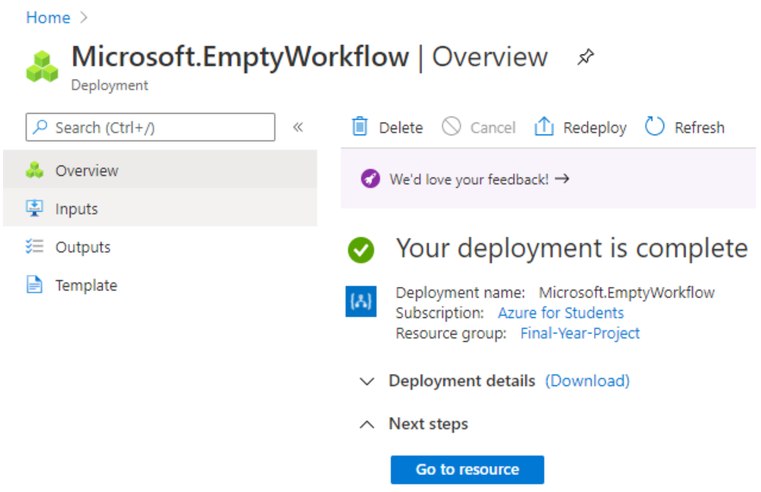

No logic apps to display
Create workflows leveraging hundreds of connectors and the visual designer.
Learn more
Create logic app

Figure 5: Creating a simple logic app.

The user will then choose the options for the logic app. Such as what subscription to use, logic app name, resource group and time zone/region. Below is how to go to the resource within the logic app. This is done by selecting the blue button labelled "Go to resource".


Home >

Microsoft.EmptyWorkflow | Overview

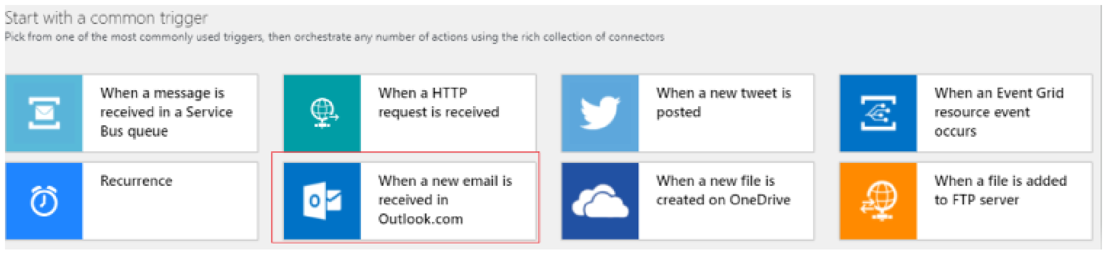
Deployment

Your deployment is complete

Deployment name: Microsoft.EmptyWorkflow
Subscription: Azure for Students
Resource group: Final-Year-Project

Figure 6: Going to resource for the logic app.

The next step involves pulling an email from Outlook. Select the red squared box to access the emails from outlook.

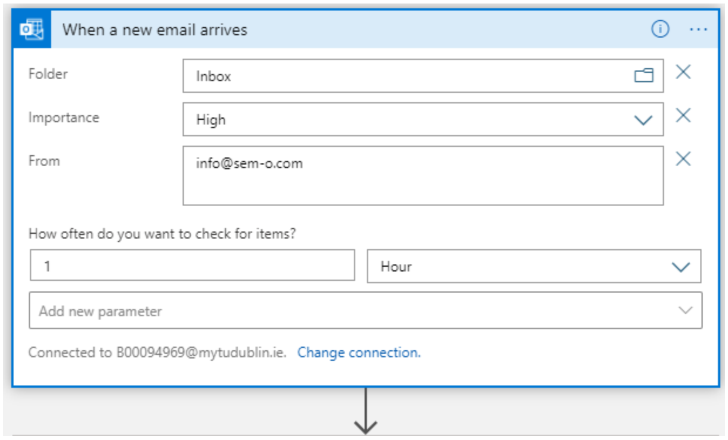

Start with a common trigger

Pick from one of the most commonly used triggers, then orchestrate any number of actions using the rich collection of connectors

When a message is received in a Service Bus queue
When a HTTP request is received
When a new tweet is posted
When an Event Grid resource event occurs
Recurrence
When a new email is received in Outlook.com
When a new file is created on OneDrive
When a file is added to FTP server

Figure 7: Selecting the outlook option.

The following options were chosen to check for emails from info@sem-o.com. These emails typically contain power outage reports from across the county.


When a new email arrives

Folder: Inbox

Importance: High

From: info@sem-o.com

How often do you want to check for items?

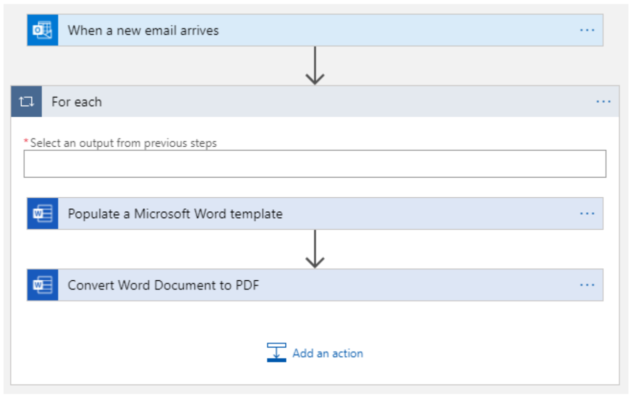
1 Hour

Add new parameter

Connected to B00094969@mytudublin.ie. Change connection.

Figure 8: The chosen parameters for when a new email arrives.

An example of a simple logic app that creates and populates a word document when an email is sent from info@sem-o.com is shown below.


When a new email arrives

For each

* Select an output from previous steps

Populate a Microsoft Word template

Convert Word Document to PDF

Add an action

Figure 9: Finalising a simple logic app.

Design model for the Project.

For a better understanding of the project, a block diagram was created. Initially the student created an outlook account adequately named for the project, projectsemo@outlook.ie, this email account was soon dropped due to the student having more credit available on their student email, with a $200 credit limit. Typically, a block diagram is made for understanding the parameters of the problem. There was different version for a designed block diagram, they were each different to the previously designed model. The first design was put into a diagram that used Entity relationship modelling for a more visual understanding. The design was created using the online diagram maker, Draw.io. The created design can be seen below: [6]

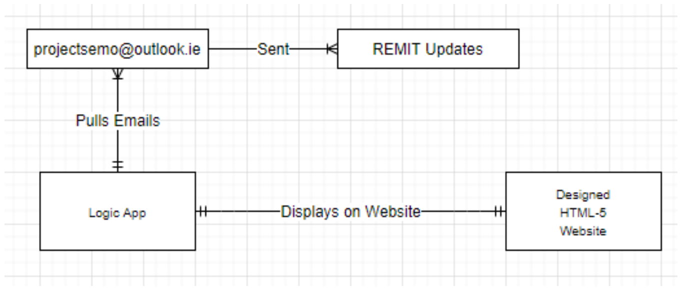


Figure 10: ER model designed block diagram.

The student email, “B00094969@mytudublin.ie”, gets sent REMIT updates stating how long power outages will occur. An example of what the REMIT email looks like can be seen in Figure 13. The email contains information regarding power outages across the country, such as was the outage planned or how many Megawatts of electricity are available.

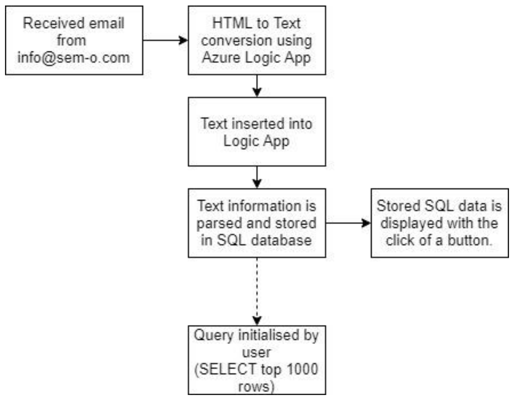

Received email from info@sem-o.com
HTML to Text conversion using Azure Logic App
Text inserted into Logic App
Text information is parsed and stored in SQL database
Stored SQL data is displayed with the click of a button.
Query initialised by user (SELECT top 1000 rows)

Figure 11: Block diagram for project.

These emails are good for understanding the patterns within Ireland's power grid. Such patterns can include areas that need improving with their power input on the power grid. These areas are susceptible to power outages due to the stormy conditions that both Ireland and Northern Ireland are known to have.

Once the emails are sent to "B00094969@mytudublin.ie", they are then pulled into Microsoft's Azure logic app. The created logic app by the user will enable the REMIT information to be parsed and analysed, once the information is contained within the logic app. A user designed HTML website will display the collated data from Azure and then further inspected or viewed by other users of the website.

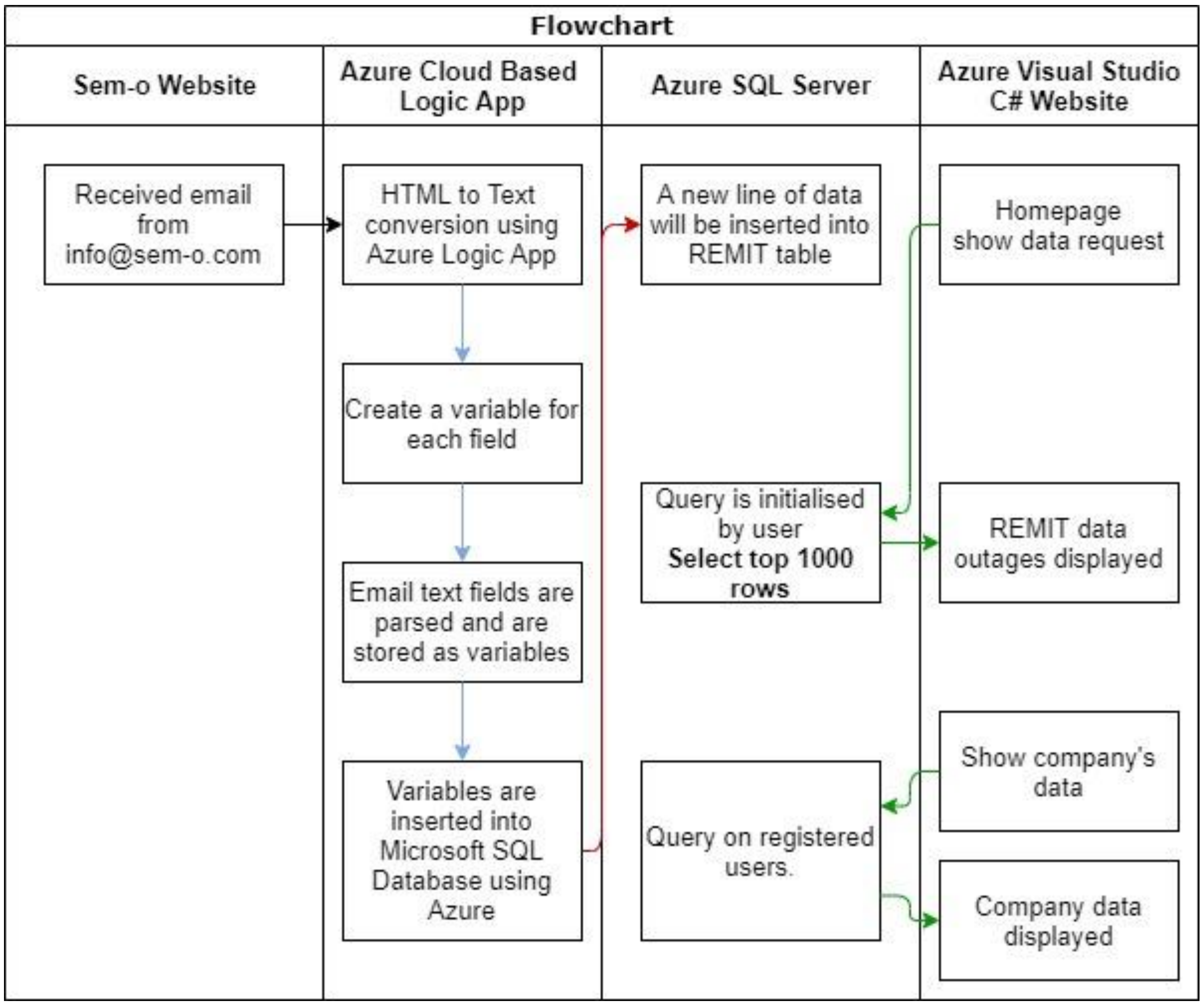
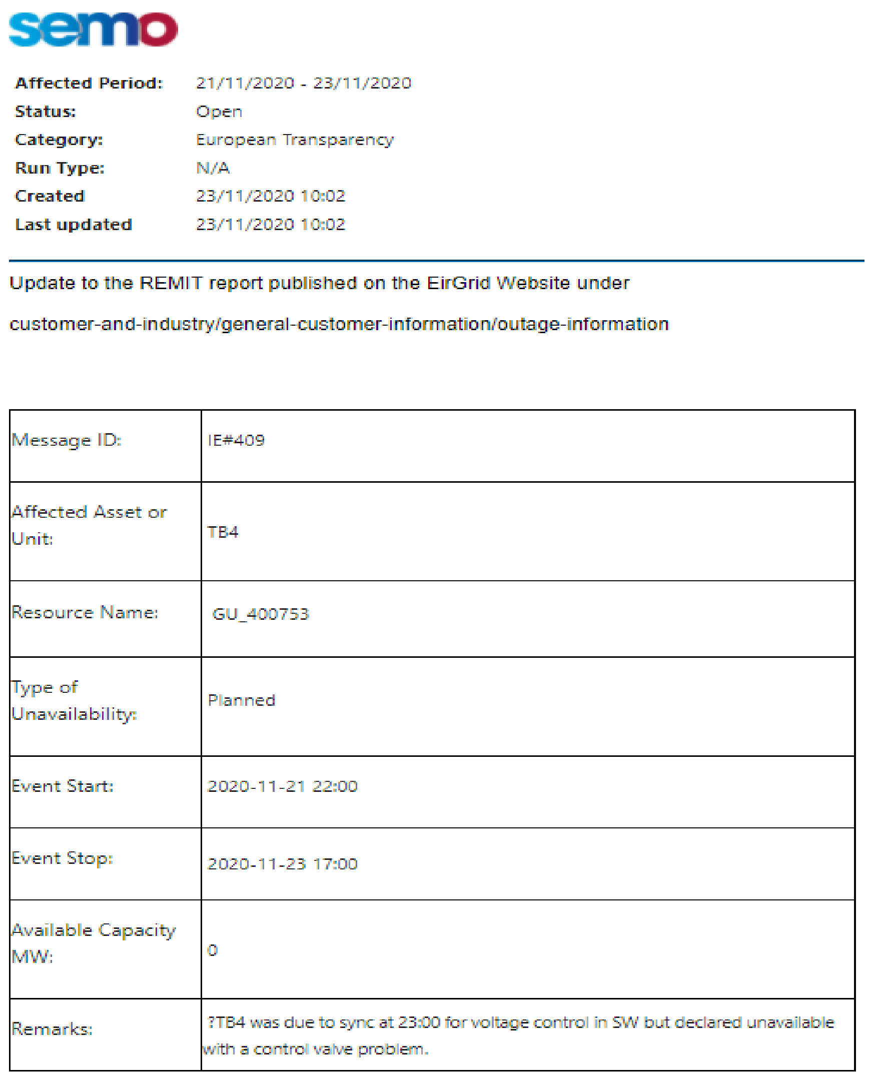

Flowchart
Sem-o Website
Azure Cloud Based Logic App
Azure SQL Server
Azure Visual Studio C# Website
Received email from info@sem-o.com
HTML to Text conversion using Azure Logic App
Create a variable for each field
Email text fields are parsed and are stored as variables
Variables are inserted into Microsoft SQL Database using Azure
A new line of data will be inserted into REMIT table
Query is initialised by user Select top 1000 rows
Query on registered users.
Homepage show data request
REMIT data outages displayed
Show company's data
Company data displayed

Figure 12: Detailed block diagram



Affected Period:	21/11/2020 - 23/11/2020
Status:	Open
Category:	European Transparency
Run Type:	N/A
Created	23/11/2020 10:02
Last updated	23/11/2020 10:02

Update to the REMIT report published on the EirGrid Website under

customer-and-industry/general-customer-information/outage-information

Message ID:	IE#409
Affected Asset or Unit:	TB4
Resource Name:	GU_400753
Type of Unavailability:	Planned
Event Start:	2020-11-21 22:00
Event Stop:	2020-11-23 17:00
Available Capacity MW:	0
Remarks:	?TB4 was due to sync at 23:00 for voltage control in SW but declared unavailable with a control valve problem.

Figure 13: Screen clipping of email from REMIT.

These emails display the types of unavailability, the times the incident occurred, and how much available MW capacity is available. These emails are sent to subscribers of SEMO.

Europe's Electricity Market vs. Ireland's Electricity Market.

27 interconnected countries changing the face of the wholesale electricity market is unprecedented. Formerly 28, now with the inclusion of Brexit. Countries such as Ireland, France, Germany, and Netherlands make up some of the European Union Countries.

Germany, a European Union Giant, relies heavily on importing fossil fuels, such as coal, gas and oil with a net spend of 93.5 billion, Figures from 2012. [7]

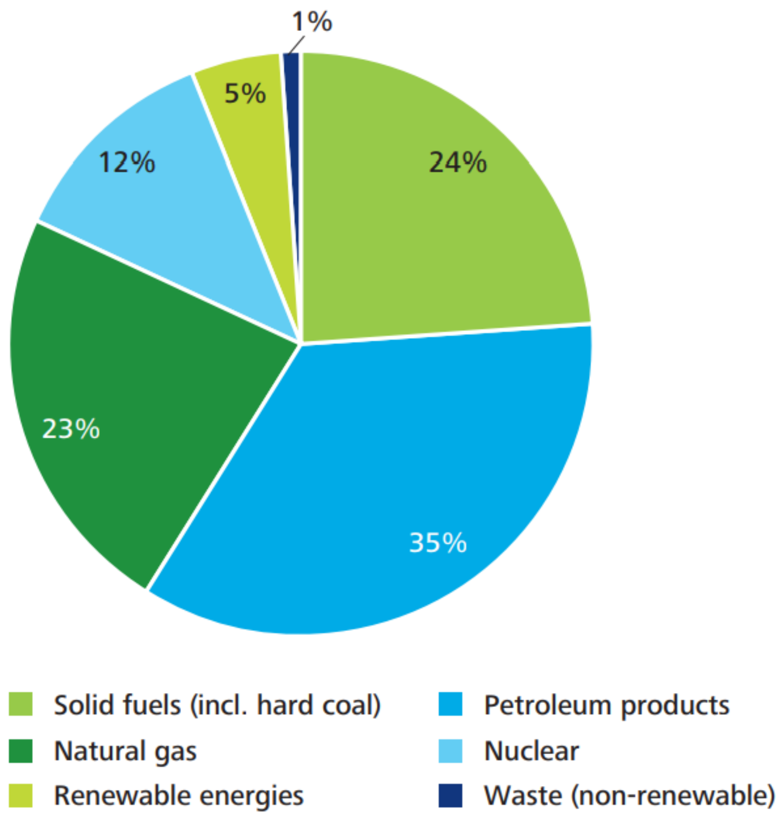


Figure 14: Energy consumption of Germany circa 2012. **[8]**

Germany has adopted an aggressive approach for energy transition known as the Energiewende. This approach is making Germany more environmentally friendly, reliable, and affordable. The new system that Germany have adopted relies heavily on renewable energy, particularly wind, hydro and solar. Germany intends to reduce greenhouse gasses by 80-95% by 2050 . Germany are a large producer of electricity and export to other European countries around them.

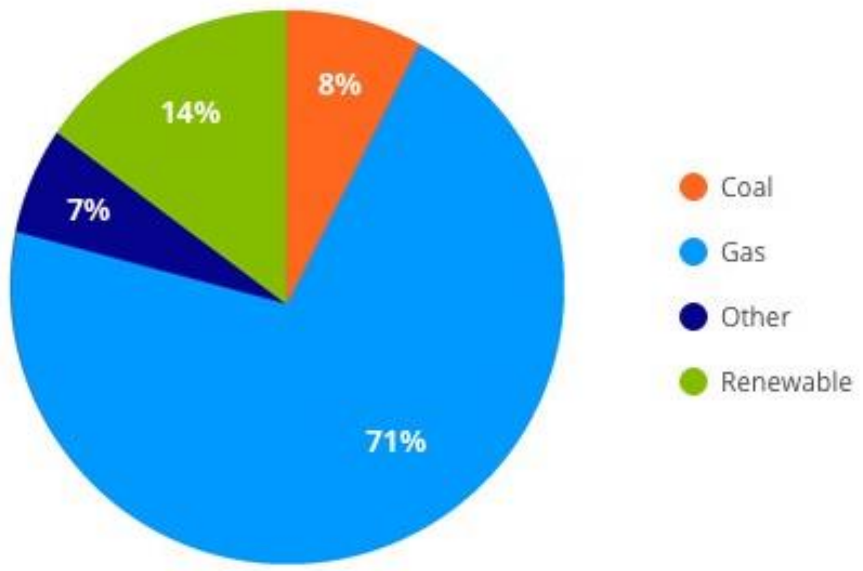


Figure 15: Ireland’s energy consumption circa 2018. **[9]**

It is evident from the comparison of Ireland and Germany, that Ireland has a more sustainable hold on renewable electricity. However, Ireland has a population of near 5 million people, Germany has a population of roughly 83 million. Germany in theory is 16.6 times larger in population. This means that Germany would require a large amount of space to build wind, solar, and hydro farms sustainable enough to provide enough electricity for the entire nation. [10]

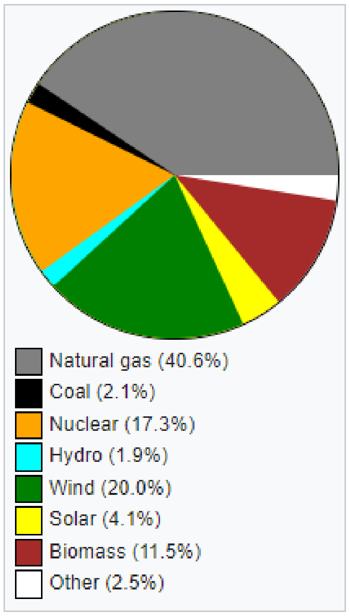


Figure 16: Britain's energy consumption circa 2019 **[11]**.

The United Kingdom (Great Britain) also has a national electricity grid like Ireland, this national grid covers most of the mainland within Great Britain, and the surrounding islands of Britain. Britain is beginning to display strong growth with renewable energy, compared to Ireland/Northern Ireland. The UK's use of fossil fuels is starting to plunder. Coal generators are primarily used within Winter with the aim of stopping pollution.

Like Ireland, The UK have a wholesale market where suppliers can purchase electricity from electric generators. In Britain, the average market price for MW/h ranges from £40 to £50. It is also within Britain's interests to reduce the cost of wholesale electricity to the public.

What is SEMO?

SEMO is a joint venture between Ireland's EirGrid and Northern Ireland's SONI. Both EirGrid and SONI share the same values when it comes to providing electricity to the public. They both want to offer sustainable, clean, and cheaper energy to the members of the public. Both EirGrid and SONI embarked on a joint venture known as SEMO. SEMO stands for Single Electricity Market Operator. SEMO created a new project known as the Integrated Single Electricity Market, simply known as I-SEM. The new all-island market was designed by and is currently overseen by the SEM committee. The SEM committee consists of the Commission for Energy Regulation, the Utility Regulator, and independent members along with a deputy independent member.

Originally, SEM was established in November of 2007. SEM has since been replaced by I-SEM. When SEM was first founded, it was known for being the first market of its kind. It is said that free trade across borders is the foundation upon the single European market. As of October 1st, 2018, I-SEM officially replaced SEM. It is anticipated that I-SEM will bring many benefits:

- Access to cheaper and more affordable sources of electricity.
- A more open and more efficient European electricity market that can deliver benefits to its consumers.
- A basis for development of new market types. That can manage the risk for investment. [12]

What is SEMOpx?

SEMOpx can provide day-ahead and intraday electricity market trading for both Ireland and Northern Ireland. SEMOpx is a part of SEM. SEMOpx can assist and offer guidance to its participants in achieving their trading goals. The want to provide knowledge with some cost-effective services, this is done using their experience in trading and the experience of their trading partners. [13]

There are three markets available to the all-island participants:

- Day-Ahead Market.
- Intraday Auctions Market.
    - Intraday Auctions 1 & 2.
    - Intraday Auctions 3.
- Intraday Continuous Market.

Each one of the markets above are all different from one another. Each market will be broken down into what time it opens and closes at, what order types they use, and what currencies are available to buy or sell with.

Day-Ahead Market:

The Day-Ahead Market takes place every calendar day. The auctions begin at 11:00 A.M. Within this auction, members of SEMOpx will trade electricity in 24 one-hour Trading periods for the timeframe 23:00 P.M until 23:00 P.M the following day. The Market remains open for 19 days for the given day that is selected. The currencies used in the Day-Ahead market are GBP and Euro. There are two order types available to traders, Simple orders, and Complex orders. A Simple order is trading for one individual hourly period. A Complex order is for one or more trading periods in the selected day. [14]

Intraday Auctions Market:

There are three daily auctions within the Intraday Auctions Market. Intraday Auctions 1 & 2 are coupled with the Great Britain bidding area, whereas Intraday Auctions 3 is not coupled with Great Britain’s bidding area. There are three different auctions start times for 1,2, and 3, however, each finish at the same time.

Table 3: Auction Timelines.

Auction	Auction Start Time	Trading Period	Auction Duration
IDA-1	17:30	23:00-23:00	24 Hours
IDA-2	08:00	11:00-23:00	12 Hours
IDA-3	14:00	17:00-23:00	6 Hours

The Intraday Auctions Market is a dual currency market, both GBP and Euro are used. Both simple and complex trading orders are available for the Intraday Auctions Market. [15]

Intraday Continuous Market:

This market operates every day of the calendar year, 365 days a year. Trading on this market takes place in a trading day basis, 23:00-23:00. It consists of 48 half-hours slots available. The Intraday Continuous Market only operates with Euro. Multiple order types are available for this market.

There are both simple and block order types available, however, the block order types are capable of being purchased in multiple formats:

- 24-hour blocks, 1 block available.
- 8-hour blocks, 3 blocks available.
- 4-hour blocks, 6 blocks available.
- 2-hour blocks, 12 blocks available.

Each of the blocks that are available for purchase in their respective hour blocks, all have different period times, the 24-hour block covers 23:00-23:00, whereas a 2-hour block will only cover 23:00-01:00. [16]

Chapter 3 - Implementation

The aim of this chapter is to provide a structured presentation in detail of how the student completed the project and how it was implemented. This chapter is to describe the tools and techniques that enabled the student to complete the project implementation.

The student was to create a website that displayed collated/parsed electricity data that was analysed. The student collated and monitored data for months with the use of their personalised college email. Emails were received from “info@sem-o.com”. These emails were specifically sent to SEMO subscribers and contained information on multiple things such as the asset that was affected and even the available Megawatts capacity. The student designed three major components for the success of the project, and then implemented each to work with one another. The student created a Logic app, SQL database, and HTML Web App service, all using Microsoft Azure. This was due to the project proposal stating, “The student would poll live xml or csv updates, collated, analyse on a cloud database and then display on a hosted web page or application”.

Considerations for the project.

A lot of consideration over which cloud-based platform would be best for the student was done at the beginning of the project. This was due to whichever choice made at the beginning of the project would lead the student into a different direction each time.

There was a strict deadline given to the student, this in theory meant that the student was to choose the perfect choice for them to enable project completion within the project timeframe. This was done by specifically choosing which cloud-based service was appropriate for the student.

There were three choices to choose from. However, there was only one clear winner, Microsoft Azure. Microsoft Azure was chosen over Google Cloud and Amazon Web Services due to the student receiving Credit for their account and the possibility of combining resources to work with one another.

Software.

There were several software suites available to the student to download for writing code for this project. The choices included can be seen below: [17] [18] [19]

- Microsoft Visual Studio Community Edition 2017.
- Microsoft Visual Studio Community Edition 2019.
- Microsoft SQL Server Management Studio 18.
- Visual Studio Code.

Both versions of Microsoft Visual Studio Community Edition were downloaded. However, the student found the earlier release, 2017 edition, to be more useful. This was due to the toolbox feature when designing the HTML web app. The toolbox feature displays specific controls that the user can add to their visual studio projects, this was found to be quite useful for renaming labels or even divisions within the HTML code. The toolbox window within the 2017 community edition for Visual Studio can be accessed two ways:

View > Toolbox. This will open the toolbox by selecting the View tab and then selecting the toolbox.

Ctrl + Alt + x. This windows command will allow for the user to access the toolbox without having to use a computer mouse.

The toolbox will allow for the user to not only rename tags, divisions, labels, etc. But allow for the user to also click and drag the previously mentioned HTML elements. The toolbox is only available for viewing when in the current designer window. The toolbox will disappear when the designer view has been closed.

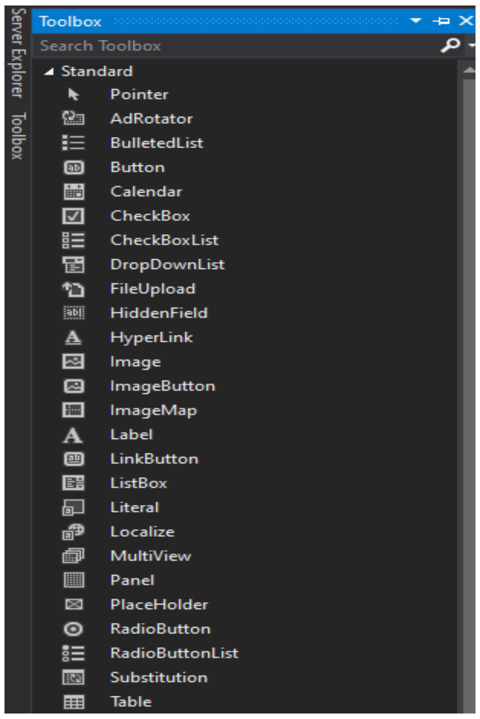


Figure 17: Toolbox in Visual Studio

Extra content can be added to the user's Visual Studio account. This includes both technologies/languages and workloads.

Workloads:

- Azure
- Web
- Mobile
- Windows

Technologies/languages:

- .Net
- Python
- Node.js
- C++

The azure feature within Visual Studio allows for the user to write and develop code locally without access to the internet. This project can then be saved and deployed to the Azure cloud by publishing the users project.

Projects in the cloud are scalable, this means that users can scale their projects depending on performance, demand, etc. It can also be used for cutting down costs of their projects, this step will be shown below.

When in Azure online, the user will see the drop-down menu from the left side of the screen. They will select the configure option to scale the size of the SQL database.

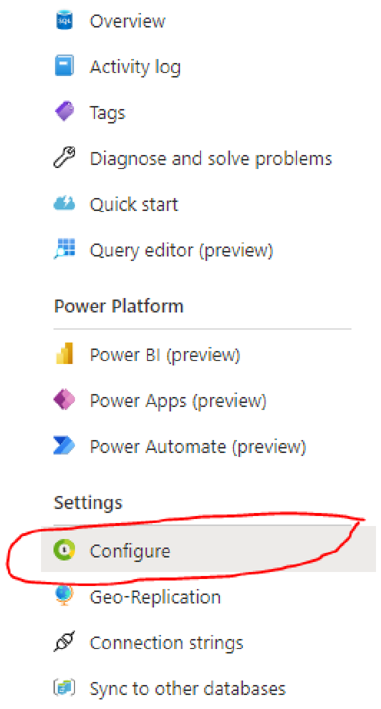
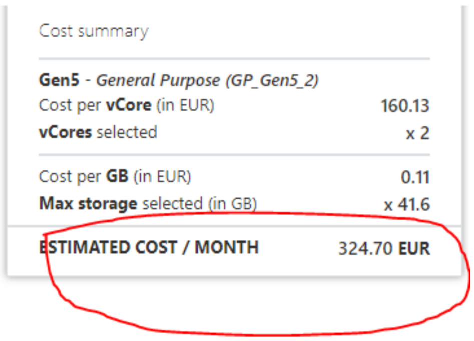


Figure 18: Scale the database.

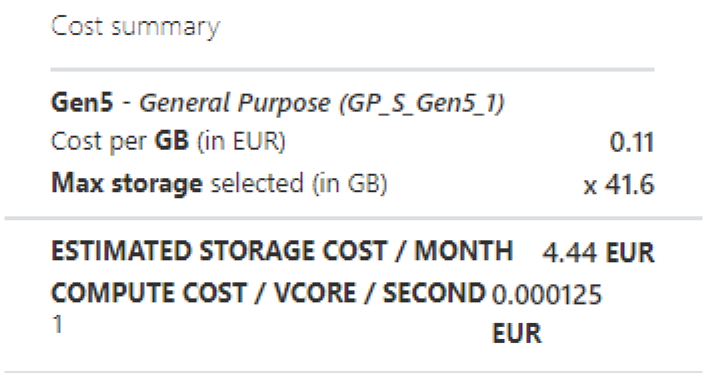
The user will choose the serverless version of the SQL database. This will save the user an estimated 320 Euro worth of Credit. If the provisioned option is chosen, the estimated amount is very high as seen below.



Cost summary

Gen5 - *General Purpose (GP_Gen5_2)*	
Cost per **vCore** (in EUR)	160.13
vCores selected	x 2
Cost per **GB** (in EUR)	0.11
Max storage selected (in GB)	x 41.6
ESTIMATED COST / MONTH	324.70 **EUR**

Figure 19: Provisioned server cost (Monthly).



Cost summary

Gen5 - *General Purpose (GP_S_Gen5_1)*

Cost per **GB** (in EUR)	0.11
Max storage selected (in GB)	x 41.6
ESTIMATED STORAGE COST / MONTH	4.44 EUR
COMPUTE COST / VCORE / SECOND 1	0.000125 **EUR**

Figure 20: Serverless database cost (Monthly).

When the serverless database is chosen, this allows for the student's choice to not only be most cost effective, but it will allow for the student to use their Azure credit in other applications, like the Web App and Logic App.

The student's Web App is also scalable, this is mainly done depending on the demand for the website. A small-scale website was used for the project, it displayed the data from SEMO, the list of registered units, and gave a brief insight into the project.

The Scale up feature is used for increasing the size of the Web App. The student will first choose the scale up feature under settings, then select the Dev/Test tab. When this is done, the F1 option is chosen over D1 and B1 as it is free.

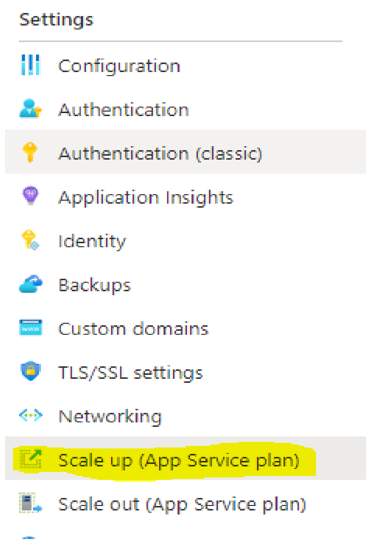
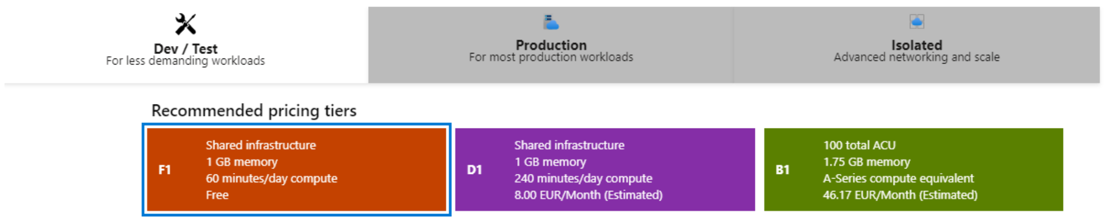


Figure 21: Scale up feature under settings tab.

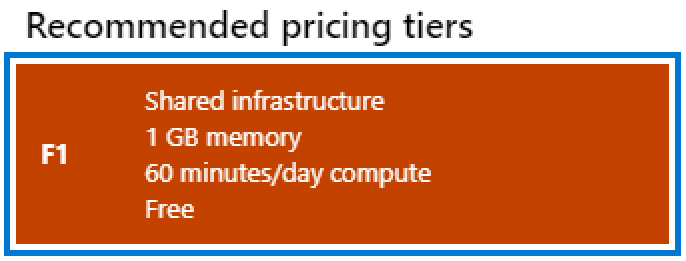

Dev / Test
For less demanding workloads
Production
For most production workloads
Isolated
Advanced networking and scale
Recommended pricing tiers

F1 — Shared infrastructure, 1 GB memory, 60 minutes/day compute, Free

D1 — Shared infrastructure, 1 GB memory, 240 minutes/day compute, 8.00 EUR/Month (Estimated)

B1 — 100 total ACU, 1.75 GB memory, A-Series compute equivalent, 46.17 EUR/Month (Estimated)

Figure 22: Selecting the F1 option.



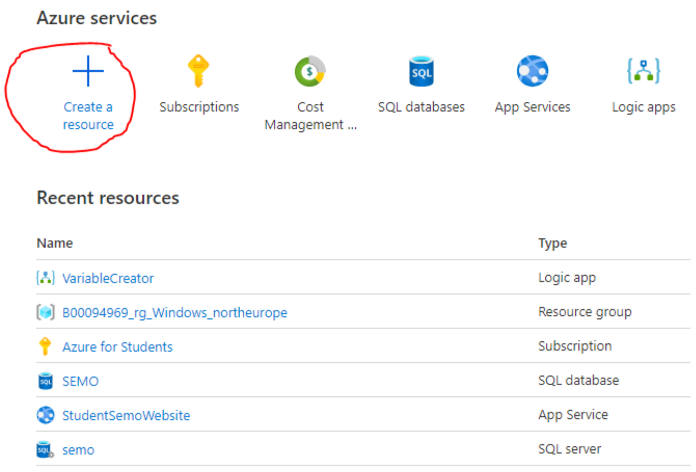
F1 — Shared infrastructure, 1 GB memory, 60 minutes/day compute, Free

Figure 23: F1 option is free to use.

If the production or isolated tab were chosen, the cost of the Web App would be driven much higher, the basic S1 option is over 60 Euro a month.

SQL Database

Creating an SQL database in Microsoft Azure was done using the following steps. The user was to log into their Microsoft account. The user will then select the create a resource option. This will then display a wide variety of choices to the user, it will include both popular choices among Azure subscribers and the entire Azure marketplace. [20]



Name	Type
VariableCreator	Logic app
B00094969_rg_Windows_northeurope	Resource group
Azure for Students	Subscription
SEMO	SQL database
StudentSemoWebsite	App Service
semo	SQL server

Figure 24: Create a resource option.

The user will select the SQL Database option available to them. If the Azure user is unfamiliar with creating databases or new resources, there is a tutorial available under the selection option.

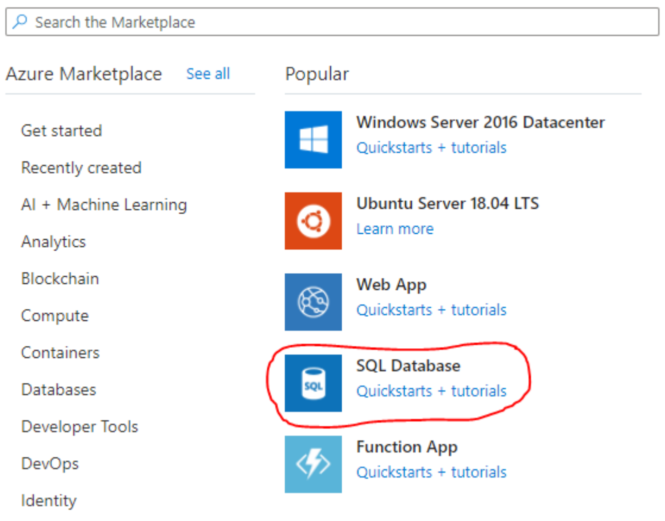
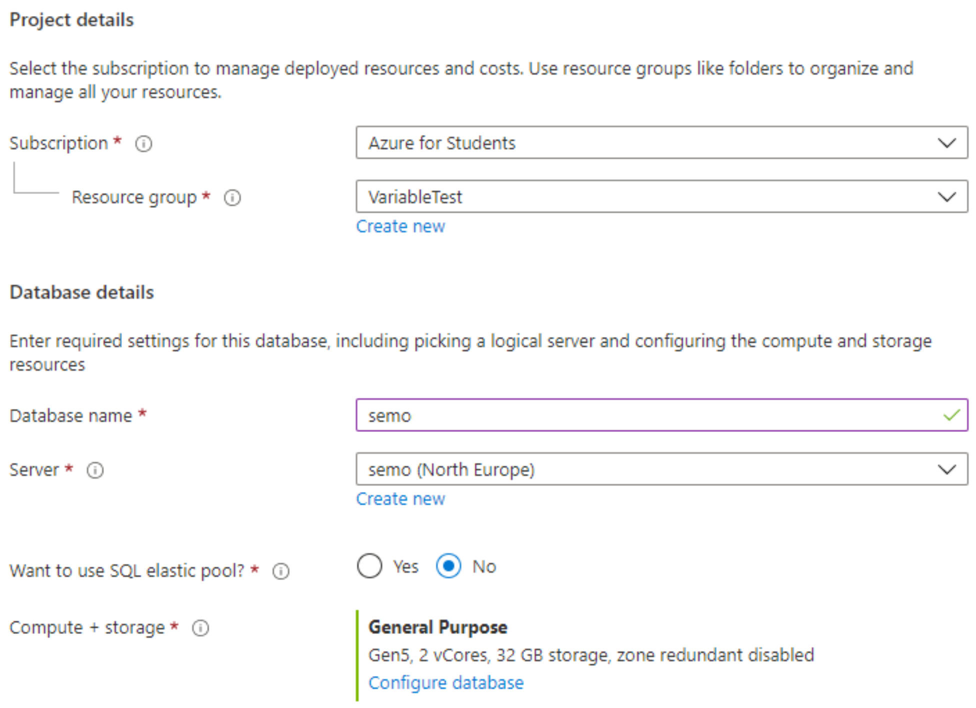


Figure 25: Selecting the SQL database option.

Once the SQL Database is selected, the user will be brought to a new page and given more options to choose from. The user will first fill out options within the basics tab of the "**Create SQL Database**" Webpage. Such options will include which subscription and resource group to use. If no resource group was previously created, the user can create a new one. The user will have to enter database details such as the database name and which server it will use. Once again, if a server has not been created or available to the user, they can simply create a new one under the create new button.

The student chose the following options for their semo database.



Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *	Azure for Students
Resource group *	VariableTest Create new

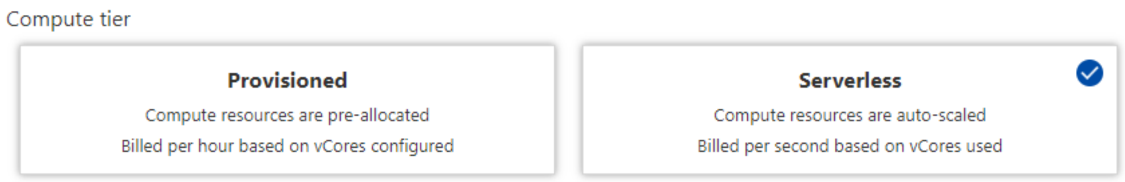
Database details

Enter required settings for this database, including picking a logical server and configuring the compute and storage resources

Database name *	semo
Server *	semo (North Europe) Create new
Want to use SQL elastic pool? *	Yes [✓ No]
Compute + storage *	**General Purpose** Gen5, 2 vCores, 32 GB storage, zone redundant disabled Configure database

Figure 26: Creating the SQL database.

The blue "Review + create" button was then selected at the bottom of the page to create and save the newly created "semo" database. The next step in the SQL database creation is to change and configure the database running costs. This was shown in Figure 18, Figure 19, and Figure 20. The serverless option should be chosen to maintain a low-cost running database. This will ensure the student can use their Azure credit on other resources.



Compute tier

Provisioned
Compute resources are pre-allocated
Billed per hour based on vCores configured

Serverless ✓
Compute resources are auto-scaled
Billed per second based on vCores used

Figure 27: Serverless option.

A server firewall is added to the database when it is created. To access the query editor within the server's database, the student was to Select the "Add client IP" option. The On button was to be selected for allowing Azure services to have remote access to the student created server. The student allowed for specific IP addresses to access their database. This included the student's home PC IP address, personal laptop IP address, and mobile phone IP address. When the IP address for all the above devices were added to the Firewall settings, the student could access the database, and create tables to store the parsed REMIT data. Multiple tables were created and tested, however, only one table had success with the "Insert Row (V2)" SQL database connection shown in Figure 54.

Within the home page of the database on Azure, the student can access a 7-day graph displaying the usage of the database. Which is useful for checking if items got stored within the database. The graph displays CPU storage that was used at the time a string was added to the Remit2 table within the SQL database.

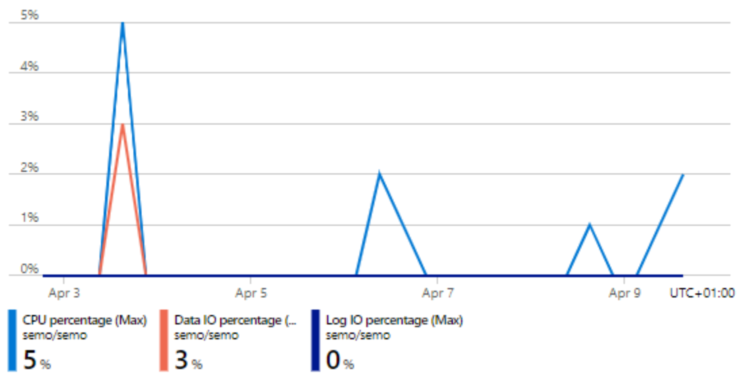


Figure 28: CPU percentage for SQL database.

Query Editor in SQL Database

To login to the query editor using sql, a login and password is required:

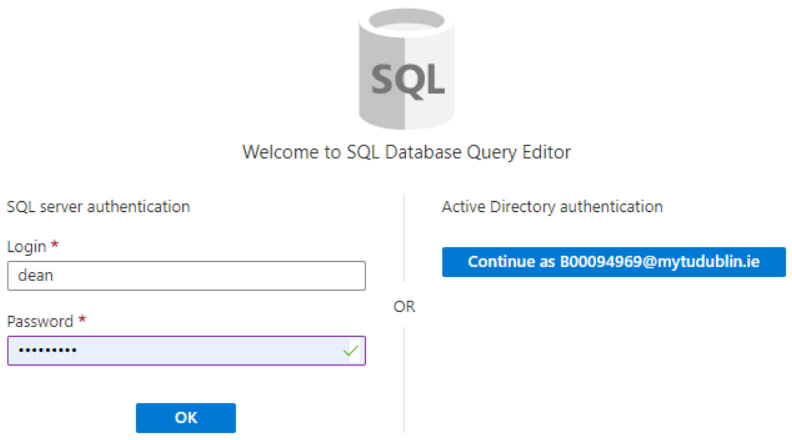
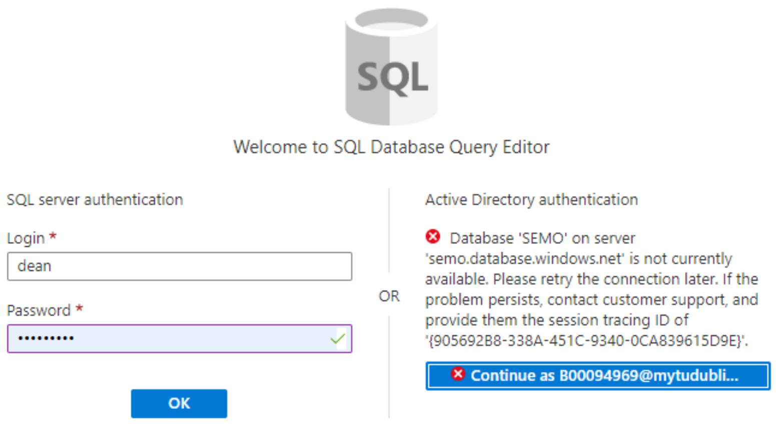

SQL
Welcome to SQL Database Query Editor
SQL server authentication
Login *
dean
Password *
•••••••••
OK
OR
Active Directory authentication
Continue as B00094969@mytudublin.ie

Figure 29: SQL database login using query editor.

Once the correct password is inputted, a green tick will appear, and the user will then be able to access the database by then pressing the blue box containing the word “OK”.

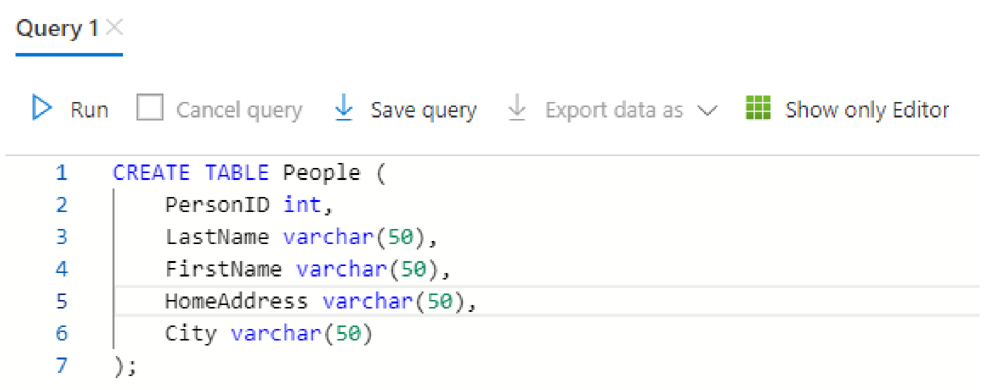

SQL
Welcome to SQL Database Query Editor
SQL server authentication
Login *
dean
Password *
•••••••••
OK
OR
Active Directory authentication
Continue as B00094969@mytudubli...

Database 'SEMO' on server 'semo.database.windows.net' is not currently available. Please retry the connection later. If the problem persists, contact customer support, and provide them the session tracing ID of '{905692B8-338A-451C-9340-0CA839615D9E}'.

Figure 30: Error when using email credentials.

Once the user has signed into the query editor on Azure they will be greeted with a new page. This page will enable the user to write new queries into the database. Such queries can include creating a new table within the database. A table is created using the create statement in SQL, the user will create and name the table (table name cannot already exist), and then create columns for the table.

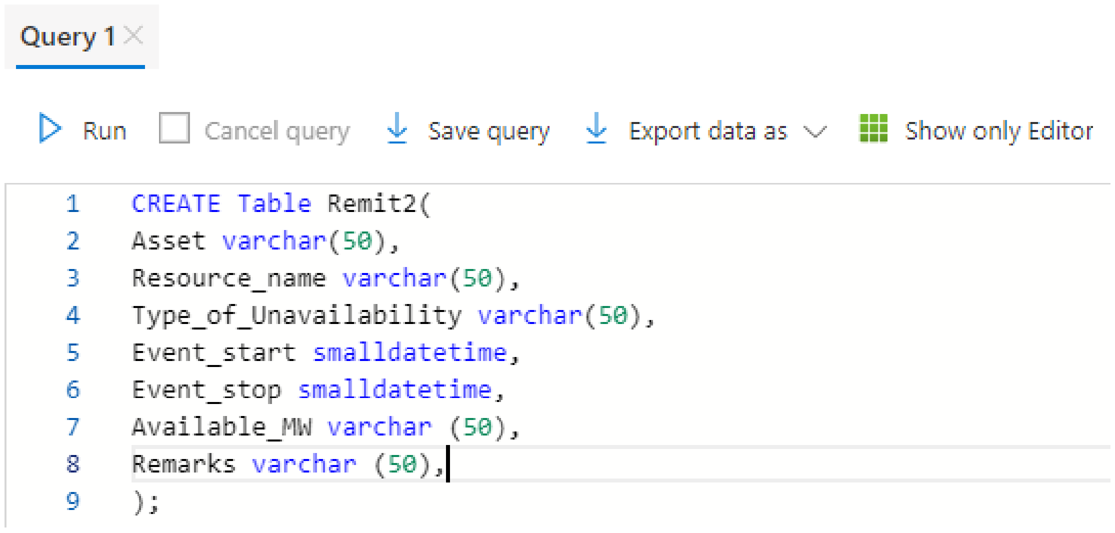
A generic example would be to create a table known as “People” and fill the columns with PersonalID, LastName, FirstName, HomeAddress, and City. Each one of these columns will be of a different variable type.



```sql
CREATE TABLE People (
    PersonID int,
    LastName varchar(50),
    FirstName varchar(50),
    HomeAddress varchar(50),
    City varchar(50)
);
```

Figure 31: Generic table creation in SQL.

This table will then be stored in the “tables” drop down menu in the Query Editor. The table that was created for the purpose of this project was adequately named Remit2. The table was created using the following statements:



```sql
CREATE Table Remit2(
Asset varchar(50),
Resource_name varchar(50),
Type_of_Unavailability varchar(50),
Event_start smalldatetime,
Event_stop smalldatetime,
Available_MW varchar (50),
Remarks varchar (50),
);
```

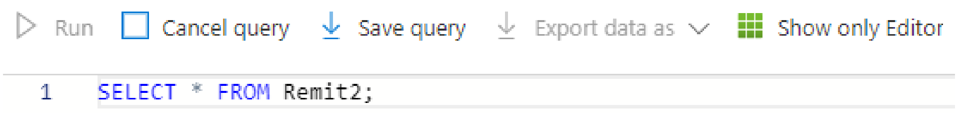
Figure 32: Remit2 table being created in SQL.

The following columns were chosen as variable characters with a maximum amount of 50 characters:

- Asset (Asset)
- Resource Name (Resource_name)
- Type of Unavailability (Type_of_Unavailability)
- Available Capacity MW (Available_MW)
- Remarks (Remarks)

Both Event start and Event stop times were created using smalldatetime element. The smalldatetime element is in an American dated format. This would display the date and time that the occurrence occurred. If the start and stop events were trying to be stored as variable characters they would not store within the table.

To select the top 1000 rows of the table, it can be done two ways. Once with a SQL select statement.



```
SELECT * FROM Remit2;
```

Figure 33: SQL Select statement.

The other way of selecting the top 1000 rows for the table is to click the three dots on the table you want to select, then choose “Select Top 1000 Rows”.

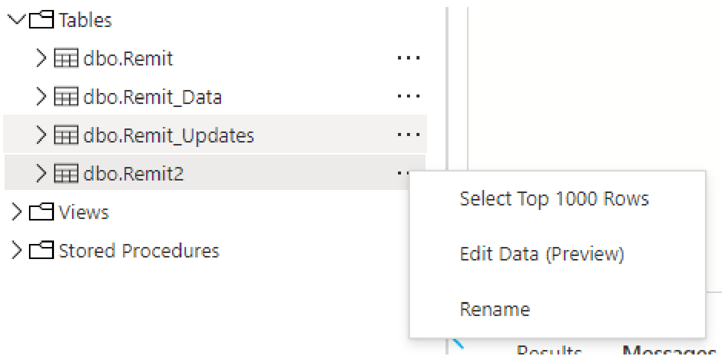
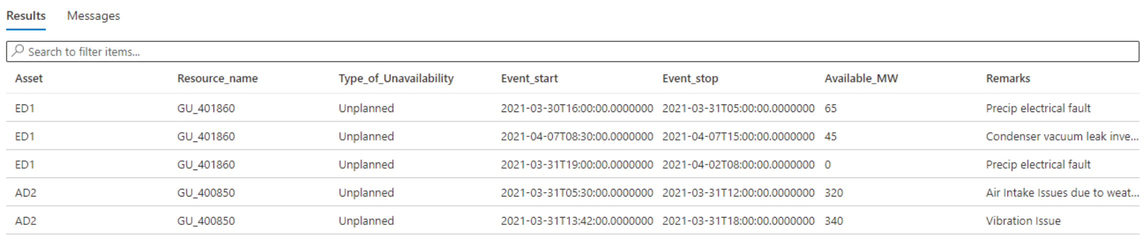


Figure 34: Selecting top 1000 rows of Remit2.

Once the top 1000 rows are selected that can be viewed in the results tab shown underneath the Query editor box.

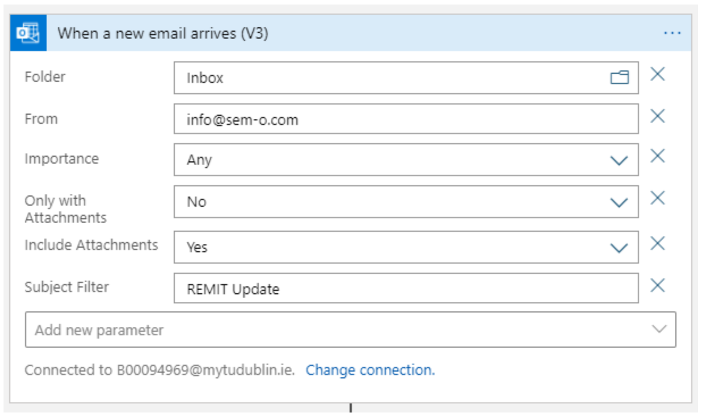


Asset	Resource_name	Type_of_Unavailability	Event_start	Event_stop	Available_MW	Remarks
ED1	GU_401860	Unplanned	2021-03-30T16:00:00.0000000	2021-03-31T05:00:00.0000000	65	Precip electrical fault
ED1	GU_401860	Unplanned	2021-04-07T08:30:00.0000000	2021-04-07T15:00:00.0000000	45	Condenser vacuum leak inve...
ED1	GU_401860	Unplanned	2021-03-31T19:00:00.0000000	2021-04-02T08:00:00.0000000	0	Precip electrical fault
AD2	GU_400850	Unplanned	2021-03-31T05:30:00.0000000	2021-03-31T12:00:00.0000000	320	Air Intake Issues due to weat...
AD2	GU_400850	Unplanned	2021-03-31T13:42:00.0000000	2021-03-31T18:00:00.0000000	340	Vibration Issue

Figure 35: Returned results for table "Remit2".

Logic App

The logic app for the project was built using Azure. A logic app is deployed and ran using logic applications within Azure. A logic app enables the student to create an automated workflow with integrated apps, data, and services. The logic simplifies how the student will design/code solutions for their app integration. logic apps are used for a variety of things within software-based programs, such as processing orders, moving/uploading files when an instance has occurred. The student designed a logic app that converts HTML to Text when an email has arrived from info@sem-o.com. Variables are then created before being stored within an SQL database known as semo. This database contains a table, named Remit2. When variables are created, they are then stored within the SQL database and can be viewed, by selecting the Top 1000 rows feature. Below is a series of images, displaying how the logic app for the project was created.

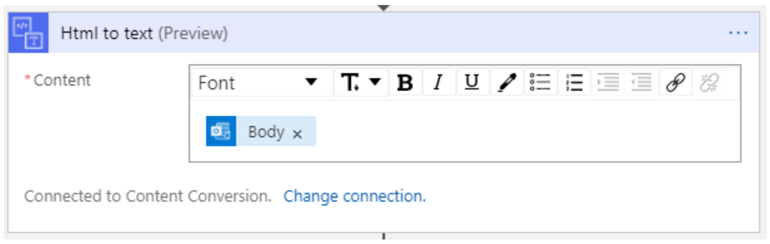


Folder	Inbox
From	info@sem-o.com
Importance	Any
Only with Attachments	No
Include Attachments	Yes
Subject Filter	REMIT Update

Connected to B00094969@mytudublin.ie.

Figure 36: When an email arrives with specific parameters.

Above is the first step/instance within the logic app. This creates the logic app. When an email from info@sem-o.com is received containing the subject REMIT Update, the email is pulled into the logic app and enables the app to begin running.

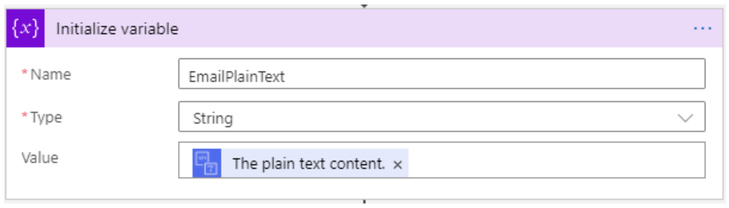

Html to text (Preview)

*Content: Body

Connected to Content Conversion. Change connection.

Figure 37: HTML to text.

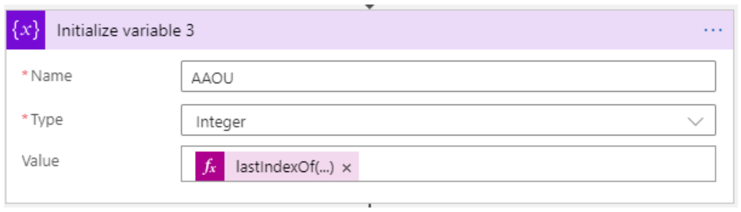
The above image displays what happens when converting the HTML to text. The body of the email, which typically contains the contents of the REMIT Update occurrence will be converted from standard HTML to plain text. This is done to create variables in the upcoming steps.


Initialize variable

*Name	EmailPlainText
*Type	String
Value	The plain text content.

Figure 38: EmailPlainText variable

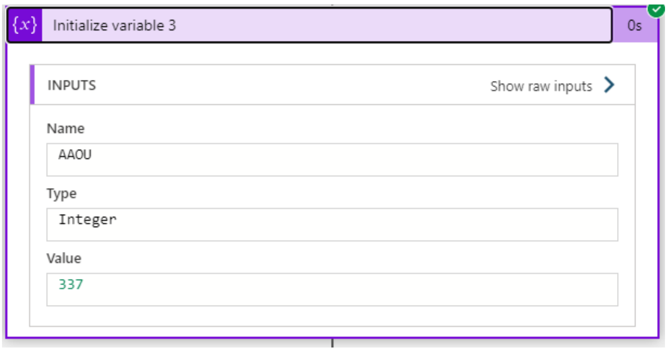
The first variable that is created is known as EmailPlainText. This variable contains the entire body of the REMIT Update email in text format rather than HTML format. This allows for the future variables to be created by pulling strings from the created integer variables.


Initialize variable 3

*Name	AAOU
*Type	Integer
Value	lastIndexOf(...)

Figure 39: Affected Asset or Unit variable

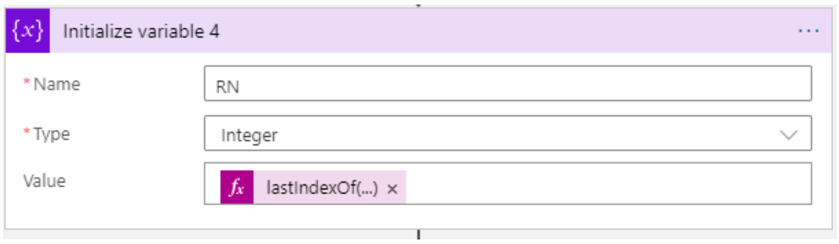
From Variables 3 to 9, The type of variable that was created was of type integer. This would assign a real number to the variable. This means that the variable can be equal to both positive and negative integers. This can be seen in a successful logic app run.



Name: AAOU
Type: Integer
Value: 337

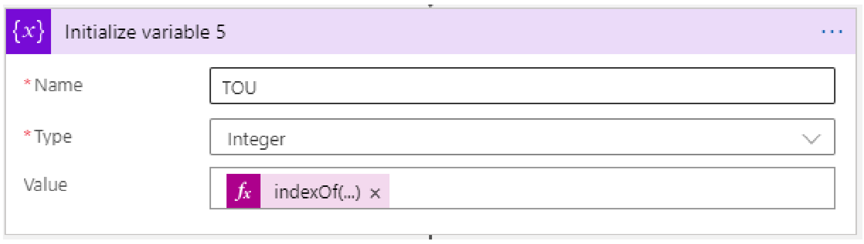
Figure 40: Successful logic app run.

Specific code was required for each variable creation. An example of the code can be seen here: *lastIndexOf(variables('EmailPlainText'),'Affected Asset or Unit:')*. This line of code was used for creating a variable where the words, "Affected Asset or Unit" are shown. This was done for all key components which were stored in the SQL database.



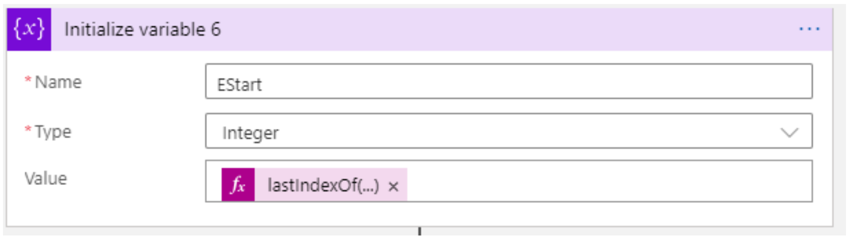
*Name: RN
*Type: Integer
Value: lastIndexOf(...)

Figure 41: Resource name variable.



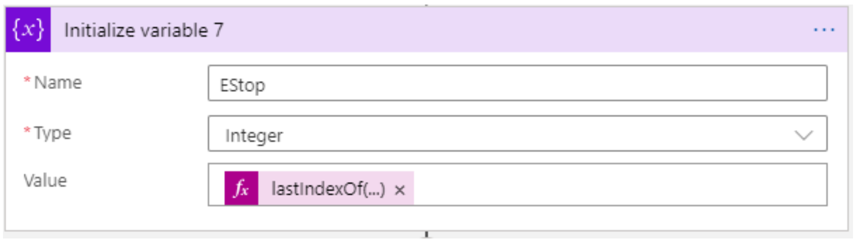
*Name: TOU
*Type: Integer
Value: indexOf(...)

Figure 42: Type of unit variable.



*Name: EStart
*Type: Integer
Value: lastIndexOf(...)

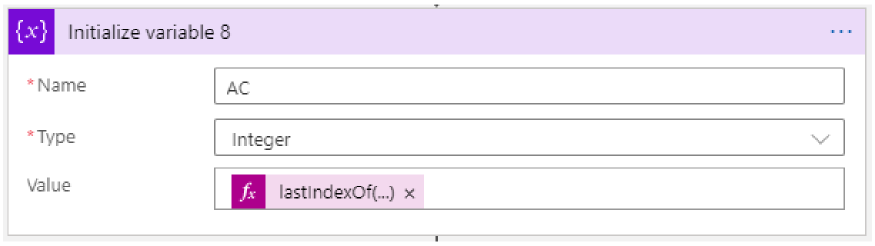
Figure 43: Event start time variable.



Initialize variable 7

*Name	EStop
*Type	Integer
Value	lastIndexOf(...)

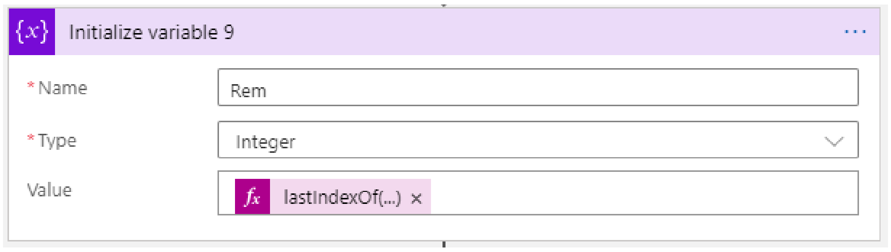
Figure 44: Event stop time variable.



Initialize variable 8

*Name	AC
*Type	Integer
Value	lastIndexOf(...)

Figure 45: Available capacity megawattage.

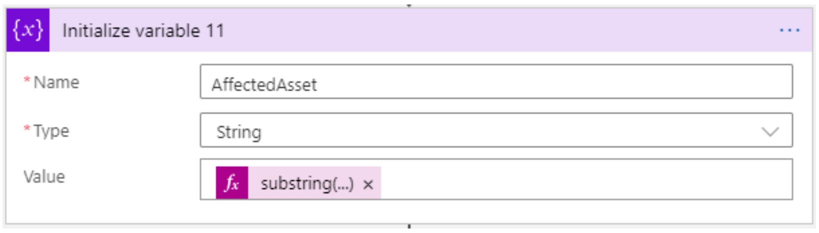


Initialize variable 9

*Name	Rem
*Type	Integer
Value	lastIndexOf(...)

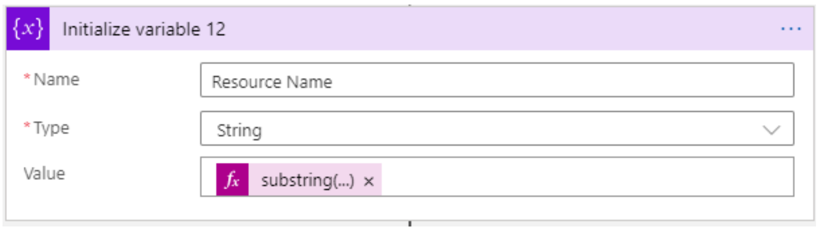
Figure 46: Remarks variable.

All the above images from Figure 39 to Figure 46, display the variables as type integer. This is basically done for character counting. As shown in Figure 40, the variable AAOU is equal to an integer of value 337, this is done for calculating the character difference between AAOU and the variable RN, for the successful Logic run there was a difference of 28 characters between them. For the following steps, integers were not used and were replaced with strings. The purpose of the variables of type string was to display the string within the created SQL database. The variables of type string contained different variable names to the previously created integer variables, this was to stop errors within the code. The string variables contained similar names to the integer variables to avoid confusion, AAOU was adjacent to AffectedAsset, RN was like Resource Name, etc. The variable names were chosen to match their counterparts.


Initialize variable 11

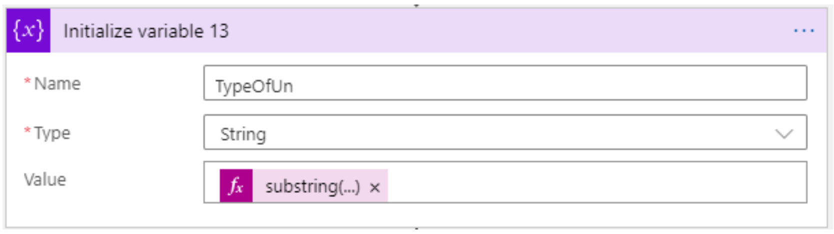
*Name	AffectedAsset
*Type	String
Value	substring(...)

Figure 47: AffectedAsset string variable.


Initialize variable 12

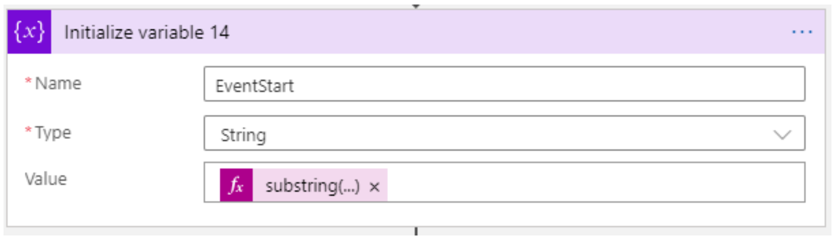
*Name	Resource Name
*Type	String
Value	substring(...)

Figure 48: Resource Name string variable.


Initialize variable 13

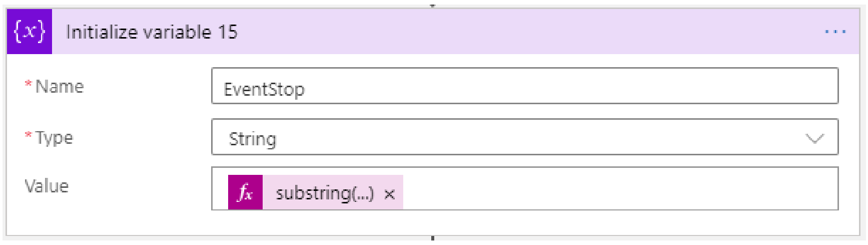
*Name	TypeOfUn
*Type	String
Value	substring(...)

Figure 49: TypeOfUn string variable.


Initialize variable 14

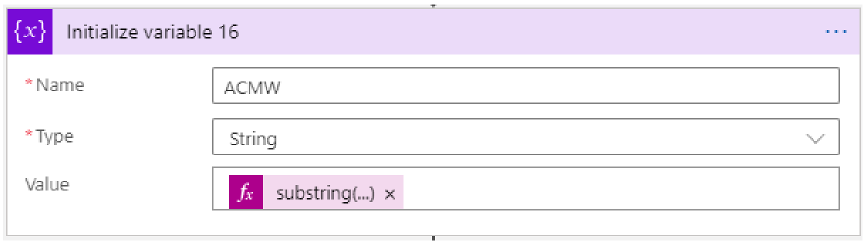
*Name	EventStart
*Type	String
Value	substring(...)

Figure 50: EventStart string variable.


Initialize variable 15

*Name	EventStop
*Type	String
Value	substring(...)

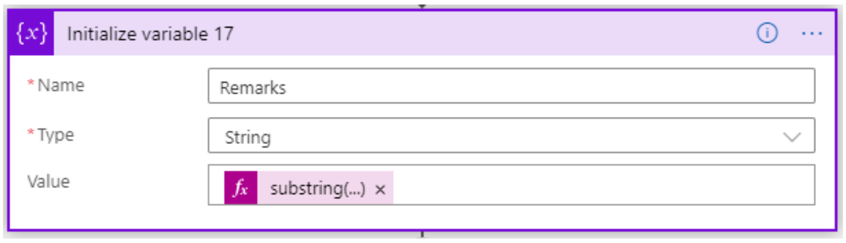
Figure 51: EventStop string variable.



Initialize variable 16

* Name	ACMW
* Type	String
Value	substring(...)

Figure 52: ACMW string variable.

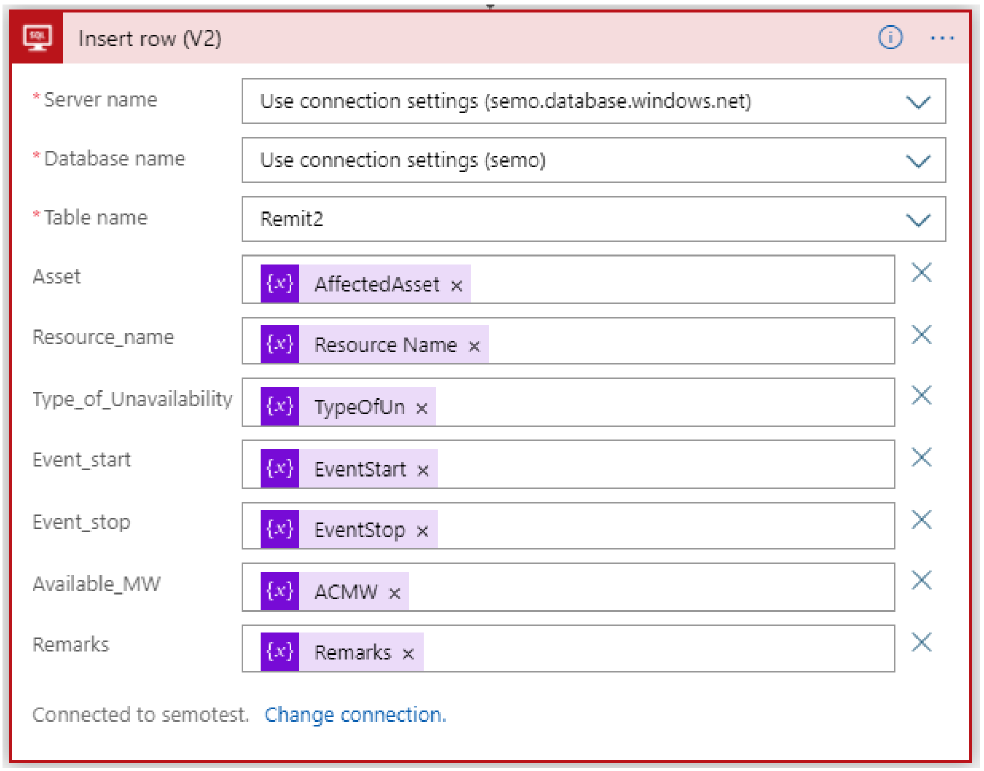


Initialize variable 17

* Name	Remarks
* Type	String
Value	substring(...)

Figure 53: Remarks string variable.

All the variables were to be created for before the final step within the logic app, the "Insert Row" feature. The insert row feature added new rows within the table "Remit2" in the SQL database "semo".



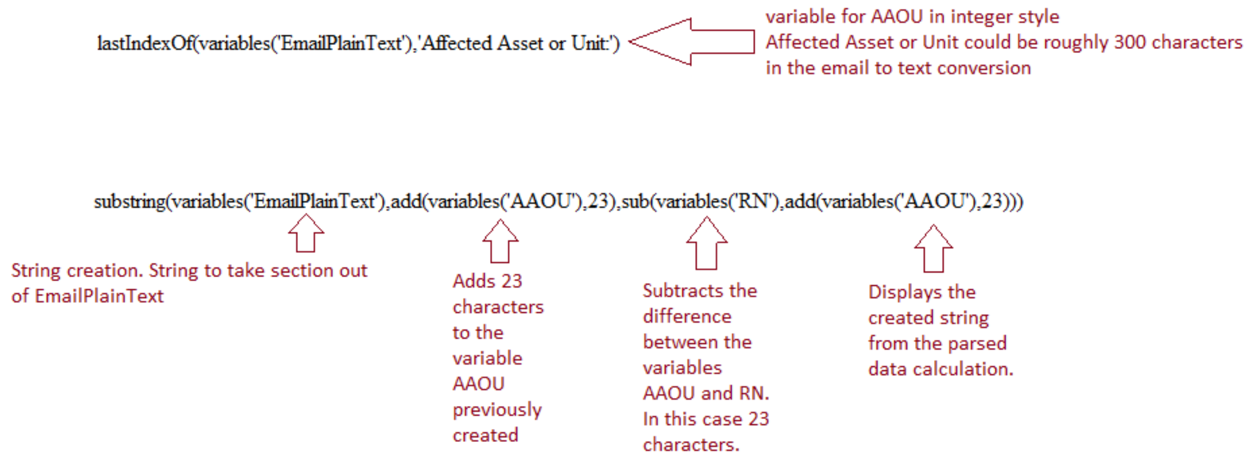
Insert row (V2)

* Server name	Use connection settings (semo.database.windows.net)
* Database name	Use connection settings (semo)
* Table name	Remit2
Asset	AffectedAsset
Resource_name	Resource Name
Type_of_Unavailability	TypeOfUn
Event_start	EventStart
Event_stop	EventStop
Available_MW	ACMW
Remarks	Remarks

Connected to semotest. Change connection.

Figure 54: Insert row (V2) in Azure.

The query editor is a common tool which is used within the Azure portal. It is directly related towards SQL queries within the student's created database, in this instance, "semo". The query editor does have a login screen. The login for the user trying to access the database, must know the server name, aka, server login and the password to match. Attempting to login with the student's ID number will result in an error.



```
lastIndexOf(variables('EmailPlainText'),'Affected Asset or Unit:')
```

variable for AAOU in integer style
Affected Asset or Unit could be roughly 300 characters in the email to text conversion

```
substring(variables('EmailPlainText'),add(variables('AAOU'),23),sub(variables('RN'),add(variables('AAOU'),23)))
```

String creation. String to take section out of EmailPlainText
Adds 23 characters to the variable AAOU previously created
Subtracts the difference between the variables AAOU and RN. In this case 23 characters.
Displays the created string from the parsed data calculation.

Figure 55: Breakdown of variable creation

As seen in Figure 55: Breakdown of variable creation, the lastIndexOf() method will return the current position for the last occurrence the specified value within a string, this states that the string is both searched from endpoint to the beginning. It returns the index which starts at the beginning. If the value never occurs it will return a negative one, -1. It checked the current position and last occurrence of "Affected Asset or Unit", within the variable "EmailPlainText".

For the bottom line of code in Figure 55: Breakdown of variable creation, the substring() method will extract the characters from a created string, that is located between two specified points, and in theory returns a new sub string. The substring() method will extract any characters between the chosen start and end points. The substring method does not change the values of the original string.

Displaying data from Azure SQL server database.

The student downloaded Visual Studio Community Edition 2017. The 2017 version was chosen over the newer 2019 version as it allowed for easier integration between the creation of the WebApp, and then linking it to the Azure account.

This enabled the student to program their web app locally with the SQL string then connected to the cloud server. Once Visual Studio is installed, the student was to sign into their Azure Account. This can be done by accessing the top right corner of the Visual studio application.

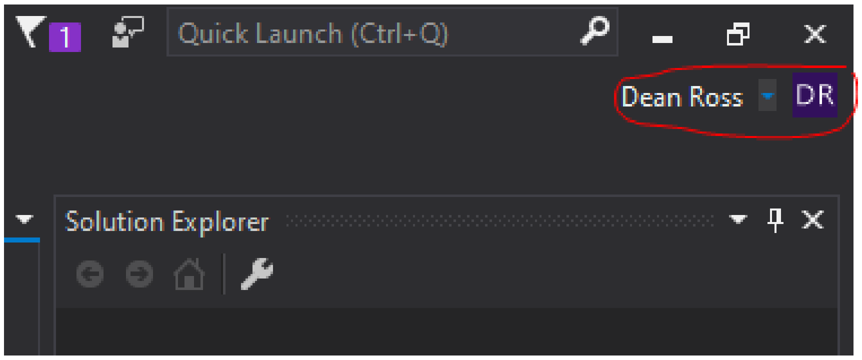


Figure 56: Visual Studio login

When the student logged in, it was possible to install multiple resources and coding languages to the Visual Studio application. One such coding language to install was C#. C# is a programming language which encompasses strong typing and static typing. It is object-oriented and component oriented. C# is a programming language which was developed by Microsoft. It runs using the .NET framework. C# is used in developing web applications, desktop applications, etc.

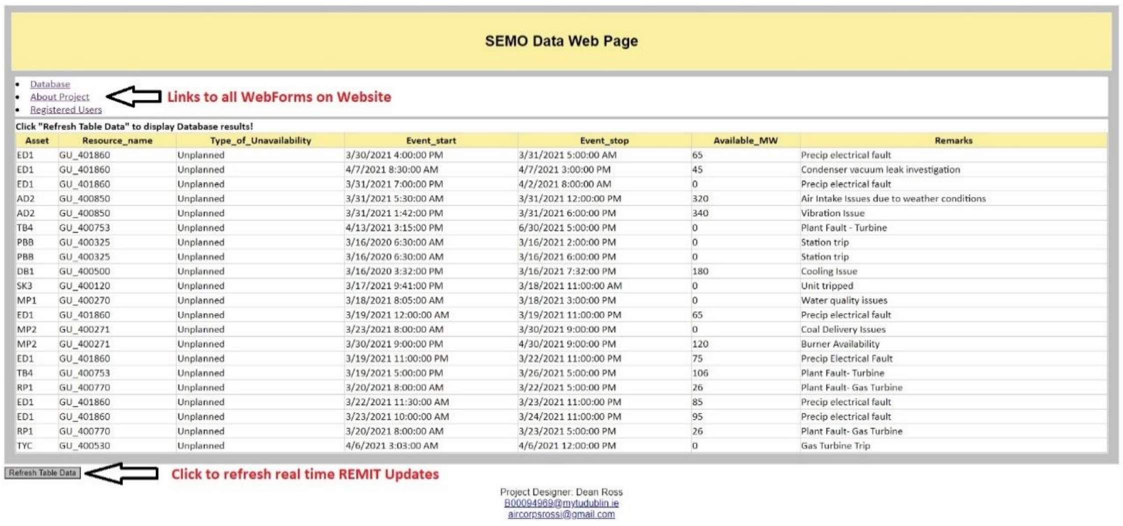
Once C# was installed the next step was to install the deploy to Azure. This was to connect the created SQL server database to the created HTML oriented Web App. This allowed for the SQL database to be displayed on the website when a button is pressed.

The student started the process of creating a new project. The project was named “Semo_Data”. The solution name was also adequately named “Semo_Data”. The next step was to create a new ASP.NET Web App. The MVC option was selected. The MVC choice enables the user to build applications whilst using a Model-View-Controller (MVC) style architecture. MVC offers multiple features to choose from. Such features include enabling fast, test-driven development for creating apps that use the latest standards.

Once the app was created, the student was to click on the bold name for the project within the solution explorer. This was on the right-hand side of the application. The add button was selected, then a new item was to be chosen. Once these were selected, the option Web Form was chosen. The student would then name the Web Form, for the project, the names were left at default to WebForm1.aspx, WebForm2.aspx, and WebForm3.aspx. Each one of these Web Forms had an intended purpose. Each Web Form followed a unique colour scheme for the entire website. The colours chosen included cream, silver, white and black. This was to keep the website modern and to allow for the website to not be harsh on the viewer's eyes.

WebForm1.aspx:

The main purpose for WebForm1.aspx was to display the REMIT Updates from the SQL database. A connection was needed to link the SQL database to the Web Form. The connection was created in WebForm1.aspx.cs. The code can be seen below in List of Software Code, under the sub section WebForm1.aspx.cs Code. A unique connection to the SQL server was established within this software code. The student was to include their SQL database ID and Password to display the REMIT Update information. The users of the website can refresh the SQL database with the use of a button. The main intention of this button is to refresh the SQL database to display any real time updates. This was shown in the project demonstration. A screen clipping from the website can be seen below, it displays the links to access the other Web Forms, the button and student signature.


SEMO Data Web Page

- Database
- About Project
- Registered Users

Click "Refresh Table Data" to display Database results!

Asset	Resource_name	Type_of_Unavailability	Event_start	Event_stop	Available_MW	Remarks
ED1	GU_401860	Unplanned	3/30/2021 4:00:00 PM	3/31/2021 5:00:00 AM	65	Precip electrical fault
ED1	GU_401860	Unplanned	4/7/2021 8:30:00 AM	4/7/2021 3:00:00 PM	45	Condenser vacuum leak investigation
ED1	GU_401860	Unplanned	3/31/2021 7:00:00 PM	4/2/2021 8:00:00 AM	0	Precip electrical fault
AD2	GU_400850	Unplanned	3/31/2021 5:30:00 AM	3/31/2021 12:00:00 PM	320	Air Intake Issues due to weather conditions
AD2	GU_400850	Unplanned	3/31/2021 1:42:00 PM	3/31/2021 6:00:00 PM	340	Vibration Issue
TB4	GU_400753	Unplanned	4/13/2021 3:15:00 PM	6/30/2021 5:00:00 PM	0	Plant Fault - Turbine
PBB	GU_400325	Unplanned	3/16/2020 6:30:00 AM	3/16/2021 2:00:00 PM	0	Station trip
PBB	GU_400325	Unplanned	3/16/2020 6:30:00 AM	3/16/2021 6:00:00 PM	0	Station trip
DB1	GU_400500	Unplanned	3/16/2020 3:32:00 PM	3/16/2021 7:32:00 PM	180	Cooling Issue
SK3	GU_400120	Unplanned	3/17/2021 9:41:00 PM	3/18/2021 11:00:00 AM	0	Unit tripped
MP1	GU_400270	Unplanned	3/18/2021 8:05:00 AM	3/18/2021 3:00:00 PM	0	Water quality issues
ED1	GU_401860	Unplanned	3/19/2021 12:00:00 AM	3/19/2021 11:00:00 PM	65	Precip electrical fault
MP2	GU_400271	Unplanned	3/23/2021 8:00:00 AM	3/30/2021 9:00:00 PM	0	Coal Delivery Issues
MP2	GU_400271	Unplanned	3/30/2021 9:00:00 PM	4/30/2021 9:00:00 PM	120	Burner Availability
ED1	GU_401860	Unplanned	3/19/2021 11:00:00 PM	3/22/2021 11:00:00 PM	75	Precip Electrical Fault
TB4	GU_400753	Unplanned	3/19/2021 5:00:00 PM	3/26/2021 5:00:00 PM	106	Plant Fault- Turbine
RP1	GU_400770	Unplanned	3/20/2021 8:00:00 AM	3/22/2021 5:00:00 PM	26	Plant Fault- Gas Turbine
ED1	GU_401860	Unplanned	3/22/2021 11:30:00 AM	3/23/2021 11:00:00 PM	85	Precip electrical fault
ED1	GU_401860	Unplanned	3/23/2021 10:00:00 AM	3/24/2021 11:00:00 PM	95	Precip electrical fault
RP1	GU_400770	Unplanned	3/20/2021 8:00:00 AM	3/23/2021 5:00:00 PM	26	Plant Fault- Gas Turbine
TYC	GU_400530	Unplanned	4/6/2021 3:03:00 AM	4/6/2021 12:00:00 PM	0	Gas Turbine Trip

Project Designer: Dean Ross
B00094969@mytudublin.ie
aircorpsrossi@gmail.com

Figure 57: WebForm1.aspx

WebForm2.aspx:

The second Web Form was used for displaying information about the project, such as the purpose of the project, and why Azure was chosen.

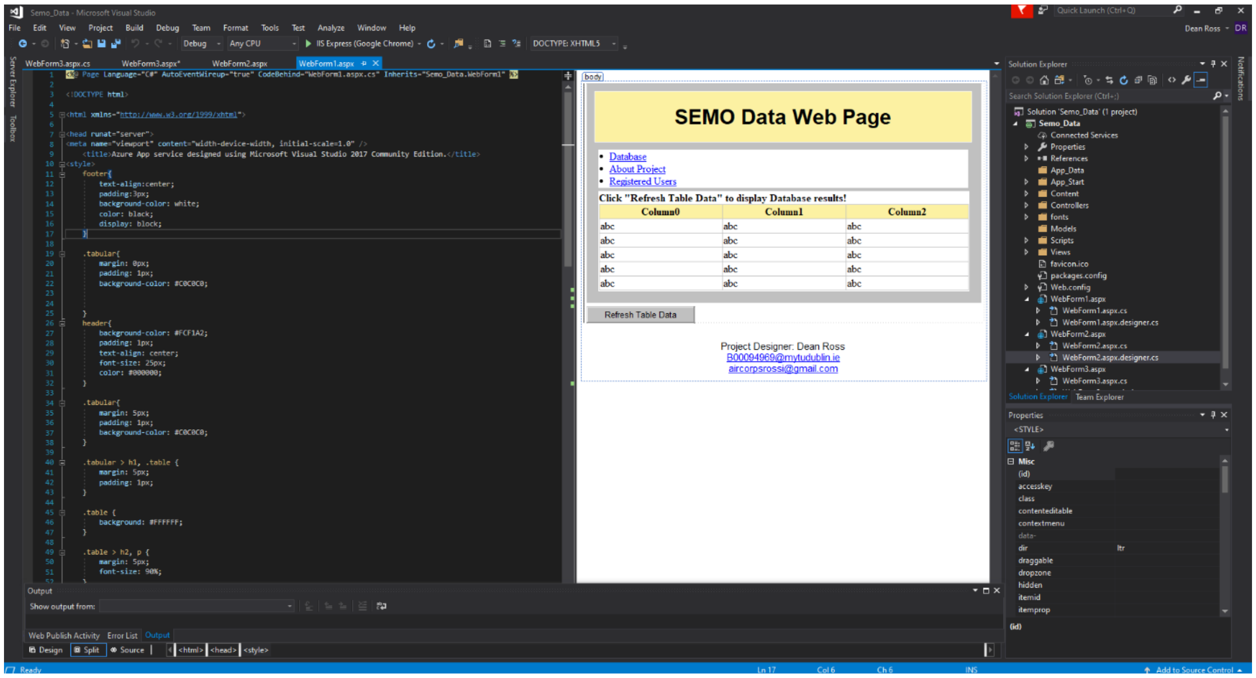
WebForm3.aspx:

The third Web Form displayed a list of all the registered units with SEMO. Companies such as the ESB, Energia, and much more can be seen in this list. This list can be displayed also using a button. Once the button is clicked, it will display the list of registered units. The list was originally downloaded as an Excel file, but was converted to a .csv file, then to a Microsoft SQL file. This was then added to the semo database that was previously created under a different table name: “dbo.Registered_Units”.

Project Testing.

Testing was done regularly throughout the project. Testing was required for each stage of the project build. The project consisted of multiple coding styles. When using Microsoft's SQL Server application, MySQL was used as the primary coding language. When Microsoft's Visual Studio was required, both C# and HTML were used. Testing was done on all applications, this even included Microsoft Azure. Query editors allowed for the user to test the application.

When writing the code within Visual Studio, the student was able to see real time updates of the website being built. If any error were present within the code, it would display an error message and display a coding error on the right side of the split viewer.



```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs" Inherits="Semo_Data.WebForm1" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">
<meta name="viewport" content="width-device-width, initial-scale=1.0" />
    <title>Azure App service designed using Microsoft Visual Studio 2017 Community Edition.</title>
<style>
    footer{
        text-align:center;
        padding:3px;
        background-color: white;
        color: black;
        display: block;
    }

    .tabular{
        margin: 0px;
        padding: 1px;
        background-color: #C0C0C0;


    }
    header{
        background-color: #FCF1A2;
        padding: 1px;
        text-align: center;
        font-size: 25px;
        color: #000000;
    }

    .tabular{
        margin: 5px;
        padding: 1px;
        background-color: #C0C0C0;
    }

    .tabular > h1, .table {
        margin: 5px;
        padding: 1px;
    }

    .table {
        background: #FFFFFF;
    }

    .table > h2, p {
        margin: 5px;
        font-size: 90%;
```

SEMO Data Web Page

- Database
- About Project
- Registered Users

Click "Refresh Table Data" to display Database results!

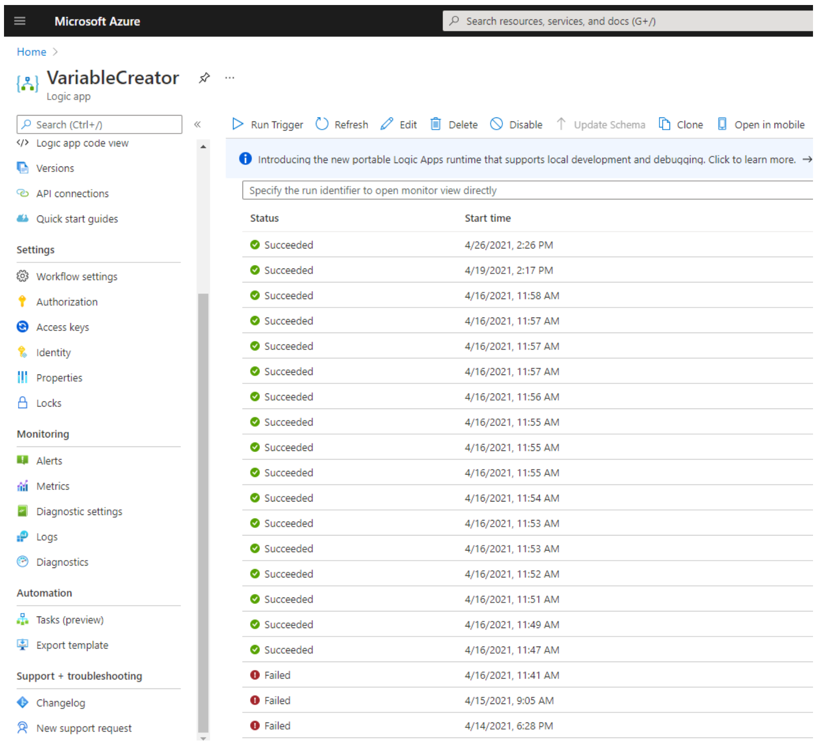
Column0	Column1	Column2
abc	abc	abc
abc	abc	abc
abc	abc	abc
abc	abc	abc
abc	abc	abc

Refresh Table Data

Project Designer: Dean Ross
B00094969@mytudublin.ie
aircorpsrossi@gmail.com

Figure 58: Visual Studio application

It was possible to check if the run history on the VariableCreator logic app using Microsoft Azure was either successful or not. However, testing was different on the logic app. To test if a logic app run was successful, the student was to send a REMIT Update from a private email account and send to the student's email account. Below, successful runs will be shown from Azure. A successful run will be shown with a green tick and the words "Succeeded". An unsuccessful run will display with a red exclamation mark and the words "Failed".



Introducing the new portable Logic Apps runtime that supports local development and debugging. Click to learn more.

Specify the run identifier to open monitor view directly

Status	Start time
Succeeded	4/26/2021, 2:26 PM
Succeeded	4/19/2021, 2:17 PM
Succeeded	4/16/2021, 11:58 AM
Succeeded	4/16/2021, 11:57 AM
Succeeded	4/16/2021, 11:57 AM
Succeeded	4/16/2021, 11:57 AM
Succeeded	4/16/2021, 11:56 AM
Succeeded	4/16/2021, 11:55 AM
Succeeded	4/16/2021, 11:55 AM
Succeeded	4/16/2021, 11:55 AM
Succeeded	4/16/2021, 11:54 AM
Succeeded	4/16/2021, 11:53 AM
Succeeded	4/16/2021, 11:53 AM
Succeeded	4/16/2021, 11:52 AM
Succeeded	4/16/2021, 11:51 AM
Succeeded	4/16/2021, 11:49 AM
Succeeded	4/16/2021, 11:47 AM
Failed	4/16/2021, 11:41 AM
Failed	4/15/2021, 9:05 AM
Failed	4/14/2021, 6:28 PM

Figure 59: logic app runs.

The testing phase for the logic app is crucial, if no testing were done, either variables would be stored incorrectly within the SQL Server database, or they would result in an unsuccessful run and not store in the SQL database at all.

A testing sheet was created on Microsoft Excel to ensure that the deliverables were working. The excel sheet is seen below and is broken up into stages for testing components on the website, logic app, and the SQL database.

Table 4: Website testing phase

Website Test					
Page	**Test Description**	**Expected Outcome**	**Result**	**Date**	**Tester Name**
1	Did the page load	Visible Homepage	Yes	12/05/2021	Dean
	Did the page links work	Loads other webpages	Yes	12/05/2021	Dean
	Did the refresh table button work	Loads REMIT data	Yes	12/05/2021	Dean
				12/05/2021	
2	Did the page load	Visible About Page	Yes	12/05/2021	Dean
	Did the page links work	Loads other webpages	Yes	12/05/2021	Dean
	Did information show	About Project was shown	Yes	12/05/2021	Dean
				12/05/2021	
3	Did the page load	Visible page for Units	Yes	12/05/2021	Dean
	Did the page links work	Loaded other webpages	Yes	12/05/2021	Dean
	Did the display table button work	List of units displayed	Yes	12/05/2021	Dean

Table 5: logic app testing phase

Logic App Test				
Test Description	**Expected Outcome**	**Result**	**Result Date**	**Tester Name**
Email Received	Email is received in logic app	Works	12/05/2021	Dean
HTML to text conversion	Converts HTML into text	Works	12/05/2021	Dean
Integer Variable	Locates position in text field, variable initialised	Works	12/05/2021	Dean
String Variable	Displays created string variable from parsed data	Works	12/05/2021	Dean
Insert Row	Inserts a new row into the database	Works	12/05/2021	Dean

Table 6: SQL Database testing phase

SQL Database Test				
Test Description	**Expected Outcome**	**Result**	**Result Date**	**Tester Name**
Row Added	New row added from logic app	works	12/05/2021	Dean
Query Editor	Open Query Editor to view rows in table	works	12/05/2021	Dean
Server Management	Open SQL server management to view rows in table	works	12/05/2021	Dean
Viewable on website	The data is viewable on hosted website	works	12/05/2021	Dean

Chapter 4 - Discussion

Overview of discussion

This section will cover the discussion of the project. The discussion will include the results of the project, and how the project succeeded. The discussion will also include aspects on how the student felt areas in which the project could have been improved upon. The discussion chapter will also cover if similar resources were used in research by others. The student also states the relevance of the experiment towards the problem at hand.

Discussion of Results

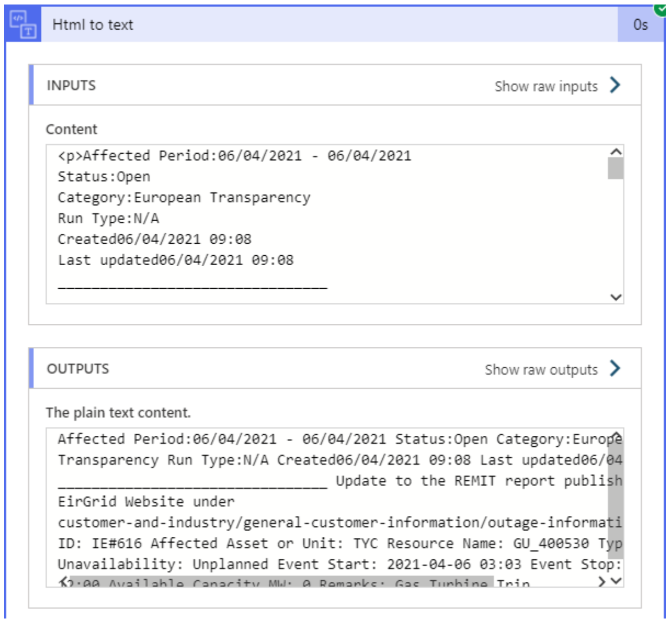
The project worked as intended. The student successfully built a website which could display the REMIT Updates from a created SQL database. A shortened version of the testing phase is shown in Table **7**: Results of project. This table displays a hypothesis for the project. The Hypothesis was primarily the testing phase for the project. Each part of the testing phase worked individually; the main goal was to combine all aspects of the testing phase and get them to work together.

Table 7: Results of project.

Operation	**Hypothesis**	**Physical Build**
Received Emails.	Working	Working
HTML to text conversion.	Working	Working
Parsing expressions.	Working	Working
INSERT Query into SQL Server Database.	Working	Working
Website using Visual Studio C#.	Working	Working
HTML , CS and PHP.	Working	Working
SQL Select Query displayed on website.	Working	Working

Results for project components

As shown in Figure 59: **logic app** runs. For the logic app run to be successful , the variables are to be inserted correctly. To check what was inserted into the logic app it is possible to click on none of the successful or failed runs. Once within the run, the logic app will display if the variables were collecting the correct information from the converted HTML to Text.



```
<p>Affected Period:06/04/2021 - 06/04/2021
Status:Open
Category:European Transparency
Run Type:N/A
Created06/04/2021 09:08
Last updated06/04/2021 09:08
________________________________
```

```
Affected Period:06/04/2021 - 06/04/2021 Status:Open Category:Europe
Transparency Run Type:N/A Created06/04/2021 09:08 Last updated06/04
________________________________ Update to the REMIT report publish
EirGrid Website under
customer-and-industry/general-customer-information/outage-informati
ID: IE#616 Affected Asset or Unit: TYC Resource Name: GU_400530 Typ
Unavailability: Unplanned Event Start: 2021-04-06 03:03 Event Stop:
2:00 Available Capacity MW: 0 Remarks: Gas Turbine Trip
```

Figure 60: HTML to text conversion

The HTML to text conversion was successful, this is shown by the green tick in the top right corner. The input of the pulled email is in HTML format, this is evident from the <p> tag, which is a paragraph tag. A paragraph is both opened and closed with <p></p>. The contents of the email were contained within the paragraph tags. The output is shown as plain text. This allows for variable creation for each email. The first variable which was to be initialised was the EmailPlainText variable. This would create the entire text field as a variable of type string. Multiple variables were created for the logic app. Some variables were of type string and others of type integer. The variables of type string would retrieve the text and were inputted into the created string variables. The variables of type integer assigned a number to the variable, this was in accordance with the location of the required string. For when the logic app created the integer fields, it assigned the variable with a number in accordance with the character positions within the

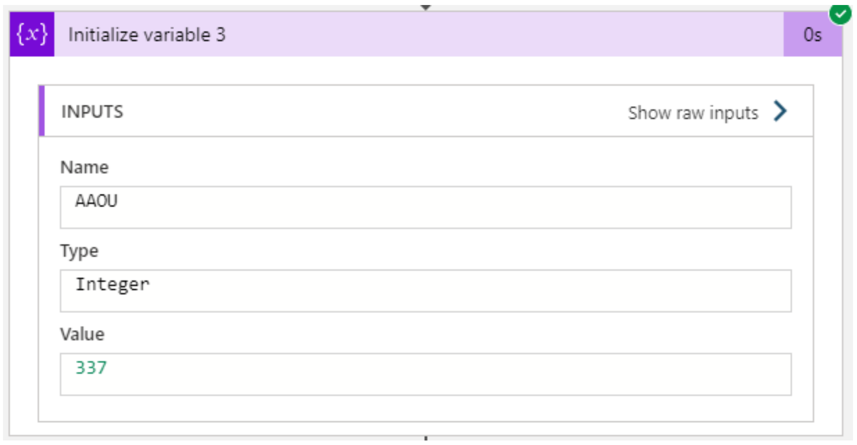
EmailPlainText variable. Variables of type integer can be shown below with their respective value. The value will increment as the next variable is initialised due to the variable being further in the EmailPlainText field. Each variable was given a name in accordance with its respective counterpart in the email. Each example is viewable below with their counterpart:

- AAOU – Affected Asset or Unit.
- RN – Resource Name.
- TOU – Type of Unavailability.
- EStart – Event Start.
- EStop – Event Stop.
- AC – Available Mega Watt Capacity.
- Rem – Remarks.

Each string variable also had a counterpart name. The variable names were close to the desired contents from the email.

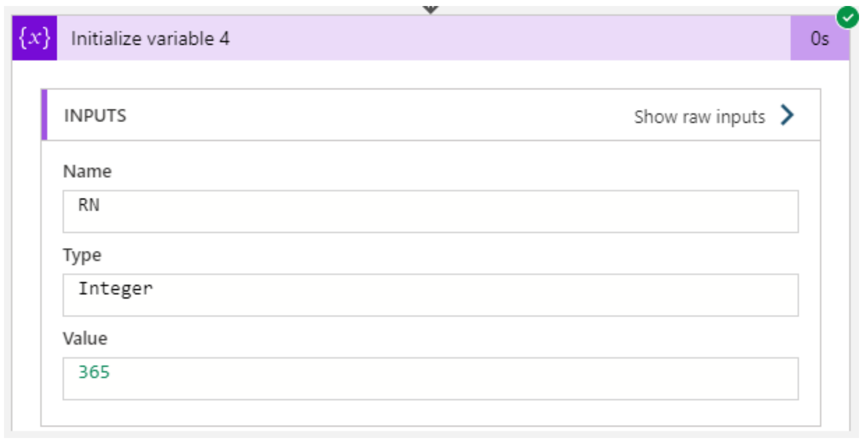
- Affect Asset – Affected Asset or Unit.
- Resource name – Resource Name.
- Type of Un – Type of Unavailability.
- EventStart – Event Start.
- EventStop – Event Stop.
- ACMW – Available Mega Watt Capacity.
- Remarks – Remarks.

For the string variables, the names of the variables were chosen to be similar as they would be displayed in the SQL server and on the website. All the variables are shown below with their respective names and the contents of the email that was sent April 26th, 2021.



Name
AAOU
Type
Integer
Value
337

Figure 61: AAOU variable initialised.


Initialize variable 4 0s

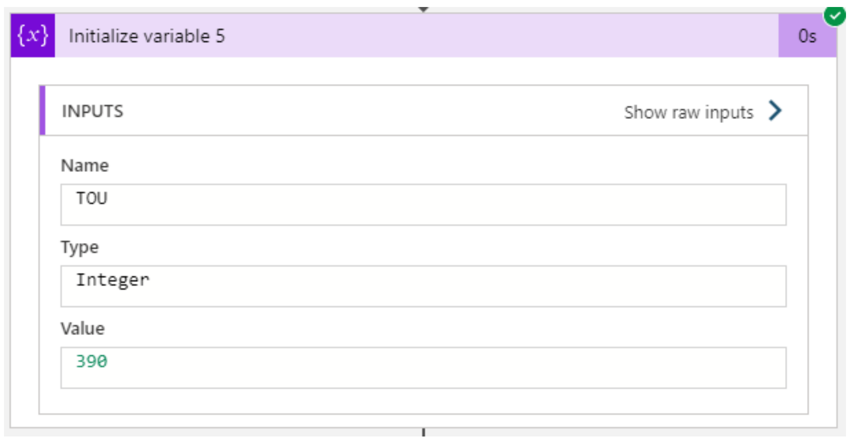
INPUTS Show raw inputs

Name: RN

Type: Integer

Value: 365

Figure 62: RN variable initialised.


Initialize variable 5 0s

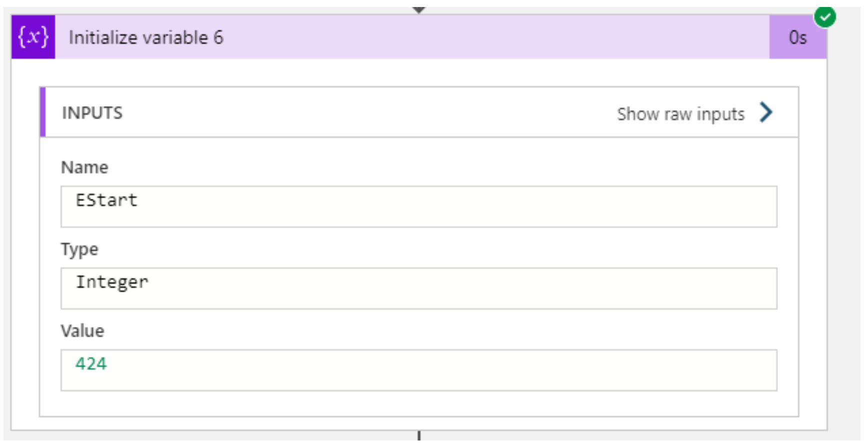
INPUTS Show raw inputs

Name: TOU

Type: Integer

Value: 390

Figure 63: TOU variable initialised.


Initialize variable 6 0s

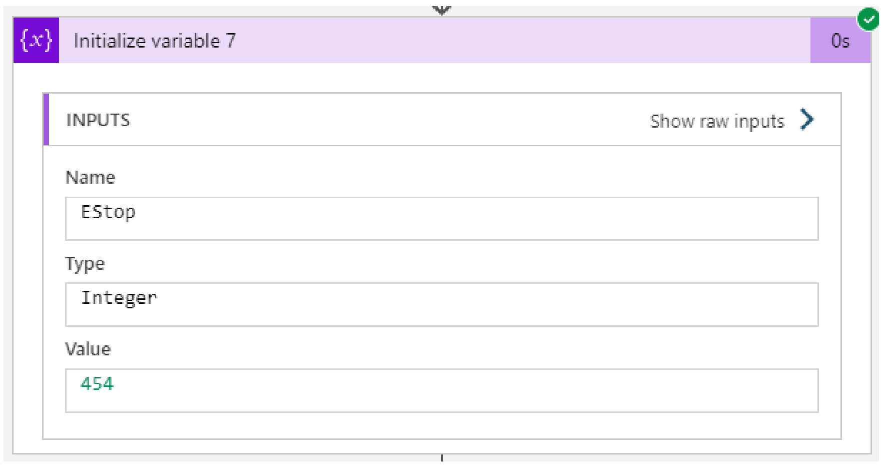
INPUTS Show raw inputs

Name: EStart

Type: Integer

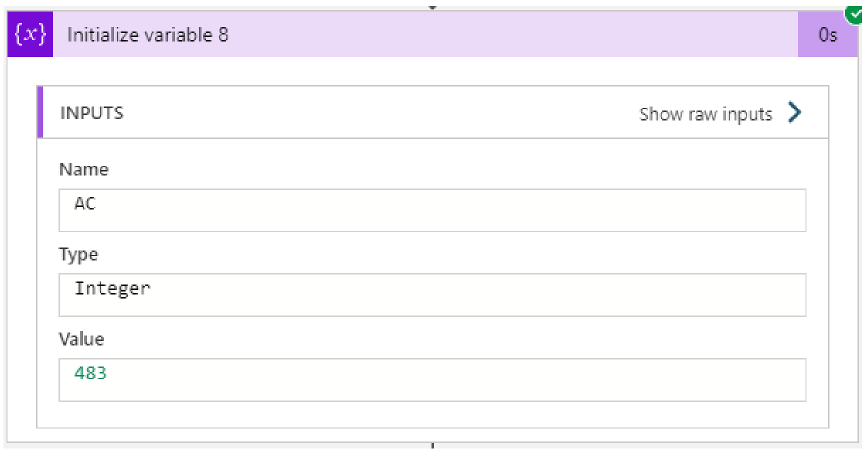
Value: 424

Figure 64: EStart variable initialised.



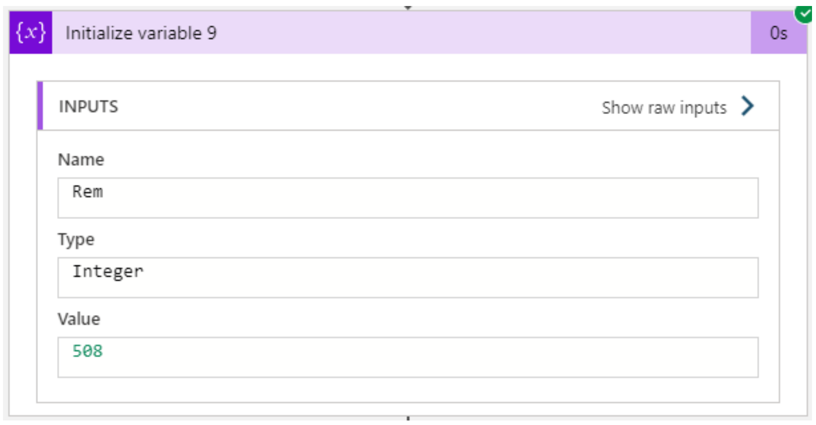
Name
EStop
Type
Integer
Value
454

Figure 65: Estop variable initialised.



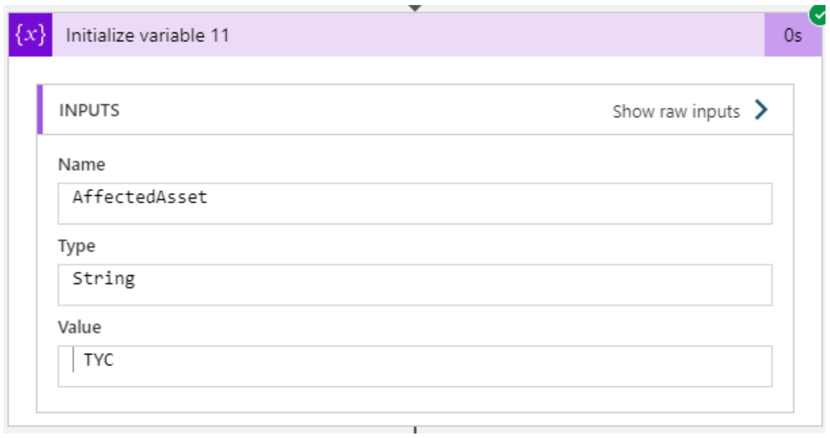
Name
AC
Type
Integer
Value
483

Figure 66: AC variable initialised.



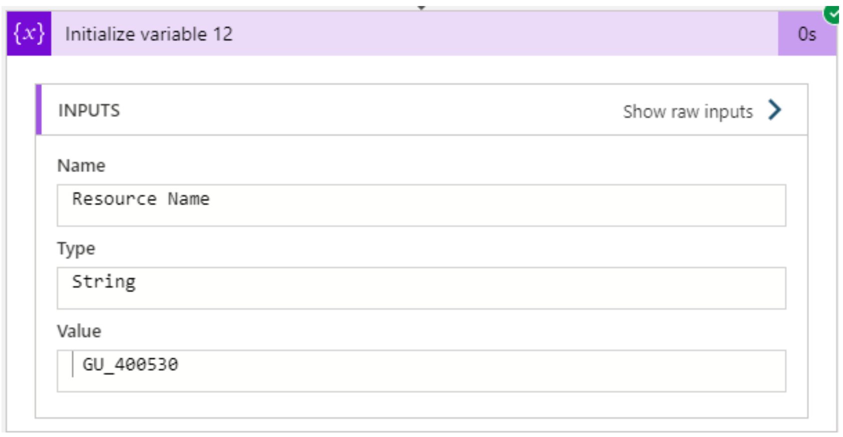
Name
Rem
Type
Integer
Value
508

Figure 67: Rem variable initialised.



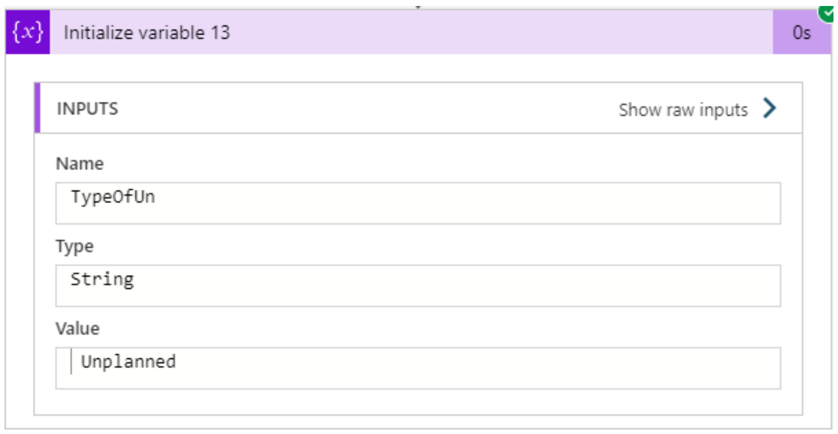
Name: AffectedAsset
Type: String
Value: TYC

Figure 68: AffectedAsset variable initialised.



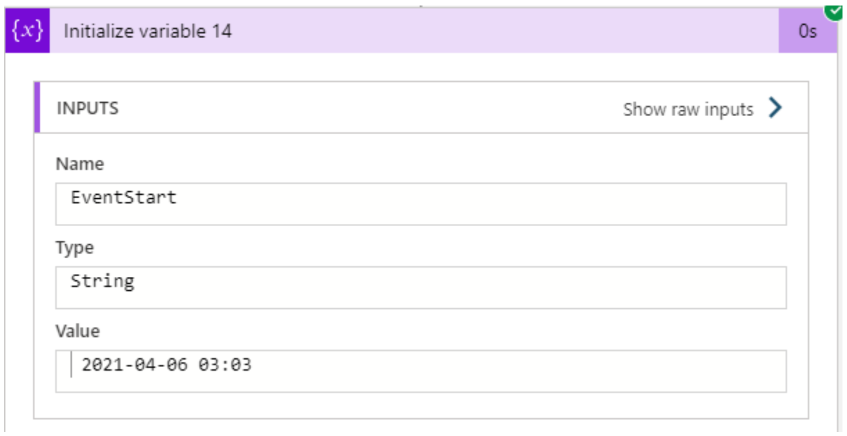
Name: Resource Name
Type: String
Value: GU_400530

Figure 69: Resource Name variable initialised.



Name: TypeOfUn
Type: String
Value: Unplanned

Figure 70: TypeOfUn variable initialised.


Initialize variable 14 0s

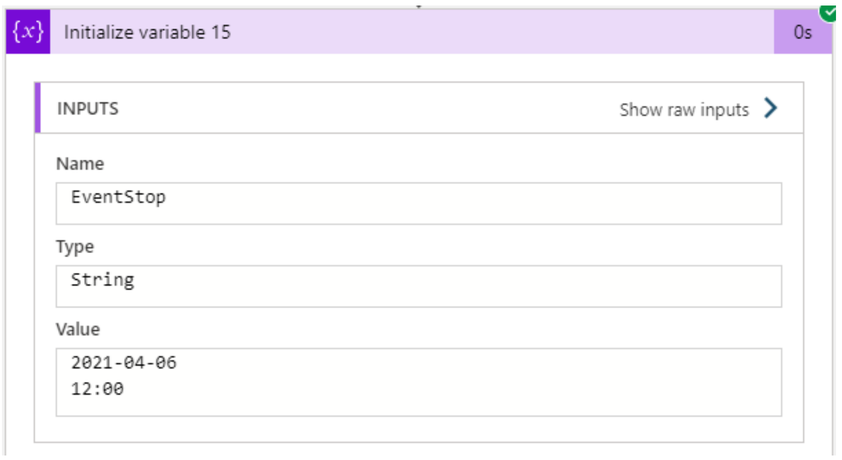
INPUTS Show raw inputs

Name
EventStart

Type
String

Value
2021-04-06 03:03

Figure 71: EventStart variable initialised.


Initialize variable 15 0s

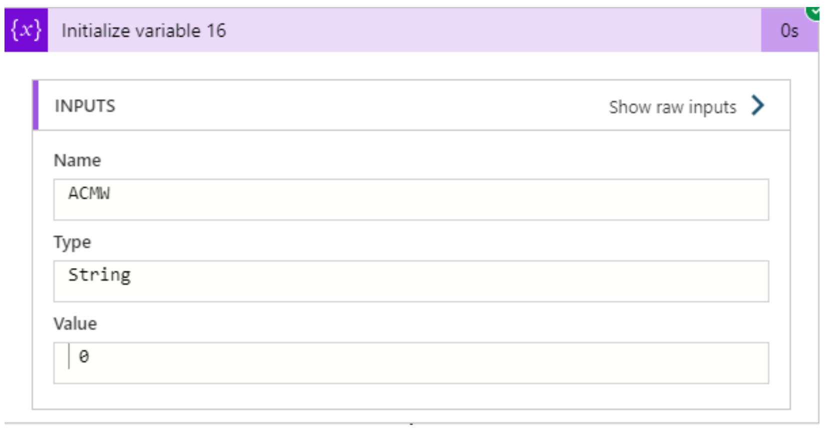
INPUTS Show raw inputs

Name
EventStop

Type
String

Value
2021-04-06
12:00

Figure 72: EventStop variable initialised.


Initialize variable 16 0s

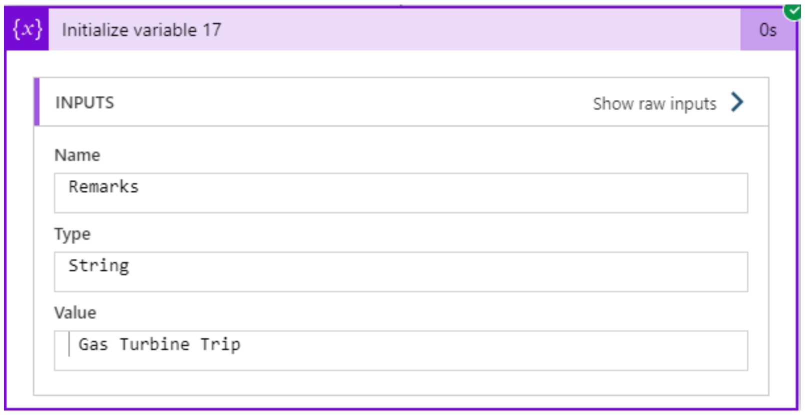
INPUTS Show raw inputs

Name
ACMW

Type
String

Value
0

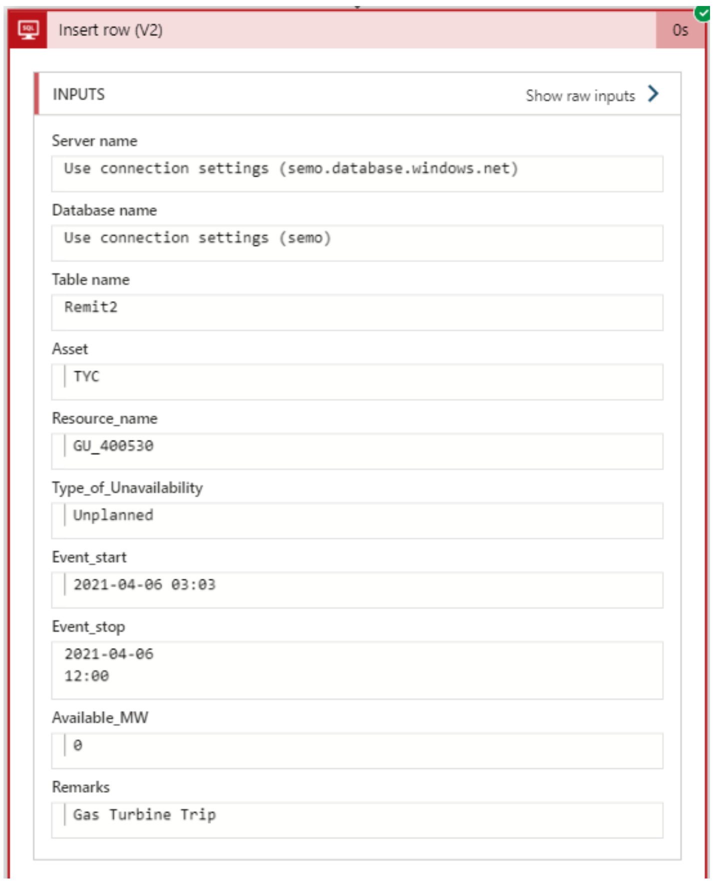
Figure 73: ACMW variable initialised.



Name: Remarks
Type: String
Value: Gas Turbine Trip

Figure 74: Remarks variable initialised.

Once all the variables were initialised, they were added to the SQL server database using the insert row (V2) feature on the logic app. The insert row feature would add all the String variables to the SQL server. This would make all the variables viewable by either using the SQL query editor on Azure, Microsoft's SQL server management, or using the website. There is two parts to the insert row feature, an input and output. The input feature will contain fields such as the Server name and the database name. When the insert row feature was created the student was to include connection strings for the SQL server. If the connection strings were not included, it would result in an error and no variables would be added to the SQL server database.


Insert row (V2) 0s

INPUTS Show raw inputs

Server name
Use connection settings (semo.database.windows.net)

Database name
Use connection settings (semo)

Table name
Remit2

Asset
TYC

Resource_name
GU_400530

Type_of_Unavailability
Unplanned

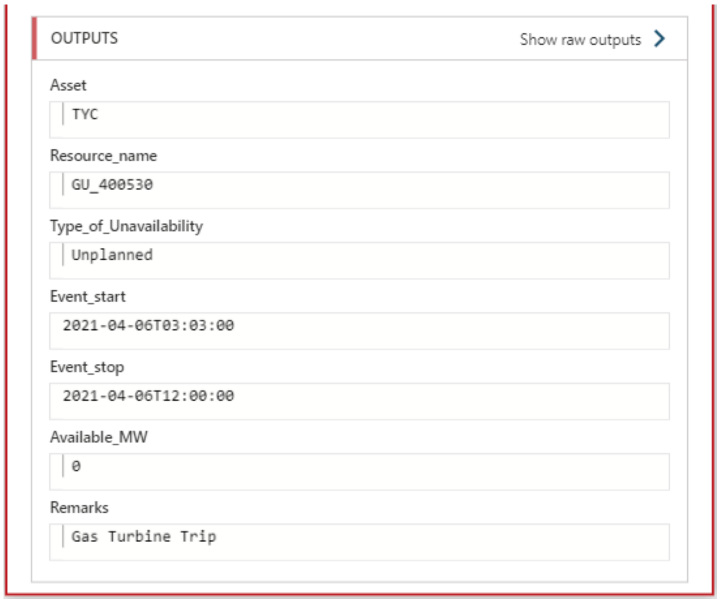
Event_start
2021-04-06 03:03

Event_stop
2021-04-06
12:00

Available_MW
0

Remarks
Gas Turbine Trip

Figure 75: Inputs for Inserting rows into SQL server database.


OUTPUTS Show raw outputs

Asset
TYC

Resource_name
GU_400530

Type_of_Unavailability
Unplanned

Event_start
2021-04-06T03:03:00

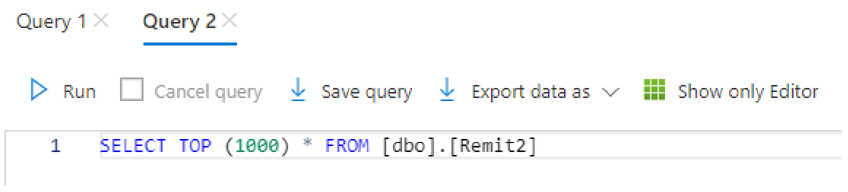
Event_stop
2021-04-06T12:00:00

Available_MW
0

Remarks
Gas Turbine Trip

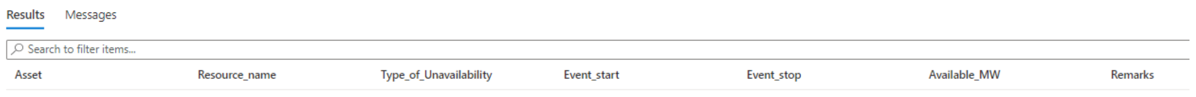
Figure 76: Outputs for inserting rows into SQL server database.

The resulting logic app run can be viewed in all the previously mention formats, query editor, Microsoft SQL server manager and the designed website. To view the logic app run in both the query editor on azure and the Microsoft SQL server manager, the user will need access to both the username and password to login. Once the user is logged in, a query will be needed to access the top 1000 rows of the Remit2 table from the database.



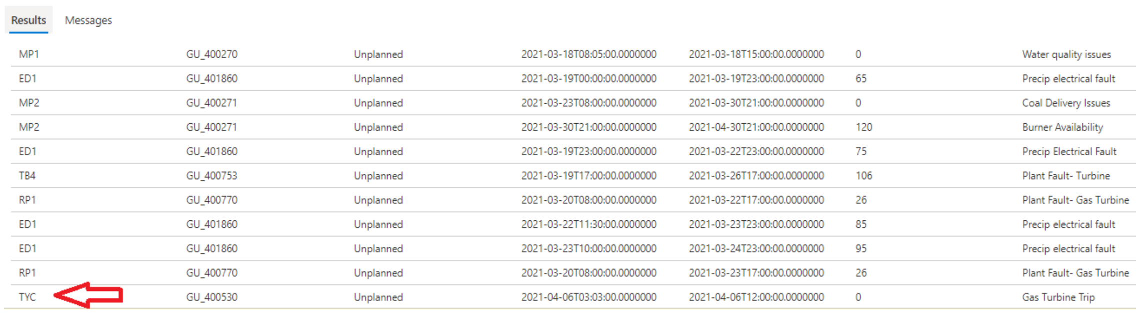
```sql
SELECT TOP (1000) * FROM [dbo].[Remit2]
```

Figure 77: Select from query in semo database using query editor on Azure.



Asset	Resource_name	Type_of_Unavailability	Event_start	Event_stop	Available_MW	Remarks

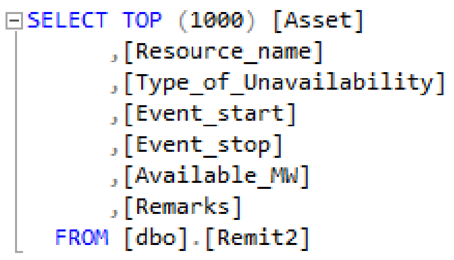
Figure 78: Variable types using Azure's query editor.



MP1	GU_400270	Unplanned	2021-03-18T08:05:00.0000000	2021-03-18T15:00:00.0000000	0	Water quality issues
ED1	GU_401860	Unplanned	2021-03-19T00:00:00.0000000	2021-03-19T23:00:00.0000000	65	Precip electrical fault
MP2	GU_400271	Unplanned	2021-03-23T08:00:00.0000000	2021-03-30T21:00:00.0000000	0	Coal Delivery Issues
MP2	GU_400271	Unplanned	2021-03-30T21:00:00.0000000	2021-04-30T21:00:00.0000000	120	Burner Availability
ED1	GU_401860	Unplanned	2021-03-19T23:00:00.0000000	2021-03-22T23:00:00.0000000	75	Precip Electrical Fault
TB4	GU_400753	Unplanned	2021-03-19T17:00:00.0000000	2021-03-26T17:00:00.0000000	106	Plant Fault- Turbine
RP1	GU_400770	Unplanned	2021-03-20T08:00:00.0000000	2021-03-22T17:00:00.0000000	26	Plant Fault- Gas Turbine
ED1	GU_401860	Unplanned	2021-03-22T11:30:00.0000000	2021-03-23T23:00:00.0000000	85	Precip electrical fault
ED1	GU_401860	Unplanned	2021-03-23T10:00:00.0000000	2021-03-24T23:00:00.0000000	95	Precip electrical fault
RP1	GU_400770	Unplanned	2021-03-20T08:00:00.0000000	2021-03-23T17:00:00.0000000	26	Plant Fault- Gas Turbine
TYC	GU_400530	Unplanned	2021-04-06T03:03:00.0000000	2021-04-06T12:00:00.0000000	0	Gas Turbine Trip

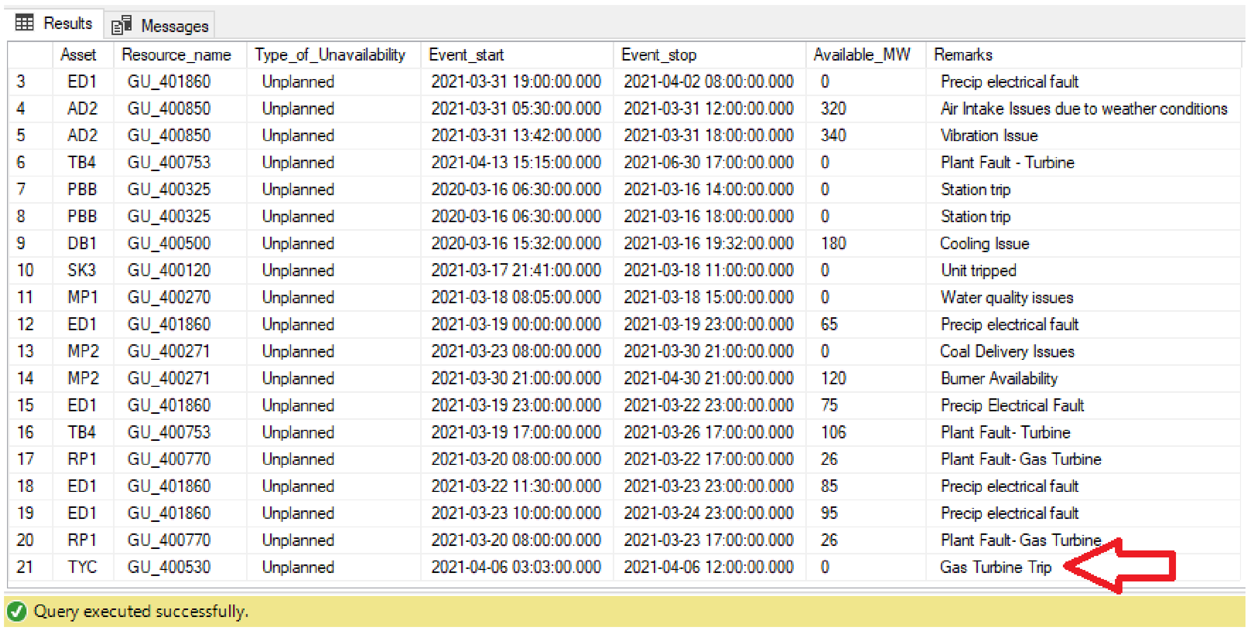
Figure 79: Successful logic app run from April 26th, 2021 on Azure.

Selecting the top 1000 rows varies slightly using Microsoft's SQL server management studio. Instead of typing a short query like Azure's query editor, a larger query is needed to be typed.



```sql
SELECT TOP (1000) [Asset]
      ,[Resource_name]
      ,[Type_of_Unavailability]
      ,[Event_start]
      ,[Event_stop]
      ,[Available_MW]
      ,[Remarks]
  FROM [dbo].[Remit2]
```

Figure 80: Microsoft SQL query for selecting top 1000 rows.



	Asset	Resource_name	Type_of_Unavailability	Event_start	Event_stop	Available_MW	Remarks
3	ED1	GU_401860	Unplanned	2021-03-31 19:00:00.000	2021-04-02 08:00:00.000	0	Precip electrical fault
4	AD2	GU_400850	Unplanned	2021-03-31 05:30:00.000	2021-03-31 12:00:00.000	320	Air Intake Issues due to weather conditions
5	AD2	GU_400850	Unplanned	2021-03-31 13:42:00.000	2021-03-31 18:00:00.000	340	Vibration Issue
6	TB4	GU_400753	Unplanned	2021-04-13 15:15:00.000	2021-06-30 17:00:00.000	0	Plant Fault - Turbine
7	PBB	GU_400325	Unplanned	2020-03-16 06:30:00.000	2021-03-16 14:00:00.000	0	Station trip
8	PBB	GU_400325	Unplanned	2020-03-16 06:30:00.000	2021-03-16 18:00:00.000	0	Station trip
9	DB1	GU_400500	Unplanned	2020-03-16 15:32:00.000	2021-03-16 19:32:00.000	180	Cooling Issue
10	SK3	GU_400120	Unplanned	2021-03-17 21:41:00.000	2021-03-18 11:00:00.000	0	Unit tripped
11	MP1	GU_400270	Unplanned	2021-03-18 08:05:00.000	2021-03-18 15:00:00.000	0	Water quality issues
12	ED1	GU_401860	Unplanned	2021-03-19 00:00:00.000	2021-03-19 23:00:00.000	65	Precip electrical fault
13	MP2	GU_400271	Unplanned	2021-03-23 08:00:00.000	2021-03-30 21:00:00.000	0	Coal Delivery Issues
14	MP2	GU_400271	Unplanned	2021-03-30 21:00:00.000	2021-04-30 21:00:00.000	120	Burner Availability
15	ED1	GU_401860	Unplanned	2021-03-19 23:00:00.000	2021-03-22 23:00:00.000	75	Precip Electrical Fault
16	TB4	GU_400753	Unplanned	2021-03-19 17:00:00.000	2021-03-26 17:00:00.000	106	Plant Fault- Turbine
17	RP1	GU_400770	Unplanned	2021-03-20 08:00:00.000	2021-03-22 17:00:00.000	26	Plant Fault- Gas Turbine
18	ED1	GU_401860	Unplanned	2021-03-22 11:30:00.000	2021-03-23 23:00:00.000	85	Precip electrical fault
19	ED1	GU_401860	Unplanned	2021-03-23 10:00:00.000	2021-03-24 23:00:00.000	95	Precip electrical fault
20	RP1	GU_400770	Unplanned	2021-03-20 08:00:00.000	2021-03-23 17:00:00.000	26	Plant Fault- Gas Turbine
21	TYC	GU_400530	Unplanned	2021-04-06 03:03:00.000	2021-04-06 12:00:00.000	0	Gas Turbine Trip

Query executed successfully.

Figure 81: Top 1000 row results.

Finding the logic app run using the website is a lot easier. No coding is involved to find the run. It is done by the click of a button. The user must click the “Refresh Table Data” button on the website to display the parsed REMIT Update data.

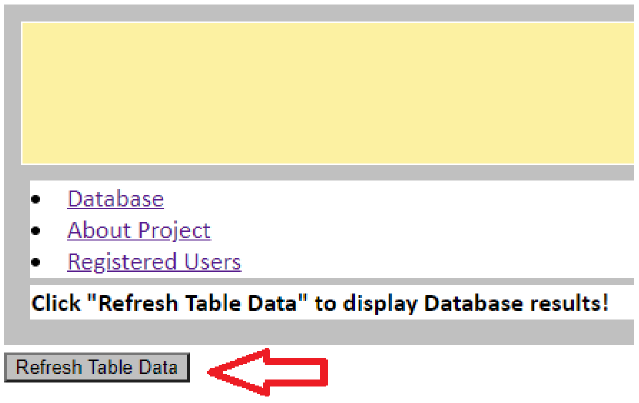
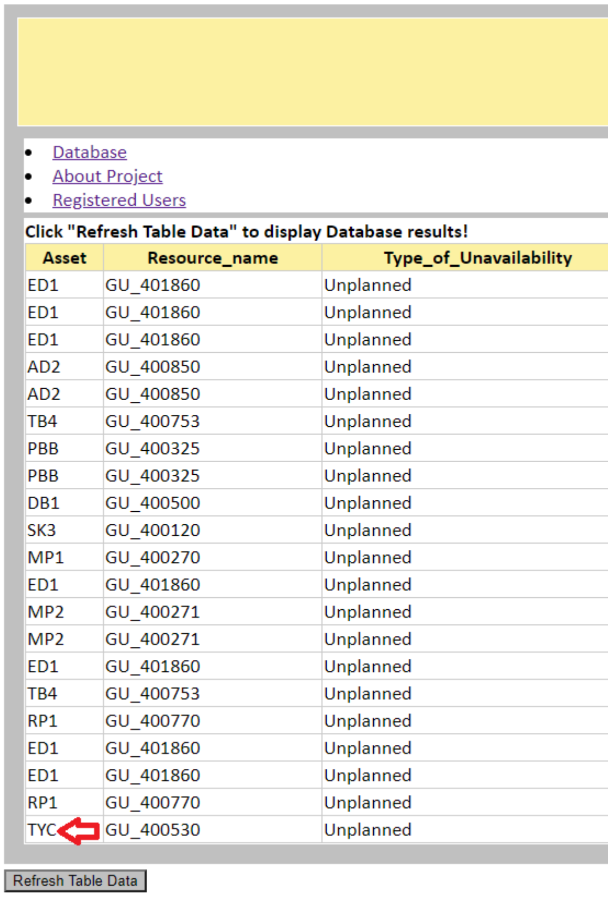


Figure 82: Refresh table data button on website.

Once the button is clicked, a table from the SQL server is displayed to the user. This is available due to the connection string through Azure.



- Database
- About Project
- Registered Users

Click "Refresh Table Data" to display Database results!

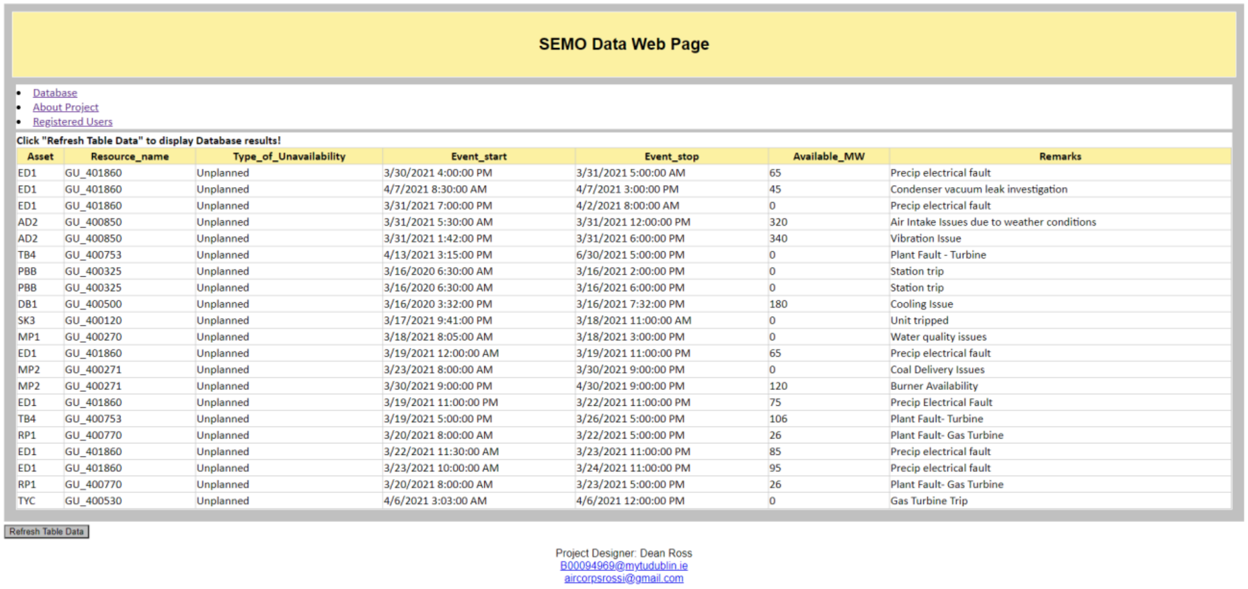
Asset	Resource_name	Type_of_Unavailability
ED1	GU_401860	Unplanned
ED1	GU_401860	Unplanned
ED1	GU_401860	Unplanned
AD2	GU_400850	Unplanned
AD2	GU_400850	Unplanned
TB4	GU_400753	Unplanned
PBB	GU_400325	Unplanned
PBB	GU_400325	Unplanned
DB1	GU_400500	Unplanned
SK3	GU_400120	Unplanned
MP1	GU_400270	Unplanned
ED1	GU_401860	Unplanned
MP2	GU_400271	Unplanned
MP2	GU_400271	Unplanned
ED1	GU_401860	Unplanned
TB4	GU_400753	Unplanned
RP1	GU_400770	Unplanned
ED1	GU_401860	Unplanned
ED1	GU_401860	Unplanned
RP1	GU_400770	Unplanned
TYC	GU_400530	Unplanned

Figure 83: logic app run displayed on website.

Website results

As mentioned prior in the report a website was required by the student for displaying parsed REMIT update data. This data is viewable in real time with the use of a logic app and SQL server database. The main purpose of the website was to display the REMIT Update data, however, the student added two additional pages to the website. Each page had an intended purpose. The home page displayed REMIT Update data, the about page displayed a quick summary of the website's purpose, and the third and final page contained another table that displayed a list of the registered units with SEMO. Both tables for the REMIT data and registered units with SEMO are viewable with a HTML button. The website is accessible using the following link: [21]

https://semowebpageproject.azurewebsites.net/WebForm1.aspx.



SEMO Data Web Page

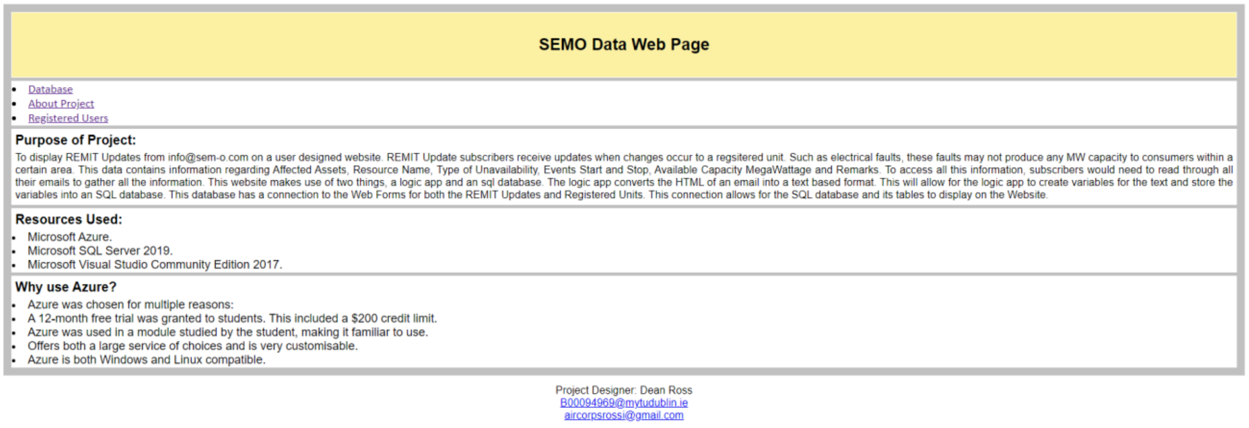
- Database
- About Project
- Registered Users

Click "Refresh Table Data" to display Database results!

Asset	Resource_name	Type_of_Unavailability	Event_start	Event_stop	Available_MW	Remarks
ED1	GU_401860	Unplanned	3/30/2021 4:00:00 PM	3/31/2021 5:00:00 AM	65	Precip electrical fault
ED1	GU_401860	Unplanned	4/7/2021 8:30:00 AM	4/7/2021 3:00:00 PM	45	Condenser vacuum leak investigation
ED1	GU_401860	Unplanned	3/31/2021 7:00:00 PM	4/2/2021 8:00:00 AM	0	Precip electrical fault
AD2	GU_400850	Unplanned	3/31/2021 5:30:00 AM	3/31/2021 12:00:00 PM	320	Air Intake Issues due to weather conditions
AD2	GU_400850	Unplanned	3/31/2021 1:42:00 PM	3/31/2021 6:00:00 PM	340	Vibration Issue
TB4	GU_400753	Unplanned	4/13/2021 3:15:00 PM	6/30/2021 5:00:00 PM	0	Plant Fault - Turbine
PBB	GU_400325	Unplanned	3/16/2020 6:30:00 AM	3/16/2021 2:00:00 PM	0	Station trip
PBB	GU_400325	Unplanned	3/16/2020 6:30:00 AM	3/16/2021 6:00:00 PM	0	Station trip
DB1	GU_400500	Unplanned	3/16/2020 3:32:00 PM	3/16/2021 7:32:00 PM	180	Cooling Issue
SK3	GU_400120	Unplanned	3/17/2021 9:41:00 PM	3/18/2021 11:00:00 AM	0	Unit tripped
MP1	GU_400270	Unplanned	3/18/2021 8:05:00 AM	3/18/2021 3:00:00 PM	0	Water quality issues
ED1	GU_401860	Unplanned	3/19/2021 12:00:00 AM	3/19/2021 11:00:00 PM	65	Precip electrical fault
MP2	GU_400271	Unplanned	3/23/2021 8:00:00 AM	3/30/2021 9:00:00 PM	0	Coal Delivery Issues
MP2	GU_400271	Unplanned	3/30/2021 9:00:00 PM	4/30/2021 9:00:00 PM	120	Burner Availability
ED1	GU_401860	Unplanned	3/19/2021 11:00:00 PM	3/22/2021 11:00:00 PM	75	Precip Electrical Fault
TB4	GU_400753	Unplanned	3/19/2021 5:00:00 PM	3/26/2021 5:00:00 PM	106	Plant Fault- Turbine
RP1	GU_400770	Unplanned	3/20/2021 8:00:00 AM	3/22/2021 5:00:00 PM	26	Plant Fault- Gas Turbine
ED1	GU_401860	Unplanned	3/22/2021 11:30:00 AM	3/23/2021 11:00:00 PM	85	Precip electrical fault
ED1	GU_401860	Unplanned	3/23/2021 10:00:00 AM	3/24/2021 11:00:00 PM	95	Precip electrical fault
RP1	GU_400770	Unplanned	3/20/2021 8:00:00 AM	3/23/2021 5:00:00 PM	26	Plant Fault- Gas Turbine
TYC	GU_400530	Unplanned	4/6/2021 3:03:00 AM	4/6/2021 12:00:00 PM	0	Gas Turbine Trip

Project Designer: Dean Ross
B00094969@mytudublin.ie
aircorpsrossi@gmail.com

Figure 84: WebForm1.aspx - "Homepage".



SEMO Data Web Page

- Database
- About Project
- Registered Users

Purpose of Project:

To display REMIT Updates from info@sem-o.com on a user designed website. REMIT Update subscribers receive updates when changes occur to a regsitered unit. Such as electrical faults, these faults may not produce any MW capacity to consumers within a certain area. This data contains information regarding Affected Assets, Resource Name, Type of Unavailability, Events Start and Stop, Available Capacity MegaWattage and Remarks. To access all this information, subscribers would need to read through all their emails to gather all the information. This website makes use of two things, a logic app and an sql database. The logic app converts the HTML of an email into a text based format. This will allow for the logic app to create variables for the text and store the variables into an SQL database. This database has a connection to the Web Forms for both the REMIT Updates and Registered Units. This connection allows for the SQL database and its tables to display on the Website.

Resources Used:

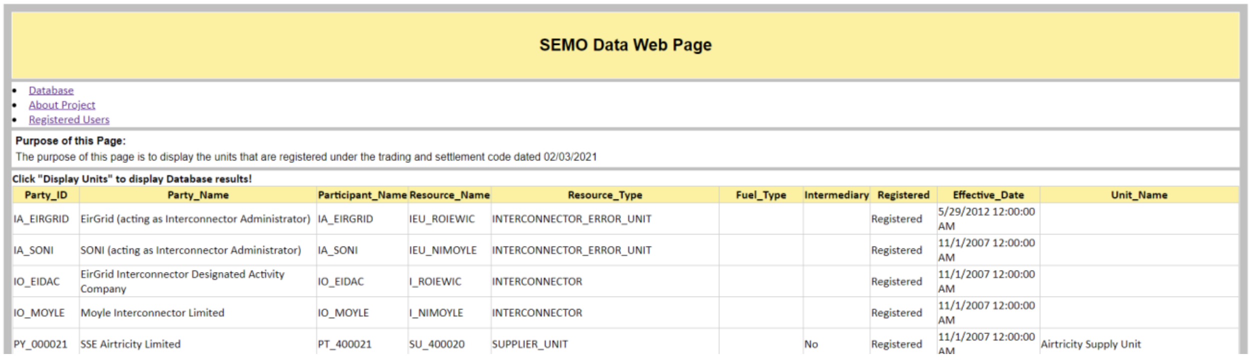
- Microsoft Azure.
- Microsoft SQL Server 2019.
- Microsoft Visual Studio Community Edition 2017.

Why use Azure?

- Azure was chosen for multiple reasons:
- A 12-month free trial was granted to students. This included a $200 credit limit.
- Azure was used in a module studied by the student, making it familiar to use.
- Offers both a large service of choices and is very customisable.
- Azure is both Windows and Linux compatible.

Project Designer: Dean Ross
B00094969@mytudublin.ie
aircorpsrossi@gmail.com

Figure 85: WebForm2.apsx - "About Project".



SEMO Data Web Page

- Database
- About Project
- Registered Users

Purpose of this Page:

The purpose of this page is to display the units that are registered under the trading and settlement code dated 02/03/2021

Click "Display Units" to display Database results!

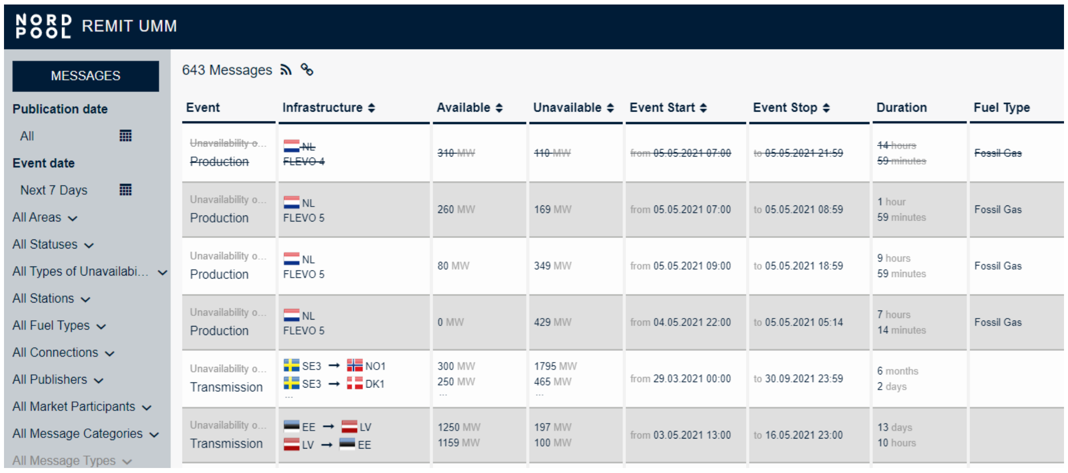
Party_ID	Party_Name	Participant_Name	Resource_Name	Resource_Type	Fuel_Type	Intermediary	Registered	Effective_Date	Unit_Name
IA_EIRGRID	EirGrid (acting as Interconnector Administrator)	IA_EIRGRID	IEU_ROIEWIC	INTERCONNECTOR_ERROR_UNIT			Registered	5/29/2012 12:00:00 AM	
IA_SONI	SONI (acting as Interconnector Administrator)	IA_SONI	IEU_NIMOYLE	INTERCONNECTOR_ERROR_UNIT			Registered	11/1/2007 12:00:00 AM	
IO_EIDAC	EirGrid Interconnector Designated Activity Company	IO_EIDAC	I_ROIEWIC	INTERCONNECTOR			Registered	11/1/2007 12:00:00 AM	
IO_MOYLE	Moyle Interconnector Limited	IO_MOYLE	I_NIMOYLE	INTERCONNECTOR			Registered	11/1/2007 12:00:00 AM	
PY_000021	SSE Airtricity Limited	PT_400021	SU_400020	SUPPLIER_UNIT		No	Registered	11/1/2007 12:00:00 AM	Airtricity Supply Unit

Figure 86: WebForm3.aspx – “Registered Units”.

There are nearly 500 units registered with SEMO as of March 2nd, 2021. The registered units webpage contains information regarding the registered units, such as who owns the unit, "Party_Name", the Resource Name for the unit and what type of fuel type the unit is, some examples include gas, coal, and wind.

Similar Research on familiar resources

As of January 2021, a website was launched by a large energy market operator in Europe, NordPool, which displays similar data to the student designed website. NordPool's website offers REMIT updates on countries across Europe's electricity market. Many countries are included in NordPool's website: https://umm.nordpoolgroup.com. Countries included on the website: Netherlands, Sweden, Norway, Ireland, Germany, and many more.



643 Messages

Event	Infrastructure	Available	Unavailable	Event Start	Event Stop	Duration	Fuel Type
~~Unavailability o... Production~~	~~NL FLEVO 4~~	~~310 MW~~	~~110 MW~~	~~from 05.05.2021 07:00~~	~~to 05.05.2021 21:59~~	~~14 hours 59 minutes~~	~~Fossil Gas~~
Unavailability o... Production	NL FLEVO 5	260 MW	169 MW	from 05.05.2021 07:00	to 05.05.2021 08:59	1 hour 59 minutes	Fossil Gas
Unavailability o... Production	NL FLEVO 5	80 MW	349 MW	from 05.05.2021 09:00	to 05.05.2021 18:59	9 hours 59 minutes	Fossil Gas
Unavailability o... Production	NL FLEVO 5	0 MW	429 MW	from 04.05.2021 22:00	to 05.05.2021 05:14	7 hours 14 minutes	Fossil Gas
Unavailability o... Transmission	SE3 → NO1 SE3 → DK1 ...	300 MW 250 MW ...	1795 MW 465 MW ...	from 29.03.2021 00:00	to 30.09.2021 23:59	6 months 2 days	
Unavailability o... Transmission	EE → LV LV → EE	1250 MW 1159 MW	197 MW 100 MW	from 03.05.2021 13:00	to 16.05.2021 23:00	13 days 10 hours	

Figure 87: NordPool's website. **[22]**

NordPool's website offers users information on the infrastructure, which was affected, how much Mega Watts are available, and even the Start and Stop times of the event.

The student's website is similar in offering users information on affected assets, event time, fuel type, and the available Mega Watt capacity.

In terms of possible upgrades for the student's website, a sidebar could be added to allow for easier navigation, the current time in whichever region the user is accessing the website, and possibly add images to the website, such as flags or company logos.

NordPool allows for participants of the European electricity market to publish and receive urgent updates. NordPool's REMIT UMM allows for users to notify the electricity market about any planned or unplanned changes with electricity generation, consumption, or even electricity transmission. Users of NordPool's REMIT UMM can view real-time events which are occurring in the electricity market, price changing and any disturbances that are caused. [23]

Overall discussion

The results demonstrate a success. All aspects of the project worked as intended. The student sought out to create a project which could display parsed data on a website or product. The data that was parsed was from SEMO. The final build could display parsed REMIT updates on a hosted website. Overall, the result was successful, the student successfully built a hosted website using Microsoft Azure, which displayed parsed data from SEMO, using a logic app and SQL server database. The student learned valuable skills throughout the course of the project. The student became knowledgeable in both front-end and back-end cloud-based engineering. When doing the front-end side of cloud-based engineering, the student built a hosted website with the use of HTML and C#. When doing back-end cloud-based developing, the student primarily focused on the logic app and SQL server, this was completed with the aid of .Net coding. The student become proficient in the use of HTML, C# and the .Net framework. One area in which the website could be improved upon is through using more JavaScript elements, such as including the current time and location, and possibly drop-down menus. Using CSS would vastly reduce the size of the code for the WebForm.aspx files.

The project is relevant due to Ireland and Northern Ireland not having a website to display REMIT updates regarding the all-island electricity market which is operated by SEMO. SEMO does however offer subscribers updates through emails, but all the information is not in one place. With the hosted website the student can provide SEMO subscribers an alternative to access all REMIT Updates at the one time. The hosted website can also provide SEMO subscribers with all the registered units in Ireland/Northern Ireland at the click of a button. NordPool offers users an alternative with REMIT Updates for the whole of Europe. At the time the project was chosen, there was no other choice or alternative for viewing all the REMIT data in one place, until NordPool launched their hosted website in January 2021.

Chapter 5 - Conclusions and Recommendations

Conclusions

This project successfully shows the importance of front-end and back-end cloud-based development, and how combining both aspects of cloud-based development is done. It is demonstrated using parsing data from an information source, such as an email, creating variables for the required information, storing it within an SQL database, then displaying and granting access to users to view information in one location on a hosted website.

The REMIT data and the live updates are viewable on the hosted website by pressing the button for refreshing the table data. This will check for a response from the SQL server if a new email has been parsed from the logic app and stored in the SQL database. If no email was sent to the student, the website will remain the same. If an email is received the website will display a new row in the table.

The project met all the deliverables for building a hosted website that was able to display the parsed and analysed REMIT data. The website displayed live remit updates, which were stored on a cloud-based SQL server using Azure.

To conclude, the project was completed successfully with an exception for a few inconveniences along the path to completion. Minor inconveniences were encountered, such as not sticking to the project timeline, this included going over the deliverable timeline by a week, which would delay moving on to another aspect of the project. Additional features added to the website would make the website more user friendly and possibly more appealing to the eye.

Recommendations

When attempting to do a project which is of similar fashion, time management, planning and coding are all important aspects. Make sure to ensure a strict schedule for deliverables within the project. This may include giving yourself two weeks to have built a successful SQL database, or to allow up to three weeks for building a website. Ensuring tighter deadlines may cause a lot of pressure to meet the deliverables on time, however, it allows for more time to work on other aspects of the project, such as writing the report, or making changes to previously written code/material.

A good thing to create would be a scope for any project. This can offer a visual timeline for the project. A Gantt chart was used to follow the deadlines for the deliverables. A Gantt chart was created for both the first semester and second semester for the project, it was helpful for keeping the student on track and to fulfil the deliverables on time.

One key aspect to the project was the testing phase, it may seem desirable to build the project all in one go, however, it is vital to test each component to the project. Testing was carried out on all aspects for the project. It was important to test each stage before building and moving on to the next one.

Research is key to any project, and the same goes for building a website which displayed parsed data. It was important to become knowledgeable in HTML, C#, .Net framework, SQL, and many more. All this research allowed for a greater understanding of the project and what was needed to succeed. Once the fundamentals for each resource was studied, the student could move on to more advanced techniques within each coding language or the next aspect within the project.

In terms of improvement for the project itself, a couple of features could be added to allow for a better and more fluid user experience overall:

- Search bar or search feature could allow for the user of the website to search for a registered unit, without using Google Chrome or Firefox’s built-in search feature.
- Possibly make a mobile phone app and readily available on the google play store, instead of just using a website to display the REMIT Update data.

References

[1] EIRGRID, “About EirGrid Group,” [Online]. Available: https://www.eirgridgroup.com/about/. [Accessed 13 11 2020].

[2] SONI, “About SONI,” [Online]. Available: https://www.soni.ltd.uk/about/. [Accessed 20 11 2020].

[3] F. Logisticks, “5 Reasons to Choose Microsoft Azure Cloud for Your Enterprise,” 29 04 2020. [Online]. Available: https://dzone.com/articles/5-reasons-to-choose-microsoft-azure-cloud-for-your. [Accessed 21 11 2020].

[4] S. Carey, “AWS vs Azure vs Google Cloud: What's the best cloud platform for enterprise?,” 23 01 2020. [Online]. Available: https://www.computerworld.com/article/3429365/aws-vs-azure-vs-google-whats-the-best-cloud-platform-for-enterprise.html. [Accessed 24 11 2020].

[5] e. l. D. M.-M. P.-M. and i. , “What is Azure Logic Apps,” 26 04 2021. [Online]. Available: https://docs.microsoft.com/en-us/azure/logic-apps/logic-apps-overview#:~:text=Logic%20Apps%20is%20a%20cloud,%2Dbusiness%20(B2B)%20scenarios.. [Accessed 04 12 2020].

[6] Draw.io, [Online]. Available: https://app.diagrams.net/. [Accessed 20 04 2021].

[7] Department for Business, Energy and Industial Strategy, “Digest of United Kingdom Energy Statistics 2020.,” [Online]. [Accessed 28 12 2020].

[8] Deloitte, “European Energy Market Reform Country Profile: Germany,” [Online]. Available: https://www2.deloitte.com/content/dam/Deloitte/global/Documents/Energy-and-Resources/gx-er-market-reform-germany.pdf. [Accessed 06 12 2020].

[9] Gas Networks Ireland, “Power Generation,” [Online]. Available: https://www.gasnetworks.ie/corporate/company/our-business/power-generation/. [Accessed 06 12 2020].

[10] Federal Ministry for Economic Affairs and Energy, “The Energy of the Future,” 11 2015. [Online]. Available: https://www.bmwi.de/Redaktion/EN/Publikationen/vierter-monitoring-bericht-energie-der-zukunft-kurzfassung.pdf?__blob=publicationFile&v=16. [Accessed 28 12 2020].

[11] Department for Business, Energy & Industrial Strategy, “DIGEST OF UNITED KINGDOM ENERGY STATISTICS 2020,” 25 - 30 07 - 07 2013 - 2020. [Online]. Available: https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/924591/DUKES_2020_MASTER.pdf. [Accessed 28 12 2020].

[12] EirGrid, “The Integrated Single Electricity Market Project,” [Online]. Available: https://www.eirgridgroup.com/customer-and-industry/i-sem/#:~:text=The%20SEM%20was%20established%20in,the%20world%20when%20it%20opened.&text=Free%20trade%20across%20borders%20is%20the%20foundation%20of%20the%20single%20European%20market.. [Accessed 04 01 2021].

[13] SEMOpx, “What can SEMOpx do for you?,” [Online]. Available: https://www.semopx.com/. [Accessed 04 01 2021].

[14] SEMOpx, “Day-Ahead Market,” [Online]. Available: https://www.semopx.com/markets/day-ahead-market/. [Accessed 04 01 2021].

[15] SEMOpx, “Intraday Auctions Market,” [Online]. Available: https://www.semopx.com/markets/intraday-market/. [Accessed 04 01 2021].

[16] SEMOpx, “Intraday Continuous Market,” [Online]. Available: https://www.semopx.com/markets/ict-market/. [Accessed 04 01 2021].

[17] Microsoft - Visual Studio, “Introduction,” [Online]. Available: https://tutorials.visualstudio.com/vs-get-started/intro. [Accessed 26 04 2021].

[18] Microsoft, d. D.-E. C. v.-k. and P. , “Download SQL Server Management Studio (SSMS),” 20 04 2021. [Online]. Available: https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver15. [Accessed 26 04 2021].

[19] W3Schools, “C# Tutorial,” [Online]. Available: https://www.w3schools.com/cs/. [Accessed 26 04 2021].

[20] W3Schools, “SQL Tutorial,” [Online]. Available: https://www.w3schools.com/sql/. [Accessed 29 04 2021].

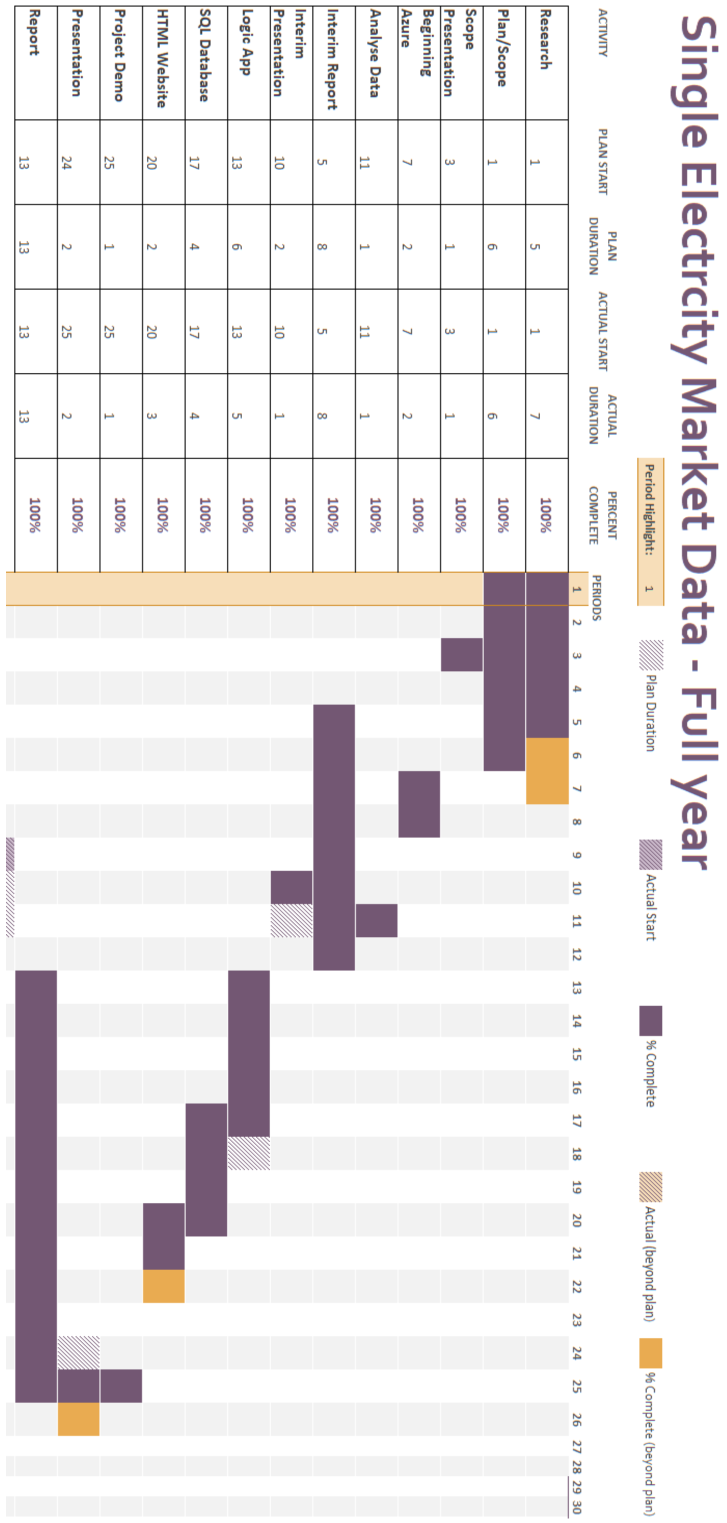
[2 W3Schools, “HTML Tutorial,” [Online]. Available: https://www.w3schools.com/html/. [Accessed 26 04 2021].

1
]

[22] NORDPOOL, “REMIT UMM,” [Online]. Available: https://umm.nordpoolgroup.com/#/messages?publicationDate=all&eventDate=nextweek. [Accessed 04 05 2021].

[23] NordPool, “Compliance - REMIT UMM,” [Online]. Available: https://www.nordpoolgroup.com/services/compliance/umm/. [Accessed 04 05 2021].

Appendix A Project Planning



ACTIVITY	PLAN START	PLAN DURATION	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE
Research	1	5	1	7	100%
Plan/Scope	1	6	1	6	100%
Scope Presentation	3	1	3	1	100%
Beginning Azure	7	2	7	2	100%
Analyse Data	11	1	11	1	100%
Interim Report	5	8	5	8	100%
Interim Presentation	10	2	10	1	100%
Logic App	13	6	13	5	100%
SQL Database	17	4	17	4	100%
HTML Website	20	2	20	3	100%
Project Demo	25	1	25	1	100%
Presentation	24	2	25	2	100%
Report	13	13	13	13	100%

Figure 88: Full Gantt chart.

Appendix B Salient Extracts from Project Diary

Semester 1:

Week 1: Introduction to the problem and the project supervisor, Marie Armstrong, was done over email due to the unforeseen circumstances in Ireland now. Research on the single electricity market in Ireland and Northern Ireland began.

Week 2: First Project meeting with project supervisor occurred on Microsoft Teams, weekly meetings were held at 10:00 A.M on Mondays. Discussions were had regarding SEMO, REMIT, and the electricity market.

Week 3: A scope presentation was held on Monday the 9th of November, the student presented on what the project is, what the student will do, technologies that will be required for completing the project, and the success criteria.

Week 4: The student began researching technologies required for the project, including what cloud-based platform will be used. Azure was chosen over AWS and Google Cloud, due to familiarity with the service and a student credit limit was offered.

Week 5: Interim report and Interim presentation began, the student prioritised writing the report for the remainder of the semester and decided to do the bulk of the coding for the project in semester 2.

Week 6: Research on other markets in Europe were conducted and added to the report. This enabled the student to write about and compare both Ireland and other markets across Europe, such as Germany and the UK. An Azure logic app was created, it demonstrated the basic concept of a logic app, this was recorded and added to the report.

Week 7 – 8: The student focused on writing the interim report, presentation and finalised some simple logic app testing using Microsoft Azure. The student also created a test SQL database, to learn SQL queries, such as selecting rows and creating tables.

Semester 2:

Week 1: A new timetable was given to the student, this meant that the student had a new time slot for project meetings, the new time slot was on Monday at 14:00 P.M. The student would meet with his project supervisor for the remainder of the semester at the scheduled timeslot.

Week 2 – 5: A new database was created, "semo". A new logic app was also created, this logic app gathered incoming emails from info@sem-o.com and converted the HTML to text. It would create variables of type string and integer, then would store the variables in the SQL database, "semo".

Week 6 – 8: The hosted website was created; this was done using Visual Studio using C# and HTML. The website was then published to Microsoft Azure as an app service, meaning it is now a hosted website using Azure,

Week 9 – 12: The report, presentation, and any fixes/alterations to the website was done within these weeks. Not many changes were implemented to the website. The primary focus was writing the report, creating the presentation, and doing the video demonstration for the project supervisor.

Appendix C Design Drawings & Component Specifications

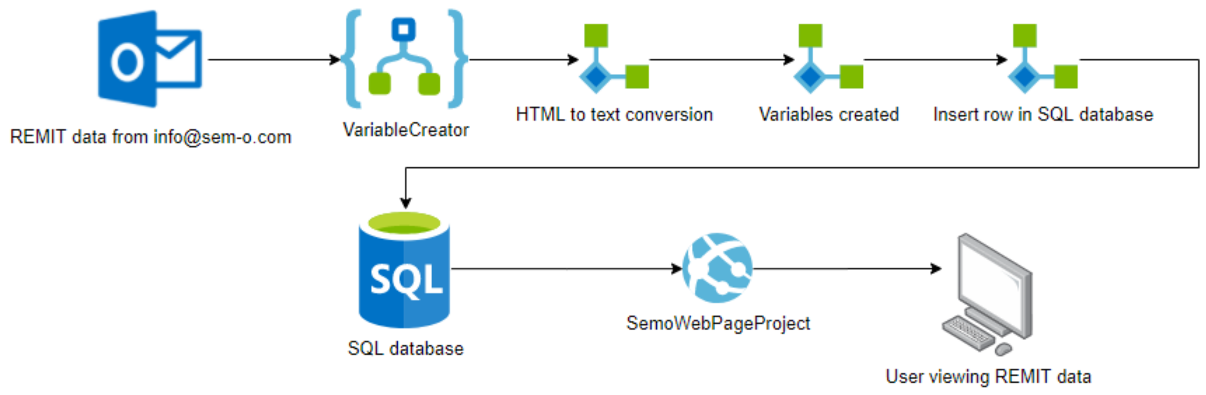


Figure 89: Plan for project.

A visual standpoint was required for the project. Above is a quick demonstration on how the user of the website would view the data, with all the previous steps included.

Appendix D List of Software Code

Logic App code:

This code was used for creating Variables in the logic app "VariableCreator".

```
lastIndexOf(variables('EmailPlainText'),'Affected Asset or Unit:')

lastIndexOf(variables('EmailPlainText'),'Resource Name:')

lastIndexOf(variables('EmailPlainText'),'Type of')

lastIndexOf(variables('EmailPlainText'),'Event Start:')

lastIndexOf(variables('EmailPlainText'),'Event Stop:')

lastIndexOf(variables('EmailPlainText'),'Available Capacity MW:')

lastIndexOf(variables('EmailPlainText'),'Remarks:')


substring(variables('EmailPlainText'),add(variables('AAOU'),23),sub(variables('RN'),add(variables('AAOU'),23)))

substring(variables('EmailPlainText'),add(variables('RN'),14),sub(variables('TOU'),add(variables('RN'),14)))

substring(variables('EmailPlainText'),add(variables('TOU'),23),sub(variables('EStart'),add(variables('TOU'),23)))

substring(variables('EmailPlainText'),add(variables('EStart'),12),sub(variables('EStop'),add(variables('EStart'),12)))

substring(variables('EmailPlainText'),add(variables('EStop'),12),sub(variables('AC'),add(variables('EStop'),12)))

substring(variables('EmailPlainText'),add(variables('AC'),22),sub(variables('Rem'),add(variables('AC'),22)))

substring(variables('EmailPlainText'),add(variables('Rem'),8))
```

WebForm1.aspx

```
<%@    Page    Language="C#"    AutoEventWireup="true"    CodeBehind="WebForm1.aspx.cs" Inherits="Semo_Data.WebForm1" %>


<!DOCTYPE html>


<html xmlns="http://www.w3.org/1999/xhtml">


<head runat="server">
<meta name="viewport" content="width-device-width, initial-scale=1.0" />
    <title>Azure App service designed using Microsoft Visual Studio 2017 Community Edition.</title>
<style>
    footer{
        text-align:center;
        padding:3px;
        background-color: white;
        color: black;
        display: block;
    }

    .tabular{
        margin: 0px;
        padding: 1px;
        background-color: #C0C0C0;
    }
    header{
        background-color: #FCF1A2;
        padding: 1px;
        text-align: center;
        font-size: 25px;
        color: #000000;
    }

    .tabular{
        margin: 5px;
        padding: 1px;
        background-color: #C0C0C0;
    }

    .tabular > h1, .table {
        margin: 5px;
        padding: 1px;
    }

    .table {
        background: #FFFFFF;
    }

    .table > h2, p {
        margin: 5px;
        font-size: 90%;
    }
</style>
</head>

<!-----------------------------------------------------------------------------------------
------->

<!--- This is a Break in the code, this is to distinguish the difference between head of
the html code and body of the HTML website --->
<body>
    <form id="form1" runat="server">
    <article class="tabular">
    <article class="tabular">
    <article class="table">
        <header>
            <h1 style="font-style: normal; font-family: Arial, Helvetica, sans-serif">SEMO
Data Web Page</h1>
        </header>
    </article>
    <article class="tabular">
        <article class="table">
        <nav>
            <li>
```

```
                <a href ="WebForm1.aspx">Database</a>
            </li>
            <li>
                <a href ="WebForm2.aspx">About Project</a>
            </li>
            <li>
                <a href ="WebForm3.aspx">Registered Users</a>
            </li>
        </nav>
    </article>
        <article class="table">
            <b>Click "Refresh Table Data" to display Database results!</b>
                            <asp:GridView ID="GridView1" runat="server" style="width:100%"
BackColor="White" BorderColor="#CCCCCC">
                            <FooterStyle BackColor="#CCCC99" ForeColor="Black" />
                            <HeaderStyle   BackColor="#FCF1A2"   ForeColor="Black"   Font-
Bold="true" />
                            <PagerStyle        BackColor="White"        ForeColor="Black"
HorizontalAlign="Center" />
                            <SelectedRowStyle     BackColor="#CC3333"     Font-Bold="True"
ForeColor="White" />
                            <SortedAscendingCellStyle BackColor="#F7F7F7" />
                            <SortedAscendingHeaderStyle BackColor="#484848" />
                            <SortedDescendingCellStyle BackColor="#E5E5E5" />
                            <SortedDescendingHeaderStyle BackColor="#242121" />
                            </asp:GridView>
                        </article>
                    </article>
    </article>
    </article>
        <p style="font-style: normal; font-family: Arial, Helvetica, sans-serif">
            <asp:Button ID="Button2" runat="server" OnClick="Button2_Click" Text="Refresh
Table Data" Style="background-color:#C0C0C0" />
        </p>
    </form>
</body>
<footer>
    <p  style="font-style:  normal;  font-family:  Arial,  Helvetica,  sans-serif">Project
Designer: Dean Ross<br />
```

```
        <a href="mailto:B00094969@mytudublin.ie" style="font-style: normal; font-family: Arial, Helvetica, sans-serif">B00094969@mytudublin.ie<br /></a>
        <a href="mailto:aircorpsrossi@gmail.com" style="font-style: normal; font-family: Arial, Helvetica, sans-serif">aircorpsrossi@gmail.com</a>
    </p>
</footer>
</html>
```

WebForm1.aspx.cs Code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data.SqlClient;
using System.Data;
namespace Semo_Data
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        private const string V = "Select Affected_Asset,Resource_Name,
Type_of_Unavailability, Event_Start, Event_Stop, Available_MW_Capacity, and Remarks from
Table Remit2";
        private bool isPostBack;


        protected void Button2_Click(object sender, EventArgs e)
        {
            bindRemit();
        }


        private void bindRemit()
        {
            var dataTable = new DataTable();


            SqlConnection semocon = new
SqlConnection("Server=tcp:semo.database.windows.net,1433;Initial Catalog=SEMO;Persist
Security Info=False;User
ID=dean;Password=Project21;MultipleActiveResultSets=False;Encrypt=True;TrustServerCertifi
cate=False;Connection Timeout=30;");
            {
                semocon.Open();


            using (var sqlCommand = new SqlCommand("Select * from Remit2", semocon))
            {
                using (var sqlReader = sqlCommand.ExecuteReader())
                {
                    dataTable.Load(sqlReader);

                    GridView1.DataSource = dataTable;
                    GridView1.DataBind();
                }
            }
        }
    }
        protected void Page_Load(object sender, EventArgs e)
        {
            if (!isPostBack)
            {

            }
        }
    }
}
```

WebForm2.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm2.aspx.cs" Inherits="Semo_Data.WebForm2" %>


<!DOCTYPE html>


<html xmlns="http://www.w3.org/1999/xhtml">


<head runat="server">
<meta name="viewport" content="width-device-width, initial-scale=1.0" />
    <title>Azure App service designed using Microsoft Visual Studio 2017 Community Edition.</title>
<style>
    footer{
        text-align:center;
        padding:3px;
        background-color: white;
        color: black;
        display: block;
    }

    .tabular{
        margin: 0px;
        padding: 1px;
        background-color: #C0C0C0;



    }
    header{
        background-color: #FCF1A2;
        padding: 1px;
        text-align: center;
        font-size: 25px;
        color: #000000;
    }

    .tabular{
        margin: 5px;
        padding: 1px;
        background-color: #C0C0C0;
    }

    .tabular > h1, .table {
        margin: 5px;
        padding: 1px;
    }

    .table {
        background: #FFFFFF;
    }

    .table > h2, p {
        margin: 5px;
        font-size: 90%;
    }
</style>

</head>

<!------------------------------------------------------------------------------------------
------->

<!--- This is a Break in the code, this is to distinguish the difference between head of
the html code and body of the HTML website --->
<body>
    <form id="form1" runat="server">

    <article class="tabular">
    <article class="tabular">
    <article class="table">
        <header>
            <h1 style="font-style: normal; font-family: Arial, Helvetica, sans-serif">SEMO
Data Web Page</h1>
        </header>
    </article>
```

```
        <article class="table">
        <nav>
            <li>
                <a href ="WebForm1.aspx">Database</a>
            </li>
            <li>
                <a href ="WebForm2.aspx">About Project</a>
            </li>
            <li>
                <a href ="WebForm3.aspx">Registered Users</a>
            </li>
        </nav>
    </article>
    <article class="table">
            <h2 style="font-style: normal; font-size:20px; font-family: Arial, Helvetica, sans-serif">Purpose of Project:</h2>
        <p style="font-style: normal; text-align:justify; font-family: Arial, Helvetica, sans-serif">
            To display REMIT Updates from info@sem-o.com on a user designed website. REMIT Update subscribers receive updates when changes
            occur to a regsitered unit. Such as electrical faults, these faults may not produce any MW capacity to consumers within a certain area.
            This data contains information regarding Affected Assets, Resource Name, Type of Unavailability, Events Start and Stop,
            Available Capacity MegaWattage and Remarks. To access all this information, subscribers would need to read through all their emails
            to gather all the information. This website makes use of two things, a logic app and an sql database. The logic app converts
            the HTML of an email into a text based format. This will allow for the logic app to create variables for the text and store
            the variables into an SQL database. This database has a connection to the Web Forms for both the REMIT Updates and Registered Units.
            This connection allows for the SQL database and its tables to display on the Website.
        </p>
    </article>
    <article class="table">
           <h2 style="font-style: normal; font-size:20px; font-family: Arial, Helvetica, sans-serif">Resources Used:</h2>
            <li style="font-style: normal; font-family: Arial, Helvetica, sans-serif">Microsoft Azure.<br /></li>
            <li style="font-style: normal; font-family: Arial, Helvetica, sans-serif">Microsoft SQL Server 2019.<br /></li>
            <li style="font-style: normal; font-family: Arial, Helvetica, sans-serif">Microsoft Visual Studio Community Edition 2017.<br />
    </article>
    <article class="table">
            <h2 style="font-style: normal; font-size:20px; font-family: Arial, Helvetica, sans-serif" >Why use Azure?</h2>
            <li style="font-style: normal; font-family: Arial, Helvetica, sans-serif">Azure was chosen for multiple reasons: <br /></li>
            <li style="font-style: normal; font-family: Arial, Helvetica, sans-serif">A 12-month free trial was granted to students. This included a $200 credit limit. <br /></li>
            <li style="font-style: normal; font-family: Arial, Helvetica, sans-serif">Azure was used in a module studied by the student, making it familiar to use.<br /> </li>
            <li style="font-style: normal; font-family: Arial, Helvetica, sans-serif">Offers both a large service of choices and is very customisable. <br /> </li>
            <li style="font-style: normal; font-family: Arial, Helvetica, sans-serif">Azure is both Windows and Linux compatible. <br />
        </article>
    </article>
    </article>
    </form>
</body>


<footer>
    <p style="font-style: normal; font-family: Arial, Helvetica, sans-serif">Project Designer: Dean Ross<br />
        <a href="mailto:B00094969@mytudublin.ie" style="font-style: normal; font-family: Arial, Helvetica, sans-serif">B00094969@mytudublin.ie<br /></a>
        <a href="mailto:aircorpsrossi@gmail.com" style="font-style: normal; font-family: Arial, Helvetica, sans-serif">aircorpsrossi@gmail.com</a>
    </p>
</footer>
</html>
```

WebForm3.aspx

```
<%@    Page    Language="C#"    AutoEventWireup="true"    CodeBehind="WebForm3.aspx.cs"
Inherits="Semo_Data.WebForm3" %>


<!DOCTYPE html>


<html xmlns="http://www.w3.org/1999/xhtml">


<head runat="server">
<meta name="viewport" content="width-device-width, initial-scale=1.0" />
    <title>Azure  App  service  designed  using  Microsoft  Visual  Studio  2017  Community
Edition.</title>
<style>
    footer{
        text-align:center;
        padding:3px;
        background-color: white;
        color: black;
        display: block;
    }

    .tabular{
        margin: 0px;
        padding: 1px;
        background-color: #C0C0C0;



    }
    header{
        background-color: #FCF1A2;
        padding: 1px;
        text-align: center;
        font-size: 25px;
        color: #000000;
    }

    .tabular{
        margin: 5px;
        padding: 1px;
        background-color: #C0C0C0;
    }

    .tabular > h1, .table {
        margin: 5px;
        padding: 1px;
    }

    .table {
        background: #FFFFFF;
    }

    .table > h2, p {
        margin: 5px;
        font-size: 90%;
    }
</style>

</head>

<!-------------------------------------------------------------------------------------------
------->

<!--- This is a Break in the code, this is to distinguish the difference between head of
the html code and body of the HTML website --->
<body>
    <form id="form1" runat="server">
    <article class="tabular">
    <article class="tabular">
    <article class="table">
        <header>
            <h1 style="font-style: normal; font-family: Arial, Helvetica, sans-serif">SEMO
Data Web Page</h1>
        </header>
    </article>
        <article class="table">
        <nav>
            <li>
                <a href ="WebForm1.aspx">Database</a>
            </li>
            <li>
                <a href ="WebForm2.aspx">About Project</a>
            </li>
            <li>
                <a href ="WebForm3.aspx">Registered Users</a>
            </li>
        </nav>
    </article>
    <article class="table">
            <h2  style="font-style:  normal;  font-family:  Arial,  Helvetica,  sans-
serif">Purpose of this Page:</h2>
        <p style="font-style: normal; font-family: Arial, Helvetica, sans-serif">
            The purpose of this page is to display the units that are registered under the
trading and settlement code dated 02/03/2021
        </p>
    </article>
    <article class="table">
        <b>Click "Display Units" to display Database results!</b>
                            <asp:GridView ID="GridView2" runat="server" style="width:100%"
BackColor="White" BorderColor="#CCCCCC">
                            <FooterStyle BackColor="#CCCC99" ForeColor="Black" />
                            <HeaderStyle   BackColor="#FCF1A2"   ForeColor="Black"   Font-
Bold="true" />
                            <PagerStyle        BackColor="White"        ForeColor="Black"
HorizontalAlign="Center" />
                            <SelectedRowStyle     BackColor="#CC3333"     Font-Bold="True"
ForeColor="White" />
                            <SortedAscendingCellStyle BackColor="#F7F7F7" />
                            <SortedAscendingHeaderStyle BackColor="#484848" />
                            <SortedDescendingCellStyle BackColor="#E5E5E5" />
                            <SortedDescendingHeaderStyle BackColor="#242121" />
                            </asp:GridView>
    </article>
    </article>
    </article>
    <p style="font-style: normal; font-family: Arial, Helvetica, sans-serif">

        <asp:Button ID="Users" runat="server" OnClick="Button_User" Text="Display Units" Style="background-color:#C0C0C0" />

    </p>

    </form>

</body>

<footer>

    <p style="font-style: normal; font-family: Arial, Helvetica, sans-serif">Project Designer: Dean Ross<br />

        <a href="mailto:B00094969@mytudublin.ie" style="font-style: normal; font-family: Arial, Helvetica, sans-serif">B00094969@mytudublin.ie<br /></a>

        <a href="mailto:aircorpsrossi@gmail.com" style="font-style: normal; font-family: Arial, Helvetica, sans-serif">aircorpsrossi@gmail.com</a>

    </p>

</footer>

</html>
```

WebForm3.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data.SqlClient;
using System.Data;


namespace Semo_Data
{
    public partial class WebForm3 : System.Web.UI.Page
    {
        private const string R = "Select Party_ID, Party_Name, Participant_Name,
Resource_Name, Resource_Type, Fuel_Type, etc from List_of_Registered_Units";
        private bool isPostBack;


        protected void Button_User(object sender, EventArgs e)
        {
            bindUser();
        }
        private void bindUser()
        {
            var userTable = new DataTable();


            SqlConnection usercon = new
SqlConnection("Server=tcp:semo.database.windows.net,1433;Initial Catalog=SEMO;Persist
Security Info=False;User
ID=dean;Password=Project21;MultipleActiveResultSets=False;Encrypt=True;TrustServerCertifi
cate=False;Connection Timeout=30;");
            {
                usercon.Open();


                using (var sqlCommand = new SqlCommand("Select * from Registered_Units",
usercon))
                {
                    using (var sqlReader = sqlCommand.ExecuteReader())
                    {
                        userTable.Load(sqlReader);
                        GridView2.DataSource = userTable;
                        GridView2.DataBind();
                    }
                }
            }
        }
        protected void Page_Load(object sender, EventArgs e)
        {
            if (!isPostBack)
            {

            }
        }
    }
}
```