# 4. Experiment Development

## 4.1. Introduction

This chapter goes into the development details of NDMA. This was from developed architecture, the designed layout and the tools used for the application functionality. It also delves some of the development reasons regarding the feature requirements of the application. It also describes the technical hurdles that found during the development lifecycle itself. Once these difficulties were found, what actions were taken to resolve them are discussed too. The URL for NDMA GitHub repository is <https://github.com/WilliamCareySemiColon/NDMA/tree/MasterPathVersionThree>.

Aside from this documentation, there is a descriptive readme file on the repository. It goes into the various details of NDMA connection. This includes which activity classes are connected to which layout files. It also talks of the external systems which are used for the application. The readme was designed to assist in the reviewing of the software. The reviewing of the software could need extra clarification, when reviewing NDMA. It was cover some of the documentation the developer may have missed themselves.

The brand names for usage of the application was Edamam Food Database and Syncfusion. Both are external resources used in NDMA and are privately owned. The reason behind Edamam food database was access to the nutritional information and the associated imagery. The images would allow for better selection of the food products for user to select from. Edamam also offered standalone basic access to the application services for free too. The reason behind Syncfusion system is it was the only platform that supported Xamarin Android in development. To expand, there was no other platform that allowed chart developed that allowed a free timeframe, or that offered free student package.

Both systems required a license or identifiers to access their resources. This was why user profiles were created on both websites for NDMA requirement purpose. Without the use of the license, both the food database and the chart system could have not been used for the purpose of NDMA. Both were essential requirements for NDMA. The link for Edamam website and documentation can be found at the following website: <https://developer.edamam.com/> . The link for Syncfusion website and documentation can be found at the following website: <https://www.syncfusion.com/> .

## 4.2. Software Development

## 4.2. Software Development

The development of NDMA’s software was constructed under a plan that was continuous reviewed for various reasons. Such reasons will be elaborated under the different key sections which they are applicable. NDMA’s architecture was developed under the Agile Feature Prototyping methodology. Agile Feature methodology was created to maximise meeting the requirements of NDMA.

The architecture itself was a Three Tier Client Server architecture. The three tiers of NDMA consisted of the presentation layer, application layer and the database layer. These layers were devised to separate out the functionality, the presentation and the data aspects of NDMA. These were separated to ensure each layer didn’t interfere with any of the other two layers functionality.

The Client Server architecture was divided into two different responsibilities. The client-side role was to interact with the user. This would be either getting input from the user to complete a task, or getting data from external servers and displaying such information to the user. The server side of the application, in the case of NDMA, is remote data storage which is accessed by the application. The remote storage data resulted from either the cloud database or an external API.

In the case of the three tiers, the plan was to use different technologies for each tier. In the presentation tier, the technology of Xamarin was used. In the application layer, the .Net framework was employed. Finally, the database layer was handled by the Azure Cloud Resources remotely and SQLLite locally. Both are to be synced in order to maximise user experience regarding their data usage.

The actual technology for NDMA was using Xamarin only. Some parts of the middleware were developed using pure C# language, not the .Net framework. The localhost variety of data layer was also not developed. While the creation of the cloud environment was successful, other struggles meant it could not be integrated into the application properly. All the individual reasons for the breakdown of each segment will be discussed in their appropriate area. This means the reasoning behind the cloud environment, not properly integrated into the system, will be discussed in the data layer section.

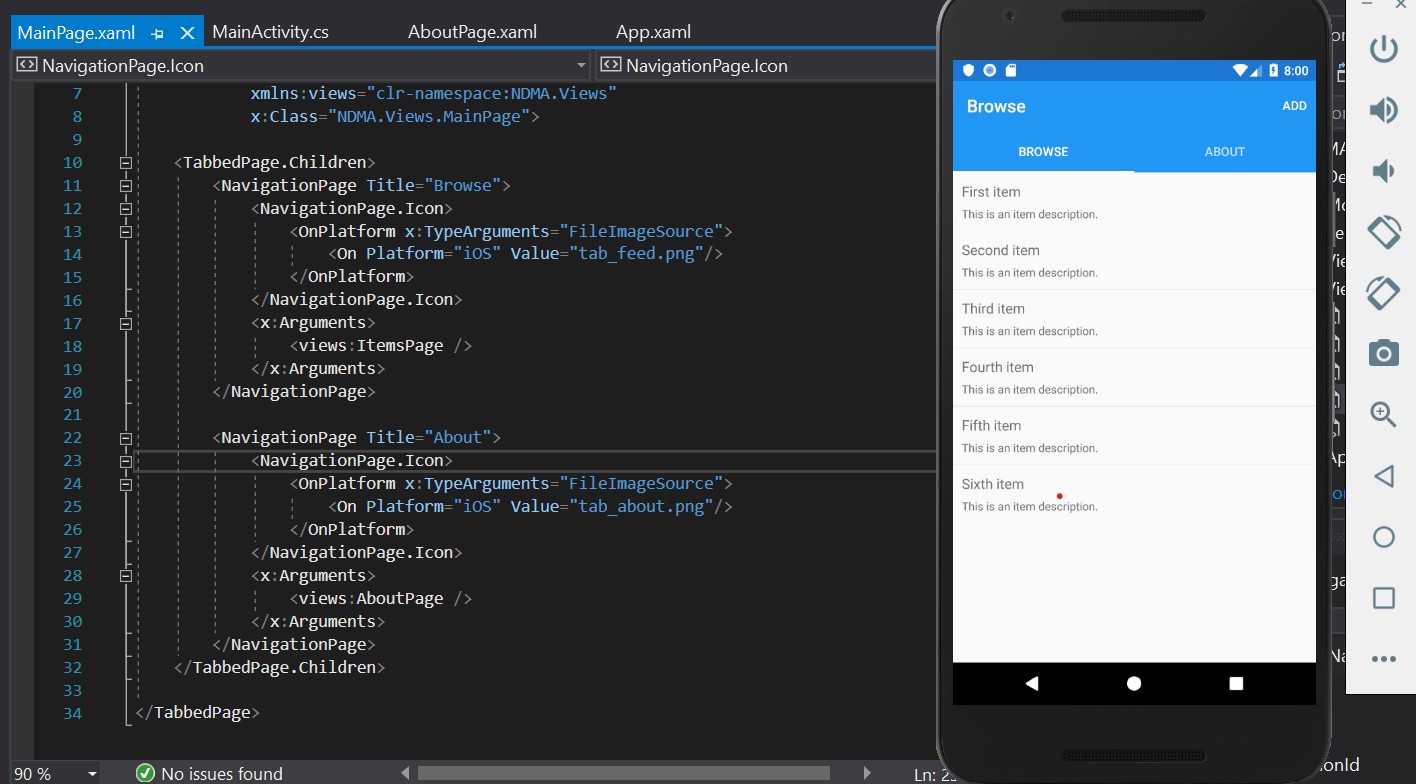
To ensure the application functioned properly, test data were created and used inside the system. This external data would be inside its own environment and accessed by the activity classes, or Application layer classes, to ensure the functionality were as designed.

The GitHub environment was used as the source control for the application. This is to ensure the ongoing development of the application can be tracked. Errors and Features can be developed in intervals as a result of the source control. One problem was found from using GitHub as the source control. When a new feature is conceptual designed, a new branch would be created from the source control to develop the feature. When attempting to merge the new branch feature, the project would unload completely. The only way to make the application work was to move to a different branch, revert to a previous version of the application and continue from there.

**Before Christmas Design**

The was no development that was really built before Christmas. This was due to the pressure from other college assignments and further research on NDMA which was needed. As a result, the was only a throwaway prototype which was developed for the interim presentation. The research was conducted due to the lack of understanding of NDMA and its scope at the time. The prototype was developed in a few days. The prototype concept was derived from the default Xamarin types of the application. It was furthered developed to display the concepts of NDMA requirements.

The research was split into two areas. The first was how to properly tackle NDMA requirements. This would be the technical requirements and associated feature requirements. The second was the collected datasets used inside the project. The original decision was to make predicative analysis a main feature for NDMA. This was soon to be discovered to not work as there was no dataset which produced result for analysis. The only datasets that was publicly available was food types. As a result, the decision was made to change the complexity area from predicative analysis to UI design.



**After Christmas Design**

Through January and February, the dataset was continued to be investigated. This was due to the uncertainty of which best suited NDMA requirements. None contained imagery which was needed for NDMA UI design. It was also at this stage, the requirements needed for the application, and how to implement them. Due to the timing constraint, there wasn’t time to learn how to use Xamarin Forms Technology. The technology was changed to Xamarin Android instead. Xamarin Forms was the initial chosen technology. This was due to the framework allows development on multiple platforms (iOS, Android) by sharing the same code base. Xamarin Android was more familiar technology but only develops on Android platforms. This allowed immediate development to occur for NDMA.

After the technology was chosen, Endaman API was discovered. This allowed access to a complete food database that contained images. Because the data was cleaner and more usable for NDMA, it was selected over the dataset contents. The cloud environment Azure started to provide trouble at this stage. This was with the student subscription used at the time. Reading the package led to misunderstandings of the use of Azure services. The products were consuming the credit at a faster rate then expected. The cloud team were conducted about these issues. However, they provided no assistance themselves due to the timing interval they replied to my requests. They seemed to be based inside America as they contacted me during early hours of morning.

As a result, research into AWS happened for their services. A need for a credit card for the hesitation to accept cloud resources assistance. This was required for Azure too. The choice was to return to Azure as there was more familiarity with their technology in the given moment. The development was continued for NDMA from that point onwards. The last issue discovered was all the requirements needed for NDMA to be professional completed would have taken a team a year or two to implement. The project at the time only had over a month to complete, so a lot of the desired decisions for NDMA had to be dropped. This was so the functionality of NDMA can be completed.

## 4.3. Front-End

The Front-End of the application is one of the main focuses of NDMA development. The chosen area of complexity is the user interface for NDMA. This choice came from evaluating two both industry applications. The two applications evaluated were Fitbit and MySugr. These applications were evaluated using the 7 modern mobile usability and the 10 Neilson’s heuristics.

Both applications have one area in which could reduce the user experience. This was the exact measurements as textual input for the logging purpose. The logged details for Fitbit are the specifics of the food consumed. This included: name, brand, amount served, calories and different nutrient composites used (a combination of Macronutrients and micronutrients). The MySugr logged details are blood sugar, insulin, food types, type of food time used (breakfast, lunch, dinner etc) and much more.

This would indicate the targeted user base are people whom require exact measurements for the logging purpose. For other users, this is not user friendly to them. This is different for NDMA, as the target user base does not need to input the exact measurements. Instead, an approximate value system would be inputted using buttons and dropdown options. This would ensure the UI of NDMA logging system more friendly for those whom don’t need exact measurements. This would be the contrast to the two professional applications.

Fitbit and MySugr were scored high in the other areas of evaluation. In such, ideas were borrowed from them too for NDMA UI. An example was the chart system that was used to display the logged trends. This would be combined with the design of the advisor system to develop NDMA. Both will be used to provide an in-depth analysis of the nutritional logged input provided by the users.

The UI Design layout for NDMA is simple and consistent throughout the application. The looks of the buttons, the overall look and the colour scheme are the same throughout NDMA. This is to ensure minimal learning curve for the users of the application itself.

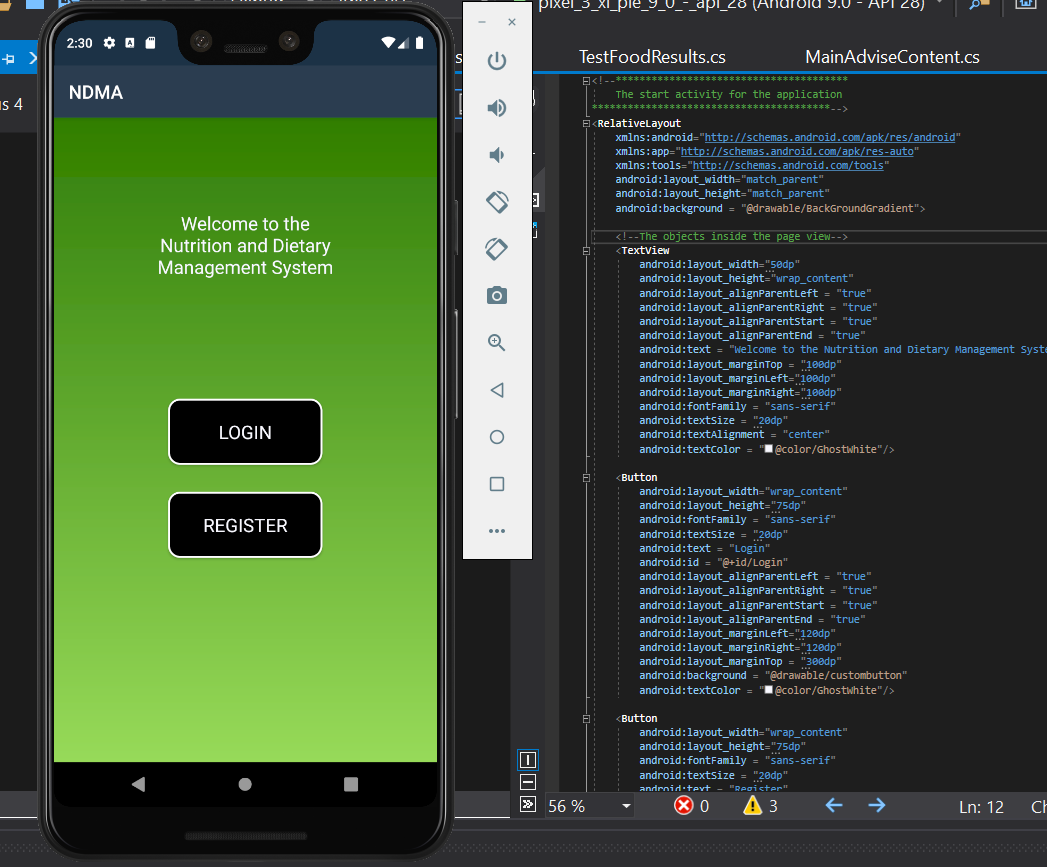
The technology used for the layout are Xamarin XML activity files. This was chosen to use modern comprehension tools to develop the application efficiently (For example, button means button for the application development). The two types of layout files are Relative Layout or linear layout, which were interchangeable depending on the layout requirements of the associated activity.

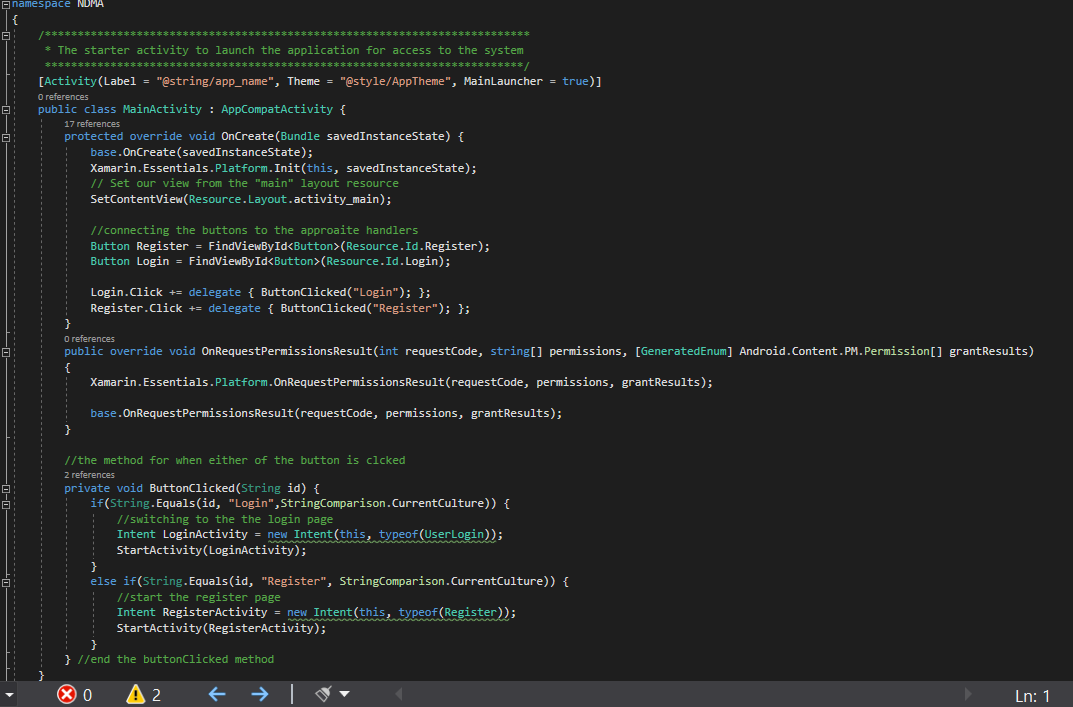
The technology used to assist the XML files are the activity classes. These classes are what makes NDMA interactable with the users. The functions of these classes are like controllers in terms on getting the requests and carrying them out. This is from getting user input or fetching data from internal or external services / resources. Another function of the activity classes is to navigate from page to page within NDMA. For the layout look of the buttons and the background looks, resources are shared within NDMA. This is to ensure the application looks are consistent across the pages.

Within these navigations inside NDMA, some activities can communicate with one another using special method. These special methods known as “*StartActivityForResult”* and *“OnActivityResult”*. Each activity class connects with activity layout files to fully interact with the user. The activity classes were developed using C# programming language with Xamarin Android technology. In this section, each of the layout files, and their activity classes, are reviewed. The functionality, look, files and resources of each page will be discussed too.

**XML layout file for registering and login into NDMA.**

The role of this activity is to capture the input of the user whether they want to login into the application or register. This is by appending event handlers to the two buttons displayed below. Once clicked, it will redirect to the activities responsible for letting the person login or register into the application.

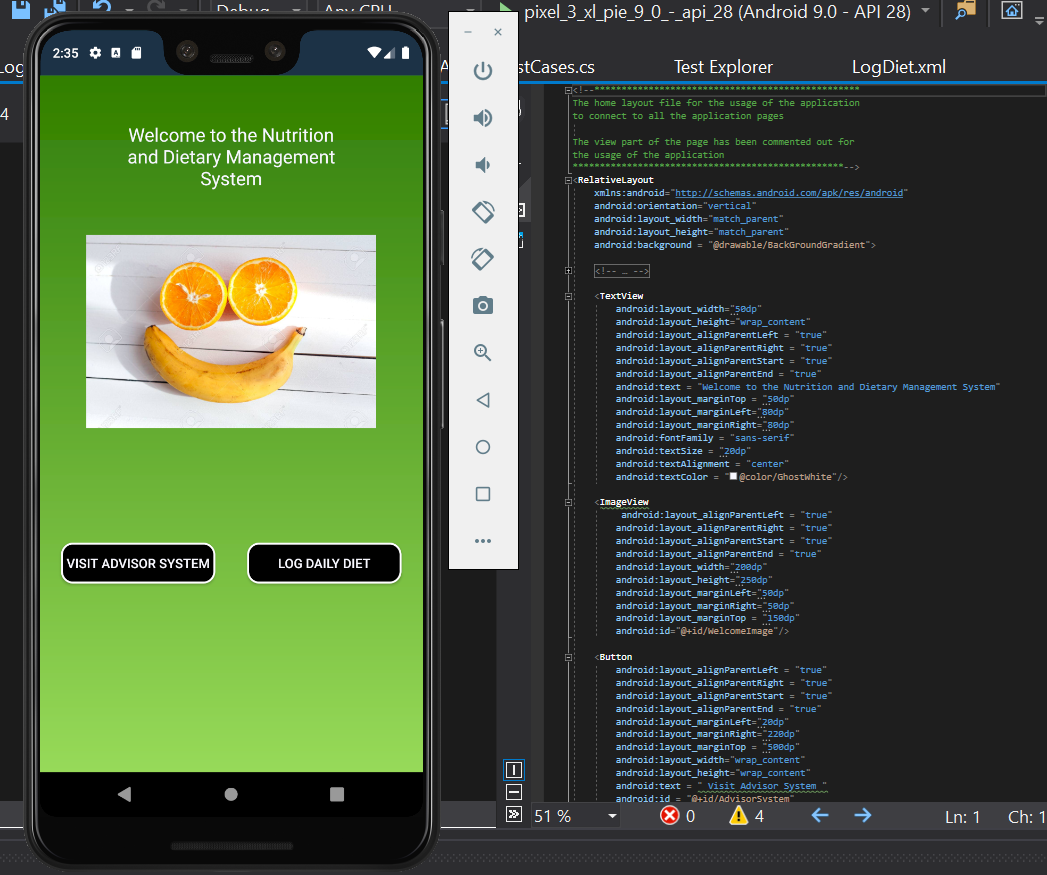




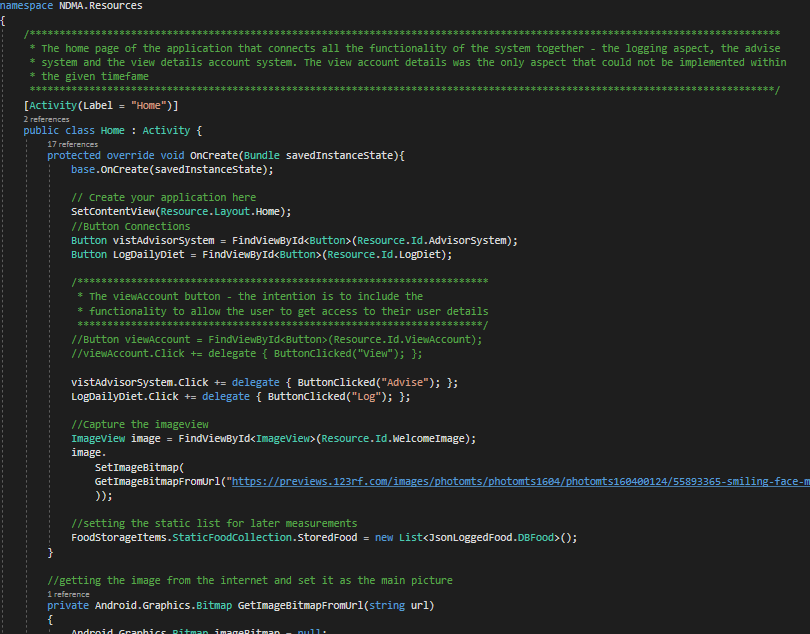
**The home screen of NDMA layout and Code**

The role of this activity is to provide access to both the logging system and the advisor system. This is completed by setting the access routes to either system through the buttons provided. There was a plan to create the “View Account Details”, “Log Daily Diet” and “Visit Advisor System” buttons. Only “Log Daily Diet” and “Visit Advisor System” buttons were created, as timing constraints and development requirements of both the logging system and the advisor system, the “View Account Details” button was not furthered developed and left out of NDMA. The “Log Daily Diet” button provides access to the logging system. The “Visit Advisor System” provides access to the advisor system.

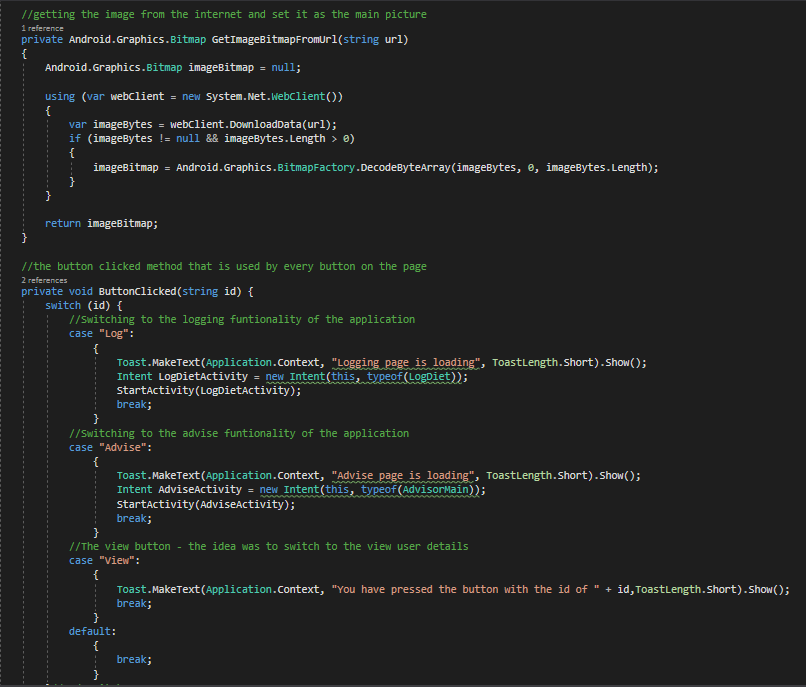
The image of NDMA is pulled from the url : <https://previews.123rf.com/images/photomts/photomts1604/photomts160400124/55893365-smiling-face-made-from-fruit-on-a-wooden-background-fruit-smile-two-halves-of-oranges-and-one-banana.jpg> . The purpose is to provide a welcome face to the user when they have logged into the application. The image is pulled through the method *GetImageBitmapFromURL* which takes a URL which is believed to be an image and convert it to bitmap format. This would allow it to be displayed on the application page.



**The setup of the home page, to access the logging system advisor system and setting the picture into the background of the application inside NDMA.**



**The methods to download the image from the WWW and the button methods to redirect to the advisor system or the logging system of NDMA.**

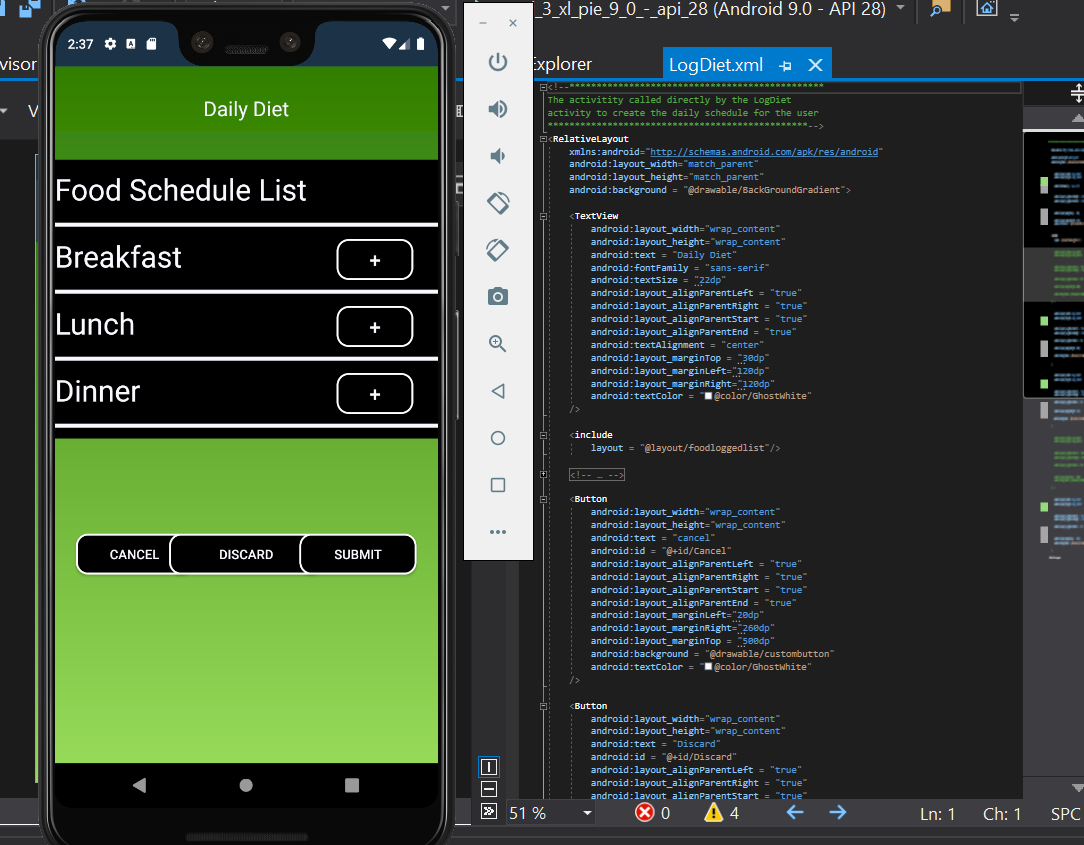


**The food schedule list as part of the Logging system of NDMA**

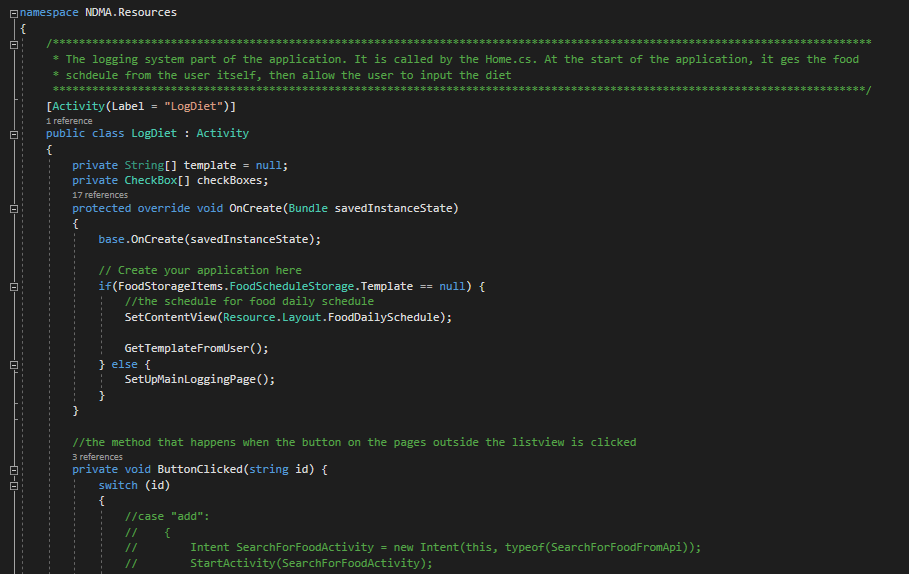
This activity provides the role of creating the food schedule for the user itself. This is conducted by a checkbox selection of the food schedule. This schedule process gets each item desired on the major food time phase selection that is checked off. Once the food schedule is created, the layout below is created. This allows a few functionalities. The visible buttons are the Cancel, Discard and Submit buttons are available. There are a number of “+” buttons, which allows the food to be logged into the current section. An example is logging breakfast into the “Breakfast” section. The difference between the cancel, and the discard button, is cancel would remove the current unsaved changes. The discard button discards all the changes. There was another button which was in the planned developments, known as save. The save button would save the current changes but not submit the log to the database. The save button was not developed due to the development complexity that would have been taken which was outside the scope. This would be stored on the database as the temporary storage. This would have be pulled and updated until the food is submitted or discarded.

The plus button, when clicked, will redirect to the search API page. When it returns to this page eventually, the food name gets stored beside the section in which they have been logged into. This version of the logging system can only log food at the time with this version. This is due to the higher development complex when more would be created. This development complex is outside the development scope of NDMA.

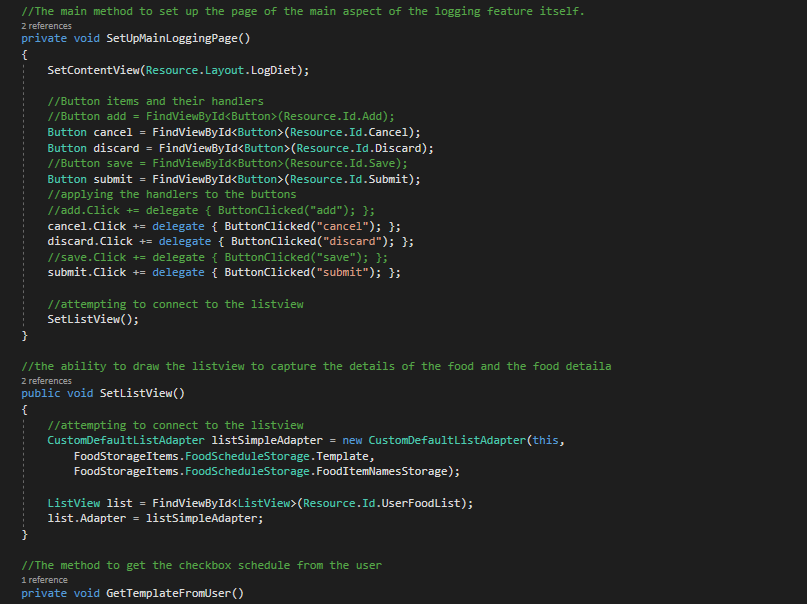
**The layout page of the food schedule**



**The process of the logging schedule for NDMA**



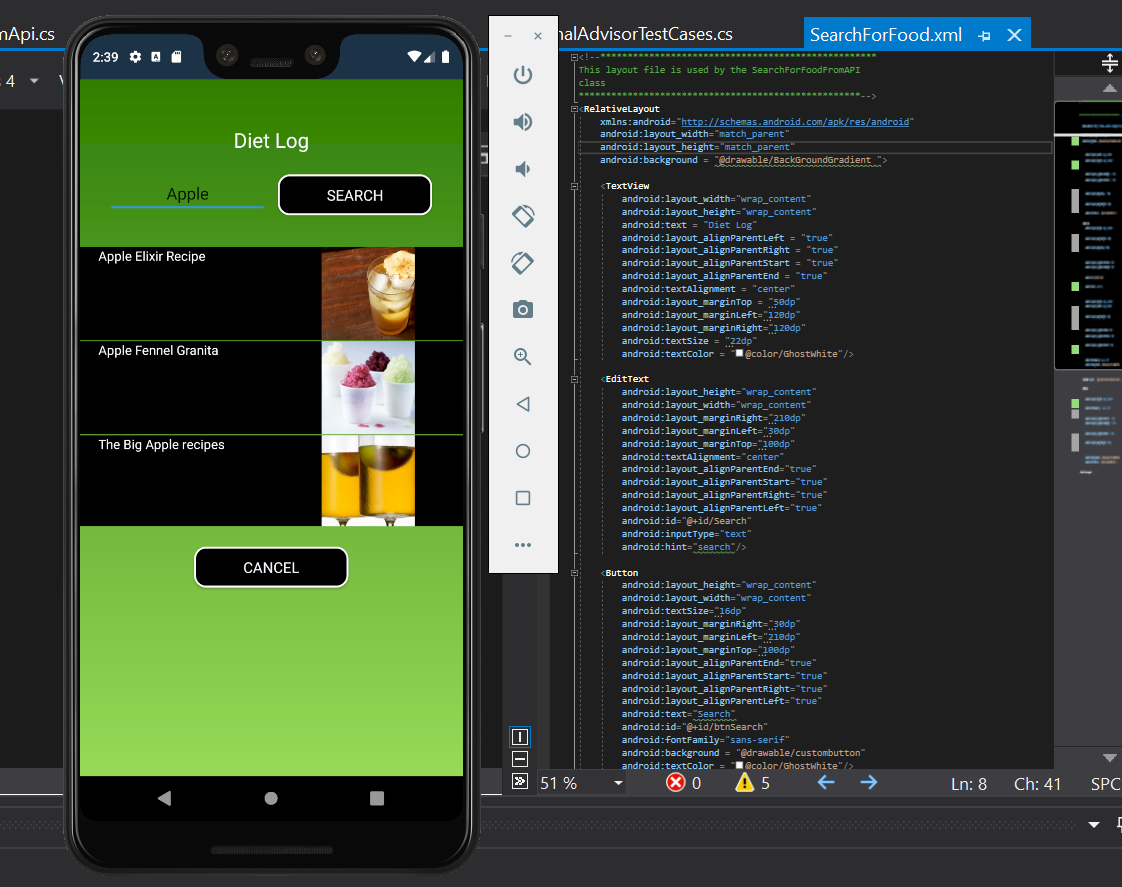
**The logging page setup code NDMA**



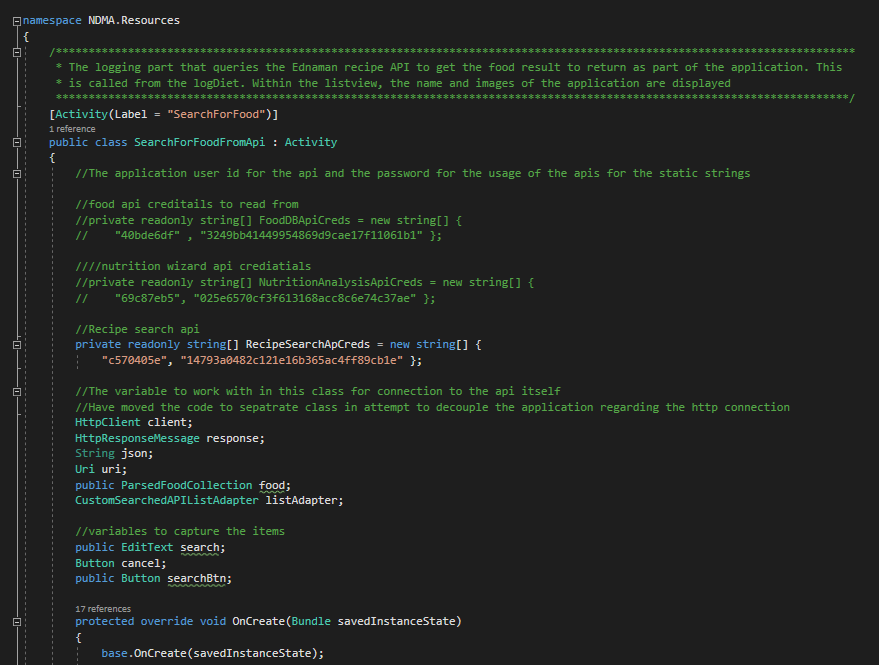
**The Search API Activity of NDMA**

This activity is designed to query the Ednamam API for the food products. This is called after the + button on the Food Schedule page. When initially loaded, only the edit text beside the Search Button and the cancel button is visible. This is until a food in inputted into the edit text section and the search button is clicked. This will search the API database for a collection of the food products and return them to the list view. The word used for the search comes from the edit text field. The displayed products details are the image and name of the product. The list view will display the different choices to the user to scroll through. Once one of them is clicked, the user will see all the ingredient of the food product. If the user accidently picks a food product, they can return to this page and select a different one. If they selected the right one, they can log that product.

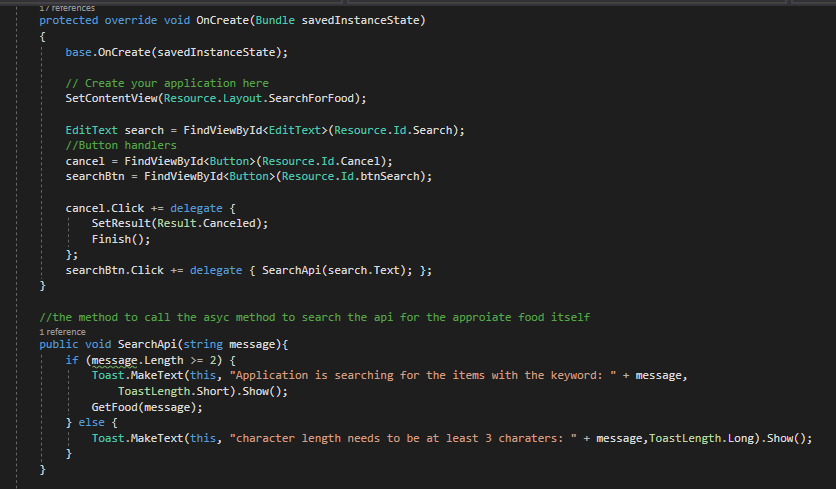
Using the Ednamam API required an App id and app key. This was retrieved by signing up to Ednamam to get access to the id and key. They were inputted into the code to get access to the database for ease of use for the application. The main food database is the recipe food database. It was the only one that uses imagery as part of the storage. This was part of the development requirement of NDMA to ease the user experience for the users.



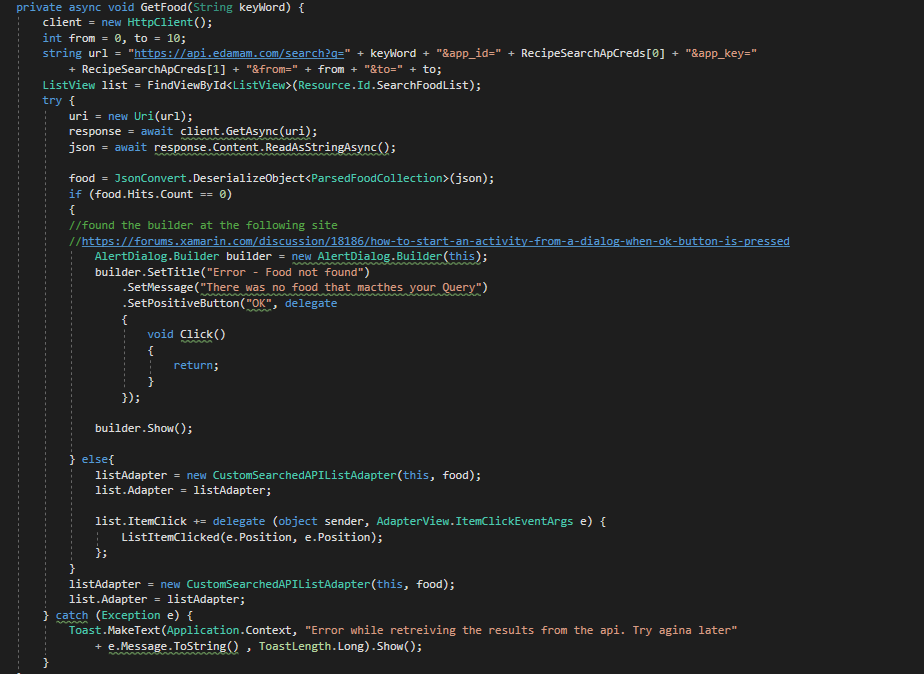
**The variables and credentials of the search for food activity class for NDMA**



**The setup of Search For Food API in NDMA**



**The search method to get the food data from the API link and set it into the list view. If it doesn’t exist, alert the user though an AlertDialog.Builder message and a clickable message to return tp the activity.**



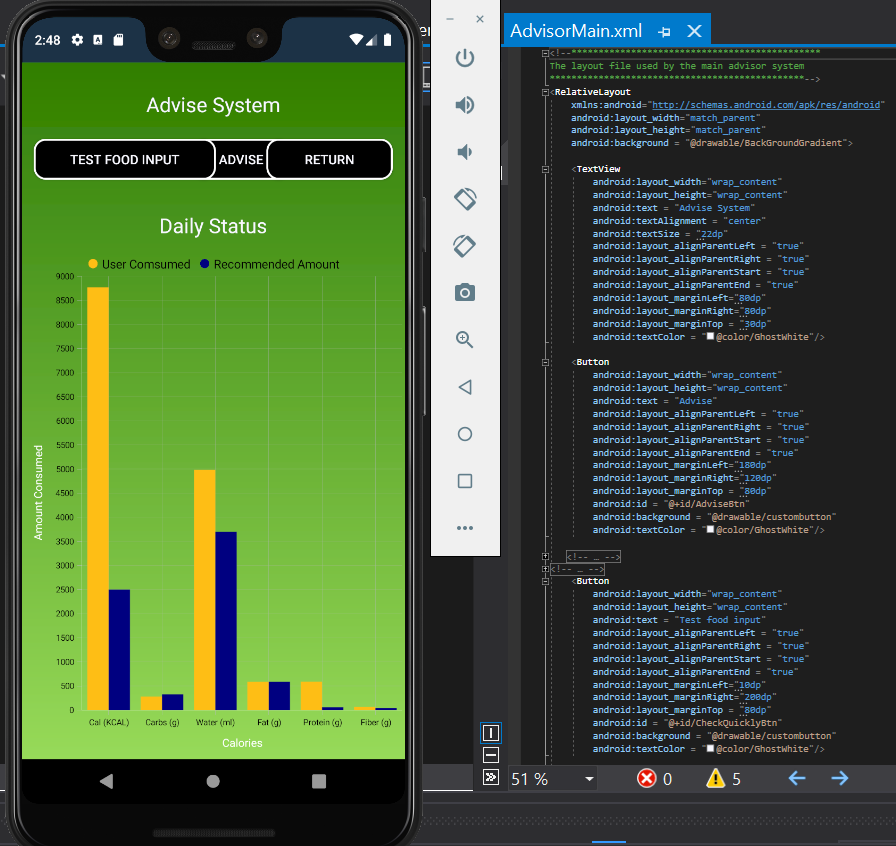
**The Advisor System of NDMA**

The advisor system activity to analysis the nutritional information from the user. This is by getting the logged input, pass it into the nutritional advisor and getting feedback from it. It also gets the recommended data from the nutritional advisor. This would be used as a comparison with the logged data for the chart reading.

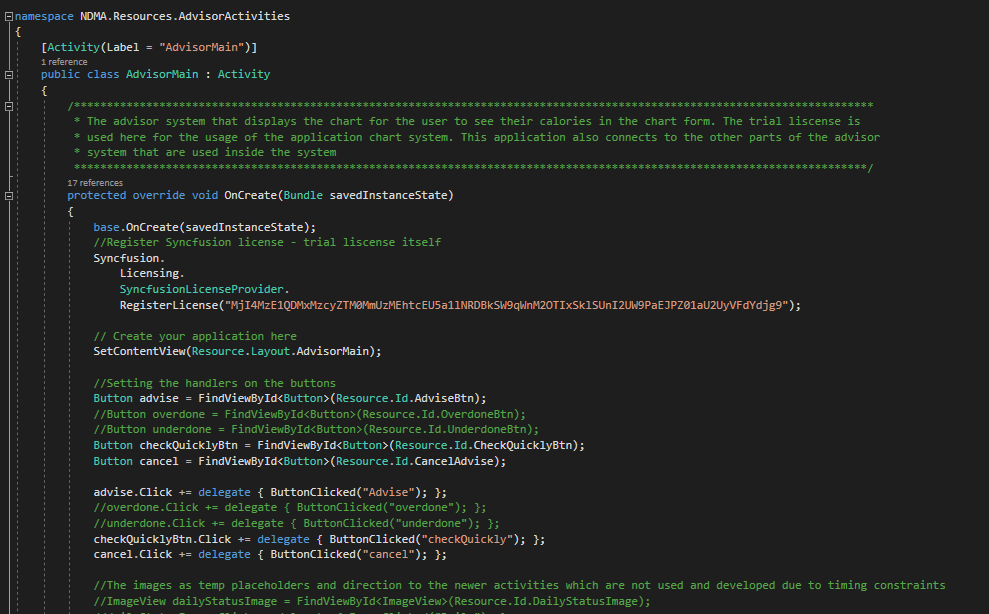
The page itself has three buttons and a bar chart. The three buttons each have a different functionality. The “Test Food Input” button allows the user to test some food data against the advisor system. This is without affecting the currently logged data. The advice button gets the textually advice based off the currently logged data. The return button returns to the home page. The chart displays the macronutrient contents of the inputted food data vs what is recommended.

The chart system is from an organisation known as SyncFusion. To use their services, a license must be acquired by registering for their services. There is then a trial license for 30 days or different types of licenses that can be purchased. For this activity, the license is register before any work on the chart begins to ensure it works efficiently.

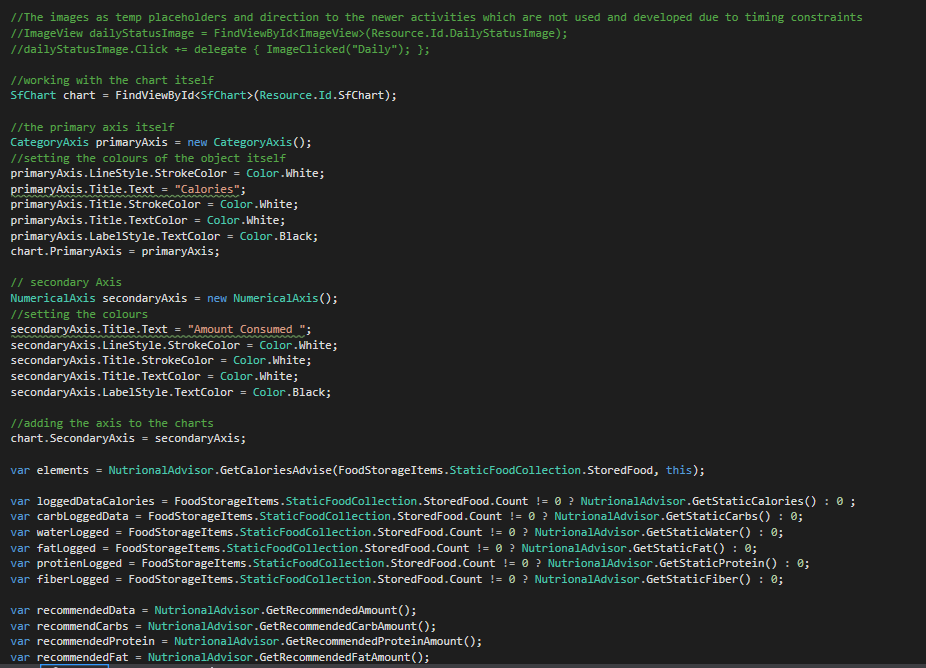
**The layout of the advisor system of NDMA**

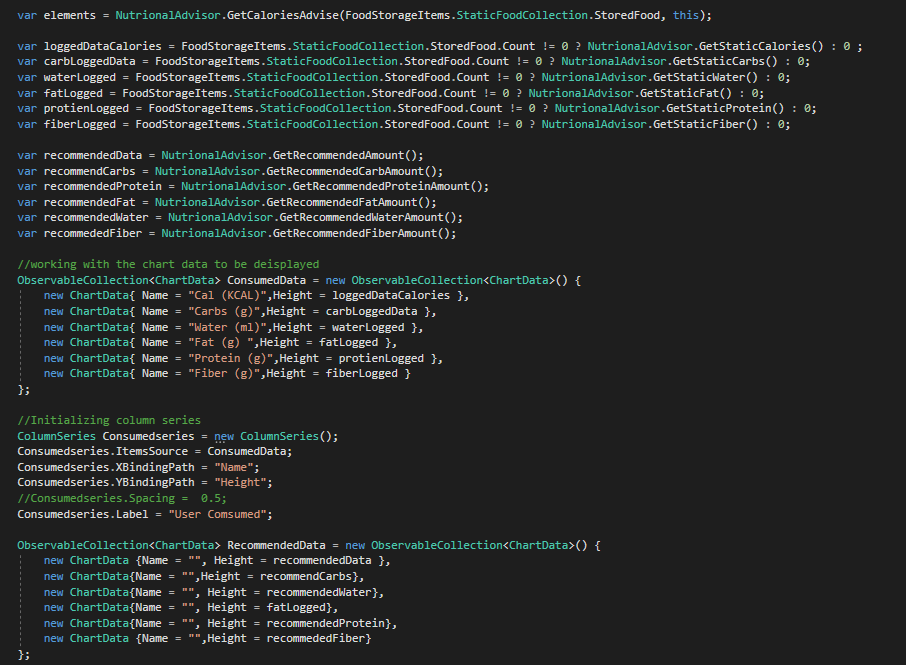


**The Advisor system activity class and the register of the license and setting the button handlers**



**The development of the chart system from the advisor system into the advisor activity class**





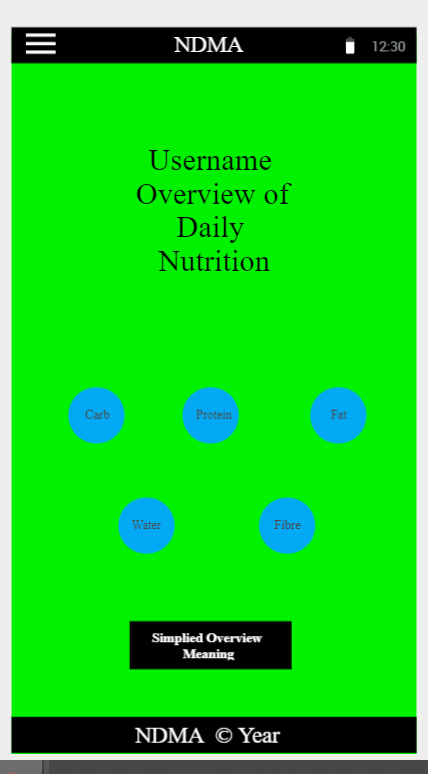
**The user interface functionality**

The design of NDMA front view is to capture the user logged food input through the simple interface. This is done by navigating to the logging system. Through this system, the person searches for the food product which they would like to log. This product is captured from an external API and returned to the mobile application for the user to select. Once the food is logged, the user can then navigate to the advisor system to get feedback on their nutritional input. The advice provided back is both textually and graphically, where the graphically element is both charts and imagery.

One of the desired items as part of the development of NDMA was to create a navigational bar. This would allow the user to access to the different parts of the systems at a quicker speed. The issue that came with the attempt of developing this was breaking NDMA functionality. This would leave the application unusable. This was found after spending a few days following an example project which had implemented the same feature. After trying twice and breaking the application at each time, the application had to be rolled back. This combined with the time needed to implement NDMA’s other features meant this task couldn’t have been carried out.

One other task that found similar issue was with the method that queried the Edamam API. The concept was to move this from the controller application to an external static class. This was to attempt to remove the dependency of the controller class to the search method. This would enable to use the same code for any parts of the application that needed access to it. When this was fully developed and initially tested, the method didn’t work in the expected way. Due to other feature requirements needed and the timeframe for the application, the decision was made to return the method to the controller class.

Any other icons that were not developed in NDMA which were planned was lack of development time. This was to commit the time to develop the advisor and logging system.



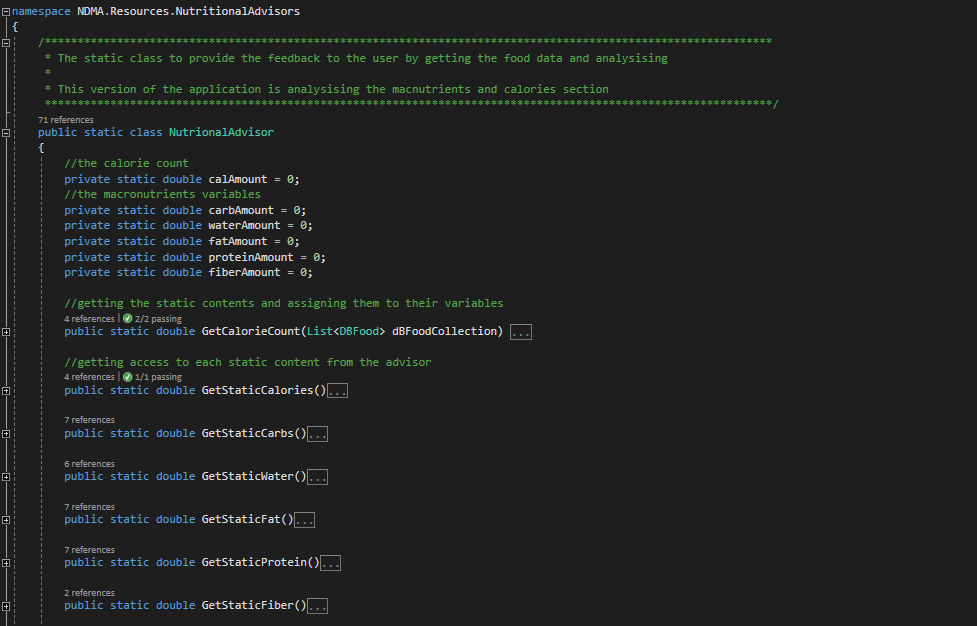
## 4.4. Middle-Tier

The middle tier of the application is the functional logic of the NDMA itself. The plan was to use the middleware to connect the front view to the data view through using the middleware. The parts of the middle were the connection to the cloud database, access to the nutritional advisor system and the application functionality itself.

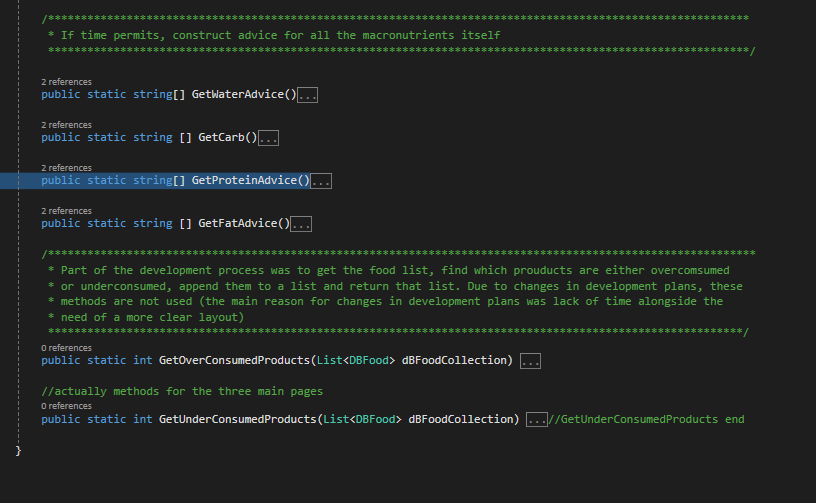
What was developed was the nutritional advisor itself and test data for the usage of NDMA. The main reason behind such decision can be described in two different ways. The first was the technical requirements to build the application properly was outside the scope of the application. This is when combined with other college studies, the timeframe was insufficient for the necessities of NDMA. This, combined with the un-expectation of the coronavirus arrival in Ireland, created an uncertain atmosphere of college credentials. The other issue was the financial implications of Azure services with the cloud environment (this will be expanded on in the database layer section).

The nutritional advisor role to analysis nutritional data and provide feedback based off the data. It would analysis the nutritional data by getting such data from the activity class. It then captures the macronutrient section for storing and analysis purposes. Once completing both, it will then provide feedback to the activity based off the inputted data. The planned way to do this would be evaluated against the user personal details (age, height, weight, sex etc), their personal goals and their activity routine. The activity routine could be described as lightly active, moderately active or highly active in terms of exercise. Due to the timing restrictions, it just does a comparison against the recommended daily amount per nutrient. This was stored in the application test data.

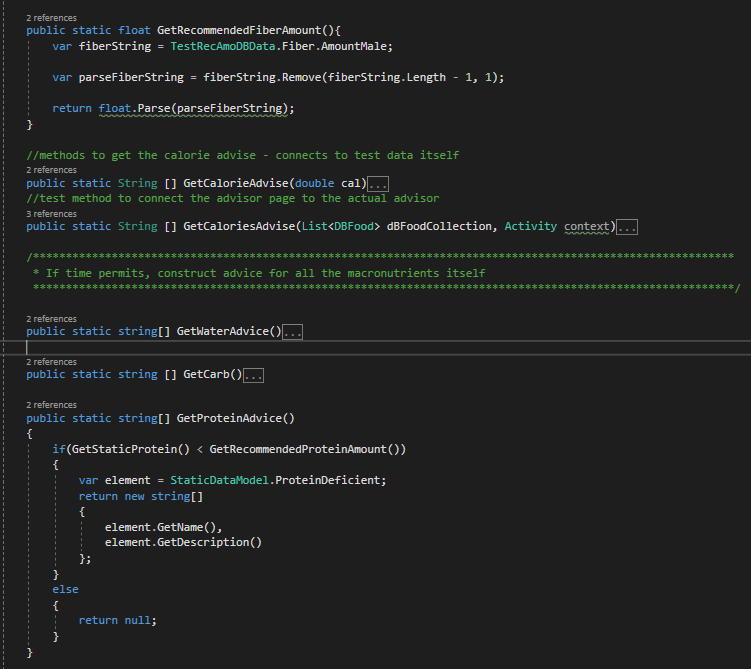
**The nutritional advisor, which holds the static members and the methods**



**The methods to get the advice based off the nutritional** **input, such as protein**



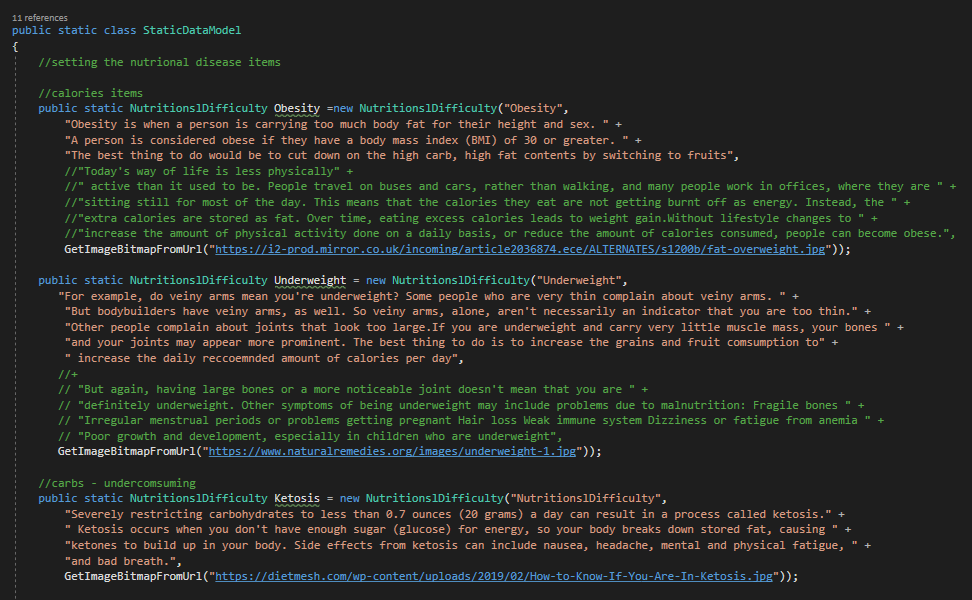
**The implementation of the get methods, such as the RDA and the advice**



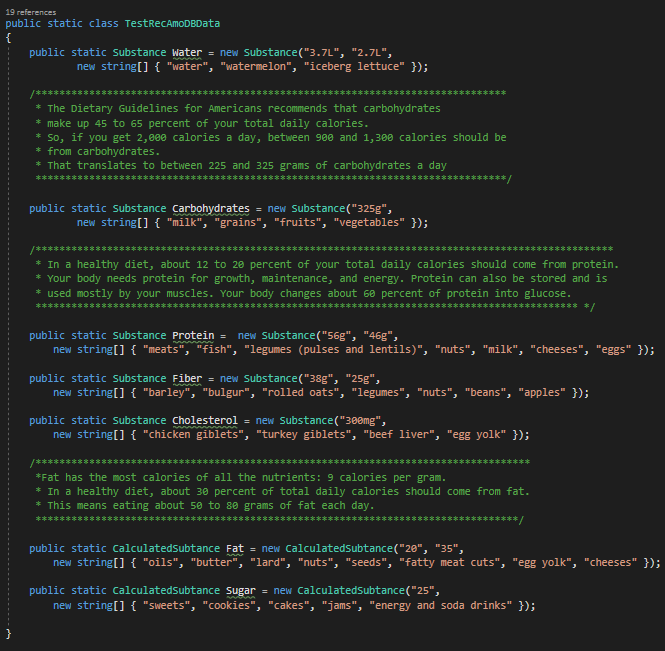
The following data, and nutritional information, was taken from numerous websites, which were available on public websites. For the nutritional Diseases, the references (insert references will be inserted here) are found. For the recommend daily allowance, a data set was used as the data source and the references numbers (insert references numbers here) are found. All the following code are the test data used to enable NDMA to work with the data effectively. The test data were used due to two reasons. The first was the timing constraints of NDMA development scope. This is when combined with other project work and assessment made development time difficult for NDMA. The second reason was the fact the configured cloud environment was deleted in the process of development.

The test data used are nutritional difficulties, food data and user detail. This is to ensure NDMA can function as designed overall. The final image was test data created to allow the functionality of NDMA to work. This would regard, in this version of NDMA, the login function and the advisor system. In order, the following three images displays: the nutritional information used inside NDMA, the recommended daily allowance of each macronutrient (and its food sources) and lastly, some test data as an example of details used by the user.

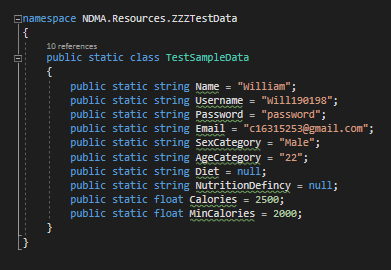
**The nutritional difficulty test data**



**The Substance of the nutritional data, which were used for the test data**



**Other test data used inside NDMA for the functionality and analysis**



## 4.5. Back-End

The back end is the data storage of NDMA. This would ensure the user had access to the most updated information for the application usage. The data from the user would be captured from the user through the front view, passed through the middle ware to ensure the format is correct and stored in the data layer. The data from the front view would be mainly personal details of the user. Such personal details would be the height, weight etc and a username to ensure only the user can access the data. The logged food data is also considered personal data in NDMA. The usage of the personal details would allow the user to personalise NDMA to their taste.

The technical structure around NDMA backend would be a combination of a cloud database and local database. The cloud database would be configured on Azure services using their SQL database and data servers. The local database would be implemented using SQLLite technology. To ensure the user experience is maximised, the two databases would be synced with each other. Since part of the technical requirements are linking the user to their logged food, tables inside the database need to connect to each other. Thus, a relational database management scheme was developed for the application.

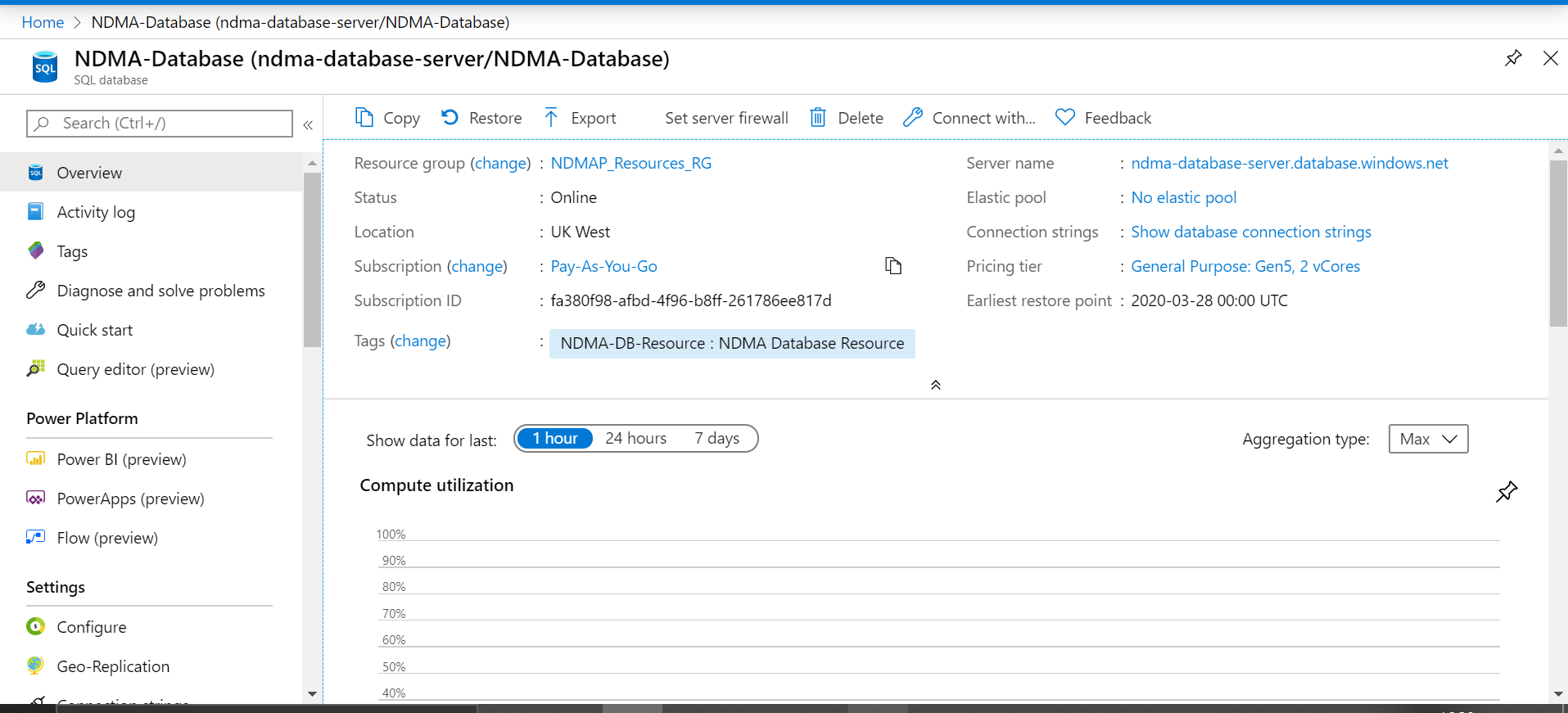
Part of the early stages of NDMA was to move the dataset contents to the database. This was by analysing each dataset contents to find which best matches NDMA requirements. The reference for the datasets analysed are (insert reference numbers here). The chosen dataset came from the “Nutritional values for common foods and products” on Kaggle site. This dataset was chosen due to the dataset containing the macronutrients contents needed. After selecting this dataset, the next step was to clean up the data for computational purposes. This is by cleaning up the data and storing in a staging localhost database. Once the data requirements were sufficient for NDMA usage, it would be moved to the cloud database. This was completed on Jupyter Notebook using the Python language. The issue found with the dataset however was the lack of images used for the application. Due to it being a significant necessity of NDMA, images that closet matched the contents from numerous online resources.

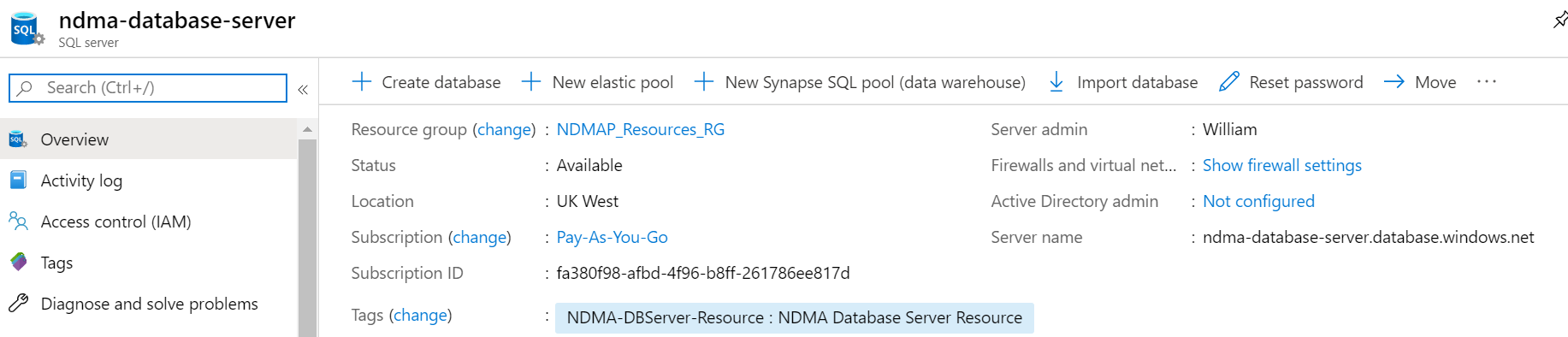
The focus was changed after exposure to Edamam Apis. Their food resources were found to match the requirements of NMDA more. There were three APIs to choose from Edamam services. Specifically, the recipe API had access to images, and a higher level of food content, as part of the system. As a result, this service was immediately prioritised for NDMA services. As a result, the localhost database contents did not move to the cloud environment. Due to the timing constraints around NDMA development, the localhost contents have not been investigated since the decision not to continue with it. Due to the same reason, the SQLLite has not been developed within the NDMA development scope.

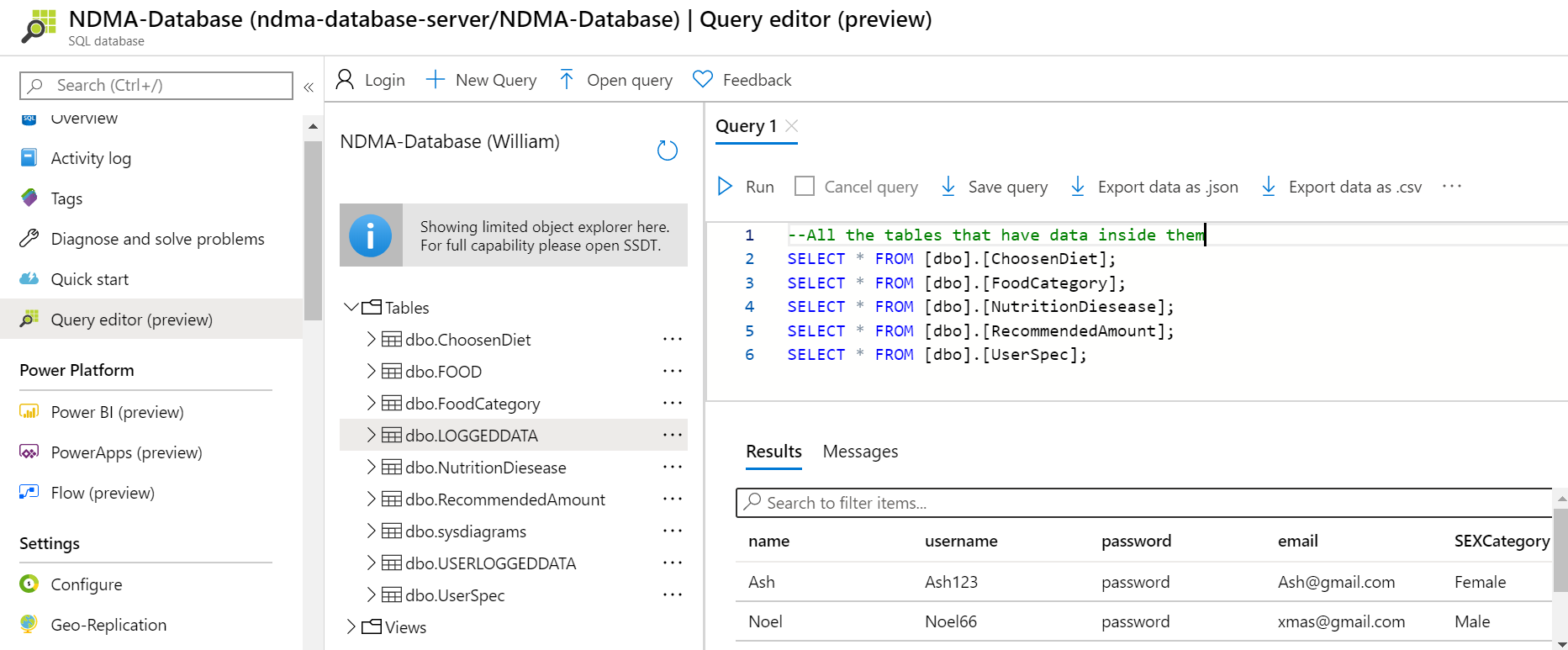
The configuration of the cloud started out in a mixed result. Initially, it was configured online and deployed to the UK environment. Completing this, the task was to create the schema and fill the schema with the needed data. After the investigation, the result came back to install Microsoft SQL Server management studio 18. This tool would connect to the database remotely and create the schema from the local source. Once the schema was finalised, NDMA front – end was continued to be developed on.

A problem was later found regarding the financial services of Azure Cloud servers. The cost for the usage became too high for potential payment. This was discovered after going through both student package and pay-as-you-go package. The mistake was different on both sides. Through the student package, it was misread on the free services and the free credits were consumed. The cloud support team was contacted to resolve this, which were of little help to understand the circumstance. So, analysis was done on the pay-as-you-go subscription to see if it had offered the services needed for NDMA. After seemingly matching the technical requirements (which was free for 12 months), Azure started to charge again. This was for a part of the services that wasn’t clearly highlighted to be part of the free package. Due to the inability to pay for the services, an email was received that the services used were deleted. AWS was considered for the cloud database services. However, due to the time which this happened, test data was created to continue NDMA functionality. There was insufficient time to create AWS services nor create the SQLLite services.

**The configuration of Azure Services before deletion**







**The analysing of the dataset before switching over to Edamam services – Part A**



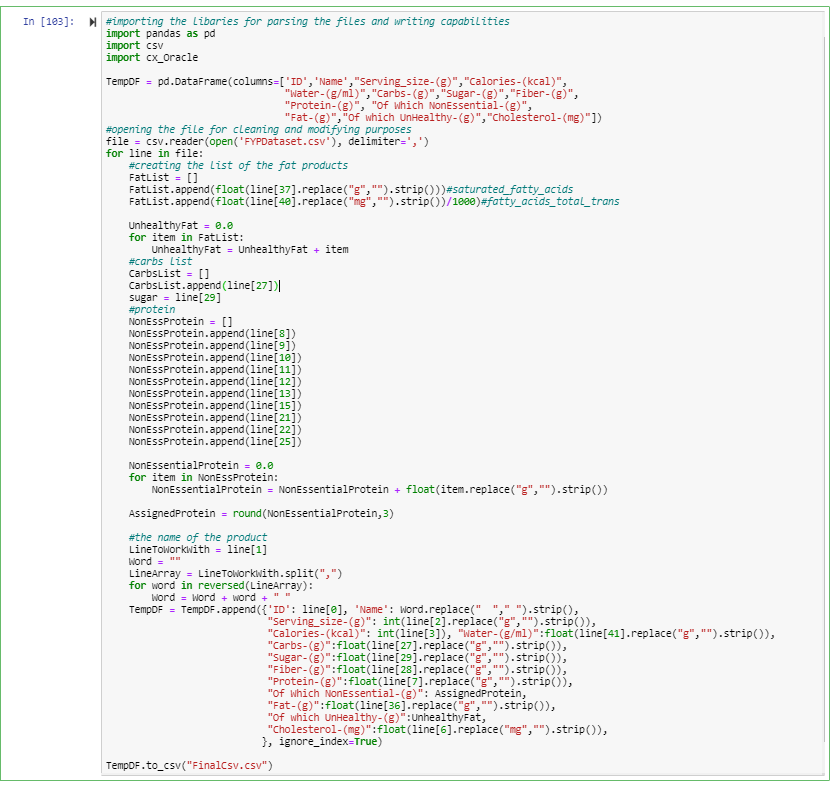
**Part B**



**Part C**



**The process of cleaning and appending the data itself**



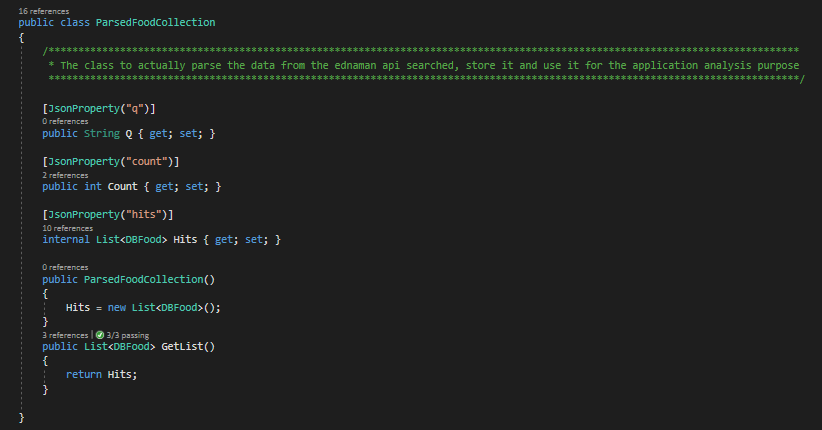
**The staging of the contents of the dataset before moving it to the dataset**



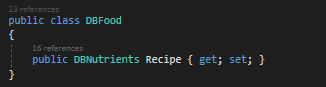
## 4.6. Edamam API Storage

When querying the Edamam Recipe API, custom classes were created based off the contents of the JSON string returned. This was to ensure NDMA was able to use the API as part of the logging system. It also allowed the analysing of the macronutrients to be efficiently conducted by the advisor system itself. The classes of NDMA need to match the field names of the JSON string returned. This is to ensure the created classes of NDMA absorbed the JSON contents properly. The data type also needed to be configured to map over the indented contents of the json string too. Once the classes were developed and completed, the conversion from string to the designed classes was efficient to complete. It also enabled the logging system to track the food effectively. The advisor system can also effectively analysis the macronutrient contents of the food through these classes.

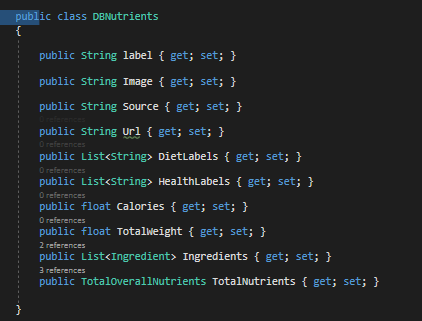
**The Start class to absorb the contents of the food recipe JSON string from the Ednaman API. The Q is the queried string passed through to search for the food. The count is the number of returned elements that match the queried string and the hits is the collection of the food that matches the queried string**



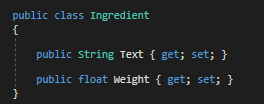
**The root area of the food collection that captures each of the returned food details.**



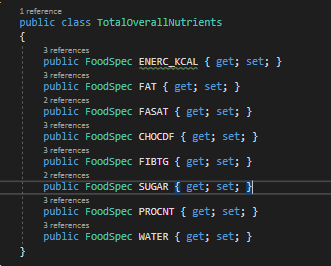
**The food details of the return food recipe, such as the name, image and the ingredients**



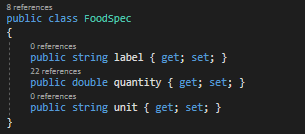
**The ingredients details, which is their name and their weight**



**The macronutrient details of each food. The contents is calories (ENERC\_KCAL), Carbohydrate (CHOCDF), Fat (FAT), Saturated fat (FASAT), protein (PROCNT), Fiber (FIBIG) Water (Water) and Sugar ( Sugar).**



**The macronutrient details. This is the name, the amount of the nutrient (for example, 325) and the measurement used (For example, litre)**



## 4.7. Conclusions

This chapter went over the development details of NDMA. It covered the technologies associated with the different layers of the application. Xamarin tools for the front view, C# for the middleware and SQL for the backend of the application. It discussed both the plans for the development of NDMA and the actual development. It also went over some of the difficulties found during the development of NDMA. Two major significant ones found were timing issues and financial issues, which mainly resided in the data layer. Finally, the insight into the external resources of Syncfusion and Edamam resource uses for NDMA were analysed. This was to allow their integration into NDAM as efficient as possible. Methods to do this was to evaluate their requirements against NDMA, create profiles on both sites, access their generated credentials and apply them to NDMA.