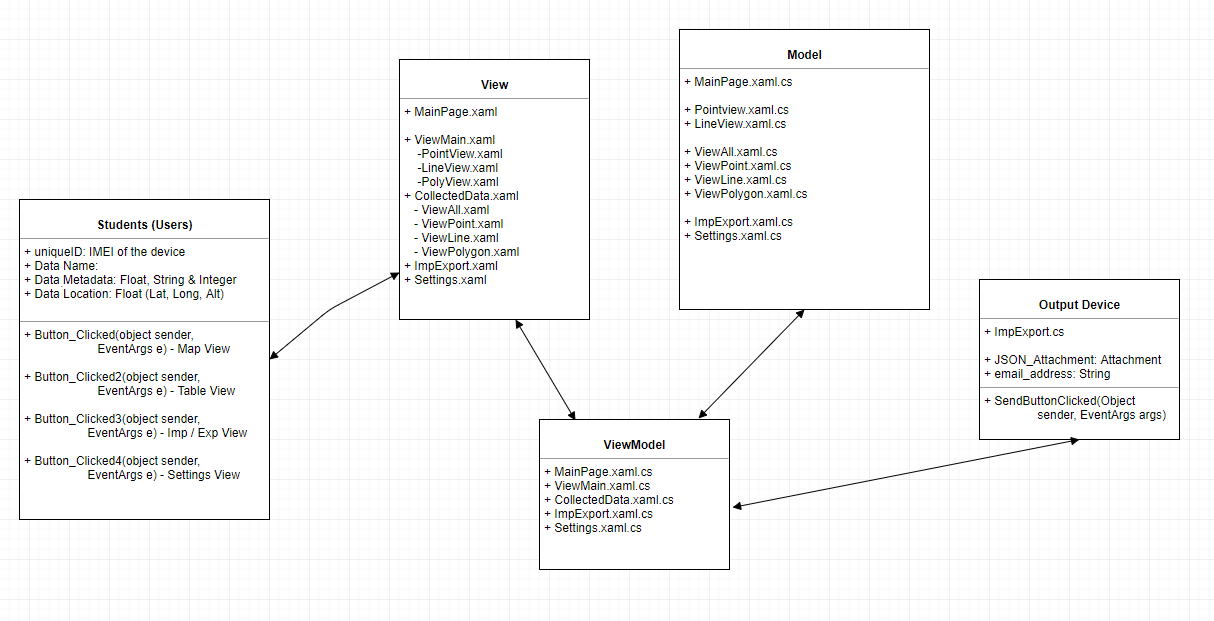
**Intended Software Architecture**

Above was the proposed architecture from report 1, which was intended to be implemented in our final prototype.



However, during the development of our project, we have leaned towards the below UML diagram of our new software architecture, which follows the best practices of Model-View-ViewModel (MVVM) software design pattern. In the previous software architecture, it only displayed and explained how classes functioned and its relationships between each classes, but not how the application as a whole, was processed.

As seen in the revised MVVM software architecture UML. Users will use the application for collecting geo-location data using GPS. In the Students (Users) Table, students will have their current device’s IMEI as the unique identifier for the ownership of the dataset, they will then enter data information such as, data name, metadata about the data and its automatically gathered location data (Latitude, Longitude and Altitude) using the “Watcher” button which refreshes the location data upon click.

The model table represents collection of classes which explains and manages the business logic and the data models, which are known as the .CS files.

The view table identifies and configures the UI display to the user, it will get the business logic (or the model) and interpret it into a user friendly interface, written in the .XAML file.

The View-Model table is responsible for handling methods, commands and other classes that affects the state of the “View” table. It will manipulate the “Model” and “View” separately using the “View-Model” in between.

The main reason, that we have implemented our application in the MVVM software architecture is, firstly, it utilizes the “Observer Design Pattern”, where it clearly separates the logic between the UI and application process and it is most commonly used in situations where data binding is in use.

As the “View-model” class manipulates the information processed and delivered through the “Model” class, then passing it through to the “View” for the users to see. As one of the MVP features was being able to display user collected data into a tabular list format (part of the required MVP feature). Using MVVM, the application would accept the data given by the user and process / store from the “Model” table and it would be given to the “View-Model” where it is manipulated and updated before passing through to the “View” table for the users to see, meaning both “View” AND the “Model” is affected by the user input.