

Self Review

By AbdelRahman



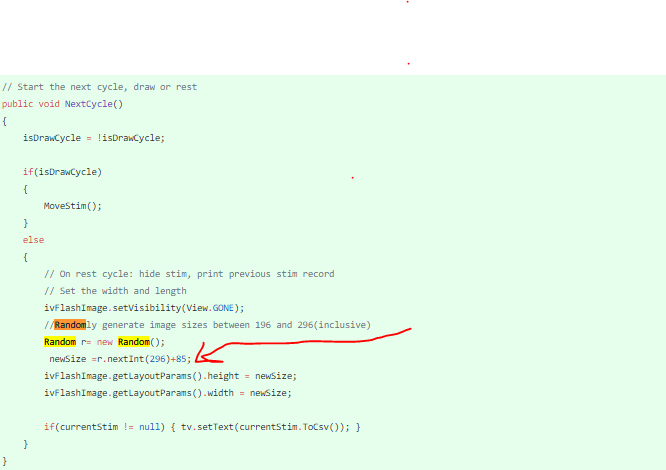
# Technical Proficiency

Project1 in the community group consisted of a lot of coding and being exposed to an array of different languages to which all posed a challenge and means to grow one vocabulary of languages. Moreover, the progressive nature of project1 was that it allowed us to gradually ease into the software development environment that involved much group work and intragroup communication.

As part of the client's needs, there was 4 assigned tasks that are the following: the Visual Scan tool, Aya App, Internet of things Database API and a VR rendition of the Visual Scan tool. These tasks will form the bases for answering the self-review questions.

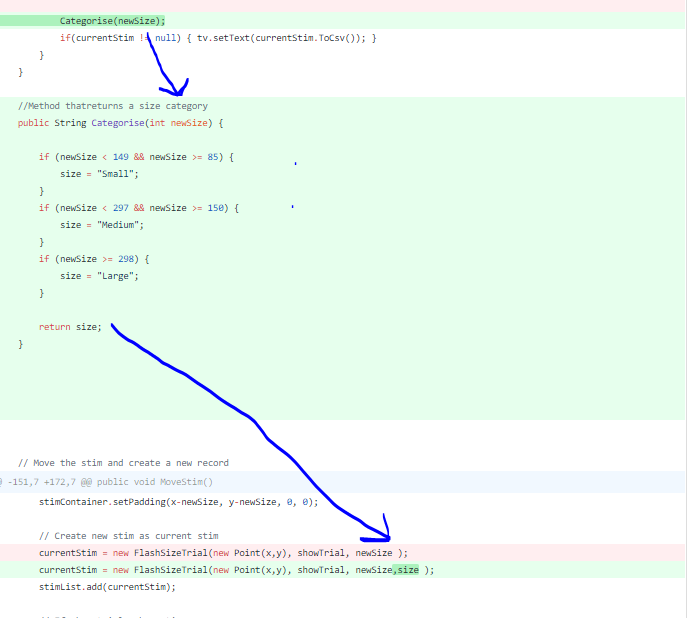
## Question 1: What is the overall quality of your code like?

### Task1: The visual Scan tool-FlashSizeImage.

As an initial transition to project1 paper, my task was to code the random generation of Image sizes that would appear on the screen in the visual scan flash tool. Integration of a new class flashSize fragment class with the required functionality needed another class /object called the FlashSizeTrial that defined its movement, and CSV file output. I injected a random generator and stored it in a variable called newSize. This value was used to set both the height and width of the image view drawn to the screen. This enabled the size of the image view to be controlled by changing the random upper and lower limit parameters.

<https://github.com/OtagoPolytechnic/CommSoftTasks/commit/e6353acb6070a6dd08533d57dbc559924d2950f1>

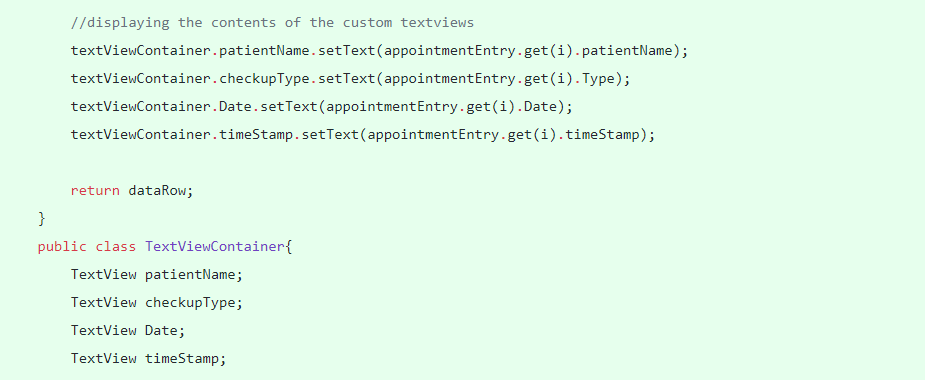
After the first initial meeting, the client needed information in the CSV to be more meaningful. This was remedied by adding functionality to display the size criteria of the imaged tapped. A simple method that checks the passed in random size integer and returns the corresponding size string value. Following the preparation of the flashTrial constructor and its overriding toCSV toString to accept a size parameter, the categorize methods size property is seamlessly included in the class instances. The Simplicity of this code meant that it can be reusable and easily incorporated into additional classes with similar functionality.

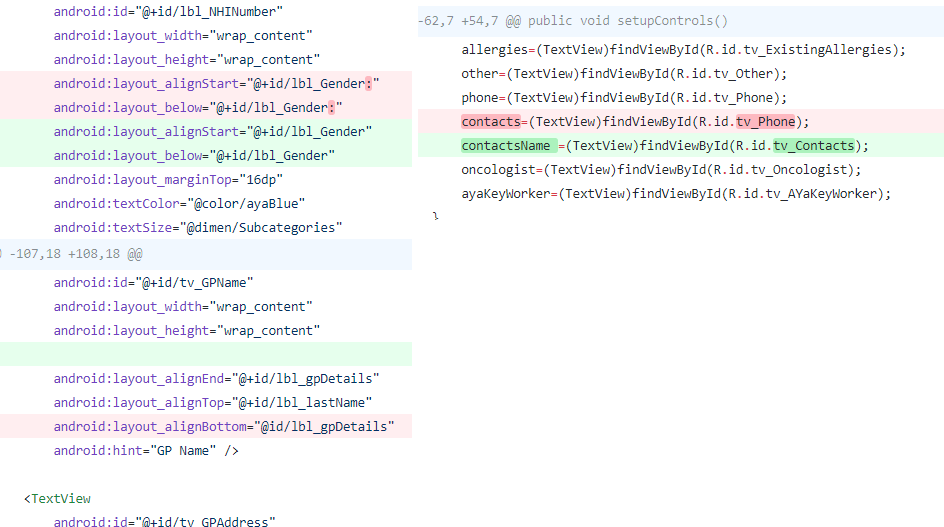


### <https://github.com/OtagoPolytechnic/CommSoftTasks/commit/d00b2da6c6c746cd9c4a14fef259b4945cbc5225>

### Task2: The Aya App.

Progressing from the Visual Scan tool, it was time for the group to onboard a medical passport app that was presented by the District Health Board. Fortunately, design students drafted page layouts for the application.

In the instance of coding the Appointments functionality that enabled users to add/edit and delete appointments with timestamps, we needed to inherit the gitView public method from the public interface adapter to populate a custom listview containing the patient's information. This custom adapter required a list of textViews that could have been declared as a global variable restricting its use within the parent activity. This is resolved by creating a simple Textview container class (courtesy of Samantha and adapted for this class) instantiated inside the method and its elements set by the retrieved XML appointment entries. This increases code modularity and tidiness as the classes sole purpose is to hold elements. As the AYA Apps activities increase, so is the need for a more efficient way to switch between intents. Having a separate button handler for each intent could prove cumbersome when adding more activities. This code conveniently creates an instance of the Intent when switching on the XML elements. As a result, the class method startActivity is executed according to the intent variable.

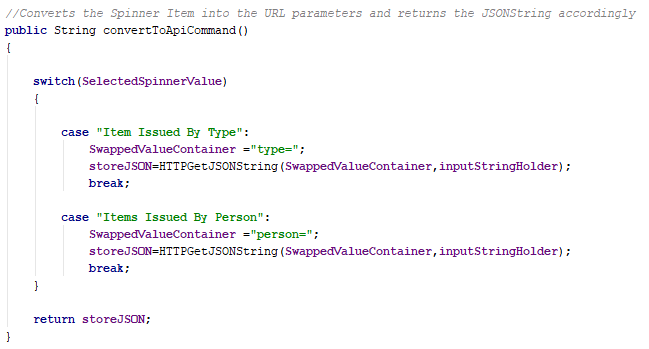
Also, the app consisted of the Edit Health Information registration form that had many text field and text view elements. Because of this, there was much need for a trivial- uniform labeling of their element ids. Having the id name and element they constituted separated by an underscore meant for easy search and fixes.

### Task3: Internet of Things database API.

Later in the semester, a client needed a means to monitor their hardware inventory and location. The ASP.net API is a framework that makes it an ideal platform for building HTTP services including RESTful applications. This allows for an easy transition to building a Mobile version for the IOT platform that simply makes a series of HTTP URL requests hitting the Database for JSON metadata.

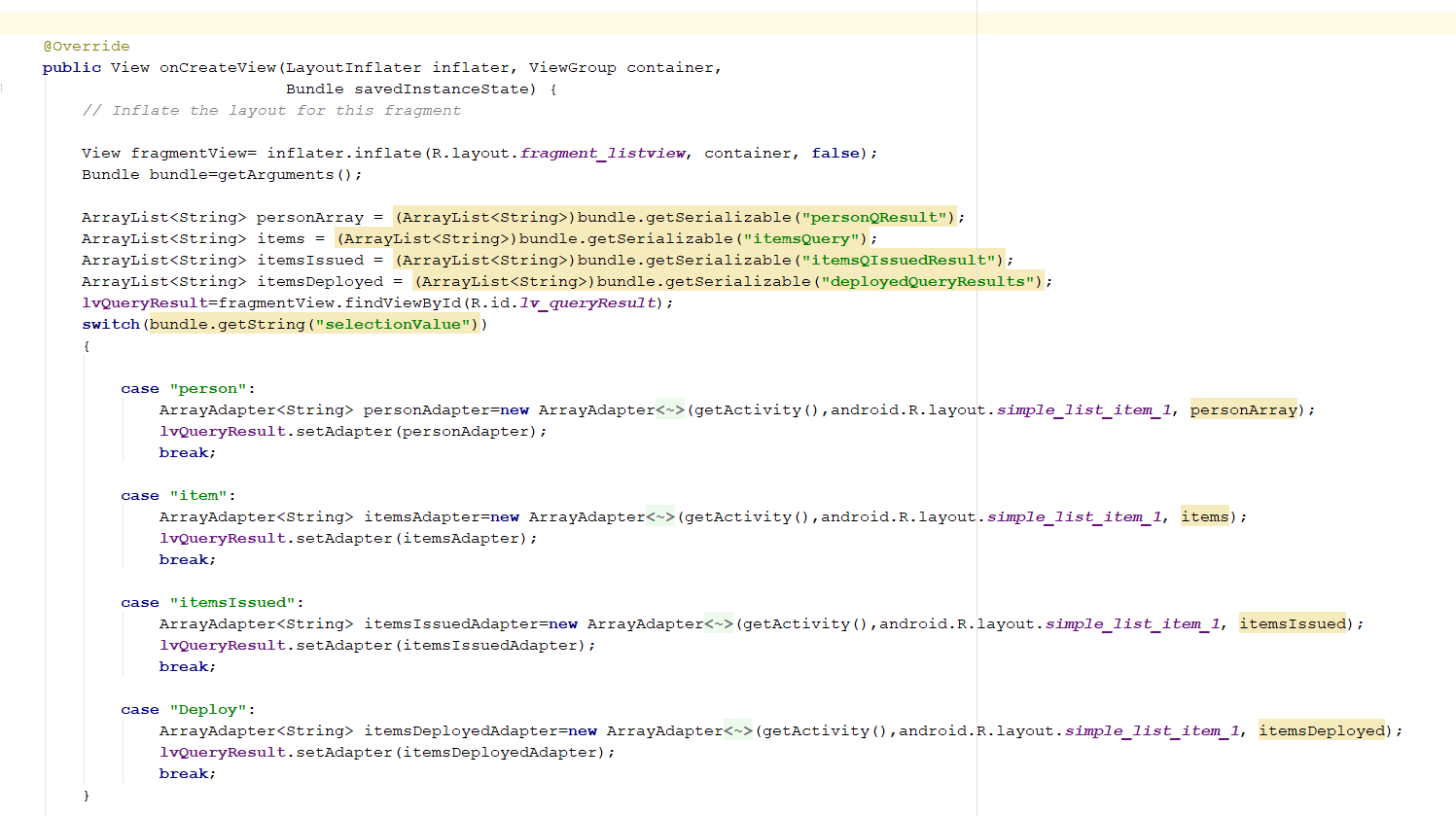
An API get request was needed to view all the items returned. However, database dependencies in the items table with the type model and subTypeModel tables made it's difficult. This was solved by iterating through the items table and selecting its properties based on its ID’s. Foreaching over the collection in turn Loops through and connects the tables to the intermediary Item table. And because the itemModel Table is a foreign key of newSubtype, that becomes its parameter. This is believed to be good quality code as the get method accepts an ID, makes a database query and selects on where the ID in the URL get request is located and matches it with the items table ID. This contains a dynamically populated list of values in the Item table.

 Next, it was my job to build a mobile app that makes HTTP requests, processes the response and displays it in a listview. However, the requirement was to display not only the information in the Person Entity, but code logic was needed to switch between the asynchronous threads to display the Items, ItemsIssued, ItemsDeployed depending on the values of the dropdown. After completing what is perceived an easy task became an ever more time-consuming endeavor when adding a search function to the application. The SearchDatabase class contained flow logic needed to grab and append the value of the text value followed by the drop downs’ category selection.

What is the highlight representation of modularized code is the below example logic that takes the value of the spinner and assigns it a new value parameter “SwappedValueContainer”. This is then passed in along with the search string “inputStringHolder” that is added to the HTTP URL of HTTP worker. The code below does suffer from lack of Method cohesion. This can be identified by the dependency this method has on the returning value of another method call “SelectedSpinnerValue”. However, the simplicity of this code is its redeeming quality as it can be understood without little context.

Switch on listview

All of the category information needs to be displayed in a listview on the same activity by means of selecting the category from the drop-down and then typing a search for the category in the text field. This was accomplished by adding the listview to a fragment so that it would display the URL response table information dynamically upon submitting the search. However, because the fragment\_listview is a separate activity, bundling of JSON data is needed for cross activity transfer. On the listview fragment, the receiving bundled information corresponding to each table category information is then stored into an ArrayList of string. Sending the Array of JSON objects to the List View fragments required a switch statement consisting of different array adapters that would accept the bundled array depending on the Spinners selection type. What is qualitatively sound about this code is the separation and of the listview fragment from the main activity. It wasn't as simple as using one array adapter for the 4 selection view types. Rather than extending from the fragment class separate from the activity but within the same namespace, this code is compacted into a separate class. This ensures there is the least amount of common coupling between the classes. They do not share any global variables but the serializable ArrayList that are necessary for data transfer. This improves debugging by better localization of errors. It is relatively easier to fix packaged code compared to its cohesive counterpart. Going back and fixing errors in the main activity will not drastically affect fragment functionality.

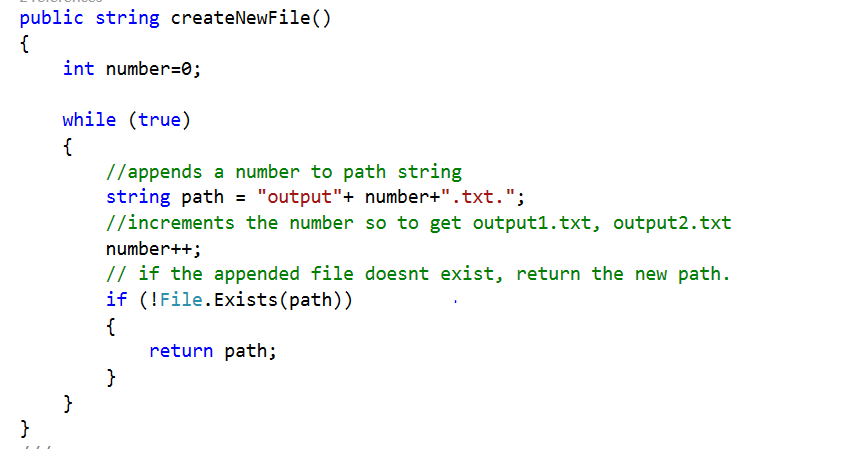


Dropdown arrayadapter switcher

### Task4: Unity Virtual Scan Wander Tool.

After completing the IOTS App, we decided to make a virtual reality rendition of the mobile visual scan tool. We used Unity as our development platform as it is equipped with a wide range of VR supporting class libraries including button input for VR peripherals and the camera view. Unity was our choice because the classes are seamless to integrate into our Visual Scan base. Also, because Unity is primarily written in c#, and its support for popular Virtual Reality technology, this seemed more of a fitting option.

After completing the core mechanics of the game, the client needed some way to monitor their performance. A simple solution for this was to output into a text file the X and Y coordinates, also including the Boolean value representing client interaction when they detect a color change by tapping the space button. Conversely, there was a slight issue with the output text file getting overridden with every new instance of the game. The code below would concatenate an incrementing integer into the string path. Whilst it checks if a pre-existing file does not exist. Based on this it will return the new modified path. This function is called in the endgame() where it creates a new file on loading a new scene. Below I believe to be good quality code in the sense that it is a short method used to do one task. Also, the name of the method entails enough information to call the method without looking at its body.

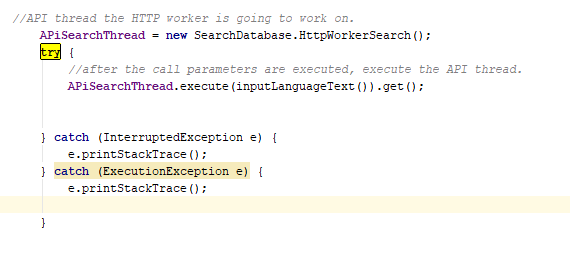


Creates a new file

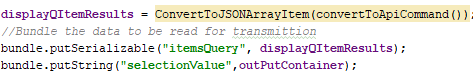
## Question 2: How well did you follow best practices in development?

### Task3: The IOTS App.

In general software development, proper error handling is perceived to be good programming practice. In the IOT App, The method responsible for input text field processing is first initialized. The addition of the .get() makes sure all the logic following follow the inputLangeText() method is run before executing the asynchronous thread. The execution thread below is wrapped in a short try/catch block that will handle the errors pertaining to that thread. The implementation of error handling advocates robust code that does not easily break. In the case that it does, exception handling provides helpful information on the source of failure.



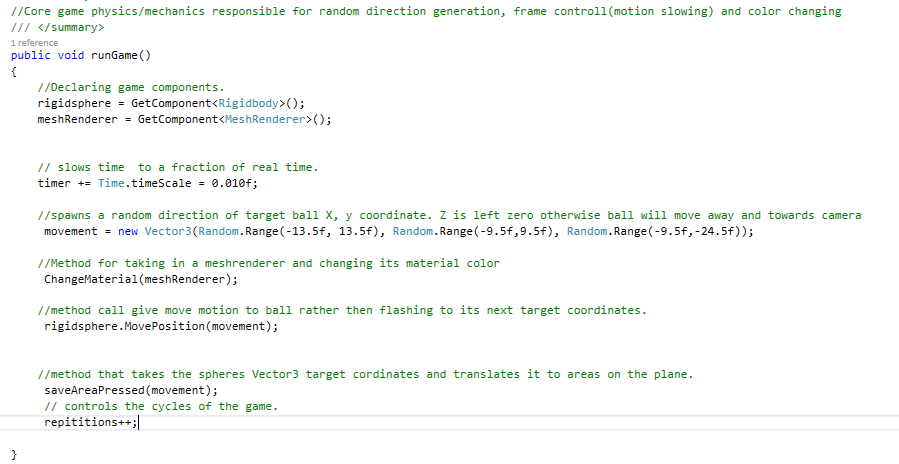
In the ViewDatabase class of the IOTDatabaseApp, below is a globally stored method “ConvertToJSONArrayItem” that will take the method call contertToAPIcommand return JSON data and convert it to a JSON array. Although, calling methods within another is not ideal. However, storing the result of the JSON array converter method into a variable means its return value can be used anywhere else in the activity. This is an instance of good development practice with code reusability in mind.



Code reuse

### Task4: Visual Scan tool.

There has been some time since coding in c#, and especially when coding in a fresh language on a new platform, it becomes a necessity to provide easy recollection of the code functionality. This was done by adding line by line code commenting so it also makes it easier for someone to resume where I left off. In addition, the below code for handling the slowed down motion of the ball in the wander tool is relatively new and complex. This made commenting very important so that someone else can understand the logic for easier code handling.

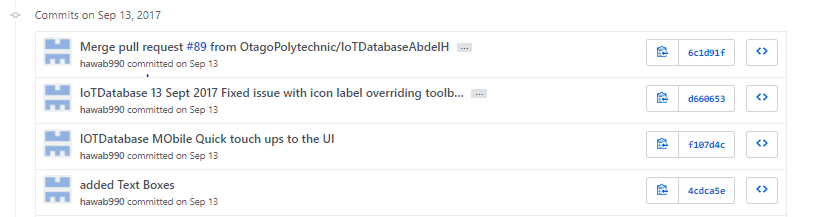


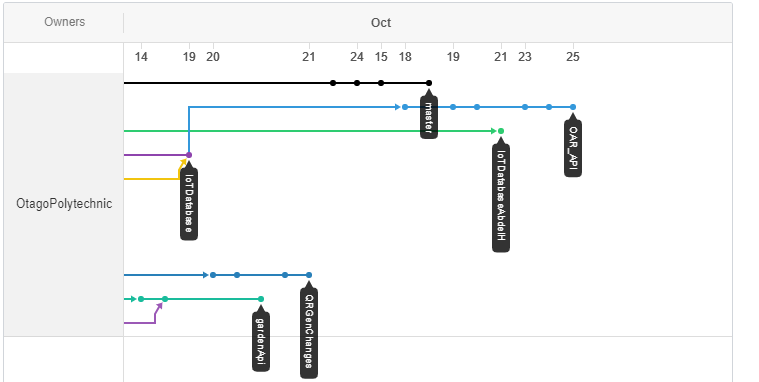
Code Commenting

## Question 3: How well did you use appropriate version control?

Over the entirety of this semester, the group has been adhering the git hub commit message rules that include firstly the master branch currently being worked on, followed by the date, and a lengthy commit message that details the changes made. This greatly clarifies the process of pulling the correct updated version of the branch. Descriptive commit messages assist with reviewing contribution in the future. Also, it is favorable to commit big and commit often. Abundant commits help save time when rolling back to a previous version of the code if errors should arise.

Because the IOT Database API was quite large consisting of many models and controllers, this needed a more productive and safer way of having multiple developers to work on the same model concurrently. Branching allows for multiple developers to be work on the API while the integrity of the master IOT Database branch is protected. In essence, to avoid merge conflicts, once modifications are made to the branch “IOTDatabaseAbdel” is when a pull request occurs. This branch is then authorized to assume the new IOT Database branch.





## To what extent do you think you contributed an equal portion of the overall project?

I believe to some degree I have contributed an equal portion to the overall project. I had a firm presence in the stages of the software development lifecycle of the AYA Passport application. I attended all the client and scrum meetings from its initial stages to its deployment. More so, alongside Samuel and Samantha I made significant contributions to the functionality of the application. This includes writing the Health information page and the appointment page. However, the first assigned task was primarily to reinforce my understanding of the requirements analysis and implementation of code stage as part of the SDLC. Furthermore, after numerous meetings with Mary, I managed to accomplish the task of receiving informative feedback regarding the Visual Scan Tool.

I completed my tasks assigned by the team to complete sections of the IOT web API controllers such as the ItemsController and the ItemsIssued Controller. In addition to this, I fulfilled my task to complete a Mobile version of the IOTDatabase that made a series of HTTP calls from Web API and applied search functions on the containing information.

Lastly, using unity, I was able to design a VR version of the Visual Scan Wander Tool with varying degrees of success. Ultimately, I believe my input was significant to each products fruition.