EAD 2

CA 2

Student Number: X00138256

Student Number: X0012725

Student Name: Luke Morgan

Student Name: Aaron O Conor

**Introduction**

We decided for EAD 2 CA2 we would build an app where a user can see sellers and the electronics of the seller on the app. The idea is that the user would be able to see a list of electronic that they have also sold a list of the sellers and their information. Users should be able to put electronic up for sale with information added for that item such as the name, price, brand name, the size of the electronic and the colour. The electronic are connected to the user by Id. You can check to see if the seller is any good by there user rating which is a variable in the seller object. You can also check what the seller has sold in the past. Users can use these numbers to judge the sellers to see if they are good or not. Also, the users can see the location of the seller so that they can decided if they want to go to that seller or another one.

**Github**

The link to the Github repository is <https://github.com/lukemorgan98/EAD2CA2.git> Github was the website we used to store our repository. This helped us to work on various parts of the project without overlapping each other we used this repository to keep track of work that we were getting done. We had committed our work to the repository whenever we had finished the task or objective that we set out to do. This was a simple way to check what was done and what was not done. To avoid many errors being committed to the Github repository we only committed when we the task or objective was finished and was working. Therefore, there is not that many commits with the overall project. The reason that there one person had more commits than the others is because we would work together using discord and let one person code while the other helps them with it so that way everything was a 50/50 project and no one person did all the work so that is why there is some more commits from one person to another person but the work was done together as a team.

Chart

Description automatically generated

**Database Schema**

Electronic Schema

In this schema we can see that we have 7 variable Id, name, price, brandname, size, colour and sellerId. The Id keeps track of the Id number of the item of clothing. Name is a string this hold the name of the product that the seller has given to that electronic an example would be “IPhone” or “LG Tv”. Price is a double and it represents the price of the item of the electronic. BrandName is a string that holds the name of the item. Size is a string at it holds the size of the product. Colour is a string and it hold the colour of the product. SellerID is an int and it holds the seller Id this links with the sellers in the seller schema. It is a foreign key and is what can be used to reference the sellers for the specific item for electronics.

Graphical user interface, text, application, email

Description automatically generated

**Seller Schema**

In this schema we can see that we have 5 Id, name, location, rating, electronicsSold, Id is an int that keeps track of the identification number of the seller and of the electronics. Name is a string and it hold the name of the seller. Location is a string that stores the location of the seller that the user wants it is usually “Naas” and “Dublin”. Rating is an int that stores the rating of the customer e.g “3”. This rating is out of 5 and is a good indicator as to the authenticity of the seller as well as it’s a good authenticity as to the credibility of the seller as it shows how new they are to the market or if they have experience in the market in terms of being a seller.

Graphical user interface, application

Description automatically generated

**Service**

**Sellers**

**Get Sellers**: This gets the list of sellers in the database

Text

Description automatically generated

**Get Sellers/id**: This is used to search for the seller’s information by their Id value from the database if a null value is entered by the user a 404 code is shown.

Text

Description automatically generated

**Put Sellers**: Uses the seller id to allow the seller to update the information which is needed if their information is wrong or outdated. If there is no seller with the id entered a 400 message appears. After the database gets checked if the seller exists or not and if not is sends a 404.

Text

Description automatically generated

**Post Sellers**: Is used to create new sellers into the database by using addSeller. You enter the values of the new seller when and you execute it, it then saves the changes. Once it is saved to the database it is then returned as the id of the seller.

Text

Description automatically generated

**Delete Sellers:** For this you input the id of the seller you would like to delete from the database if no id is entered you receive a 404 message. Once an id is removed the database is then saved.

Text

Description automatically generated

**Swagger API UI – Sellers**

Graphical user interface, application, email

Description automatically generated

**Electronics**

**Get Electronics**: Calls the database and returns all the different electronics information.

Text

Description automatically generated

**Get Electronic**: This calls the database by the id that is used if you are looking for a certain electronic. If there is no electronic with that id then you will get a 404 message.

Text

Description automatically generated

**Put Electronic**: This call is made to update the electronic database using the electronic id if the id is outdated or incorrect information. If there aren’t any electronic with the id, then a 400 message is returned. After it saves it to the database then it checks if the electronics exists if not a 404 message is returned.

Text

Description automatically generated

**Post Electronic**: Is used to create new electronic into the database by using addelectronic. You enter the values of the new clothes when you execute it, it then saves the changes. Once it is saved to the database its returned as the id of the electronic

Text

Description automatically generated

**Delete Electronic**: This is where you input the id of the database if no id is entered a 404 message is returned. Once an id is removed the database is saved.

Text

Description automatically generated

**Swagger API UI – Electronics**

Graphical user interface, application

Description automatically generated

**Azure API Overview**

**Graphical user interface, text, application, email

Description automatically generated**

**App**

**ScreenShots**

Graphical user interface, application, Teams

Description automatically generated

This is the first page that the user is greeted with when opening the app. Here you can see the logo and the drop down for the different languages this will allow you to select between to different languages. Under all that you can see the search button that allow you to search for an item and click the button beside it.

A picture containing text, indoor, electronics, computer

Description automatically generated

The show Electronics button brings you to this page where the data has been parsed from the URI web API link. If we click on the search bar and enter a name of the electronics, then we can see the specific electronic we searched for as we can see below. I searched for Microsoft and the item showed up

A picture containing text, electronics, computer

Description automatically generated

If you click on the seller page, it brings you too the seller page where if you click on the search bar then hit the search button you get to see the sellers

A picture containing text, indoor, electronics, computer

Description automatically generated

**Testing Report – Expresso**

We were unable to get the unit testing to work correctly because of import files not working however we did get the code in place for it, and this is what it looks like

**Search Wrong Clothes Test** - Testing to see if error messages appeared as wrong clothes were searched for.

Text

Description automatically generated

And something like this

**Search Wrong Clothes Test** - Testing to see if error messages appeared as the wrong seller was searched for.

Timeline

Description automatically generated

**Language Change Test** - Testing to see if what was searched for is what was shown

Graphical user interface, text, application

Description automatically generated

And the final one being

Graphical user interface, text

Description automatically generated

**Internationalisation**

The language we decided to add to is French. We wanted to add French as we felt it is a big language as many people speak it. To do this we added another string.xml file that was more tailored towards the French language. Below you can see the difference between the two when click on the spinner you get to select France and that will change from English to French.

Graphical user interface, application, Teams

Description automatically generated

Vs

Graphical user interface, application, Teams

Description automatically generated

**Miscellaneous**

For our linting tool we used SonarLint. We used this tool to make sure that as we typed there would be live reporting of parsing and compilation errors. This tool helped us with code completion and writing appropriate code,