**ACKNOWLEDGEMENTS**

Our team would like to thank Aptech for the knowledge being imparted into us in this fast growing IT era. This project has improved our comprehension on Android Development. We would also like to thank our faculty and Android instructor Mr Nwosu Polite for his guidance and support wiith this Android Development eProject.

We are also grateful to each of the team members that provided their time, knowledge and hardwork despite this pandemic facing the world in world in working together to pursue the goals outlined in this project and complete it .

**SYNOPSIS**

This project, Lenovo Laptops, is an Android application that shows the different laptops and brands made by the company Lenovo. It allows you to:

* view laptop brands made by Lenovo
* view categories of latops under the brands
* view specifications of each laptop available in app

**USER GUIDE**

The application starts off with a home page that allows users browse the different laptops by brands. These brands include Yoga, IdeaPad, ThinkBook and ThinkPad. Upon clicking these images, the user can browse the different categories.

The brands page includes a short clip that plays at the top of the page as well as the title of the brand and a dropdown to select the different sub-brands that the brand has. Upon selecting a sub-brand from the dropdown, the list of laptops below will adjust based on the sub-brands. Upon selecting any of the laptops in the menu, the laptop page for that laptop will open.

The laptop page for each laptop starts off with an image at the top of the laptop followed by its title and a brief description of the laptop. The specifications of the laptop is displayed after the details of the laptop. Lastly, a list of similar laptops to the one displayed is shown at the bottom of the page.

**DATABASE STRUCTURE**

This application does not utilize the inbuilt SQLite database in Android. The application instead uses ArrayLists of the different models in the application across all its views. The different models and their fields are as follows:

**PRODUCT MODEL**

String laptop\_sub\_brand;  
String laptop\_name;  
String laptop\_imageRef;  
String laptop\_brand;  
String laptop\_description;  
Boolean hasMultipleProcessors;  
ArrayList<ArrayList<String>> specs;

**BRAND MODEL**

String brandName;  
String[] subBrands;  
String details;  
String videoRef;

**DEVELOPER’S GUIDE**

The “main” module of the app holds the source code for the application. It is broken down into the assets, java and res folders.  
  
**ASSETS FOLDER**

The assets folder contains the JSON files that contain the content that are used in the app. There are the “products.json” and the “brands.json” files.

* The products.json file contains the details of all the products in the app. This file is parsed with a JSON parser class and added to the allProducts ArrayList.
* The “brands.json” file contains the details of the brands in the app. |This file is parsed with the JSON parser class and added to the allBrands ArrayList.

**JAVA FOLDER**

The java folder contains the main logic of the application. The folders are split into the following folders following the MVC pattern: Adapters, Controllers, Models, Pages and Util folders.

* The Adapter folder contains the adapters that are used with all the recycler views in the application. They include:
  + the LaptopAdapter - this is used to display the list of laptops used in the LaptopPage, the HomePage and the BrandPage of the application. It uses a view that showcases the laptop’s image and a small textview at the bottom that displays the name of the laptop.
  + The SpecAdapter – this is used to display the list of specifications on each laptop page. It uses a view that showcases the contents of each specification in two textviews within a linearlayout. The first contains the kind of specification (e.g Processor, Graphics) and the second contains the specification itself.
  + The SearchAdapter – that is used to display the list of search results on the search results page when a user makes a search.
* Controller folder contains the logic to interact the database to retrieve data for each page, either drink, category or search. It includes just one controller class:
  + ProductController – Used to query the database for all the data pertaining to laptops. It is also responsible for search queries. It includes the following methods:
    - generateSeedProducts – this method is responsible for seeding the ArrayList allProducts, with all the data from the JSON files in the assets folder. This method is only called once when the application is installed to prevent duplication.
    - ProductController constructor – this method instantiates a new ProductController object in the calling method using the context of the calling activity. The returned object can be used to query the database with its methods.
    - searchProducts – this method takes a search query string and searches the allProducts ArrayList for products that have a name, brand or sub-brand that matches the string. All products that match are returned in an ArrayList.
    - getProduct – this method returns a product from the allProducts arrayList at a particular index.
    - getRandomProducts – this method returns an arrayList of products from random indexes in the allProducts arraylist. This method takes a limit parameter that sets the amount of products to return.
    - searchProductsBySubBrand – this method loops through the allProducts arraylist to find products that have a sub-brand similar to the search string parameter. It then returns an arraylist of results if any.
    - The getters and setters for all the fields in this class are also within the class.
  + The BrandController – this class contains methods that retrieve data from the allBrands arraylist. The methods include:
    - getAllBrands – this method is a getter method for the allBrands arraylist.
    - getBrand – this method takes a parameter, index, and returns a brand from that index in the allBrands arraylist.
    - generateSeedBrands – this method is used to seed the allBrands arraylist with details from the JSON files in the asset folder. This method is called only once at the beginning of the application install to prevent duplication in the arraylist.
* The models folder contains the models for the different types of data that are pulled from the database. These models contain the different fields in formats that are usable across different views in the application. They include:
  + The BrandModel – this method contains the fields:
    - brandName - to represent the name of the brand.
    - subBrands – a string array containing the different subBrands in this brand.
    - Details – to contain the details of the brand.
    - videoRef – to contain the name of the video file for the brand in the raw folder of the application.
  + The ProductModel – this method contains the fields:
    - Laptop\_name – to represent the name of the product.
    - Laptop\_sub\_brand – the sub-brand of the product.
    - Laptop\_brand – the brand of the product.
    - Laptop\_imageRef – the image reference of the product in the drawable folder of the application.
    - Laptop\_description – contains the brief details of the product.
    - hasMultipleProcessors – a Boolean field to check if the product offers processors of the AMD/Intel variant.
    - specs – an arraylist of specs - a list of two strings: the specType and the specDetails.
    - The BrandModel also contains the getters and setters for the fields that it contains as well as a constructor for creating a BrandModel object with all these fields.
* The Pages folder contains the different views that the application uses. These include:
  + The BrandPage – this view contains the logic to display each individual brand’s page from a click on the corresponding imageview on the homepage. It contains the following:
    - The SearchView for each brandpage that accepts the search query and sends it off to the SearchableActivity for processing.
    - The brandLaptopsView for each brandpage that lists all the laptops that belong to that brand. The arraylist that is used by the LaptopAdapter is obtained using the searchProducts method of the ProductController class, passing the brandName as the parameter.
    - The videoView for each brandpage that displays the video of the current brand that is contained in the raw folder of the application.
    - The spinner (dropdown) that allows the user to select the different sub-brands and update the brandLaptopsView to reflect the different sub-Brands accordingly.
  + The HomePage – This activity is responsible for the view the user sees on opening the application. It contains the following:
    - The searchBtn that opens up a searchView to be used to serve search queries to the SearchableActivity for processing.
    - The recyclerview responsible for displaying some random products on the homepage. The LaptopAdapter that is used by the recyclerview, uses the getRandomProducts method of the ProductController to pull random products to the view on every start of the activity.
    - The different imageViews that have onClickListeners to start up the corresponding LaptopPage activity.
  + The LaptopPage – This activity is responsible for creating the views for each individual laptop. It uses the laptopID extra within the calling intent to find the corresponding LaptopModel within the allProducts arraylist using the ProductController’s getProduct method. The textview, and imageView for the laptop page are then set using their corresponding fields in the LaptopModel. The laptopSpecsView uses the specs field within the LaptopModel to create its SpecAdapter which in turn populates the laptopSpecsView with the corresponding specs of the LaptopModel. The similarLaptopsView is updated with a list of products using the ProductController’s searchProducts method passing in the laptop’s brand as a parameter. The returned list is passed to the LaptopAdapter that is used to populate the similarLaptopsView.
  + The SearchPage – This view is responsible for processing searchQueries from other activities across the application and displaying the. The doMySearch method in this activity takes the query from the search dialog or searchView from other activities and calls the ProductController’s searchProducts method passing the query as a parameter. The search results are then used with the SearchAdapter to display it in the searchResultsView. It also checks if there are no searchResults and updates the view accordingly with a “No search results” textview.
  + The SearchableActivity – this is a class that the other views in this application inherit from to be able to implement the listener for the SearchPage.
  + The SplashPage – this activity is the first activity that runs on the application. It’s only use is to display the splash page for the application before moving on to the HomePage activity.
* The util folder contains the following:
  + AppStart class - this class extends from an Application class. This class is responsible for all the functions that should be called on the installation of the application. It contains the generateSeedProducts and generateSeedBrands methods of the ProductController and the BrandController respectively. These methods are run at the start of the application install to prevent duplication of resources.
  + Helpers class – this class holds logic that is common to multiple classes in the application. The only method it holds is the toJSONString method that parses the contents of the JSON file to a JSON string.
* The res folder contains the different resources that are used with the application. These include :
  + images within the drawable-?? folders,
  + the layouts for the different fragments and activities,
  + the menu folder that outlines how the nav menu should be implemented,
  + the mipmap that also contain lower quality images and icons used in the application,
  + the navigation used to draw out the navigation controls and path of the navigation MainActivity’s fragments.
  + the values folder that contains all the hardcoded strings, styles and colors and their corresponding IDs that are designed for the application.
  + and an xml folder to contain any extra non-standard layouts that are used by the application.