CA400 - Technical Specification

Bróga Nua - Irish Sneakers Application

Olan Buckeridge

15461022

Supervisor: Gareth Jones

Introduction

Project Motivation

The aim of this project is to develop an Android application for Irish sneakerheads. Nowadays the sneaker industry is growing every year with the resale industry alone worth over €1bn. Due to the vast growth and spread of the internet it can be somewhat difficult to find the products you are searching for with retailers stocking different products. E-commerce is extremely popular in today's society. Consumers can conveniently purchase products online and receive them next day.

This application will be beneficial to Irish sneakerheads that purchase products both online and offline. As a sneakerhead it is an area that I'm very passionate about. I am constantly getting messages off people I know and people off Instagram wondering where to get certain pairs of sneakers. I want to create an app that offers consumers a place to find their sneakers without having to search through multiple sites.

The application will use web scraping on Irish retailers. This data will be used to create a catalog for the product. This interactive app will provide an intuitive experience for consumers allowing filtering of product and displaying the best deals for the user. I also intend to include a section for sneaker raffles and limited product. Sneaker collaborations and extremely limited runs of sneakers are becoming increasingly popular. This section will push a notification to users about a limited release that is happening in Ireland.

Business Context

The sneaker market is quite niche but it growing hugely every year. Creating an app that will allow customers to find everything in one place will save people time and money while supporting Irish businesses.This application could be used by pure sneakerheads which seek the latest sneakers for the best price. It would allow users to filter sneakers and find the best option and colourway for them. The limited releases section would ensure that they won't miss out on a chance at purchasing the highly sought after sneakers.

I also think this application would be useful for a broad market of people who are looking to purchase gifts. They may know a particular brand or model that someone is looking for and by using our application they will be easily able to select the correct sneaker for the best price. I plan on adding student options too as certain retailers will apply student discounts which will affect where has the best price for students across the country.

User Roles & Objectives

This sneakers application has a broad target audience as outlined in my business context. For the purpose of this final year project I will target the sneakerheads and students. The majority of these people are between the age of 16-25, however, there is a quite a large older audience of people that have been collecting sneakers for years that must be taken into account. People in this age range generally should not have problems with using technology and are familiar with online shopping. The objective of this application is for the user to save time and money.

From the user's perspective, the application would have be intuitive and have easy navigational access. Students lead a busy lifestyle and the main characteristics they want in an application is quick, responsive and reliable applications. They do not want to spend a lot of time clicking through different websites trying to find the best deal. The application should have a clean material design that not only looks great but is accessible too.

I would love to add improved functionality to this system as the application grows and develops in the future. For example I would love to add a potential "Wish List" that the users could add to while shopping. This could allow users to share the list with others for gift ideas and to have products to save for. The "Wish List" could also be used to receive notifications when an item from your list drops in price so the student can get a great deal. Another feature I would like to add would be student discount integration. In Ireland there are two main student sites used for discounts which is UniDays and StudentBeans. These allow users to verify there student status through university and college logins and then provide discounts to various sites.

Glossary

Data Mining
Data Mining is the process of analyzing and evaluating large datasets in order to identify patterns and useful information in the data to solve problems. It has a goal to extract information from a dataset and transform it into understandable insights that can be further used e.g. to predict future trends and allow enterprises to make better decisions.

Web Scraping
Web scraping, web harvesting, or web data extraction is data scraping used for extracting data from websites.

Algorithm
An algorithm is a set of rules, procedure or formula used for solving a problem.

UI

The user interface (UI) is everything designed into an information device with which a person may interact.

Sneakerhead
A sneakerhead is a person who collects trades and/or admires sneakers as a form of hobby. Sneakerheads, like most collectors, are passionate and dedicated to their subject. Many are very knowledgeable about the origins and history of sneakers.

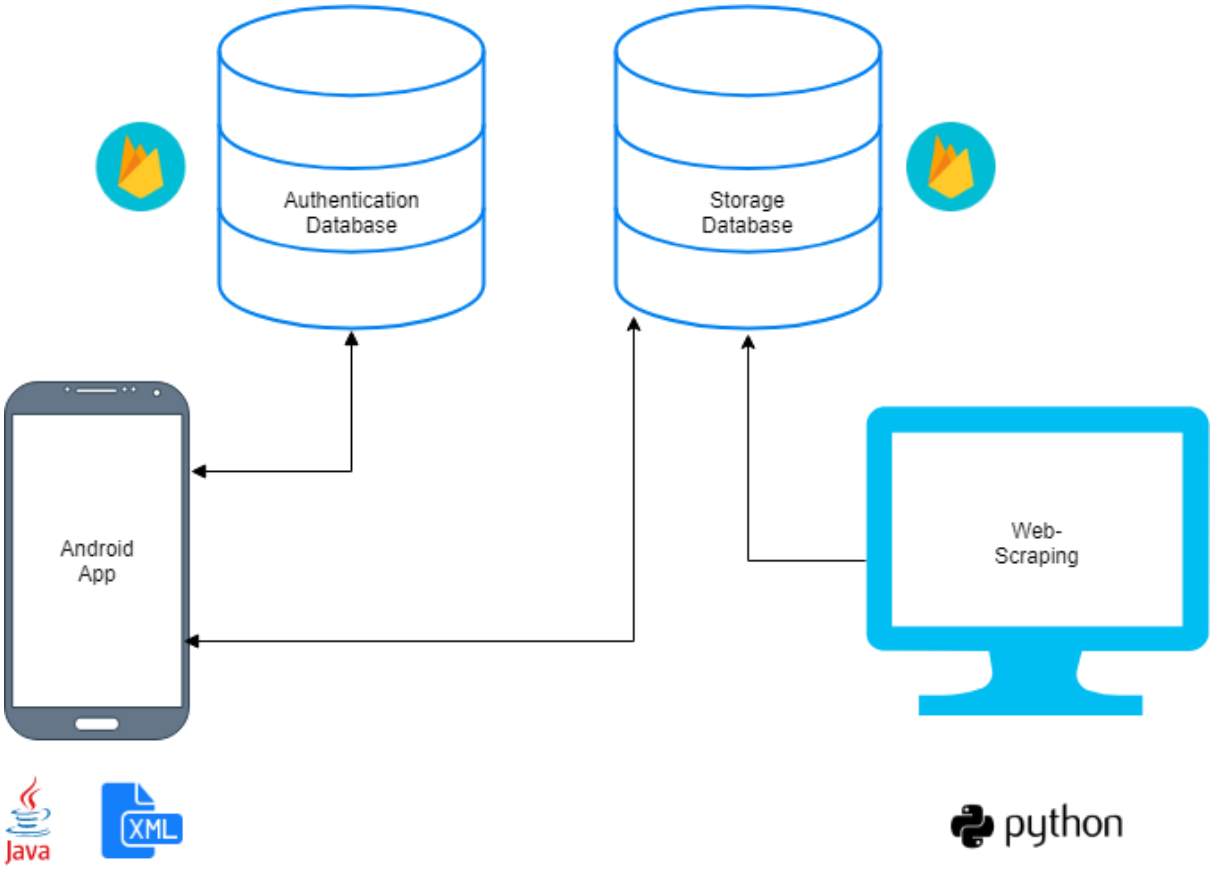
E-commerce
E-commerce is the process by which business and consumers buy and sell goods online.

API
Application Program Interface - is a set of subroutine definitions, protocols for building applications. It manages how two pieces of software interact and communicate with each other.

GUI
Graphical User Interface - It the user interface that allows the use of icons or other visual elements to interact with electronic devices other than using textual information.

Design

System Architecture



The system architecture involves a few key components as seen on the Component Diagram above. It consists of the following:

Android Application

The Android application is the primary component of the project, although it would not be able to operate without the help of the other components. The application was developed in Android Studio using Java and XML for the front-end. It uses Firebase to connect to the database and pull information.

Authentication Database

The Authentication Database is used to store user login information (email and password). This allows authentication for the user and it stored securely on Firebase.

Storage Database

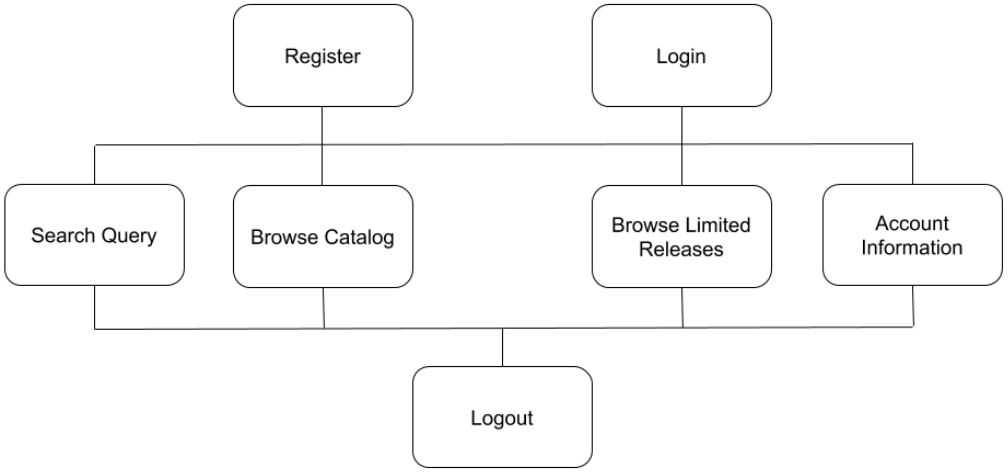
The Storage Database is broken into three sections: User, Catalogue and Limited Releases. When the User signs up, they will enter information like Name, Email and Age which is

stored in the User section. The Catalogue and Limited Releases section is built using data from the Python scripts and contains all the products.

Web-Scraping

The web-scraping is developed in Visual Studio Code using Python. It will query websites from Irish retailers and Instagram to retrieve information and populate the database.

High-Level Design



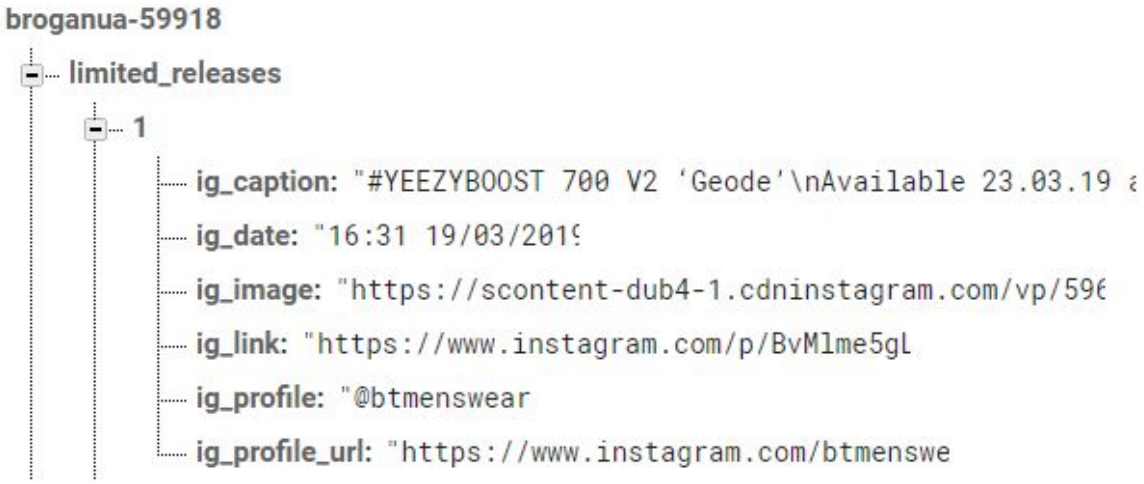
The high-level design of the systems functionality can be seen in the diagram above. The user will begin the application by registering with their information. Upon successful registration, they can proceed to login. Once logged in the user can navigate through the application by viewing the different sections in the application. Search for product, browse the catalog, browse limited releases and view/edit account. The user will be able to log out of the application too.

Database

Products

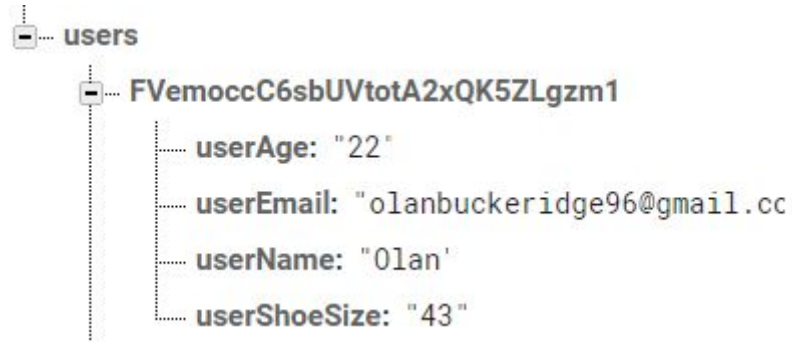
The diagram above show how the database is structured for the products on Firebase. Each product has a unique ID number and within that ID it contains information about the product used to display in the catalogue section of the Android application.

Limited Releases



The diagram above show how the database is structured for the limited releases on Firebase. Each product has a unique ID number and within that ID it contains information about the product used to display in the limited releases section of the Android application.

Users



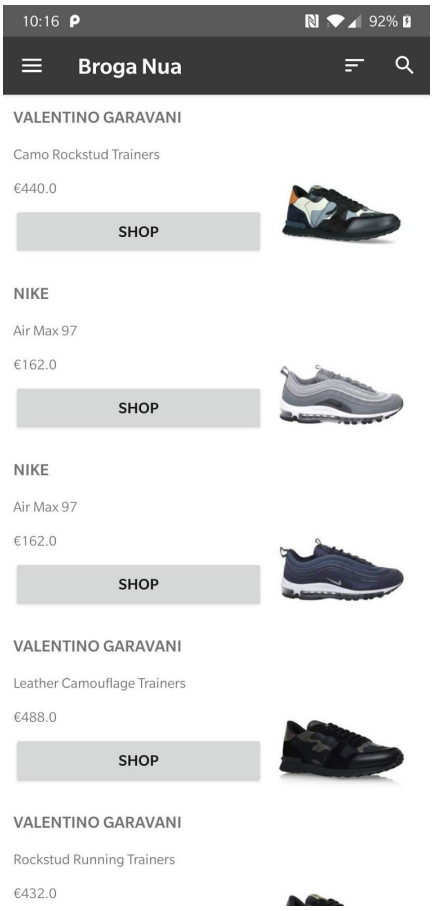
The diagram above show how the database is structured for the users on Firebase. Each user has a unique ID and within that ID it contains information about the user used to display in the profile section of the Android application.

Implementation

Catalogue

```
// Load and display products
progressBar = (ProgressBar) view.findViewById(R.id.progressBar);
progressBar.setVisibility(View.VISIBLE);
recyclerView = (RecyclerView) view.findViewById(R.id.catalogRecycler);
recyclerView.setLayoutManager(new LinearLayoutManager(getContext()));
productsList = new ArrayList<Products>();


// Retrieve products from Firebase.
reference = FirebaseDatabase.getInstance().getReference().child("products");
reference.addValueEventListener(new ValueEventListener() {
   @Override
   public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
       for (DataSnapshot dataSnapshot1 : dataSnapshot.getChildren()) {
           Products p = dataSnapshot1.getValue(Products.class);
           productsList.add(p);
       }
       //Collections.shuffle(productsList);
       adapter = new MyAdapter(getContext(), productsList);
       recyclerView.setAdapter(adapter);
       progressBar.setVisibility(View.INVISIBLE);

   }
```

CatalogueFragment.java



The CatalogueFragment.java is used to display the product catalog.

The products are retrieved from Firebase and selected from the products section. It interacts with the product adapter to build the arraylist required to display the products.

Once the products have been retrieved they are displayed in a recycler view of product cards.

I wanted to keep a clean user interface that focused on the products and only displayed the information important to the user such as Brand, Model, Price.

```
public Products(String brand, String model, double price, String retailer, String image,
String link) {
   this.brand = brand;
   this.model = model;
   this.price = price;
   this.retailer = retailer;
   this.image = image;
   this.link = link;
}
```

Products.java

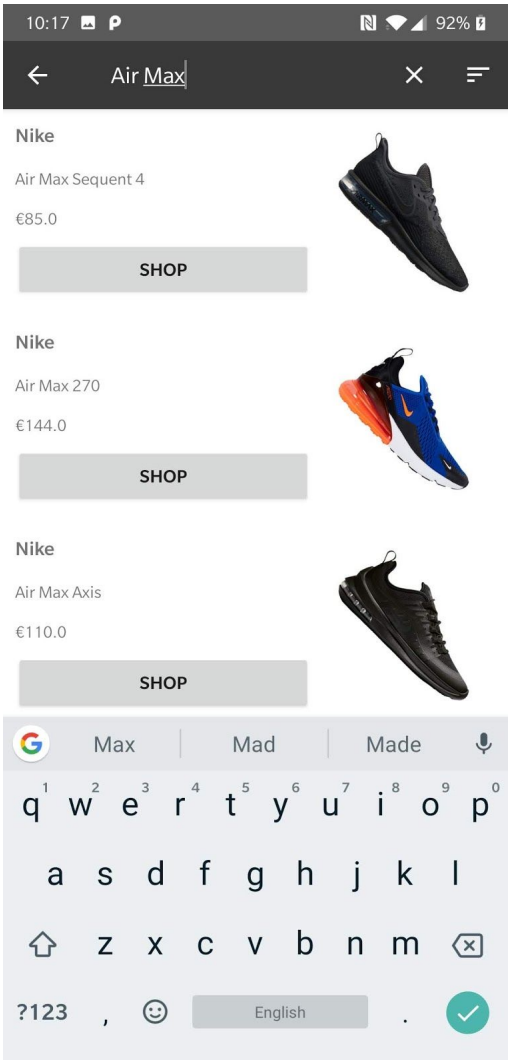
The Products.java class creates a template for the adapter to pull information from the Database.

```
public MyAdapter(Context c, ArrayList<Products> p)
{
   context = c;
   products = p;
   productsFull = new ArrayList<>(products);
}

@Override
public Filter getFilter() {
   return testFilter;
}
private Filter testFilter = new Filter() {
   @Override
   protected FilterResults performFiltering(CharSequence constraint) {
       ArrayList<Products> filteredList = new ArrayList<>();
       if (constraint == null || constraint.length() == 0){
           filteredList.addAll(productsFull);
       } else {
           String filterPattern = constraint.toString().toLowerCase().trim();

           for (Products item: productsFull) {
               if ((item.getModel().toLowerCase().contains(filterPattern)) ||
(item.getBrand().toLowerCase().contains(filterPattern)||
(item.getRetailer().toLowerCase().contains(filterPattern)))) {
                   filteredList.add(item);
               }
           }
       }
       FilterResults results = new FilterResults();
       results.values = filteredList;
       return results;
   }
   @Override
   protected void publishResults(CharSequence constraint, FilterResults results) {
       products.clear();
       products.addAll((ArrayList) results.values);
       notifyDataSetChanged();
   }
};
```

MyAdapter.java

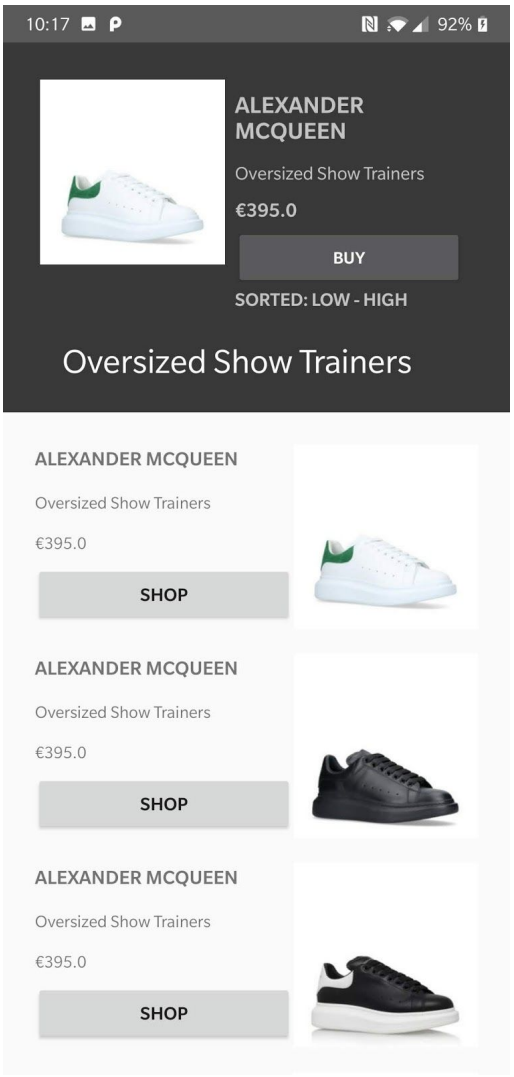
The adapter is used to retrieve the arraylist of products from Firebase primarily.

It also contains the search function which filters the list of products based on the user query.




```
// Receive Product from Catalogue fragment

String brand  = getIntent().getExtras().getString("brand");
final String model = getIntent().getExtras().getString("model");
String price = "€" + getIntent().getExtras().getDouble("price") ;
String image_url = getIntent().getExtras().getString("image") ;
ArrayList<Products> productsList =
(ArrayList<Products>)getIntent().getExtras().getSerializable("productsList");

// Display filtered product list.
productsRecyclerView = findViewById(R.id.productsRecycler);
productsRecyclerView.setLayoutManager(new LinearLayoutManager(ProductActivity.this));
Collections.sort(productsList, new PriceSorter());
adapter = new MyAdapter(ProductActivity.this, productsList);
adapter.getFilter().filter(model);
productsRecyclerView.setAdapter(adapter);
```

ProductActivity.java



The ProductActivity.java is used to display a product comparison page taken from the product catalogue.

It displays the initially selected product at the top of the page along with relevant information.

The user can purchase the sneakers using the ‘Buy’ button.

Below displays a comparison of all available models with the prices from low to high.

```python
# opening up connection, grabbing the page
uClient = uReq("https://www.lifestylesports.com/ie/mens-trainers/?sz=72&start={}".format(items))
page_html = uClient.read()
uClient.close()
# html parsing
page_soup = soup(page_html, "html.parser")
# grabs each product
containers = page_soup.findAll("div", {"class":"product-tile with-quick-buy"})
for container in containers:
    item_ref = db.reference('/products/{}'.format(products))
    # Brand Container
    brand_container = container.findAll("a", {"class":"name-link"})
    brand = brand_container[0].div.text.strip().split("\n",1)[0]
    # Model Container
    name_container = container.findAll("a", {"class":"name-link"})
    if "Mens" in name_container[0].text.strip():
        model = name_container[0].text.strip().split("Mens ",1)[1]
    elif "Adults" in name_container[0].text.strip():
        model = name_container[0].text.strip().split("Adults ",1)[1]
    # Price Container
    price_container = container.findAll("div", {"class":"product-pricing"})
    current_price = price_container[0].text.strip()
    price = current_price.split("\n")[0]
    # Image Container
    image_container = container.findAll("div", {"class":"product-image"})
    image_container[0].find("img")
    img = image_container[0].find("img")
    if img.has_attr('data-src'):
        img_url = img["data-src"]
    else:
        img_url = img['src']
    # Link Container
    link_container = container.findAll("a", {"class":"quickbuy"})
    prod_url = link_container[0]['href']
    retailer = "Life Style Sports"
    item_ref.set ({
        'brand': brand,
        'model': model,
        'retailer': retailer,
        'price': float(price.strip("€")),
        'image': img_url,
        'link': prod_url
    })
    products += 1
```

getProducts.py

Above is an example from the python code that scrapes products from Irish retailers.
It opens up a connection to the retailer and goes through the HTML locating the relevant information for products.

Once all the information has been retrieved, it is uploaded to the Firebase database for storage so that the Android application can interact with it.

Limited Releases

```
profile = 'btmenswear'
posts = instaloader.Profile.from_username(L.context, profile).get_posts()

for post in posts:
    item_ref = db.reference('/limited_releases/{}'.format(products))
    if ('RAFFLE' in post.caption or 'First come' in post.caption or 'a chance' in post.caption):
        link = "https://www.instagram.com/p/"+post.shortcode
        ig_profile = "@"+profile
        profile_url = "https://www.instagram.com/"+profile
        img_url = post.url
        caption = post.caption
        post_date = post.date.strftime('%H:%M %d/%m/%Y')
        products += 1
        item_ref.set({
            'ig_profile': ig_profile,
            'ig_profile_url': profile_url,
            'ig_image': img_url,
            'ig_caption': caption,
            'ig_date': post_date,
            'ig_link': link
        })

    else:
        pass
retailer_no += 1
```

getLimitedReleases.py

Above is an example from the python code that scrapes posts from Irish retailers.
It opens up a connection to the Instagram and goes through the javascript locating the relevant information for products.

Once all the information has been retrieved, it is uploaded to the Firebase database for storage so that the Android application can interact with it.

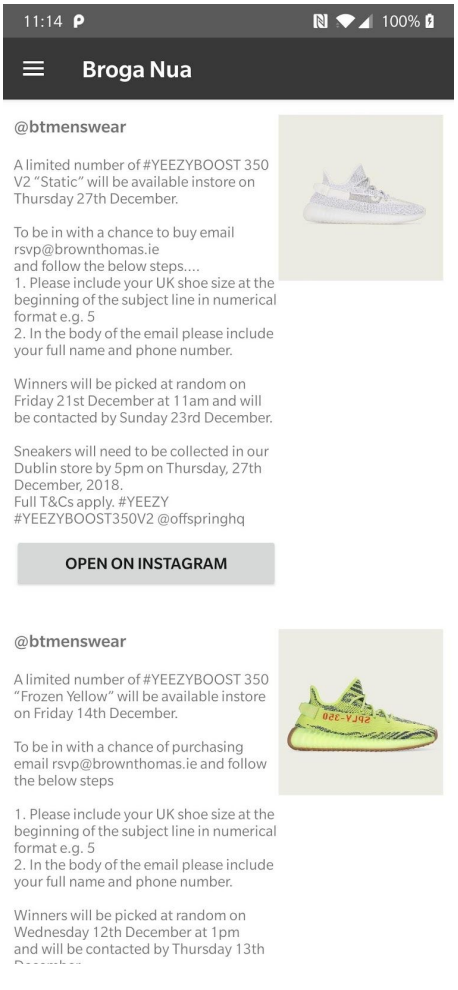
```
public Releases(String ig_profile, String ig_profile_url, String ig_link, String
ig_image, String ig_caption, String ig_date) {
   this.ig_profile = ig_profile;
   this.ig_profile_url = ig_profile_url;
   this.ig_link = ig_link;
   this.ig_image = ig_image;
   this.ig_caption = ig_caption;
   this.ig_date = ig_date;
}
```

Releases.java

The adapter is used to retrieve the arraylist of releases from Firebase primarily.

```
// Load and display products
recyclerView = (RecyclerView) view.findViewById(R.id.newsRecycler);
recyclerView.setLayoutManager(new LinearLayoutManager(getContext()));
releasesList = new ArrayList<Releases>();

// Retrieve products from Firebase.
reference = FirebaseDatabase.getInstance().getReference().child("limited_releases");
reference.addValueEventListener(new ValueEventListener() {
   @Override
   public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
       for (DataSnapshot dataSnapshot1 : dataSnapshot.getChildren()) {
           Releases r = dataSnapshot1.getValue(Releases.class);
           releasesList.add(r);
       }
       adapter = new ReleaseAdapter(getContext(), releasesList);
       recyclerView.setAdapter(adapter);
   }

   @Override
   public void onCancelled(@NonNull DatabaseError databaseError) {
       Toast.makeText(getContext(), databaseError.getCode(), Toast.LENGTH_SHORT).show();
   }
});
```

ReleasesFragment.java



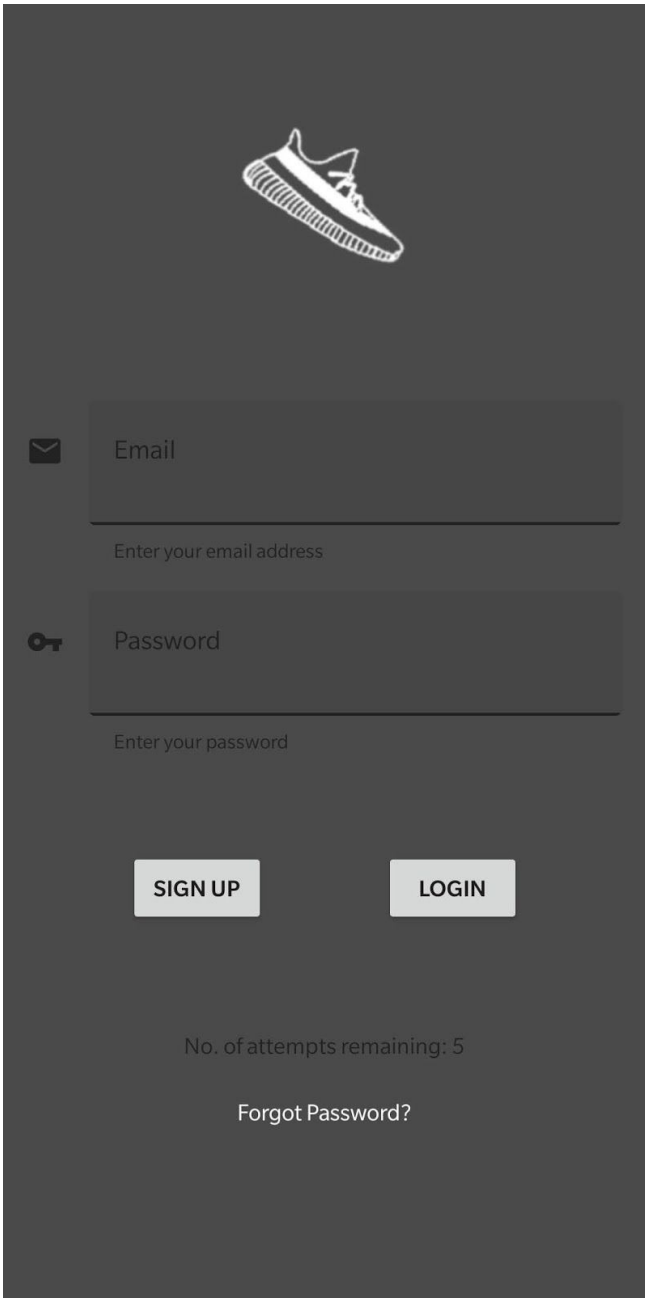
The ReleasesFragment.java is used to display the limited releases section.

The products are retrieved from Firebase and selected from the limited releases section. It interacts with the releases adapter to build the arraylist required to display the products.

Once the products have been retrieved they are displayed in a recycler view of product cards.

The user interface focuses on the products and only displayed the information important to the user.

Authentication & User Profile



```
private void validate(String userName, String userPassword) {

   progressDialog.setMessage("Validating user...");
   progressDialog.show();

   firebaseAuth.signInWithEmailAndPassword(userName,
userPassword).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
       @Override
       public void onComplete(@NonNull Task<AuthResult> task) {
           if (task.isSuccessful()) {
               progressDialog.dismiss();
               checkEmailVerification();
           }
           else {
               Toast.makeText(LoginActivity.this, "Login Failed.",
Toast.LENGTH_SHORT).show();
               attempts--;
               Info.setText("No. of attempts remaining: " + String.valueOf(attempts));
               progressDialog.dismiss();
               if (attempts == 0) {
                   Login.setEnabled(false);
               }

           }
       }
   });
}
private void checkEmailVerification () {
   FirebaseUser firebaseUser = firebaseAuth.getInstance().getCurrentUser();
   Boolean emailFlag = firebaseUser.isEmailVerified();

   if (emailFlag) {
       finish();
       startActivity(new Intent(LoginActivity.this, MainActivity.class));
   }
   else {
       Toast.makeText(this, "Please verify your email.", Toast.LENGTH_SHORT).show();
       firebaseAuth.signOut();
   }
}
```

LoginActivity.java

The login activity is used to verify the user information.

```
private void sendUserData () {
   FirebaseDatabase firebaseDatabase = FirebaseDatabase.getInstance();
   DatabaseReference myRef =
firebaseDatabase.getReferenceFromUrl("https://broganua-59918.firebaseio.com/users/").chi
ld(firebaseAuth.getUid());
   StorageReference imageReference =
storageReference.child(firebaseAuth.getUid()).child("Images").child("Profile Picture");
//User ID/Images/profile_pic
   UploadTask uploadTask = imageReference.putFile(imagePath);
   uploadTask.addOnFailureListener(new OnFailureListener() {
       @Override
       public void onFailure(@NonNull Exception e) {
           Toast.makeText(RegistrationActvity.this, "Upload failed",
Toast.LENGTH_SHORT).show();
       }
   }).addOnSuccessListener(new OnSuccessListener<UploadTask.TaskSnapshot>() {
       @Override
       public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {
           Toast.makeText(RegistrationActvity.this, "Upload successful",
Toast.LENGTH_SHORT).show();
       }
   });
   UserProfile userProfile = new UserProfile(age, email, name, shoeSize);
   myRef.setValue(userProfile);
}
```

RegistrationActivity.java

The registration activity allows the user to input all details for example profile picture, name, email. It uploads data to Firebase upon completion and sends the user a verification email.

Problems and Resolution

Learning New Technology

Before this year, I had never used Python or implemented a database for a project. Although I had dabbled with Python in some modules, it was a big undertaking to build the back-end and database using Python scripts.

This was my second Android application but I learned a lot from my last project that allowed me to be more mindful when designing and developing the application. In order to get a grasp on web scraping with Python and learning to build my Database, I used YouTube Tutorials and blog posts to help me.

Web Scraping

Web-scraping and data retrieval was completely new to me. I encountered a lot of issues while beginning to retrieve product data.

One of the biggest issues I had was figuring out how to retrieve information from multiple pages. I used Python with BeautifulSoup (BS) in order to scrape the data but BS can only read one page at a time. Most websites use AJAX to allow infinite scrolling web pages until all products are displayed but BS will not continuously scroll. This mean that only the first 12 products for example were being retrieved.

The solution to this problem was editing the URL to change what products were being loaded and using a loop to iterate through the retailers web pages until all the products were retrieved and stored in the database.

Databases

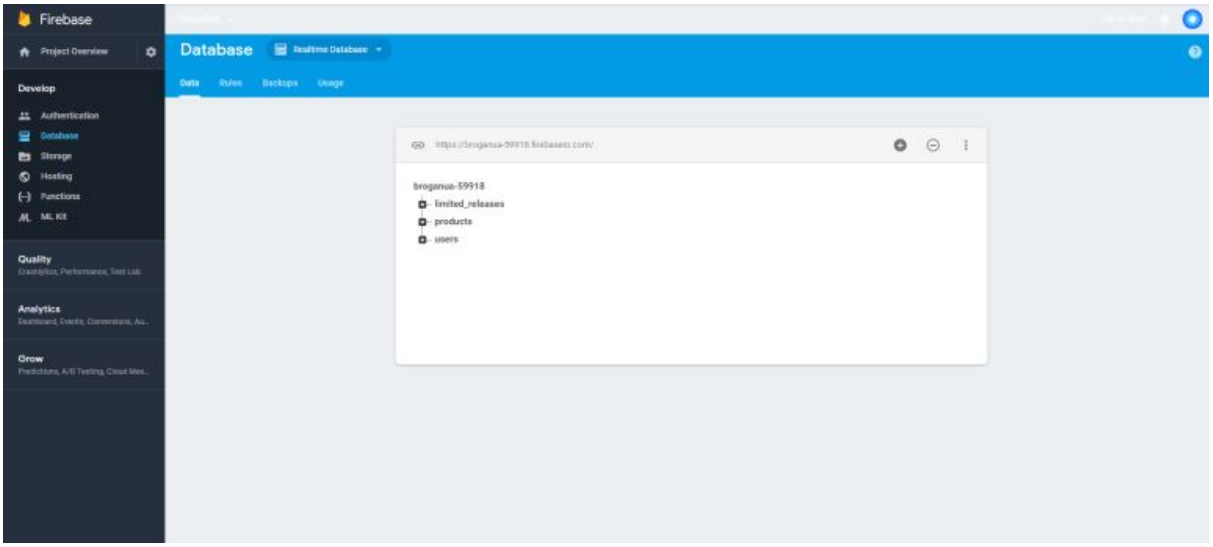
mysql> select * from products;

id	brand	model	retailer	price	images
1	'Puma'	' RS-X Re-Invention '	'Nowhere'	'110.00'	'cdn.shopify.com/s/files/1/1610/0407/products/369579_04_540x540.jpg?v=1551786517'
2	'Puma'	' RS-X Re-Invention '	'Nowhere'	'110.00'	'cdn.shopify.com/s/files/1/1610/0407/products/puma-rs-x-reinvention-green-369579-05_540x540.jpg?v=1551708840'
3	'Nike'	' Zoom Alpha '	'Nowhere'	'140.00'	'cdn.shopify.com/s/files/1/1610/0407/products/nike-zoom-alpha-retro-BQ8800-400-5_540x540.jpg?v=1551103404'
4	'Nike'	' Zoom Alpha '	'Nowhere'	'140.00'	'cdn.shopify.com/s/files/1/1610/0407/products/Nike-Zoom-Alpha-Retro-BQ8800_003-5_4c6e413a-e0eb-41af-bf76-aabc699963ad_540x540.jpg?v=1551103221'
5	'Nike'	' Air Force 1 '07 Premium '	'Nowhere'	'130.00'	'cdn.shopify.com/s/files/1/1610/0407/products/Nike-Air-Force-1-07-Premium-3-Pink-AT4144-600_540x540.png?v=1551098008'
6	'Puma'	' SF CELL Venom '	'Nowhere'	'130.00'	'cdn.shopify.com/s/files/1/1610/0407/products/370338_02_540x540.jpg?v=1551094643'
7	'Puma'	' SF CELL Venom '	'Nowhere'	'130.00'	'cdn.shopify.com/s/files/1/1610/0407/products/370338_01_540x540.jpg?v=1551094515'
8	'Nike'	' Air Force 1 '07 Premium '	'Nowhere'	'130.00'	'cdn.shopify.com/s/files/1/1610/0407/products/Nike-Air-Force-1-07-Premium-3-Black-AT4144-001_540x540.png?v=1550596194'
9	'New Balance'	' M9919 '	'Nowhere'	'220.00'	'cdn.shopify.com/s/files/1/1610/0407/products/M9919EC_540x540.jpg?v=1551096525'
10	'New Balance'	' M1500EC '	'Nowhere'	'210.00'	'cdn.shopify.com/s/files/1/1610/0407/products/M5400EC_5_540x540.JPG?v=1551096542'
11	'Puma X Mita'	' Cell Venom "Stealth"'	'Nowhere'	'160.00'	'cdn.shopify.com/s/files/1/1610/0407/products/mita-sneakers-puma-cell-venom-stealth-release-1_540x540.jpg?v=1550151415'
12	'Nike'	' Blazer Mid '77 '	'Nowhere'	'100.00'	'cdn.shopify.com/s/files/1/1610/0407/products/BQ6806-600_540x540.jpg?v=1549731637'
13	'Nike'	' Air Max 95 '	'Nowhere'	'170.00'	'cdn.shopify.com/s/files/1/1610/0407/products/CD1529-001_540x540.jpg?v=1549550216'
14	'Nike'	' Air Max 98 '	'Nowhere'	'180.00'	'cdn.shopify.com/s/files/1/1610/0407/products/CD1538-100_540x540.jpg?v=1549549912'
15	'NikeLab'	' Zoom Vomero 5 SP '	'Nowhere'	'150.00'	'cdn.shopify.com/s/files/1/1610/0407/products/nike-zoom-vomero-5-black-BV1358-002-4_540x540.jpg?v=1548869444'
16	'NikeLab'	' Zoom Vomero 5 SP '	'Nowhere'	'150.00'	'cdn.shopify.com/s/files/1/1610/0407/products/nike-zoom-vomero-5-white-BV1358-001-6_540x540.jpg?v=1548869428'
17	'Nike'	' Air Max 98 '	'Nowhere'	'180.00'	'cdn.shopify.com/s/files/1/1610/0407/products/IMG_6731_540x540.JPG?v=1548695607'

Initially I was intending on using MySQL to store my data for the application. I started using my Python scripts to write the information into the MySQL Database. The database was being hosted on Amazon Web Services.

After having some trouble connecting the database to my Android application, I decided to move the database to Firebase as it was early enough in development. The documentation on Firebase is very in depth and has clear information on how to connect to your Android application. I was using Firebase for authentication anyway and found this to be the best

route.



Product Comparison

While developing the product comparison section, I ran into issues with the performance of the application. It would take 5-8 seconds to load the page which was far from ideal. The rest of the application was quick so I didn't want this bringing down the application as one of the most important features.

The issue was that the activity was loading a new Firebase instance when opening. This made it download the entire database again and then it would filter that instance to the required model. The solution was to implement an adapter that allowed the database from the Catalogue to be carried over between activities, this reduced the loading time from 5-8 seconds, down to under 1 second.

Results & Future Work

Future Work

This project has massive scope for scaling in the future and monetary potential. There are multiple areas which could be implemented to improve the application. These are outlined below.

Developing Cross Platform

Developing a Cross Platform application would be hugely beneficial to the growth and reach of this application. While Android contains a huge chunk of the user market, being able to reach the iOS users and having a web application would open up the application to a wider audience.

There are no features that are specific to Android that would stop me from achieving this in the future although I felt it would be out of the scope of the final year project with time constraints and other modules.

Increasing Number of Retailers
Although this application is focused at Irish retailers (All major Irish retailers are in the application) - I believe the application could be scaled globally. This would allow localised versions to be developed and reaching a much larger market.

The issues with developing for more retailers would be resources - hardware and time. I believe the concept could be applied to many countries in the world without having to change too much in the application. Time and resources are the main factors why this could not be achieved in the project.

Identify Clothing - Machine Learning
Providing the application was scaled correcting and a large database was in place, I would love to implement a clothes identification service. This would allow users to take a picture of an item they are looking for or upload a screenshot from Instagram for example, and the application would return similar options or the exact match.

Currently I don't have the resources or infrastructure to implement this successfully but I believe it is certainly an option to aim for in the future.

Final Review

Overall, I am very satisfied with how my project unfolded and the result of my final application. All of my initial requirements have been executed successfully and was spread out which allowed me to focus on other modules while working on the project. The goal was to develop an application that would save "Sneakerheads" more time and provided an intuitive interface and e-commerce service. I feel like I have achieved this and I am happy with the final product.

Conducting user testing and researching the market has provided great benefit to the development of the product and gave me great confidence for the future of the application.

Working on a fashion/ sneakers application was always an interest of mine as that is my biggest interest outside of development. The project allowed me to gain great experience in Android development and working with e-commerce platforms.

Appendices

1. Android Studio - https://developer.android.com/studio
2. Visual Studio Code - https://code.visualstudio.com/
3. Python - https://www.python.org/
4. Java - https://www.java.com/
5. XML - https://www.xml.com/
6. Firebase - https://firebase.google.com/
7. Google Drive - https://www.google.com/drive/