CET343 – Android Assignment

Nyi Zaw

Student no. 239349271

User ID – bi55is

Computing systems engineering

Word Count: 2145

Analysis

There are many factors to consider when designing and developing mobile apps. However, the most important ones include operating systems, programming languages, memory configurations and compliance with native mobile standards. The two main players in the operating system space are iOS and Android. The standardized and regulated environment that iOS provides, which is only available on Apple products, guarantees a seamless user experience across all devices. In contrast, Android is more customizable due to its open source nature and versatility, but this also leads to device fragmentation. Target market, financial constraints and development preferences all influence which of these two options is better.

iOS is primarily written in Swift, a programming language developed by Apple that is known for its ease of learning and security features. Java has long been a popular option for Android, but Kotlin is gaining popularity due to its conciseness and innovative features. The choice of programming language is often influenced by the chosen platform, the requirements of the project and the experience of the development team. Cross-platform frameworks have become effective replacement solutions, allowing developers to build code once and publish it across iOS and Android platforms. Examples of these frameworks include Flutter, which uses Dart, and React Native, which uses

JavaScript. These frameworks may not work as well as native programming, although they save time and money.

Memory architecture is a crucial element that affects both program speed and user experience. For data persistence in iOS applications, Apple's Core Data Framework is often used, which provides a reliable way to manage the object graph and handle changes. SQLite is a popular local storage solution used by Android developers due to its efficiency and integration into the ecosystem. Platform-agnostic cloud storage options like Firebase and AWS offer scalability and seamless device synchronization. The decision between local storage and cloud storage is influenced by several factors, including the need for real-time updates, program complexity, and data sensitivity.

How effectively an app conforms to the platform's design principles and connects to the operating system depends largely on the native mobile standards. iOS emphasizes consistency, clarity and intuitive design according to the Human Interface Guidelines (HIG). Android emphasizes a strong, visual user experience and is influenced by Material Design concepts. Following these guidelines ensures a natural appearance and increases user acceptance and familiarity. Although cross-platform frameworks attempt to provide a consistent design, developers often need to customize the UI/UX to accommodate the different features of each platform, which can complicate the development process.

Given benchmarking, iOS app development offers a simplified and regulated environment, making it a great option for developers and companies that value a consistent user experience. However, the limited ecosystem structure and strict App Store guidelines can be limiting. Android has issues with device fragmentation, but its open environment makes it attractive to a broader user base and allows for greater customization. Choosing between cross-platform frameworks like Flutter and React Native or between Swift and Kotlin depends on project requirements, team experience, and development preferences. The choice of on-premises or cloud-based storage design should be made based on user expectations and application data requirements.

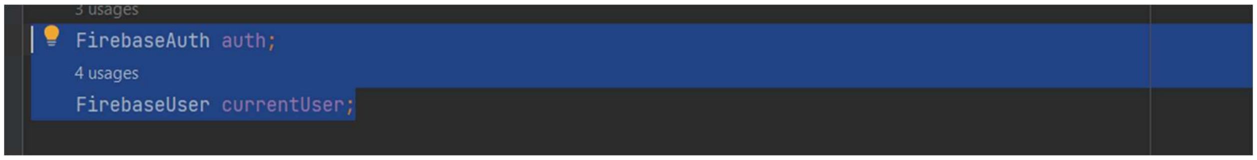
In summary, each step of the mobile app design and development process requires careful consideration of impacts. Creating powerful, user-friendly mobile apps requires balancing the advantages of iOS and Android, choosing the best programming language, implementing efficient storage solutions, and adhering to native mobile standards. The best course of action will be determined by the specific needs of the project, the target audience and the experience of the development team; This highlights the need for a methodical and strategic decision-making process.

Functionality

1. **Firebase Authentication (FirebaseAuth):**

Firebase Authentication is used for user login and session management.

auth is the entry point of the Firebase Authentication SDK, and currentUser represents the currently authenticated user.

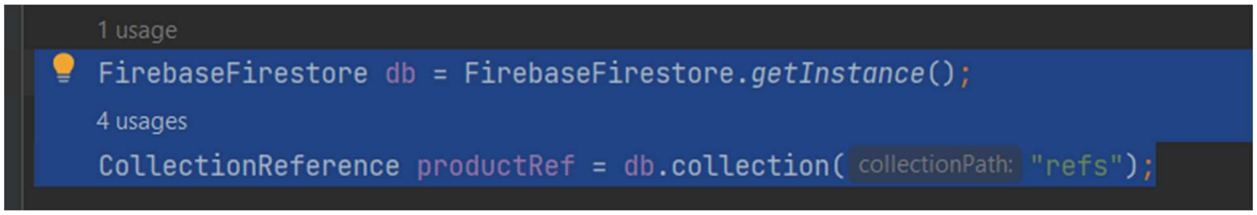


```
3 usages
FirebaseAuth auth;
4 usages
FirebaseUser currentUser;
```

2. **Real-time Database Integration (FirebaseFirestore):**

Firestore is utilized to store and retrieve product information.

productRef is a reference to the collection "refs" where product documents are stored.


```
1 usage
FirebaseFirestore db = FirebaseFirestore.getInstance();
4 usages
CollectionReference productRef = db.collection( collectionPath: "refs");
```

3. **Firebase Storage for Image Upload (FirebaseStorage):**

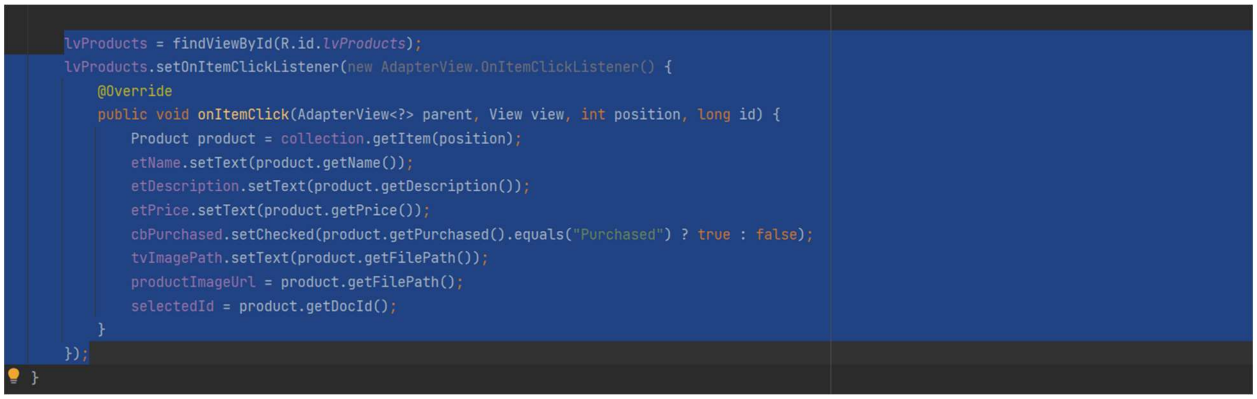
Firebase Storage is used to store and retrieve product images.

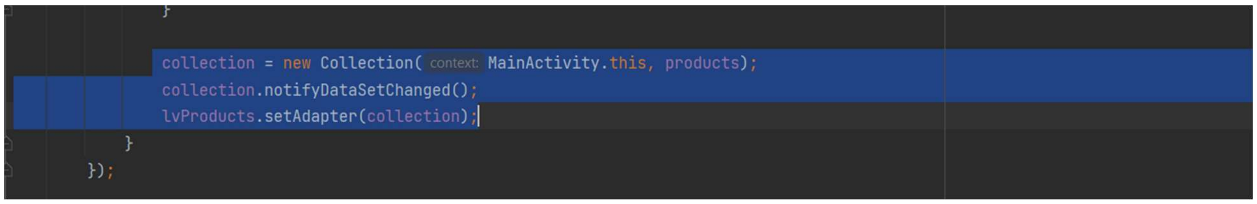
storageRef is a reference to the Firebase Storage location.

4. **ListView and Product Collection (ListView, Collection):**

The ListView (lvProducts) is populated with product data using a custom adapter (Collection).

Clicking on an item in the ListView triggers the display of product details for editing or deletion.

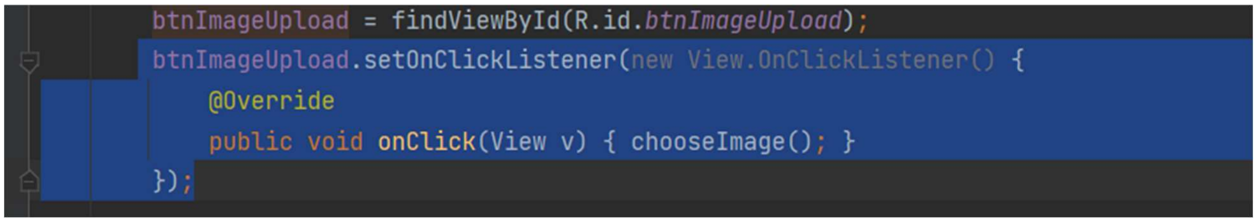

```
    lvProducts = findViewById(R.id.lvProducts);
    lvProducts.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
            Product product = collection.getItem(position);
            etName.setText(product.getName());
            etDescription.setText(product.getDescription());
            etPrice.setText(product.getPrice());
            cbPurchased.setChecked(product.getPurchased().equals("Purchased") ? true : false);
            tvImagePath.setText(product.getFilePath());
            productImageUrl = product.getFilePath();
            selectedId = product.getDocId();
        }
    });
}
```



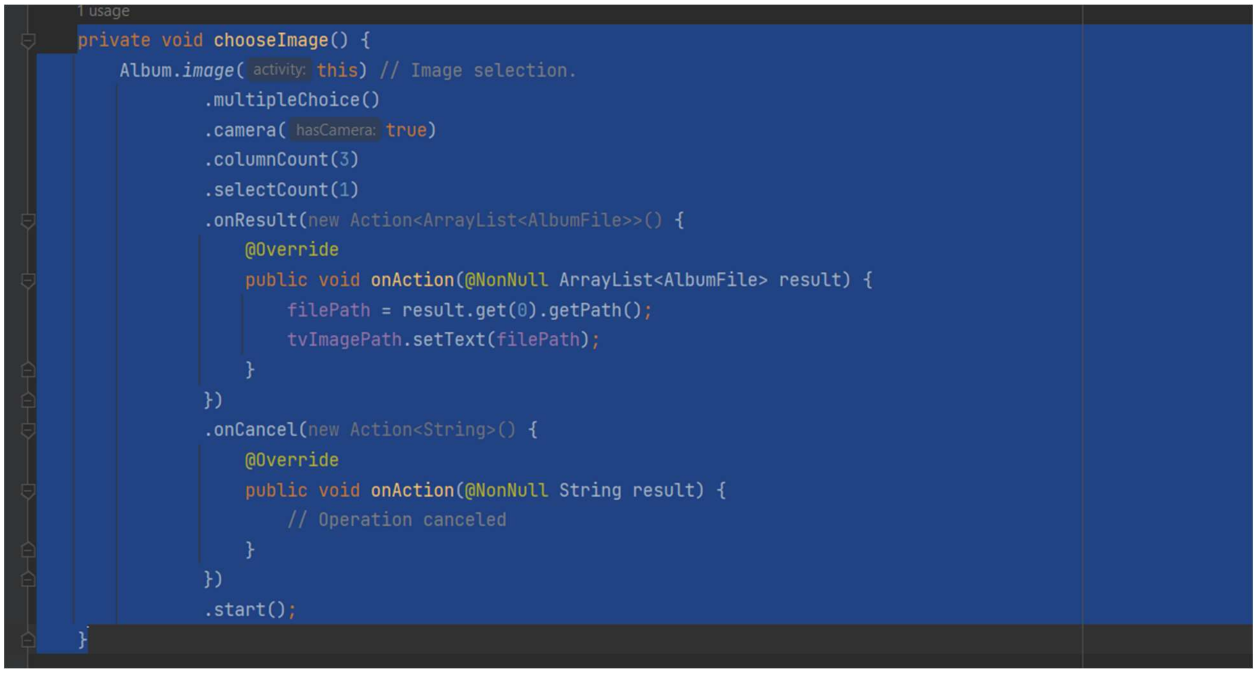
```
        }

        collection = new Collection( context: MainActivity.this, products);
        collection.notifyDataSetChanged();
        lvProducts.setAdapter(collection);
    }
});
```

5. **Image Selection using Album Library (com.yanzhenjie.album):**

Users can choose product images through the Album library, providing a user-friendly interface for image selection.



```java
        btnImageUpload = findViewById(R.id.btnImageUpload);
        btnImageUpload.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) { chooseImage(); }
        });
```



```java
1 usage
private void chooseImage() {
    Album.image( activity: this) // Image selection.
            .multipleChoice()
            .camera( hasCamera: true)
            .columnCount(3)
            .selectCount(1)
            .onResult(new Action<ArrayList<AlbumFile>>() {
                @Override
                public void onAction(@NonNull ArrayList<AlbumFile> result) {
                    filePath = result.get(0).getPath();
                    tvImagePath.setText(filePath);
                }
            })
            .onCancel(new Action<String>() {
                @Override
                public void onAction(@NonNull String result) {
                    // Operation canceled
                }
            })
            .start();
}
```

6. CRUD Operations and UI Interaction:

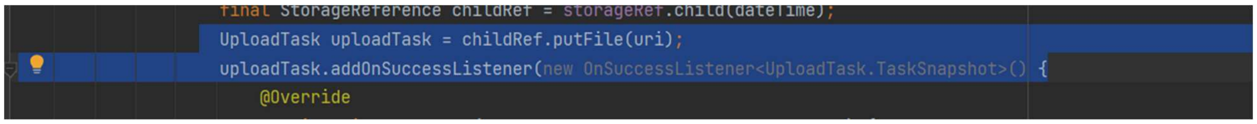
Buttons are associated with methods for adding, updating, and deleting products.

User interactions trigger these methods, leading to database updates.

7. **Handling Image Upload to Firebase Storage:**

When adding a product, if an image is selected, it is uploaded to Firebase Storage.

The download URL is then obtained and stored along with other product details.

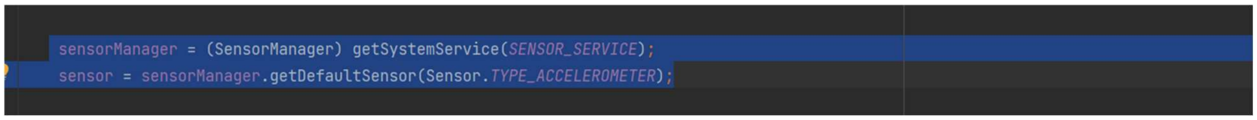


```
final StorageReference childRef = storageRef.child(dateTime);
UploadTask uploadTask = childRef.putFile(uri);
uploadTask.addOnSuccessListener(new OnSuccessListener<UploadTask.TaskSnapshot>() {
    @Override
```

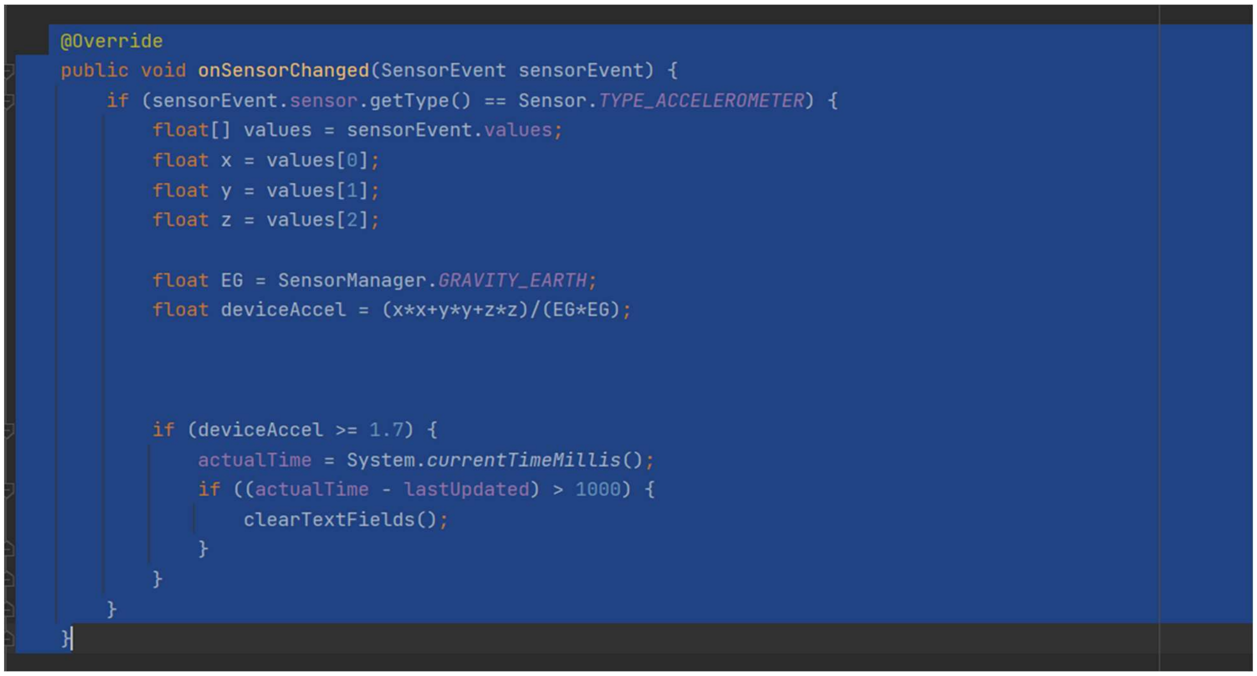
8. **Sensor Integration (Accelerometer):**

The accelerometer sensor is used to clear text fields when the device is shaken.

onSensorChanged is triggered when a significant shake is detected.



```
sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
sensor = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
```

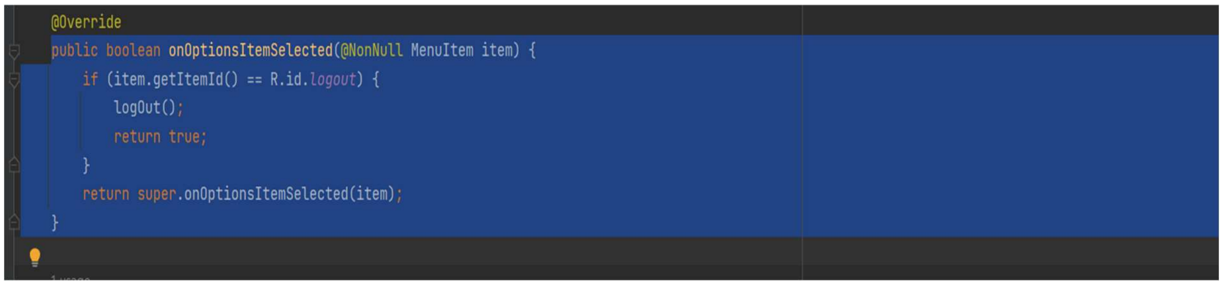


```
@Override
public void onSensorChanged(SensorEvent sensorEvent) {
    if (sensorEvent.sensor.getType() == Sensor.TYPE_ACCELEROMETER) {
        float[] values = sensorEvent.values;
        float x = values[0];
        float y = values[1];
        float z = values[2];

        float EG = SensorManager.GRAVITY_EARTH;
        float deviceAccel = (x*x+y*y+z*z)/(EG*EG);



        if (deviceAccel >= 1.7) {
            actualTime = System.currentTimeMillis();
            if ((actualTime - lastUpdated) > 1000) {
                clearTextFields();
            }
        }
    }
}
```

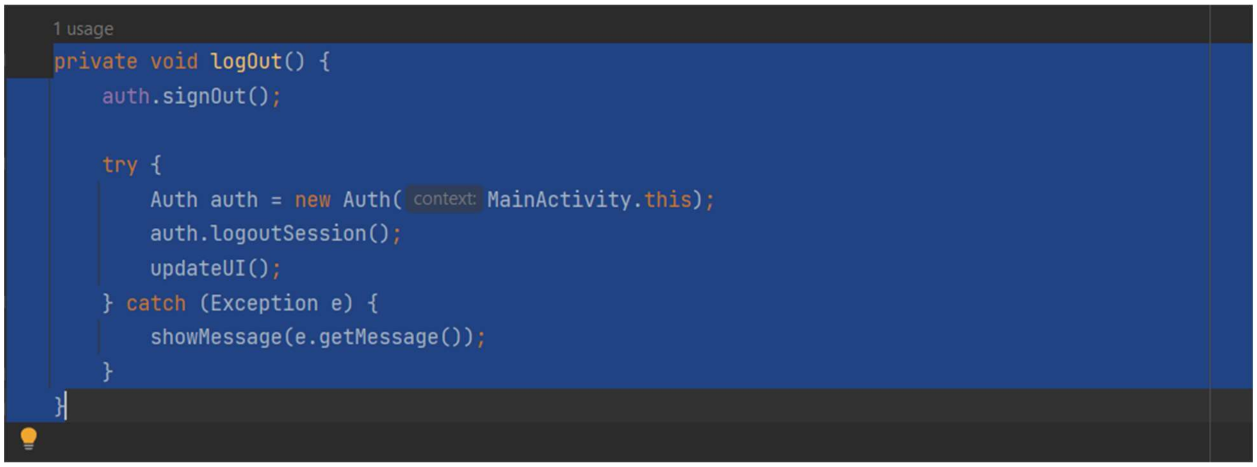
9. **Session Management and Logout:**

A logout option is available in the options menu, triggering the logOut() method.

User sessions are managed, and the UI is updated accordingly.



```
@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {
    if (item.getItemId() == R.id.logout) {
        logOut();
        return true;
    }
    return super.onOptionsItemSelected(item);
}
```



```
1 usage
private void logOut() {
    auth.signOut();

    try {
        Auth auth = new Auth( context: MainActivity.this);
        auth.logoutSession();
        updateUI();
    } catch (Exception e) {
        showMessage(e.getMessage());
    }
}
```

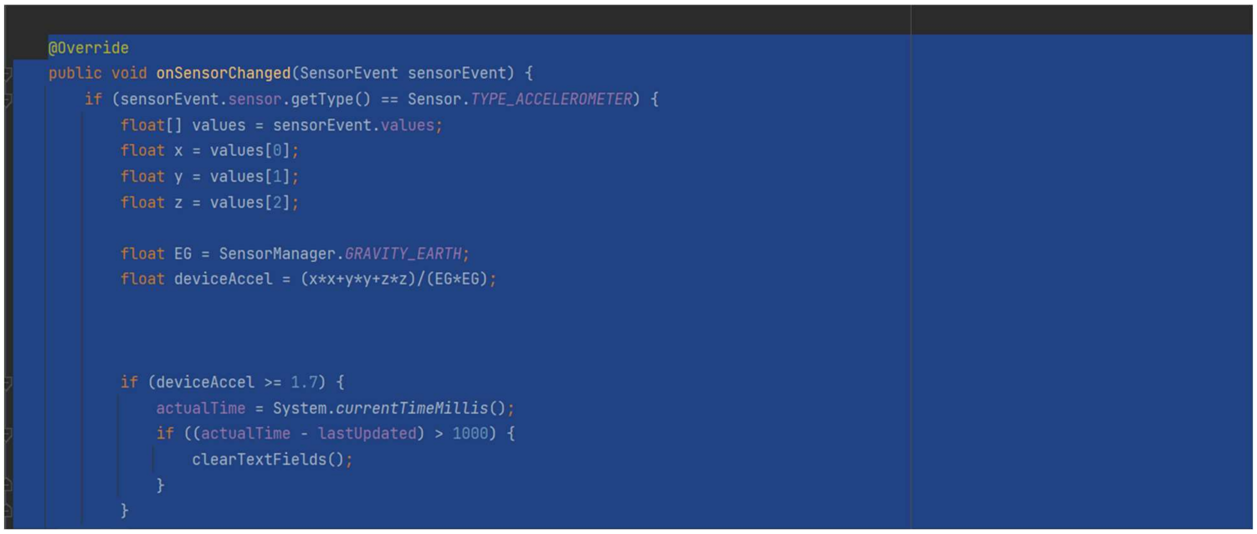
10. **Error Handling and Toast Messages:**

Throughout the code, try-catch blocks are used for error handling.

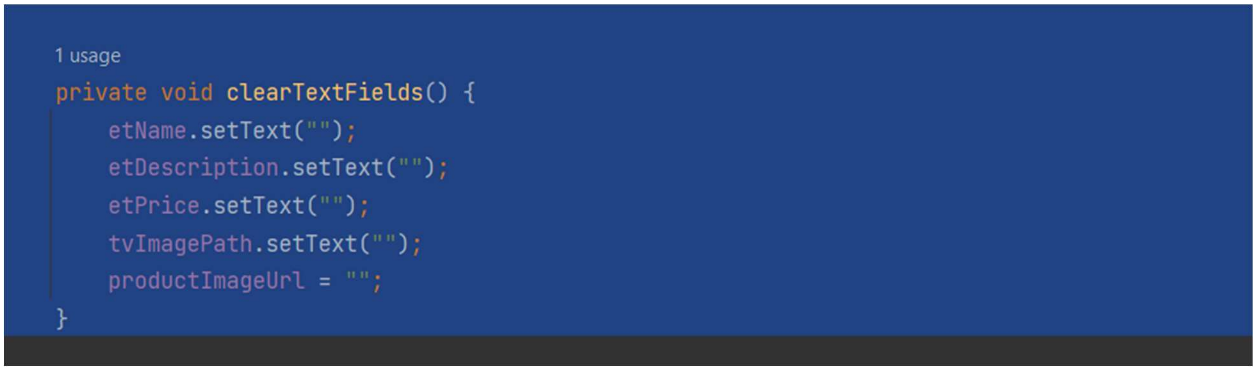
Toast messages are displayed to inform users about success or failure of operations.

11. Clear Text Fields on Shake:

The accelerometer is utilized to clear text fields when the device experiences a significant shake.



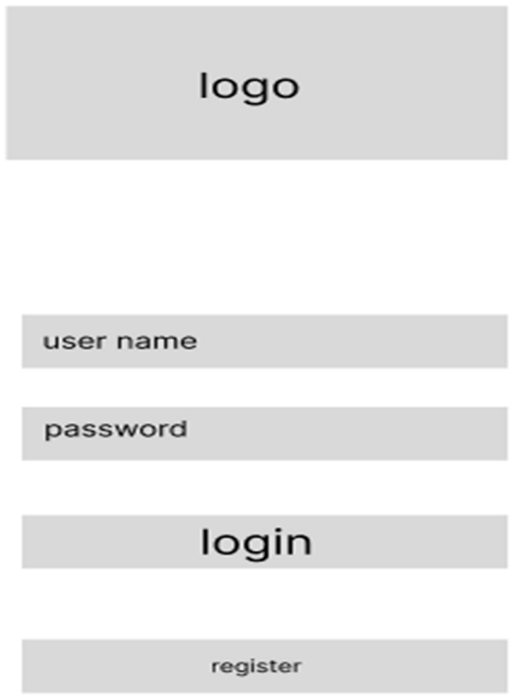
```
@Override
public void onSensorChanged(SensorEvent sensorEvent) {
    if (sensorEvent.sensor.getType() == Sensor.TYPE_ACCELEROMETER) {
        float[] values = sensorEvent.values;
        float x = values[0];
        float y = values[1];
        float z = values[2];

        float EG = SensorManager.GRAVITY_EARTH;
        float deviceAccel = (x*x+y*y+z*z)/(EG*EG);



        if (deviceAccel >= 1.7) {
            actualTime = System.currentTimeMillis();
            if ((actualTime - lastUpdated) > 1000) {
                clearTextFields();
            }
        }
```

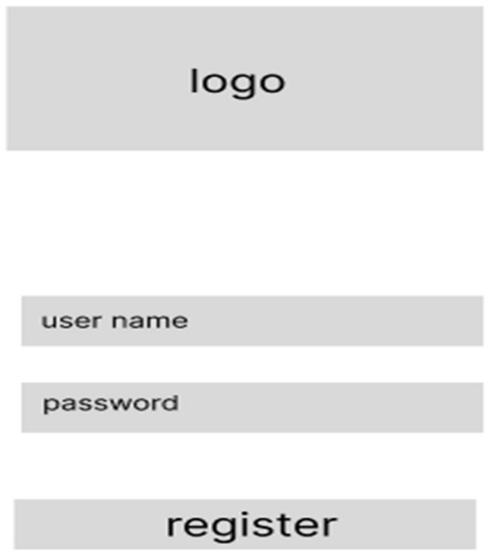


```
1 usage
private void clearTextFields() {
    etName.setText("");
    etDescription.setText("");
    etPrice.setText("");
    tvImagePath.setText("");
    productImageUrl = "";
}
```

Design

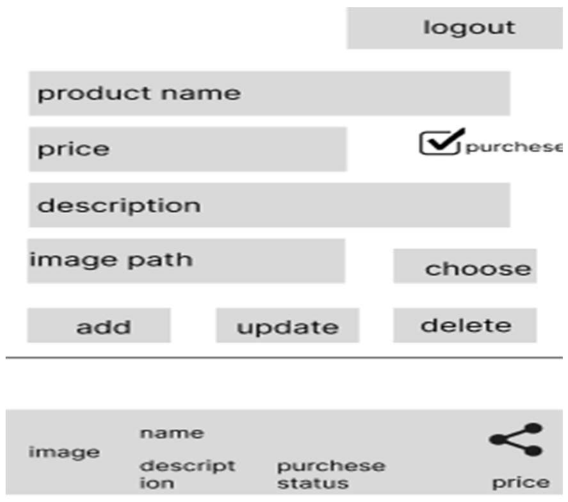
wireframe design





login page and register page

In register page user can create new account, all links to firebase to control authorization



Product page

Product can be added,update and deleted.All product information including a picture,with logout

The app wireframes were finalized after development. To save time and money, it is easier to evaluate early designs and make necessary changes before development begins if the design is good. The app was used before it was developed so I could learn through trial and error and gain the expertise needed to create the best Android app.

User authentication wireframes include screens for registration, login, and maybe a password recovery procedure. These windows provide buttons for submitting the form or switching between login and registration in addition to email and password entry boxes.

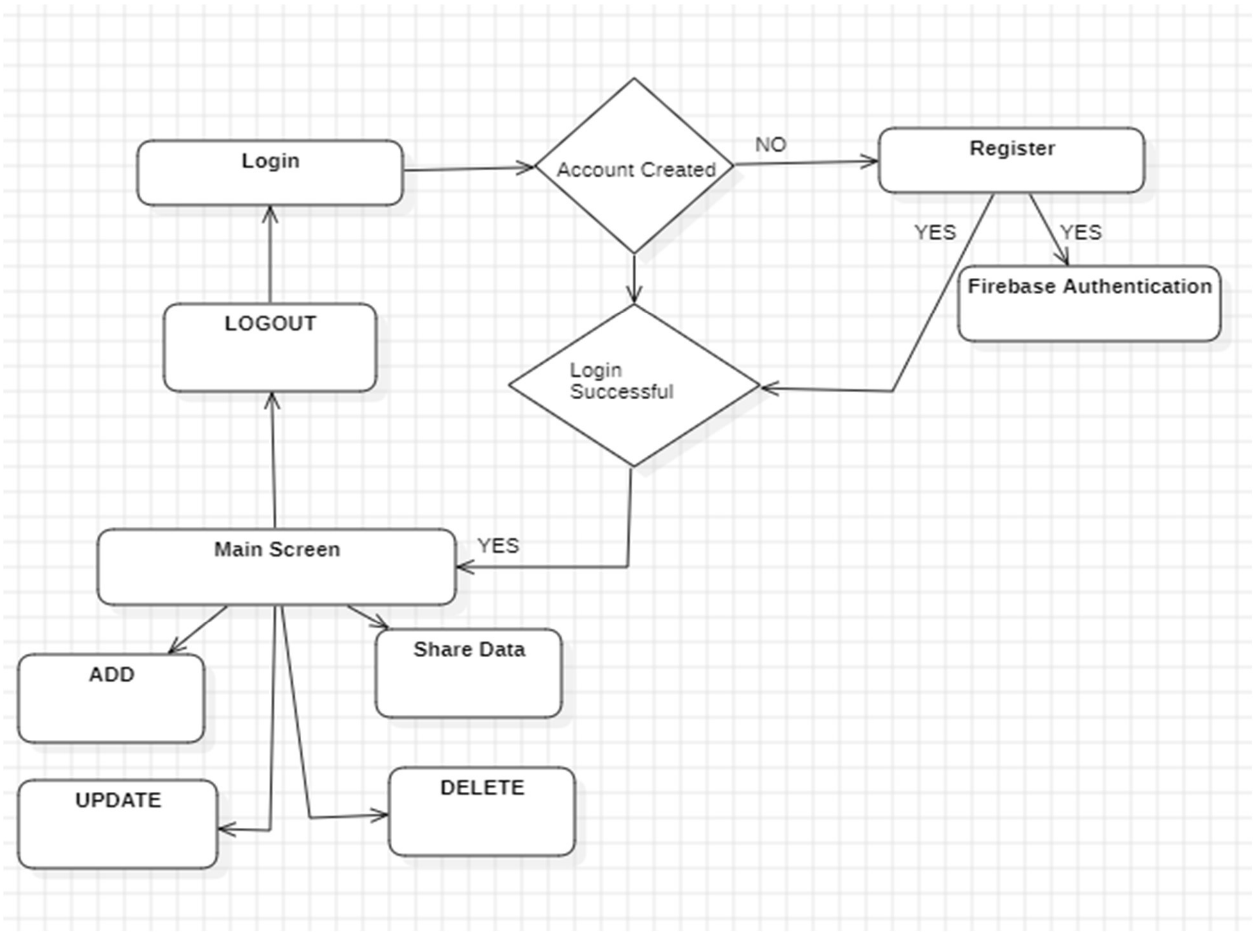
Each item is presented in the product list in an understandable and clear manner. Product names, short descriptions, prices, and checkboxes indicating whether the product was purchased are all included in the wireframe. Users can quickly identify goods by scrolling through the list.

The wireframe of the product information page shows a detailed perspective of a selected product. The product name, price, full description, an enlarged image and a check mark indicating the item as purchased are all included. In addition, there are buttons to edit and delete the product.

To add or edit products, wireframes display a form with fields for the product name, description, price, and the ability to submit an image. Users can indicate that they purchased the product by checking a box. There are obvious buttons to submit or cancel the form.

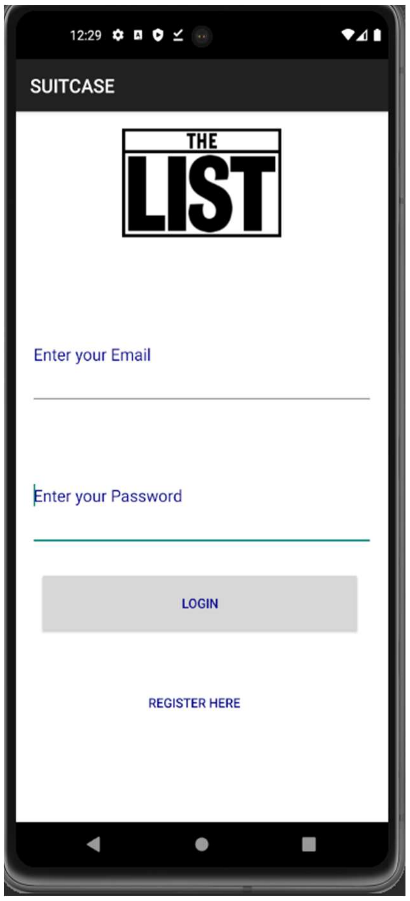
An interface that allows customers to select or take photos of their items is part of the image upload view wireframe. This may include a button to activate the camera on the device, a gallery to select from existing photos, and a panel to view the selected photo.

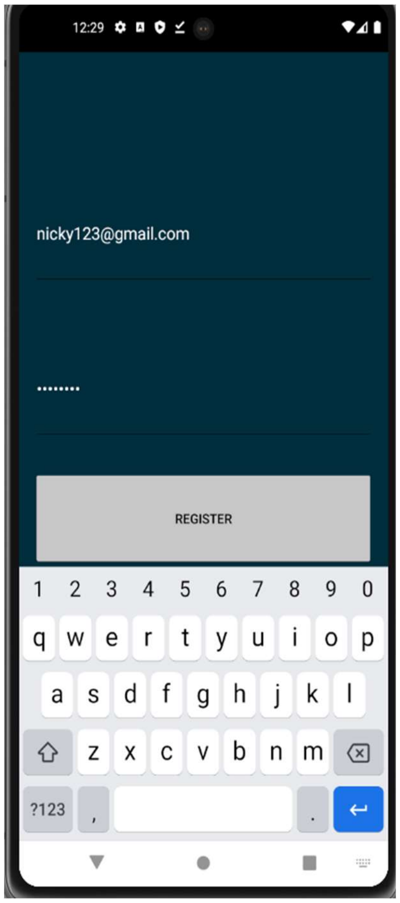
Hierarchy Design



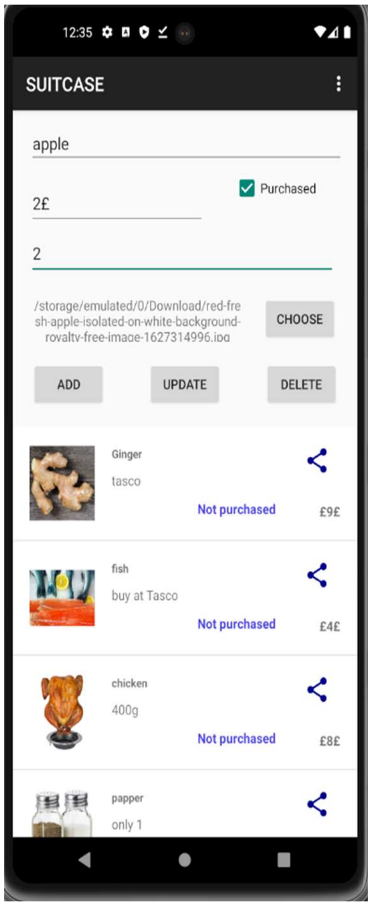
Project activity is crucial; Users can only access it through Gmail accounts, Firebase permission is set up, and users must log in and register. After successfully logging in, users will be presented with the product page where they can add, update, or remove products from their account. Here too, all item information is stored in the Firebase database and can be accessed later. The system also has a feature that allows you to send information to other people via email. If a friend or relative wants to buy the user something to help them deliver a newborn, this option comes in handy

Test Strategy and Test Results




nicky123@gmail.com

••••••••


apple

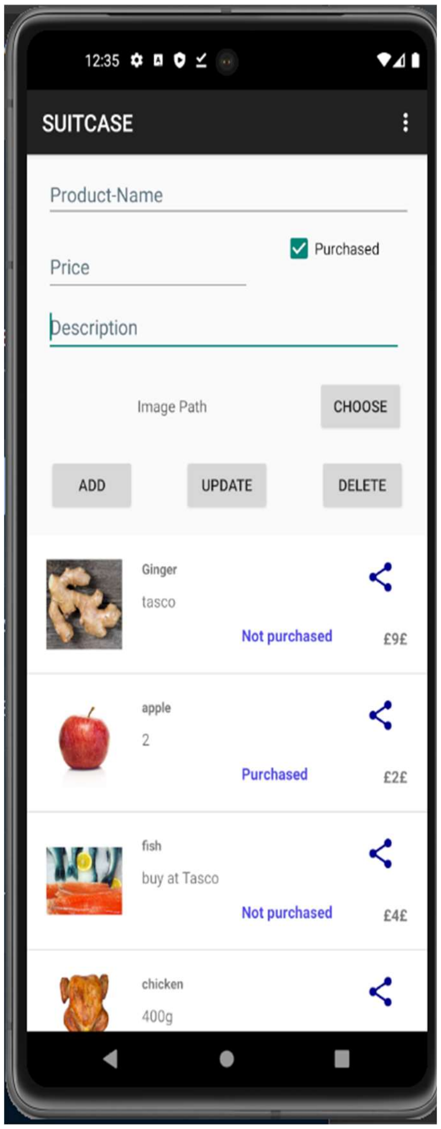
2£ ✓ Purchased

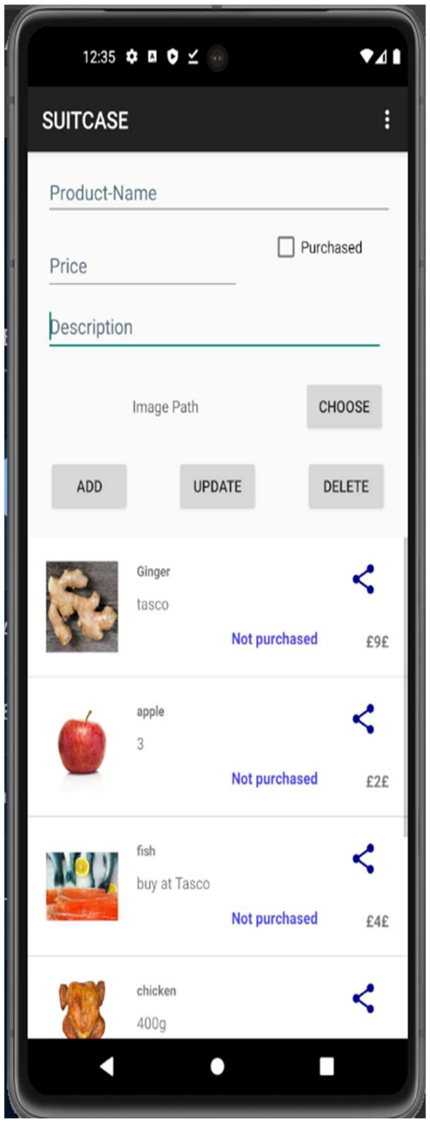
2

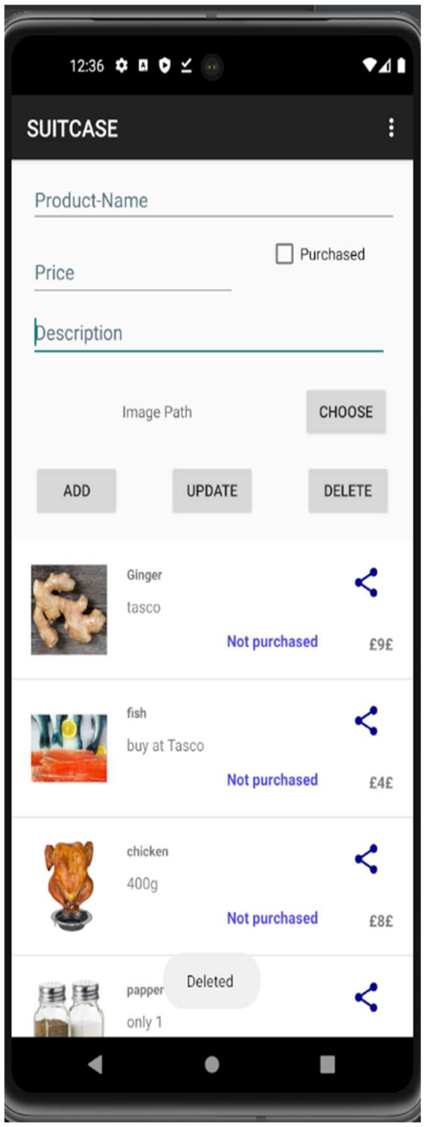
/storage/emulated/0/Download/red-fresh-apple-isolated-on-white-background-royalty-free-image-1627314996.jpg

Ginger	tasco	Not purchased	£9£
fish	buy at Tasco	Not purchased	£4£
chicken	400g	Not purchased	£8£
papper	only 1		

At the beginning of the application user need to register an account fast and login to use the application than user can put information of the project,



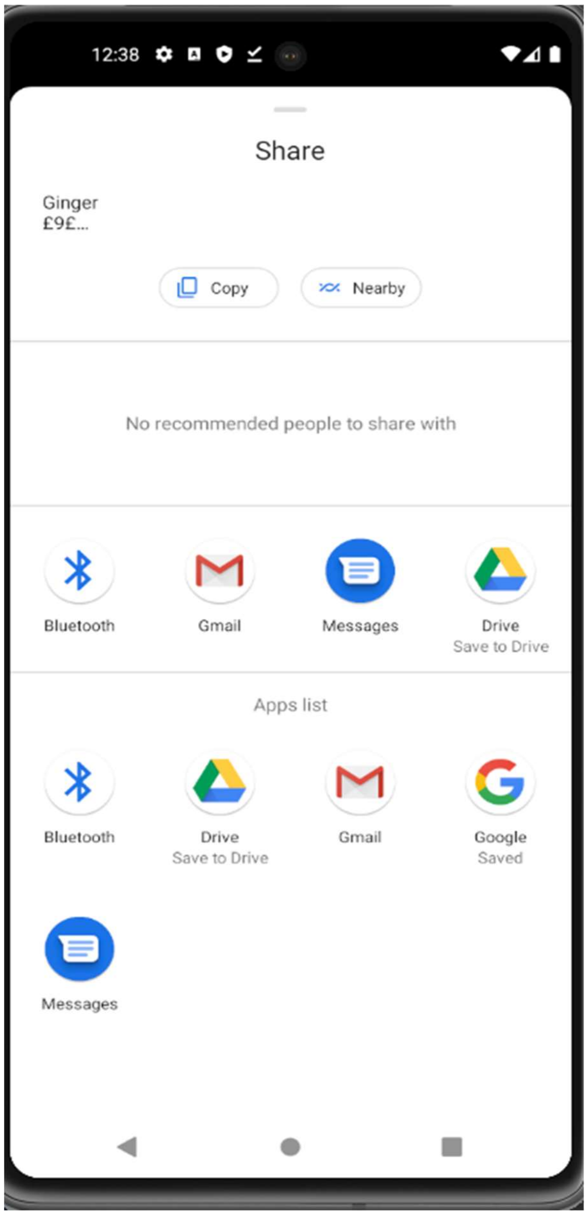


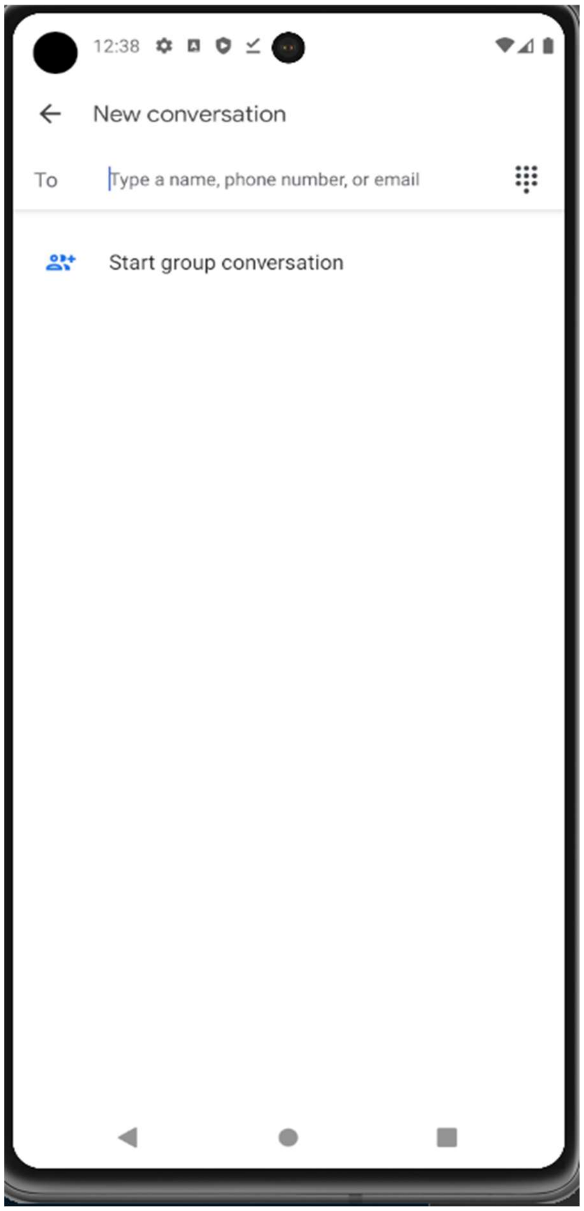


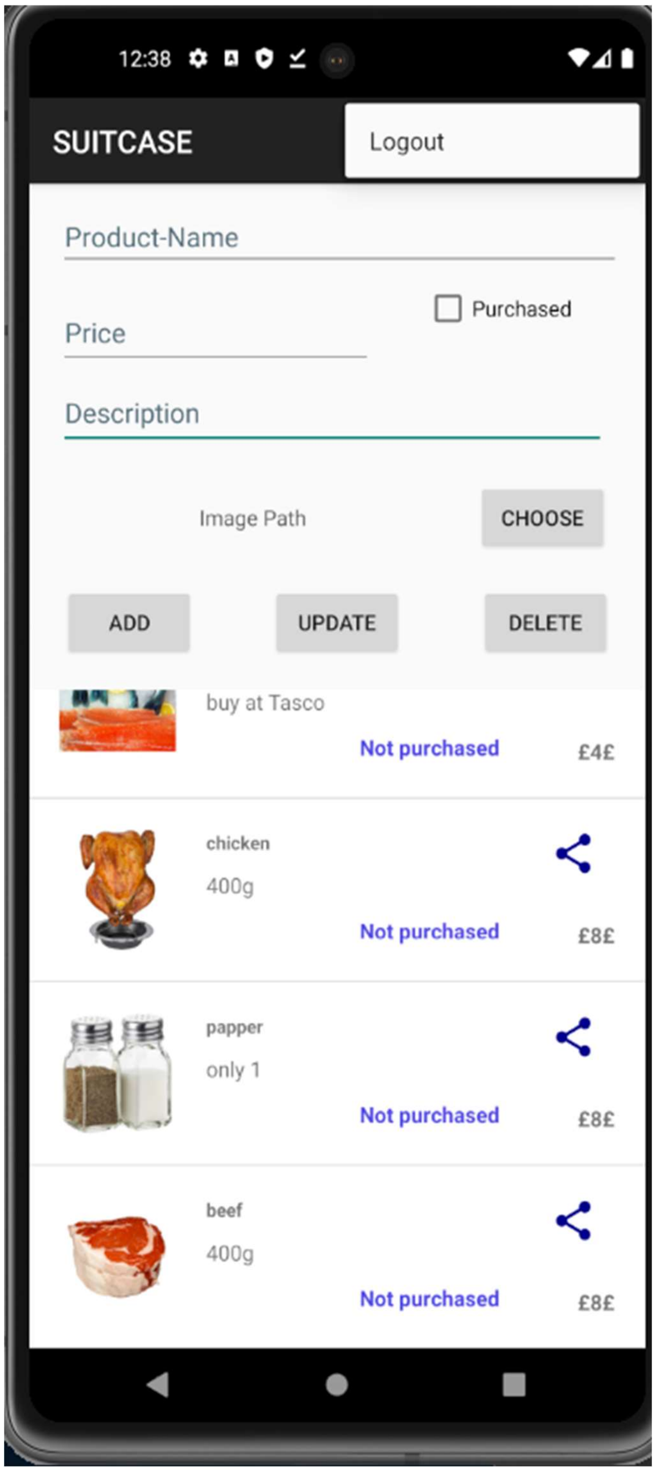
Put name,price,and description and **add** data (apple) and **update** the description of the apple data

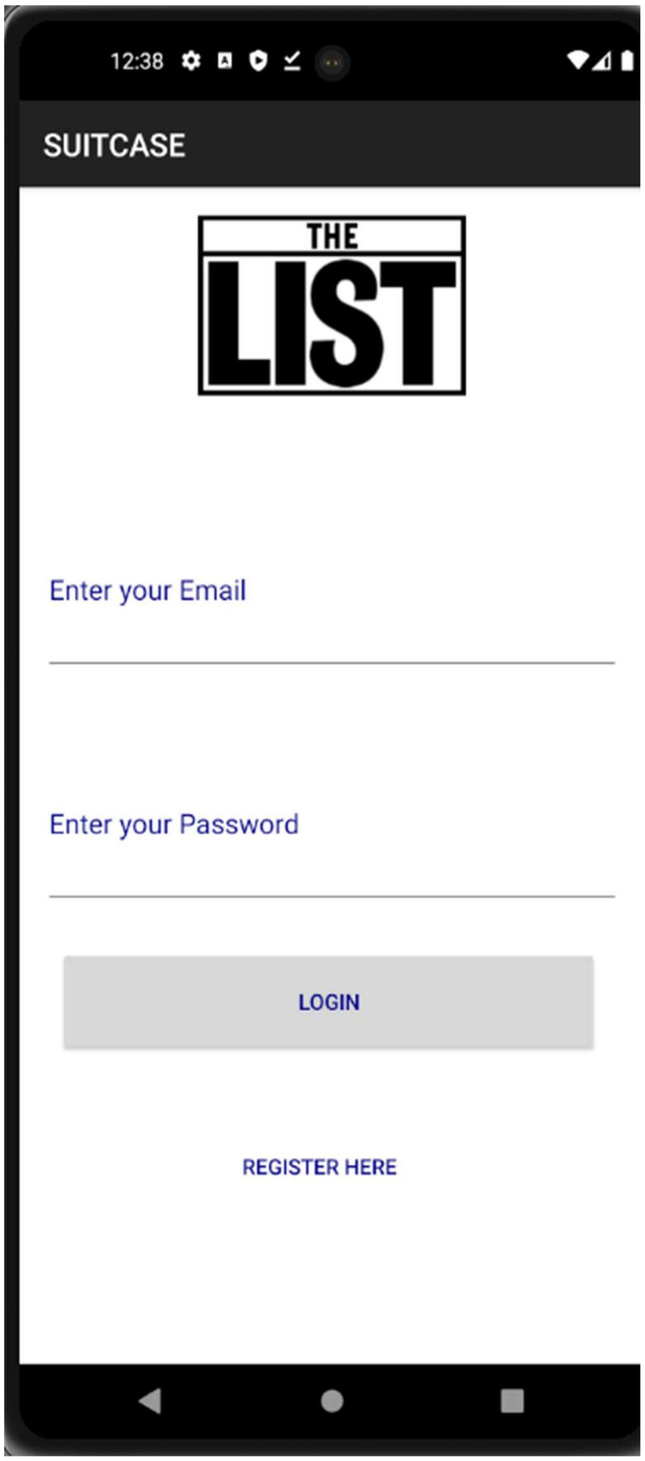
And than click on the list I want to delete and click on delete button.

Than click on share button on the right side of the list to share the details of the list by email or text messag.









After using the app user can log out by clicking log out button at the top left of the screen and can log in back to the app with another accounts.

Test Result

	Test	input	Expected Result	Actual Result	Action
1	Registration for tests	Input your name, email address, and password using your new Gmail credentials, then execute the sign-up command.	A response showing the products page, and Firebase account information received	A response logged in & new account added to Firebase	None
2	Verify the login process	Enter the right information on the login screen by using the test data from before.	Before moving on to the product activity, a response should provide access to the emulator's application.	An answer demonstrating the product's functionality	None
3	Test add product	Product details inputted with image submitted	Display product on activity list below	Product details that are successful and responsive	None

4	Test edit product functionality	Select Previous Test Data and modify all widget details.	A response with updated data that is shown on the list of products	A response edited existing data replaced with new information	None
5	Test delete functionality	using delete command to test button	The product should be removed from both the activity list and Firebase storage in response.	A response completed task & deleted test data from product activity & Firebase storage	None
6	Text message test	Use the share tool to access the text messaging capability.	Product test data should be sent by SMS in response.	reply with an effective text message	None

Evaluation

My mobile app has a number of great features, but stands out in its use of Firebase services for image storage, real-time database capabilities, and authentication. Album library integration improves user experience by providing a simple and eye-catching way to select product photos. By using addSnapshotListener for real-time updates, consumers can be assured that the shopping list is always up to date as it is constantly synchronized with the Firestore database. Well-executed CRUD activities (create, read, update, and delete) are included, and various methods help maintain code modularity. Using the accelerometer to remove text boxes adds a new and attractive feature that encourages user interaction.

While these are admirable qualities, there is still room for development. It would be easier for developers to collaborate and improve code readability if there were comments and documentation on the code. To save storage space and reduce download times, I use image compression, especially when I'm working with a large number of photos. In order for the data stored in the database to be accurate and consistent, user input must be thoroughly validated. The user experience can be improved by providing more visible feedback while asynchronous processes such as image uploads are taking place. The program would also be more robust overall if security measures, consistent UI design, and thorough testing were carefully considered.

While these are admirable qualities, there is still room for development. It would be easier for developers to collaborate and improve code readability if there were comments and documentation on the code. To save storage space and reduce download times, I use image compression, especially when I'm working with a large number of photos. In order for the data stored in the database to be accurate and consistent, user input must be thoroughly validated. The user experience can be improved by providing more visible feedback while asynchronous processes such as image uploads are taking

place. The program would also be more robust overall if security measures, consistent UI design, and thorough testing were carefully considered.

In summary, my app leverages Firebase APIs effectively and demonstrates well-thought-out UI design, providing a solid framework for a shopping list management app. The program can provide an even more professional and user-friendly experience with more features, input validation and documentation improvements.

References

N/A. (N/A). Android vs. iOS. [Online]. diffen.com. Last Updated: N/A.

https://www.diffen.com/difference/Android_vs_iOS

Truong Mai. (2021). TOP 7 BEST NATIVE APP EXAMPLE IN 2023 THAT MERCHANTS CAN LEARN FROM. [Online]. magenest.com. Last Updated: 30 September 2021

https://magenest.com/en/native-app-example/

N/A. (2022). Major Features of Java Programming Language. [Online]. interviewbit.com. Last Updated: 26 May 2022. Available at:

https://www.interviewbit.com/blog/features-of-java/

Admin. (2022). iOS Development Learning Curve: All You Need to Know. [Online]. distinguished.io. Last Updated: 23 September 2022. Available at:

https://distinguished.io/blog/ios-development-learning-curve