CET343 – Android Assignment

Drew Herrington

Student no. 1561909469

User ID - bi29xz

Computing systems engineering

Word Count: 2602

Analysis

In 2023 application building will be more accessible than ever to use, create and share with a vast range of native mobile design systems for all intelligent devices. Although operating systems control all aspects of the world's applications on smartphones and computer applications in the modern era, most adults at a stage will have been in a discussion on what mobile phone system they swear by and have a strong opinion on what is the best. Usually, this argument is between Android v iOS, as the two operating systems are top leaders in the market globally. In this analysis, I will be showing a comparison of the current native mobile designs and including operating systems, options for different programming languages, storage designs and appropriate native mobile standards.

Native mobile design, in its simplest form, is a mobile application built in a particular operating system & written in a coding language that will only run on that system. So, for example, an app built in Android OS using Java or Kotlin programming language is native to Android only and won’t appear on the Apple store or work on an iOS phone.

Users have their preferences when choosing mobile operating systems, and market share statistics show that 70% of the global market uses Android, and iOS comes second with 28%. The third closest is Samsung, with only 0.33%. However, Android are the clear leading mobile OS, run by Google and is used by a variety of mobile phone manufacturers such as Samsung, Oppo, OnePlus, Vivo, Honor and Xiaomi, boosting the number of new devices available a calendar year compared to Apple only having a select number of new models in the same period.

iOS apps can be developed in one of the two main programming languages, Swift or Objective-C. Objective-C is the primary language when writing software in iOS that provides object-oriented capabilities and a dynamic runtime. Features include excellent memory management, support for binary frameworks and Interoperability with both C++ and Objective C++. Although Swift is a modern replacement to Objective-C, starting aimed at iOS development, it has grown through the years to officially support mac OS, Windows, and Linux, with unofficially supported tools available for added development tool support to Android. Swift features include robust error handling built-in, fast and concise iteration over a range or collection and closures unified with function pointers.

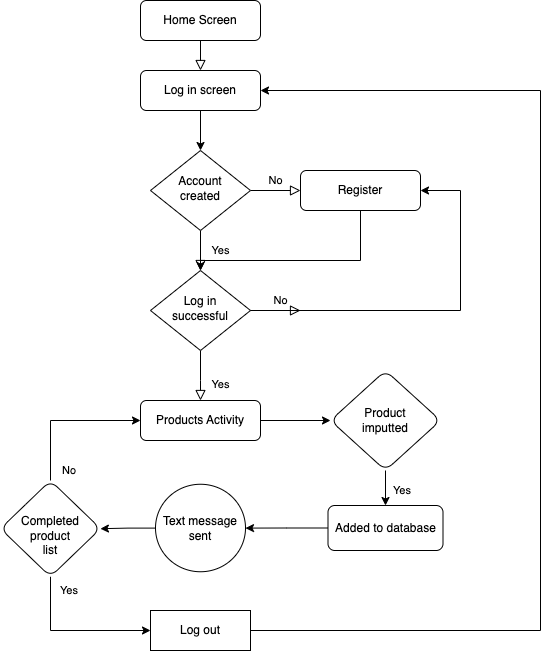
Android’s two main programming languages are Java and Kotlin, Java being a prevalent language for building applications for Android OS. Although Java was originally developed for Internet-based use and initially called Oak was pushed out for web OS in the 90s by better running web APIs from Windows. Java became irrelevant until the surge of Android mobile apps, where it excelled as it is known for its simplicity and easy understanding; Java does not contain complexities that exist in primary programming languages. Java’s features include object-oriented programming, multi-threaded, robust and portable programming. Kotlin is the newest of all the programming languages listed; Google said last year that it is “the preferred language for Android app developers”. Kotlin is multi-functional and can be used for any development, server-side, client-side web or Android. Kotlin’s features include proper function types instead of Java's SAM-conversions, invariant arrays, use-site variance without wildcards and no checked exceptions.

Security is a big talking point in this debate as smartphones now hold a user’s prized information, e.g., online banking, social media passwords, work programs, private images, and so much more. Each day cybercriminals find new ways to hack systems and misuse them by intercepting secure details and can use them to take finances and private information. iOS v Android security debate is vital to find out what device is the best to help be more secure. Firstly, app marketplace security, Apple’s app store reduces the number of applications available due to every app being closely inspected. This helps minimise malware-riddled software that could potentially cause a security issue. Android’s open Google play store allows more apps to be published than Apple’s app store, causing a higher risk of hackers creating malware through malicious apps. Settings can be modified on Androids allowing apps to be downloaded from outside the Google play store, increasing the chances of cyber-attack through downloading virus-ridden apps. Updates to patch security vulnerabilities are easier for Apple devices as the company have only their iPhone to create software updates for and can distribute to users more effectively, meaning iOS devices usually stay updated for longer. Google have many android devices to think about when rolling out updates, as distribution across multiple manufacturers and devices makes it harder to control; therefore, the frequency of updates is lower, and Android smartphones are supported for less time. Both operating systems have pros & cons but constantly work to protect users daily.

The learning curve for developers on both operating systems has to be considered; iOS developers will need 3-5 years of software development to become intermediate level at programming iOS apps, or having a software engineering background will save you at least 2-3 years of frustration. Apple developers will need to learn the essential administrative features to begin publishing apps, and you’ll need to know about developer accounts, provisioning profiles, and the app approval process, among other things. Developers from Java to Swift will find it harder to learn as its grammar is dynamically typed, like JavaScript. Swift has been known to be easier to understand than Objective C, Java, or any other programming language if new to all programming languages. Despite Swift's less verbose syntax, iOS and Android have a comparable learning curve. Both are MVC in nature and have a similar project structure. Switching between the two platforms will be much easier once you grasp the relationships between views and business logic. Similarly, once you've mastered one of the languages, you'll find it much easier to move on to the other. Finally, program debugs errors are more elegant in Swift/Objective C than in Java because if a function returns a null value in Swift or Objective C, it will not crash, and Java, on the other hand, will crash your project without revealing its source.

Design

Activity Flow



The project activity is essential; users’ access through log in & registration, Firebase authorisation is implemented, and only Gmail accounts can gain access. Once successfully logged in, the products page will appear, and users will have a list they can add, delete or edit products on their account; again, the firebase database has been programmed to record all item information to store for later use. In addition, the BabyBuy system comes with a text messaging feature to send products to others. This feature is helpful when a friend or family member wants to buy the user an item found to help with a new-born.

Wireframes



Wireframes have been completed post-development of the application. This was a big mistake as real developers would never use this method as clients or employees could not follow the progression. Good design will make it easier to review the early drafts phase to make the correct changes before development starts to save time and money. The application was implemented before design to develop my knowledge through trial & error while gaining the experience to build the best Android app possible.

Being bold, graphic, and intentional is one of the three main principles of material design. With that in mind, I created a minimalist design with bold font and used a colour palette to catch the eye but also kept in mind the application audience being parents making baby item lists. The primary colour of BabyBuy is purple & the theme has been maintained throughout. Whitespace has been utilised to grab the user’s attention to the critical aspects and focuses on the purple details. Whitespace is an essential element to improve professionalism and keep it user-friendly. To be critical & improve the design, shadows could be added to segments to get crucial depth to align with Google's material design principles. I have used no images or logos which affect the brand's advertisement; logos are a huge part of successful brands.

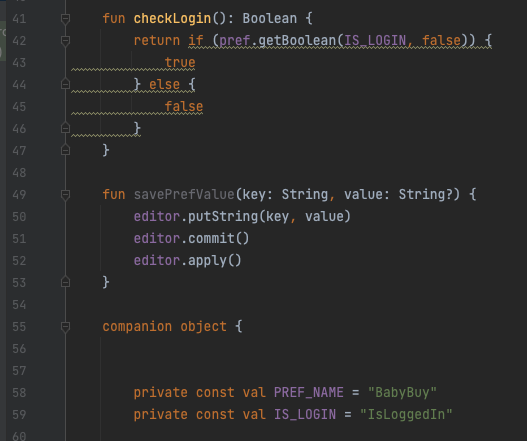
Further advancements for improving BabyBuy include incorporating motion on various functions and creating a higher level of design. Gesture control is a unique way to animate an app adding versatility. To be a better programmer for future projects, following design and implementation steps to boost communication with prospective clients. Advanced design knowledge needs to be improved for the industry standard future applications.

Functionality

Splash Screen

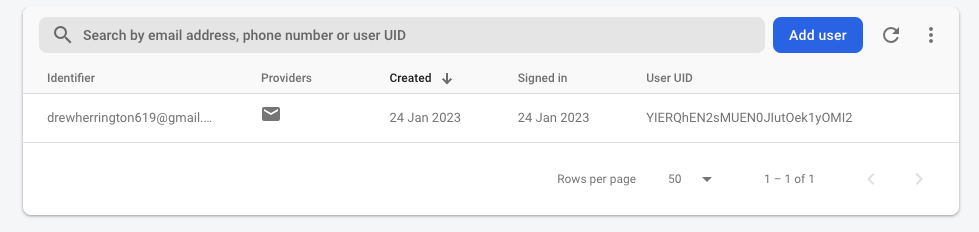


Preference Manager

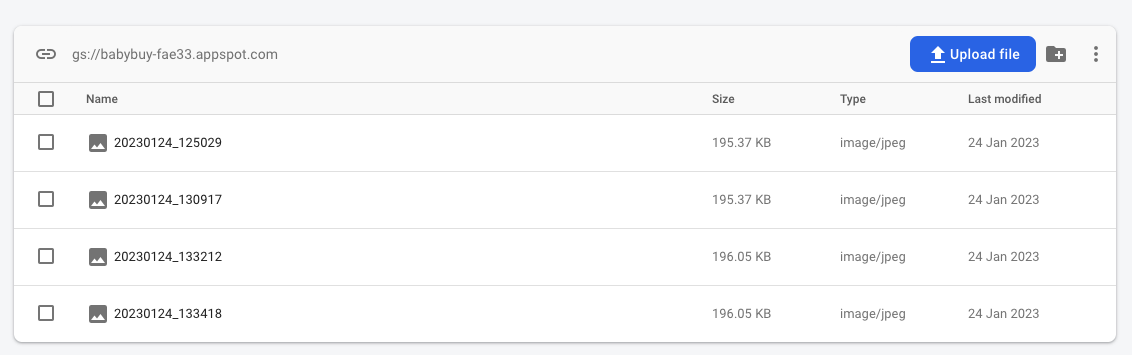


I have implemented Kotlin programming language for my splash screen opening page & preference manager, showing my diversity in program learning, not just Java. The splash screen shows the brand name with a delay of one second before heading to the Login page. The pref manager manages the auth state within the application context and is in Kotlin for compatibility purposes.

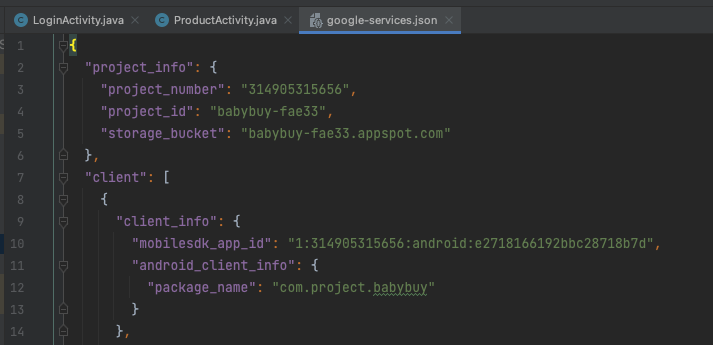
Storage



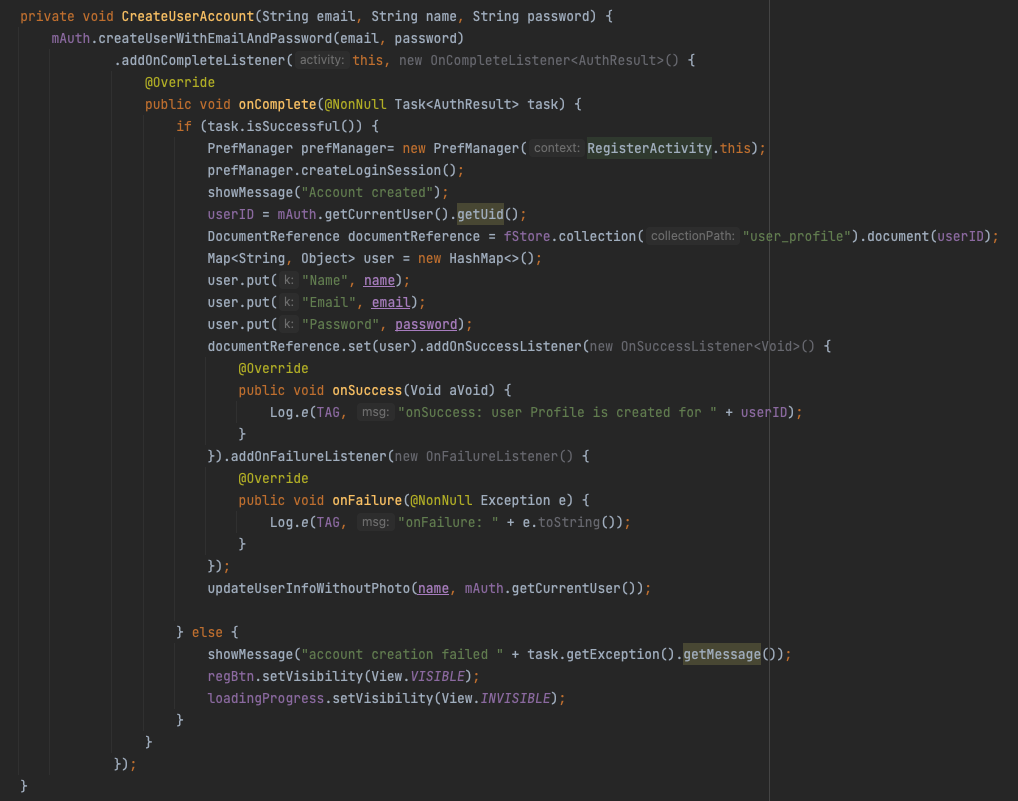
Both the authentication service and backend database are built on Firebase. A client dependency connects the application to the Firebase API by sending HTTP requests with the information provided by the app. The Google system enables the communication between the Firebase API and the app by linking Firebase to the app externally.



Firebase handles authentication and user profile management. The app's UI collects user information, which is subsequently transferred to Firebase via the client dependency and saved in Firebase's online storage. The same procedure is used to authenticate and retrieve user information. In summary, Firebase is used for user data storage and authentication, and it is incorporated into the application via a client dependency that allows connection with the Firebase API. A Limitation of the Firebase storage is that only the image appears for the administration to see, and only the user’s product data are stored on the device used, limiting access to other devices. A database with more robust API storage would be required for a general application on the Google play store. No additional data needs to be seen by the admin account for this prototype. If the client wanted an E-commerce app, listing the full product details would be required to manage orders, something to reflect on for future projects.

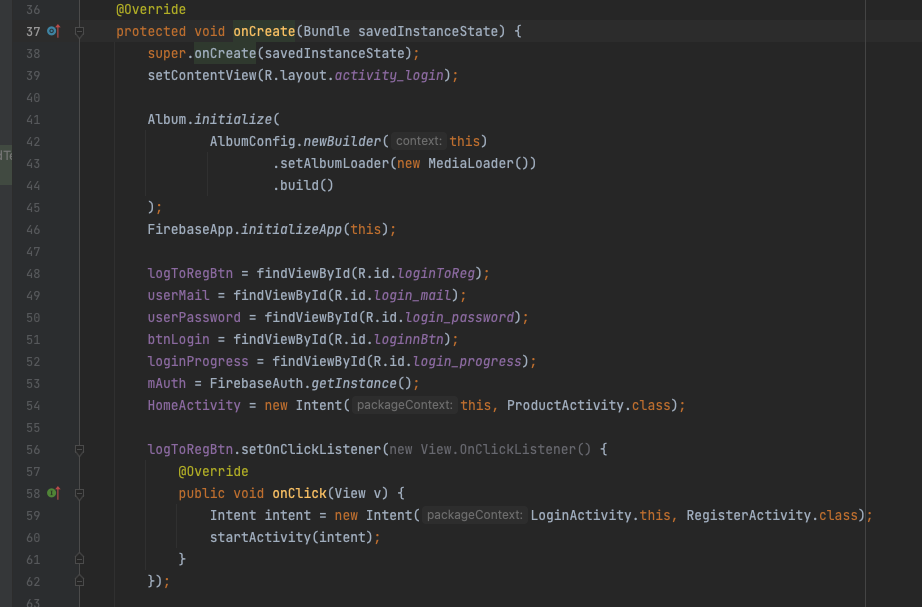


The Google services page allows the project to be connected to my Firebase account; project information identifies the properties to connect the application and database.



To create a new account, this function calls Firebase authorisation linked to my Gmail account. Using the command mAuth relates the Firebase import to the project and, on registration, adds the user's details to the accounts database.

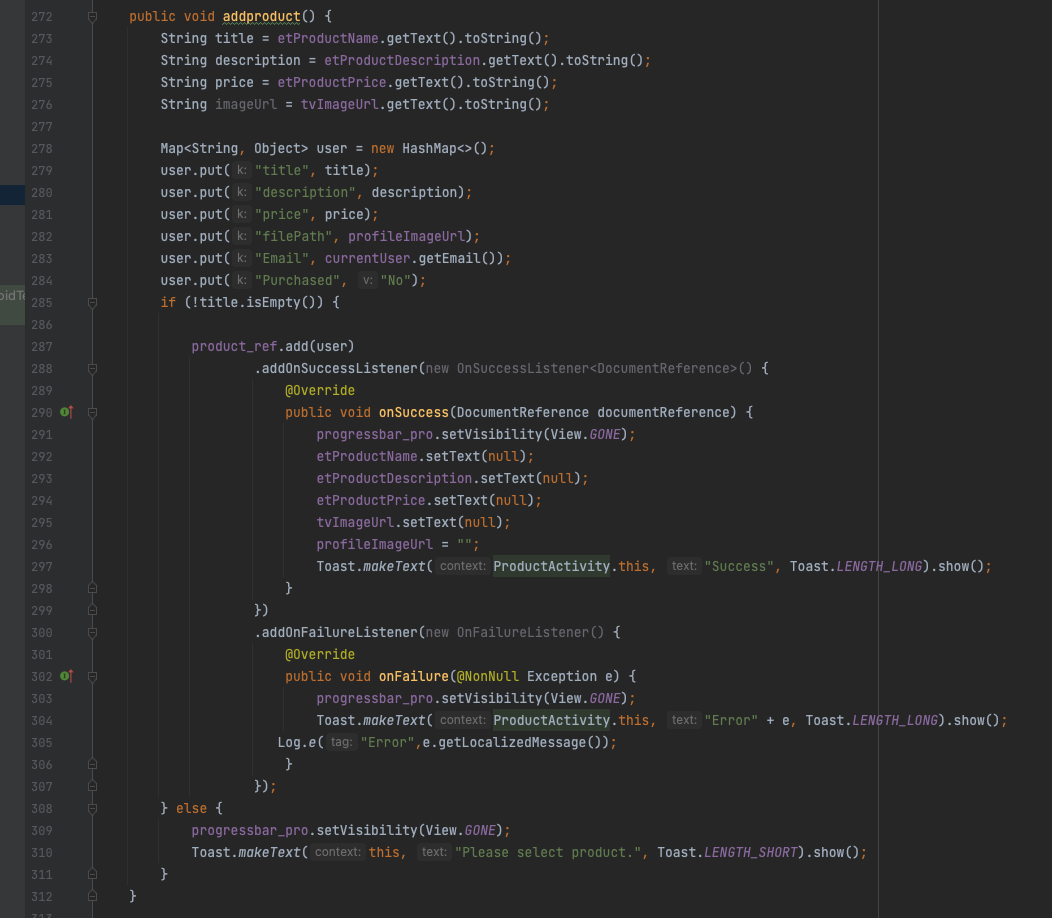
Architecture



To build Android apps, developers need to follow a set of design and architectural principles. Coding patterns promote modularity, maintainability, and testability by organising code. MVP, MVVM, and MVC are examples of popular Android architecture patterns. Using these patterns, code can be structured in an easy-to-understand, easy-to-modify, and easy-to-test manner.

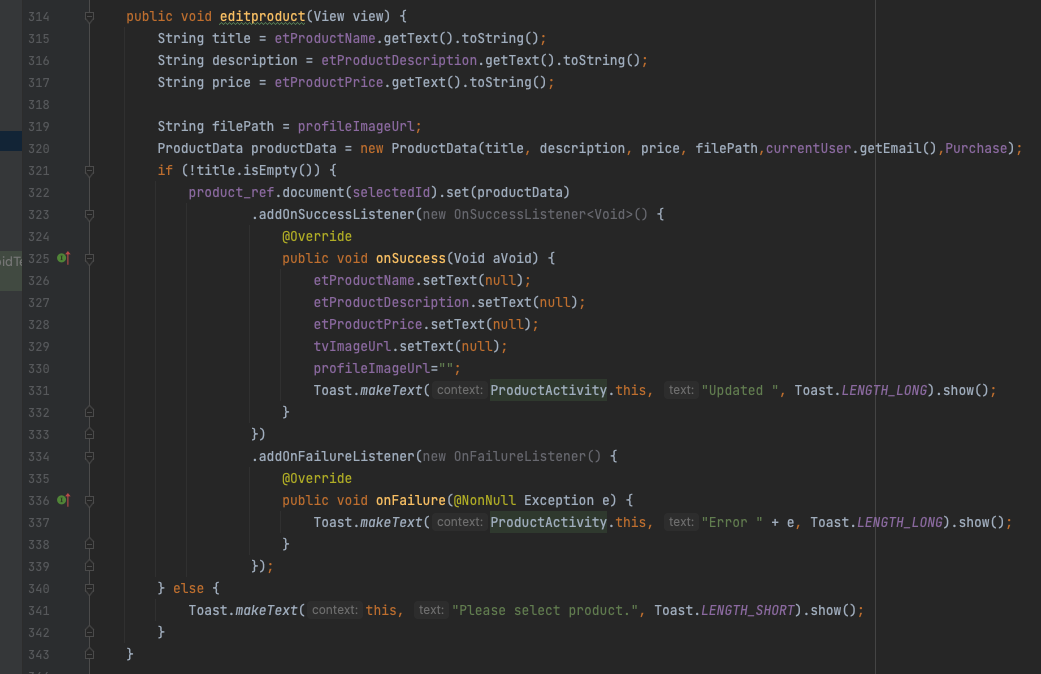
CRUD System

Add



The add Function allows the user to input items into the application that appears on the dropdown list but is also connected to Firebase storage. Items are saved for later use; all attached will remain for future log-ins.

Edit



Products can be edited on all features to change mistakes or update information, which is very helpful for spelling mistakes or items that vary in value.

Delete



The delete function erases products from both the application & Firebase storage; useful when items have been purchased and can be checked off the list.

Test Strategy

A test strategy is a method or plans for evaluating a software product or system. It describes the overall testing strategy, including the tests that will be run, the resources employed, and the testing timeline. It also specifies the objectives and goals of testing, as well as the criteria for deciding whether a test was successful and the method for reporting and recording test findings. A test plan is often created as part of the software development process. It guarantees that the product or system satisfies the defined criteria and is defect-free before being deployed to end customers. The technique is also used to detect possible hazards or problems with the product or system and to guarantee that the testing process is efficient and cost-effective. It also aids in aligning testing efforts with the development process and overall project goals. A test strategy is a document that specifies how testing will be carried out in a project to ensure that the software is of high quality and fits the demands of end users.

Test Results

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Test Data** | **Input** | **Expected Result** | **Output Result** | **Action** |
| 01 | Application test run | Running the app by processing the command in Android studio | Responsive once applied, Splash screen should be present | A response was received showing the Splash screen | None |
| 02 | Test registration functionality | Using new Gmail information for test data, input name, email & password then 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 |
| 03 | Test login functionality | Using previous test data, proceed to login page & input correct details | A response should access the application on the emulator proceeding to products activity | A response showing the products activity | None |
| 04 | Test add product information | Product details inputted with image submitted | Display product on activity list below | Product information responsive & successful | None. |
| 05 | Test edit product functionality | Click previous test data & change info on all widgets | A response with new edited data displayed on products list | A response edited existing data replaced with new information | None |
| 06 | Test delete functionality | Highlighting previous test data & using delete command to test button | A response should delete product on activity list & also in Firebase storage | A response completed task & deleted test data from product activity & Firebase storage | None |
| 07 | Text message test | Create test data & add a new product, with this info use share function to access text feature to perform message | A response should send product test data via SMS to number inputted and display test data | A response with a successful text message sent with test data | None |

Evaluation

In summary, the application is a success in terms of the project brief from the client, as the prototype application created has all the core requirements implemented. No desirable optional requirements were completed, and as a developer, I know the improvements I need to make to advance the work produced. In this timeframe, I spent many hours embedding the CRUD system and connecting it to the Firebase storage database. Lack of time management & gaining skills on the job slowed down the process, leaving the desirable features out. The project's design is fundamental and, at the bare minimum, how a professional-looking application should be published. Focusing on backend programming, I have left the appearance bland and recommend a more interactive interface to bring the project to life. Many issues occurred due to design planning needing to be completed at the right stage and diving straight into implementation with no structure to follow, which means less action. I found this the hard way with hours spent amending interface features to run with the backend coding. A unique logo must be included to symbolise the brand, represent BabyBuy, and stand out to the client. The project has been completed within the client's deadline showing I can produce application software in a developer setting and is evidence I can use these skills for the industry. Android application development has taught me more about debugging & how to produce software for Android devices, increasing my depth in Java programming and teaching me Kotlin language to progress as a developer.

References

* N/A. (N/A). Android vs. iOS. [Online]. diffen.com. Last Updated: N/A. Available at: https://www.diffen.com/difference/Android\_vs\_iOS [Accessed 3 December 2022].
* Information Technology. (2021). 5 Most Popular Operating Systems. [Online]. wgu.edu. Last Updated: April 1, 2021. Available at: https://www.wgu.edu/blog/5-most-popular-operating-systems1910.html#close [Accessed 3 December 2022].
* Truong Mai. (2021). TOP 7 BEST NATIVE APP EXAMPLE IN 2023 THAT MERCHANTS CAN LEARN FROM. [Online]. magenest.com. Last Updated: 30 September 2021. Available at: https://magenest.com/en/native-app-example/ [Accessed 3 December 2022].
* Stat Counter. (2022). Mobile Operating System Market Share Worldwide. [Online]. statcounter.com. Last Updated: December 2022. Available at: https://gs.statcounter.com/os-market-share/mobile/worldwide [Accessed 4 December 2022].
* 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/ [Accessed 4 December 2022].
* Jesus Vigo. (2020). Top 5 programming languages mobile app developers should learn. [Online]. techrepublic.com. Last Updated: 12 October 2020. Available at: https://www.techrepublic.com/article/top-5-programming-languages-for-mobile-app-developers-to-learn/ [Accessed 4 December 2022].
* Daniel Markuson. (2022). Android vs. iOS: security comparison 2023. [Online]. nordvpn.com. Last Updated: 4 January 2023. Available at: https://nordvpn.com/blog/ios-vs-android-security/ [Accessed 4 December 2022].
* 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 [Accessed 5 December 2022].