Dynamic Walls: A Wallpaper Application

MSc (Computer science)

Name (Abu Saad)

Student ID (30083959)

Supervisor (Alun King)

**Statement of Originality**

University of South Wales

Prifysgol De Cymru

Faculty of Computing, Engineering and Science

STATEMENT OF ORIGINALITY

This is to certify that, except where specific reference is made, the work described in this project is the result of the investigation carried out by the student, and that neither this project nor any part of it has been presented, or is currently being submitted in candidature for any award other than in part for the MSc award, Faculty of Computing, Engineering and Science from the University of South Wales.

Signed...........………………………………………………………...

**Abstract**

The “Dynamic Walls” project presents an original Android application that offers the users a vast collection of beautiful wallpapers obtained from the Pexels API and enhanced by the curated and search-based approaches to content selection. The design and the interface of the app are simple and easy to use, and it uses Retrofit to handle APIs and to make sure that the data is requested quickly and with minimal delays. Compatibility, performance, and user acceptance testing were performed to ensure that the app would work without a glitch on various devices and to improve the app’s functionality. Some of the important aspects that have been effectively addressed are API integration, user interface, and application performance. The project captures the importance of the user interface and the need to conduct sufficient testing on the mobile application, as well as future recommendations to improve traffic, such as the addition of search filters, image editing, and social networking features to increase usage. In conclusion, “Dynamic Walls” can be considered a well-designed application that combines contemporary design with innovative technology and has a potential for further development and improvements according to the users’ feedback and the advancements in the technology.

**Acknowledgements**

I would like to express my deepest gratitude to all those who supported me throughout the development of this project. First and foremost, I extend my sincere thanks to my supervisor for their invaluable guidance, continuous encouragement, and insightful feedback, which shaped the direction of this project.

I am also grateful to my peers and colleagues for their constructive discussions, which provided fresh perspectives and solutions to technical challenges. Additionally, I would like to acknowledge the developers and contributors of the Puppeteer package for providing the tools necessary to execute the data scraping functionality that was essential to this project.

Lastly, I owe special thanks to my family and friends, whose unwavering support and patience gave me the strength to persevere through the challenges of this project. Their belief in my abilities has been a constant source of motivation.

This project would not have been possible without the support of everyone mentioned, and I am truly thankful.

Table of Contents

[Chapter 1 8](#_Toc177594730)

[1. Introduction: 8](#_Toc177594731)

[1.1. Background: 8](#_Toc177594732)

[1.2. Significance: 8](#_Toc177594733)

[1.3. Application Name: 9](#_Toc177594734)

[1.4. Aims and Objectives: 9](#_Toc177594735)

[1.4.1. Project Aim: 9](#_Toc177594736)

[1.4.2. Project Objectives: 9](#_Toc177594737)

[1.5. Application Type: 10](#_Toc177594738)

[1.6. Detailed Project Description: 10](#_Toc177594739)

[1.7. Conclusion: 11](#_Toc177594740)

[1.8. Project Timelime: 11](#_Toc177594741)

[Chapter 2 12](#_Toc177594742)

[2. Literature Review: 12](#_Toc177594743)

[2.1. Introduction: 12](#_Toc177594744)

[2.2. Integration of Pexels API: 12](#_Toc177594745)

[2.3. User-Friendly Interface Design: 13](#_Toc177594746)

[2.4. Switching Wallpapers: 14](#_Toc177594747)

[2.5. Efficient Image Loading and Caching: 15](#_Toc177594748)

[2.6. Comprehensive Testing: 16](#_Toc177594749)

[2.7. Checking Similar Apps: 17](#_Toc177594750)

[2.8. Choice of Programming Language: 18](#_Toc177594751)

[2.9. API Security: 20](#_Toc177594752)

[2.9.1. Importance of API Security: 21](#_Toc177594753)

[2.9.2. Common API Security Threats: 21](#_Toc177594754)

[2.9.3. Security Best Practices for API Integration: 21](#_Toc177594755)

[2.9.4. Overview of APIs in Modern Application Development: 22](#_Toc177594756)

[2.9.5. Integration of APIs in Android Development: 22](#_Toc177594757)

[2.9.6. Security Concerns in API Integration: 23](#_Toc177594758)

[2.9.7. OAuth 2.0 and API Security: 23](#_Toc177594759)

[2.9.8. RESTful API Security Practices: 24](#_Toc177594760)

[2.9.9. Comprehensive Testing Methods: 24](#_Toc177594761)

[2.10. Conclusion: 25](#_Toc177594762)

[Chapter 3 26](#_Toc177594763)

[3. Methodology: 26](#_Toc177594764)

[3.1. App Design and User Interface: 26](#_Toc177594765)

[3.2. Research and Development on Pexels API: 27](#_Toc177594766)

[3.3. Implementing Curated Images: 27](#_Toc177594767)

[3.4. Implementing Search-Based Images with Retrofit: 27](#_Toc177594768)

[3.5. Comprehensive Testing and Optimization: 28](#_Toc177594769)

[3.6. Conclusion: 30](#_Toc177594770)

[Chapter 4 31](#_Toc177594771)

[4. Result and Disscussion: 31](#_Toc177594772)

[4.1. App Design and User Interface: 31](#_Toc177594773)

[4.2. Integration of Pexels API: 32](#_Toc177594774)

[4.3. Implementation of Curated and Search-Based Images: 33](#_Toc177594775)

[4.4. Comprehensive Testing and Optimization: 34](#_Toc177594776)

[4.4.1. Compatibility Testing: 35](#_Toc177594777)

[4.4.2. Performance Testing: 35](#_Toc177594778)

[4.4.3. User Acceptance Testing: 36](#_Toc177594779)

[4.4.4. User Testing Feedback: 36](#_Toc177594780)

[4.5. Discussion: 37](#_Toc177594781)

[Chapter 5 39](#_Toc177594782)

[5. Conclusion 39](#_Toc177594783)

[5.1. Recap of the Project Objectives and Methodology: 39](#_Toc177594784)

[5.2. Key Achievements and Successes: 39](#_Toc177594785)

[5.3. Discussion on the Integration of Pexels API and Retrofit: 40](#_Toc177594786)

[5.4. Impact of User-Centered Design on App Success: 41](#_Toc177594787)

[5.5. Insights from Comprehensive Testing and Optimization: 42](#_Toc177594788)

[5.6. Future Directions and Recommendations: 43](#_Toc177594789)

[References 45](#_Toc177594790)

[appendices: 50](#_Toc177594791)

**List of Figures**

[Figure 1. PROJECT TIMELIME 12](#_Toc177595931)

[Figure 2. Framework architecture (Elgendy et al., 2021) 17](#_Toc177595932)

[Figure 3. APP Start 27](#_Toc177595933)

[Figure 4. APP Desktop 30](#_Toc177595934)

[Figure 5. APP Features 30](#_Toc177595935)

[Figure 6. APP Categories 31](#_Toc177595936)

[Figure 7. Specific Category 31](#_Toc177595937)

[Figure 8. Users Feedback 38](#_Toc177595938)

# Chapter 1

# Introduction:

Among the modern trends, personalization and beautification of devices are among the most critical factors impacting the actual usage of gadgets. In terms of customization, wallpapers remain one of the most efficient and widely available means of personal and aesthetic self-identification. Recognizing this, "Dynamic Walls: In meeting the new requirements of Android users who want to easily change their wallpaper and want an application that is easy to deal with and of course visually pleasing, a new application called “A Wallpaper Application” has been created. The following introduction provides the context, rationale, and goals of the Dynamic Walls project and highlights the focus on the creativity of the project enlightening the utilization of Pexels API, the improvement of the UI/UX design, and the concern on the loading and caching of images.

## Background:

Digital wallpapers have come through improvements that define their development over the past few years. In the beginning, the wallpapers were pictures that were located on the local device storage and the selection was quite limited. However, with the help of contemporary high-resolution screens and the possibility of an internet connection, people want more various, qualitative, and accessible wallpapers. Today, users look for applications, which offer not only a large database of pictures but also have the best interface and operational speed.

Dynamic Walls was developed based on such background, as its goal was to level up the generatively defined gap between users and available solutions. Currently, Dynamic Walls takes advantage of the large collection offered in the Pexels database which is a well-known website with free stock photos and videos. The use of the Pexels API means the application can now obtain high-quality images, in real-time meaning users get a new set of wallpapers every time they want to change them.

## Significance:

The value of Dynamic Walls can be seen in the company’s desire to create an exceptionally functional and stylish wallpaper solution. Most wallpaper applications available on the market are usually not well-organized, and the images are not loaded quickly, but Dynamic Walls has a clean and simple layout and efficient caching system.

The utilization of wallpapers in an application also poses a problem with how images can be adequately loaded and cached. Large-sized wallpapers often contain high resolution and can use up a lot of the available RAM and Bandwidth which can result in a slowed-down process. Dynamic Walls gets over this issue by using complex caching techniques and leveraging Glide, a well-known image loading and caching library in Android. This makes the retrieval of wallpapers relatively easy and fast irrespective of the amount of memory in the device.

In addition, the present work focuses thoroughly on the application’s GUI, which enables effortless browsing. Some of the activities include, and probably the most important one, users can search for wallpapers of different categories using keywords and can preview wallpapers before setting them as background for the home screen or lock screen. The peculiarities of the interface provide the graphic simplicity and fantastically clear and rather clear structure that allows distinguishing it even for elementary schoolchildren and for the most non-computer-literate Java users, which is the primary objective of expanding the number of the application’s users.

## Application Name:

Dynamic Walls: A Wallpaper Application

## Aims and Objectives:

### Project Aim:

The general goal of this project is to develop an interactive wallpaper Android application. This app will be very simple to use and it will have integration with Pexels API to allow the users to get access to a rich library of stunning wallpapers. This application is designed to improve the user’s device by providing beautiful wallpapers with search and select functions to apply them to both the home and lock screens.

### Project Objectives:

* Integrate the Pexels API: The main goal is to provide the integration of the Pexels API to the Android application. This will enable the app to download a large number of images from the database of Pexels hence providing users with a large number of wallpaper options.
* Create a User-Friendly Interface: Coming up with a very simple and easy-to-understand interface is very important. The design should allow users to easily search for and choose wallpapers as well as preview them. The purpose is to guarantee that even individuals with basic IT literacy will be able to work with the application without any difficulties.
* Switching Wallpapers: It should offer options to set the wallpapers both for the lock screen and the home screen of the device. Wallpapers should be changed easily by the user to enable the user to enhance the customization of their device.
* Efficient Image Loading and Caching: The loading time of images is another area that needs to be addressed while using the app to increase the efficiency of the app and caching strategies. This will make sure that the app does not lag or consume too much data when being used by the client.
* Comprehensive Testing: That means, the app should be tested effectively to see how it performs on all the devices that run on the Android operating system. This includes; compatibility testing, performance testing and user acceptance testing to fix any problem that may be found before releasing the app.

## Application Type:

The product for this project will be an Android application that will be developed in Kotlin. It is chosen to provide the latest language features and improved performance as well as seamless integration with the Android libraries and tools.

## Detailed Project Description:

The detailed project description for "Dynamic Walls: The “A Wallpaper Application” has several important components. First, to integrate the Pexels API, one has to get an API key from Pexels, use Retrofit for network requests, and set up data models to handle API responses and functions for successful and failed requests. Second, improving the usability of the application involves developing intuitive navigation with fragments for various categories, enhancing the search function with options for search queries and suggestions, and enabling users to view wallpapers in their full size and select them for use on the home screen, lock screen, or both. For changing wallpapers, the Android’s Wallpaper Manager class is used to manage and set wallpapers with touch gestures or buttons for switching and an optional feature of changing the wallpaper after a specific time interval. Image loading and caching are managed by using the Glide library to load, cache, and display the images, which includes memory and disk caching to minimize network requests and enhance the loading time of images, and the app is also designed to handle large images without compromising on performance. This includes compatibility testing on different Android devices, performance testing on image loading time, memory usage and battery consumption using Android Profiler, and user acceptance testing with beta testers to get feedback on usability problems.

## Conclusion:

Dynamic Walls is a new application of wallpapers for Android devices that can meet the high demands of people and become a groundbreaking innovation of modern society. With the help of the Pexels API integration, friendly interface design, identification of image loading and caching optimization strategies, and vigorous testing, the application is to be delivered as the one user is to like and appreciate. The level of comparison with other similar applications helps maintain competitiveness and constant improvement of features and design in Dynamic Walls. Therefore, Dynamic Walls is expected to be the next best application for users who wish to make their devices, particularly their home screens, come alive through flexible and attractive wallpapers.

## Project Timelime:

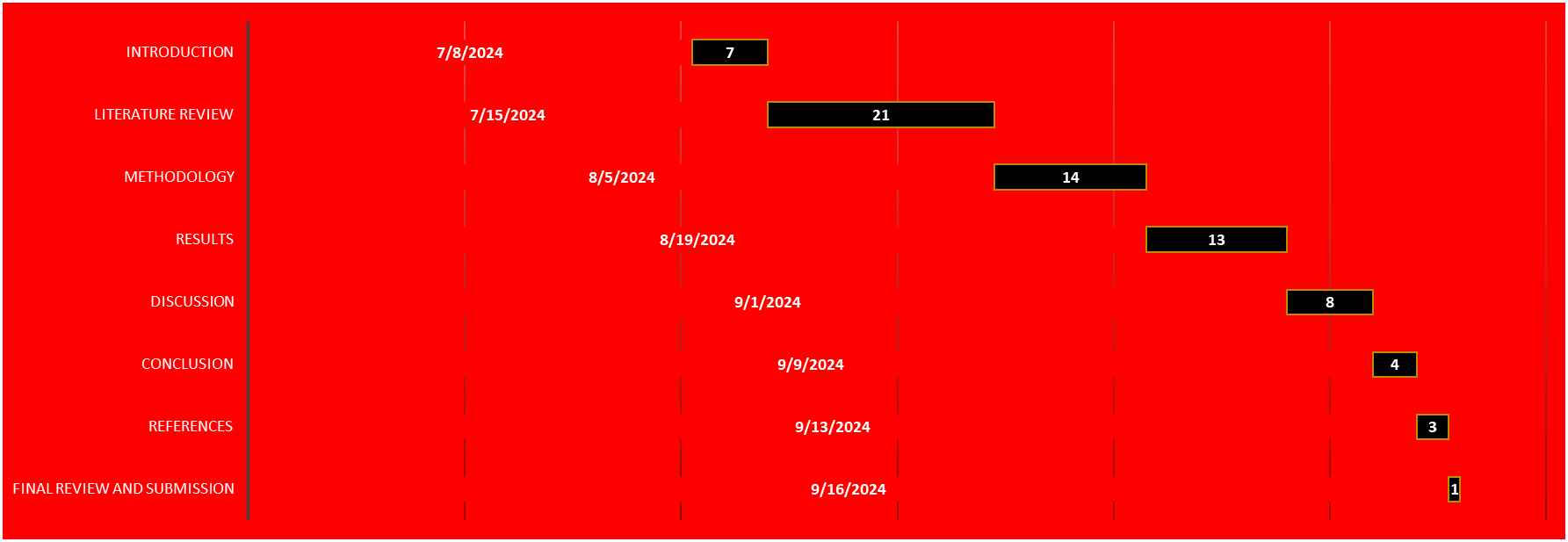


Figure . PROJECT TIMELIME

# Chapter 2

# Literature Review:

## Introduction:

This application named Dynamic Walls is intended to bring a change to the user interface of Android devices through the creation of an interactive wallpaper with the inclusion of the Pexels API. The goals of the project also comprise the creation of a clear and easily navigable interface, fast image loading and data caching, and rigorous testing. Before delving into the specifics of this project, one must look at the concepts and trends of related technologies, UX design, and performance.

There is a current study on the creation of Android applications using Kotlin, an emerging language for Android programming recommended by Google (P et al., 2022). Kotlin can be used in creating various applications for mobile devices both on the client’s side and on the server side (Quispe, 2022). Kotlin has been proven to be useful in creating specific applications like student information management applications (P et al., 2022) and electric vehicle charging station locator applications (Muddalkar et al., 2022). These applications employ Kotlin’s libraries and XML for designing, and they include such aspects as effective data handling and storage, intuitive user interface, and interaction with other services. Also, Kotlin has been used in the formulation of language learning applications that include an Arabic learning application developed using the Jetpack Compose (Dzulqarnain and Tukino, 2023). This research validates Kotlin’s ability to design and develop interesting and engaging mobile applications in different fields, which proves that Kotlin is useful in the creation of feature-rich Android applications such as wallpaper applications.

## Integration of Pexels API:

API stands for Application Programming Interface which is an essential element of contemporary application development because it helps an app to reach data and services outside the given application. Hence, the use of the Pexels API, it offers the convenience of gaining free access to a countless number of high-definition images, which, in turn, can be integrated into applications to make them look more visually appealing (Pexels, 2024). As a result, a wider option of wallpapers will be available to the users of Dynamic Walls if the application incorporates the Pexels API, which will lead to enhanced user satisfaction.

Earlier literature has pointed to API’s use as a way to enhance app content and features as the key concept. An article by (Fehling et al., 2014) used a case of how cloud API can be included in mobile applications to offer versatile and elastic services. In the same context, (Ahmad, 2017) have looked at the usability of RESTful APIs as the means to transfer information between mobile applications and external servers. These studies highlight the ways through which application programming interfaces like Pexels can improve the functionality of mobile apps.

The use of APIs in the development of Android applications has become more relevant due to the added benefits it brings. To this end, web scraping methods can be used to develop REST APIs with no database and can be incorporated in mobile applications (Mulyani, Kurniadi and Hakim, 2021). Integrating the QR Code and API technologies in the library applications shows that there is a possibility of enhancing the management of data and information (Nugraheni and Maryam, 2022). The most popular use of Retrofit library is in network management and API client in Android development and JSON parsing and serialisation (Kaura, 2024). To assist the developers, API function call and code snippet recommendation systems such as FOCUS that uses context-aware collaborative filtering of open-source software repositories have been proposed (Nguyen et al., 2022). Such improvements in the API integration and recommendation systems help in enhanced development of Android applications.

## User-Friendly Interface Design:

One of the most important things in any application is the development of a comfortable friendly interface. This should in turn make it possible to enable any user including those with low levels of informatics to be able to use the app with ease. (Norman, 2013) observes that when designing interfaces, the emphasis should be placed on the creator anticipating how the interface will be used. Norman also opposed that a good interface should be easy to understand, coherent, and should contain clear responding patterns.

The system-level investigations identified that users solved the wallpaper applications’ shortcomings by choosing the applications with an easy interface and clear categorization of wallpapers (Weichbroth, 2024). Incorporation of these principles will be done in the Dynamic Walls app by providing fragments to various categories of walls and an option for search queries and suggestions. This approach aligns with the recommended strategies in user experience design where wallpapers should be easily accessible by the users.

User-centred design (UCD) is a significant approach in the design of user interfaces for Android-based applications. The UCD process typically involves four stages: The four stages of systematic review are analysis, design, evaluation, and implementation, as pointed out by Hartawan (2022) and (Hasari, Febriansyah and Anzana, 2022). This approach involves incorporating users right from the design process to ensure that the interface that is developed fulfils the users’ requirements and choices. The UI and UX of an application are crucial factors that define the users’ perception of the application since users can be easily discouraged from using an application that has a bad design (Setiawan and Triase, 2023). UCD has been applied in various Android applications, including film synopsis apps, cinema ticket booking systems, and Quran recitation tools, and the result has been the development of interfaces that are user-friendly and appealing (Hartawan, 2022; Hasari, Febriansyah and Anzana, 2022; Mahfudh and Saputra, 2022). Some of the techniques that can be used are usability testing, for instance, the cognitive walkthrough, which can be used to test and improve the prototypes so that the final product is one that is expected by the user and provides a good experience (Mahfudh and Saputra, 2022).

## Switching Wallpapers:

Dynamic Walls the most important aspect is that one can change the walls with a lot of ease. Android comes with classes that allow the wallpaper to be managed and set programmatically with ease, the class is Wallpaper Manager (Android Developers, 2024). Thus, with the help of this class, the app will be able to provide users with the required option of setting the wallpapers for both the home screen and lock screen using simple touch motions or buttons.

Analysis by (Ali et al., 2022) pointed out that customization features of mobile applications are essential. Their research showed that the audience likes applications with adjustable preferences and options because it makes them feel like they own it. Dynamic Walls will meet this preference by incorporating an option where the users can set the program to change the wallpaper after a particular period to complement the application.

## Efficient Image Loading and Caching:

Accessibility as well as how images are loaded and cached plays a major role in determining the performance of the application and fitness of the app in the aspect of data usage. For Android applications, the Glide library is the most prevalent to work with image loading and caching facilities that are efficient and convenient (Glide, 2024). It also uses the memory and disk caching options that help in reducing network calls and improve the image loading relevancy.

Research has also indicated that issues to do with image loading and caching can cause a real problem to the performance of the app., A study by (Elgendy et al., 2021) illustrated how mechanisms for caching could lower the loading times of images and enhance the usability of the apps. Also, (Rua and Saraiva, 2023) discussed the question of memory consumption and its relationship with the performance of mobile applications. Therefore, through Glide, Dynamic Walls can be sure that images will load up fast and, in turn, run smoothly for the users.

In developing "Dynamic Walls, in the case of choosing the Glide library for loading and caching, Picasso, Fresco, and Coil were available, but Glide has been opted for improved performance and features as depicted in ‘A Wallpaper Application’. Glide is efficient in image loading from URLs, local files, and resources, with benefits that include resizing and transformations in addition to fair memory and disc caching, which reduce network calls and make image loading faster (*Glide v4 : Fast and efficient image loading for Android*, 2023). Although Picasso is more user-friendly than Glide and does not consist of some of the additional features in Glide that can slow down some of the larger images (Picasso, 2023), Despite having exciting features such as progressive JPEG loading and support for an animated GIF, Fesco is a more complex and resource-intensive framework with a steeper learning curve as compared to the rest, thus making it unsuitable for this project (Fresco An image management library., 2023). Coil is another library that has been developed more recently, with some appreciation for the modern way of working while still being quite new and not as popular as Glide among developers; however, the community support for it can be larger (Contributors, 2023). Therefore, considering the simplicity of the tool, its observance of forthcoming features, and the stability of the work on different models of Android devices, Glide could become the most appropriate solution for the formation of a smooth and continuous experience in “Dynamic Walls.”

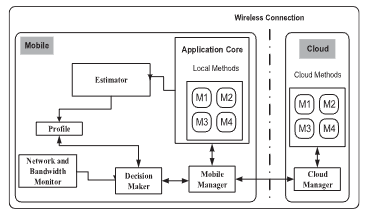


Figure . Framework architecture (Elgendy et al., 2021)

## Comprehensive Testing:

This implies that to be fully confident that the app shall be efficient in its operations, it needs to be subjected to exhaustive tests across the various coverage levels and scenarios when used. Compatibility testing, performance testing as well as user acceptance testing are some of the important steps in this process. (Gao et al., 2014) study titled ‘Usability testing of mobile apps: Techniques do and do not work’ raises concerns about the general sensitivity of conclusive testing in mobility applications. Their study discovered that intensive testing could uncover problems in an application and correct them before they could negatively affect the app once it is released to the market.

Since Dynamic Walls will be embedded in Android devices, it will have to pass through a range of compatibility tests covering various Android devices in relationship to its parameters such as the loading time of images, the use of memory as well as the battery usage. Beta testing for the application by the end users will show primary indicators of usability problems, which have to be fine-tuned and have to go through further testing before release.

## Checking Similar Apps:

To make sure that “Dynamic Walls” is differentiated and has something new to provide to its users, it is crucial to compare it with other similar applications. Here are a few popular wallpaper apps and their features: Walli-4 – 4K, HD Wallpapers & Backgrounds is an application that provides a variety of artistic and creative wallpapers suitable for both home and lock screen. Its usability features include clear categories and search, the main advantage being that the platform is based on user-submitted content, so artists can share their works. Zedge Wallpapers & Ringtones is an application that offers wallpapers, ringtones, and notification sounds with a large number of categories and themes. While it has a very cluttered design because of the abundance of content that is provided, it provides a complete solution for changing both the wallpaper and the ringtone. Backgrounds HD (wallpaper) offers high-definition images and includes wallpaper-changing options for both home and lock screens. Its main advantage is a clear and straightforward interface with a clear focus on the design, as well as the main advantage of frequent updates with new wallpapers and a large selection of wallpapers. Based on the analysis of these apps, the following opportunities and weaknesses of “Dynamic Walls” could be determined: The potential for “Dynamic Walls” could be to let the users create and save their collections of wallpapers, providing dynamic wallpapers that change depending on the time, weather, or user activity, and allowing the users to upload and share their wallpapers, creating a community of creators.

Among those, it is necessary to recognize the best-sellers and their advantages/disadvantages, as well as check the novelty of the solutions that Dynamic Walls provides. Some of the wallpaper apps are Walli, Zedge, backgrounds HD, etc which are useful for knowing the preferences and the type of wallpapers users have for their mobiles. (Gan and Balakrishnan, 2016) have it that competitive analysis enables the developers to learn about the clients and what they need as well as enables the developers to distinguish themselves from the rivals by presenting the users with features that are distinct from the ones offered by the rivals’ apps.

The content submitted by Walli’s users and the availability of various functions on Zedge underline such key aspects as user contributions and multi-functionality. Dynamic Walls are created to save collections of wallpapers that a user selects and show wallpapers change according to the time, weather, and activity of the user. This approach is related to the tendencies in the field of personalized and context-sensitive applications (Barlev, 2024).

So as to make Dynamic Walls easily recognisable in the selected application for wallpaper usage, it was necessary to foremost analyse competitors. Some of the most common apps to use contain wallpapers; some of these include Walli, Zedge, and Backgrounds HD, which helped elicit the patterns and preferences of users.

Walli also allows users to submit content so artists can present their work and users can find different artistic wallpapers. Zedge offers an app to change wallpapers and ringtones concurrently, although the interface is rather messy due to numerous downloads. Backgrounds HD is dedicated to offering high-quality images and has a simple and clean design for the application’s interface. The application updates wallpapers’ galleries quite often.

From this, we identified the following opportunities regarding the concept of dynamic walls: There must be an option for the user to create and store their favourite wallpapers in their library for convenience. Further, the idea of having wall papers that shift with the time, the weather, or the activity may also make the app more attractive. Thus, by regularly establishing alignments and differences between Dynamic Walls and other comparable apps, we will ensure its novelty and relevance in the market.

## Choice of Programming Language:

Android app development is done using a number of programming languages where Java and Kotlin are most preferred (Karjatkar et al., 2024). Kotlin, introduced as a first-class language for Android, has been widely adopted by the (Coppola, Ardito and Torchiano, 2019). Comparing Java and Kotlin, it was found that Kotlin-based projects were more popular and had a highly significant relationship to the number of GitHub stars (Sulowski and Kozieł, 2019; Coppola, Ardito and Torchiano, 2019). The shift from Java to Kotlin in the Android projects has been swift and without any issues; about 20% of the analysed projects use Kotlin (Coppola, Ardito and Torchiano, 2019). However, other frameworks, such as Flutter, developed using the Dart language are also possible for cross-platform development (Karjatkar et al., 2024). The language used in programming the application has the potential of affecting different parameters, such as the efficiency of the application, the layout of the code, and the availability of support from developers (Coppola, Ardito and Torchiano, 2019). In addition, games and applications can be designed to teach programming to students using the Android devices (Rgp and Hadi, 2019).

Selecting the most appropriate programming language is vital when designing an Android application to meet the intended goals, efficient utilization of resources, and in times of enhancement, thus the importance of selection correctly among the numerous programming languages available. Among the competitive solutions that could be employed at ‘Dynamic Walls’ for the wallpaper application, Kotlin was chosen instead of the others, namely React Native, Flutter, and Swift. Kotlin, created by JetBrains, became the most popular language for Android app building after becoming Google’s recommended language in 2017 (JetBrains, 2023). First, it can be synchronized with Android Studio while depending on existing Android libraries and Android tools, which are helpful when developing programmes because they enable efficient coding and testing (Google, 2023). Various additions in modern languages like null safety, extension functions, and coroutines make the code more efficient, less erroneous, and thus more concise and readable (JetBrains, 2023). Moreover, Kotlin is fully interoperable with Java; thus, existing Java libraries and frameworks can be used without change if the project was initiated in Java (Google, 2023). There is no difference in performance between Kotlin and Java, which means the successfully developed app operates on all Android devices, which is vital for the ‘Dynamic Walls’ application that needs fast image loading and caching (JetBrains, 2023).

React Native is a technology sponsored by Facebook that enables developing applications for iOS and Android with Javascript and React. Like any app built with a framework that is implemented on top of a native app, React Native apps are one step removed from native performance, which can be a serious disadvantage for a wallpaper app that is going to be primarily using images (Facebook, 2023). Besides, even though React Native does feature a plugin set, it is not necessarily compatible with the most recent native features and libraries, which can pose a problem if the app must accomplish extraordinary tasks (Facebook, 2023).

Flutter, which is also developed by Google, is another popular framework in cross-platform application development that uses the Dart language. It is considered high-performance because of its direct compilation into the native code; however, Dart is not as developed as Kotlin, which has been a staple language in the Android community (Google, 2023). Also, while moving to Dart can be challenging for developers already familiar with Kotlin or Java, it could slow down the work (Google, 2023).   
Something that would be completely disadvantageous for the development of the Android platform is that Swift, the language used by Apple for developing iOS, is not natively supported. Developing in Swift requires developers to have different codebases for iOS and Android; it will consequently elongate the time taken and cost too (Apple, 2023). There is no way to directly integrate Swift and Android; additional frameworks or third-party tools have to be integrated into the project (Apple, 2023).   
As a result, based on the given specific features of the ‘Dynamic Walls’ wallpaper application, Kotlin appears to be ideal. Originally, it has a modern language that supports code development in its density; it is integrated adequately with Android tools; and it has a strong synergy with Java. We may infer that the raw performance, direct access to the native libraries, and components make Kotlin the best choice to develop a premium Android application compared to the cross-platform frameworks React Native and Flutter (JetBrains, 2023; Google, 2023).

## API Security:

New technologies like the Pexels API are used together with increasingly integrated mobile applications that require APIs for communication between different software modules. These technologies allow an app to connect with other applications that use independent methods of data transmission. Nonetheless, they make it easy for malevolent users to hack into your app and get sensitive data. As such, developers need to come up with strong security systems for their applications.

### Importance of API Security:

Developers need to protect their applications from risks such as data breaches, unauthorised access, and malicious attacks when integrating APIs like Pexels. For instance, if an API is not securely implemented enough, hackers can use it to access private information or change program functionality. To facilitate the Pexels API that retrieves high-quality pictures and displays them on dynamic walls, it’s necessary to secure this API in order to avoid unauthorised use or data loss.

### Common API Security Threats:

1. Injection Attacks: These occur when an attacker injects malicious code into the API. SQL injection and cross-site scripting (XSS) are common forms of such attacks.
2. Man-in-the-Middle (MITM) Attacks: This happens when attackers intercept the communication between the client and the API, allowing them to eavesdrop, modify, or steal data.
3. Broken Authentication: If API authentication mechanisms are not secure, attackers can gain unauthorized access to protected resources.
4. Rate Limiting and DDoS: APIs are often vulnerable to Distributed Denial of Service (DDoS) attacks if there are no restrictions on the number of requests a client can make.

### Security Best Practices for API Integration:

APIs should always follow some major practices for security purposes. To make certain there’s encrypted communication preventing any sensitive information from getting lost in translation, use HTTPS for all API access. Users who would like to be authenticated before accessing the service should use strong authentication measures, OAuth 2.0 being one of them, which helps claim its safe and controlled interactions with users. Also, you should always store Pexels API keys securely in encrypted settings, together with rotating them constantly, in order to minimise the chances of being compromised. In addition, rate-limiting is necessary so that it could prevent abuse acts like DDOS by controlling how many requests are made within a certain time frame for one IP address or user account in general. Proper input validation is important to avoid injection attacks, which require cleaning all inputs before any processing takes place. API usage monitoring assists in detecting any unusual patterns, such as surges in traffic, and hence notifying about possible attacks, which will enable quick responses. Lastly, token-based systems, including JSON Web Tokens (JWT), offer stateless and signed authentication, securing how you access APIs by verifying who you are using them.

### Overview of APIs in Modern Application Development:

APIs are essential to integrate applications with external resources, thereby promoting digital transformation agendas across industries (Kumar and Falhi, 2022). However, the trend towards the use of APIs also creates a large number of threats and security risks. Deeply entrenched APIs pose an open invitation for hackers because they let them access user databases and QAs and understand the application’s working model (Kumar and Falhi, 2022; Spencer, 2022). APIs have always been considered to be very sensitive, but many users are unaware that the accounts or programs are exposed to numerous security threats (Qazi, 2023). Subpar documentation and hard-to-solve design issues are critical to the creation of potential stability weaknesses using security APIs that will make applications unsecured and fail at protecting the data of users (Mousavi et al., 2023). In response to these considerations, there are necessary measures that have to be put in place to ensure API security, such as the use of API gateways and security tools (Qazi, 2023). Further, there is a requirement for better knowledge of the developers, as well as the progressive methods for identifying and avoiding API abuse (Mousavi et al., 2023).

### Integration of APIs in Android Development:

Application Programming Interfaces (APIs) are also defining to the greatest extent the opportunities of improving user experiences and software development. APIs help in the functional rather than structural aspects and enable easier data finding and downloading as well as integration, as opposed to GUI-based pulling and pasting jobs that once took place ('What can APIs do for you?,' 2022). They can be used to build customer experience analysis tools so that customers may apply machine learning algorithms and display the outcome (Kopsiaftis et al., 2021). APIs also make software reusable by giving access to code repositories of open-source codes, although this might be a challenge to learn (Ishag et al., 2022). In this regard, literature has looked into the quality of API documentation with aspects such as understandability and completeness of API documentation. They have urged the creation of tools such as the Scout-bot that enable users to discover and learn about APIs from the gathered knowledge of the community within the Q&A platform like Stack Overflow (Ajam, Rodriguez and Benatallah, 2021). In conclusion, APIs are very desirable when it comes to developing sites and applications that are attractive to the eye and functional to the most part by offering dynamic content and also as a tool that can ease or at least make a few of the processes of development easier.

### Security Concerns in API Integration:

Current research revolves around key security concerns that relate to the use of APIs in mobile applications. Data breach and threats are principal risks, especially in automotive apps and super apps (Chatzoglou, Kambourakis and Kouliaridis, 2021; Wang, Zhang and Lin, 2023). Such risks include claiming more permissions than necessary, especially for privileges that are confidential; use of APIs that violate users’s privacy; and externally exploitable weaknesses (Chatzoglou, Kambourakis and Kouliaridis, 2021). Every networking super app offers some form of API that could provide miniapps the capability to dodge restrictions placed on them and to access the super app’s systems with higher authority (Wang, Zhang and Lin, 2023). In order to solve these problems, secure API calls are necessary in Android applications to secure the data. There are some works that have been done by the researchers, like APIScope, that helps in detecting the hidden APIs and has the potential to analyse their security effects (Wang, Zhang and Lin, 2023). Also, resources based on the OWASP API Security Project for the education of developers about threats and measures have been developed (Idris, Syarif and Winarno, 2022). These measures are intended to enhance the security of mobile applications and prevent threats associated with API consumption.

### OAuth 2.0 and API Security:

OAuth 2.0, as this is one of the most used authorisation schemes, although in many organisations it lacks a high level of security. A survey conducted among 100% OAuth identity providers reported that on average 34% of the aspects of security specification were not implemented, presuming 97% of the providers were left open to threats (Philippaerts, Preuveneers and Joosen, 2022). To this end, researchers designed Cerberus, an automated static analyser, which pointed out 47 susceptibilities in well-known OAuth libraries (Rahat, Feng and Tian, 2022). However, these are the following challenges facing OAuth 2. Further, breaking down the number 0 is still beneficial for the safe API authentication. As compared to Basic Auth, OAuth 2.0. More stable response times and requests per second were tested in load testing at 0 concluded from Wulandari, Wibowo and Susanto (2021) in this study. Furthermore, OAuth 2.0 can be effectively implemented in single sign-on systems such as Google Identity that, when integrated with REST architecture, enable users’ easy access and valid access and also promote the high speed for data transfer between client and server (Senapartha, 2021).

### RESTful API Security Practices:

A number of current research works have focused academically on examining the reasons behind patterns of API security implementation in diverse settings. In the same year, Chatterjee and Prinz (2022) carried out a study to show how Spring Security can be combined with Keycloak to prevent the attacks on microservice APIs with an efficiency of 100%. Qazi (2023) pointed out the areas of weakness that APIs have in cloud applications and stressed the importance of extensive development of security consciousness and security tools among cloud users and its service providers. In the study carried out by Munsch and Munsch (2021) with a focus on API security management, the authors discovered that the threats remain unnoticed most of the time and the perception of preparedness differs based on the professional background of the IT workers. They also described the basic measures that should be adopted to improve the security of APIs. Bass et al. (2021) demonstrated the viability of embedding the PROMIS API within three medical centres’ electronic health record systems for integrating patient-reported outcomes in clinical practice. Cumulatively, these studies highlight how critical it is for organisations to ensure adequate security of APIs and how it is possible to achieve this across the domain.

### Comprehensive Testing Methods:

Recent work has been directed toward enhancing the strategies for API security testing. Regarding test automation for REST APIs, there are studies regarding the comparison of different tools as well as approaches in order to determine the advantages and disadvantages (Kim et al., 2022). Fuzzing has become one of the popular techniques for security testing due to its ability to produce a large number of test cases for pinpointing the applications for the bugs (Zhu et al., 2022). Compatibility problems in Android apps have also been solved with the help of comparative studies of existing methods, which have shown that it is necessary to develop mechanisms for more flexible and complete detection (Liu et al., 2022). Testing methodologies for the RESTful API are problematic due to the lack of detailed internal information, which led to investigations into the frameworks and auto-generation of unit test techniques. Different types of issues and approaches have been discussed in a systematic literature review towards RESTful API testing, and proper testing is beneficial in avoiding failures in service operation and financial as well as trust loss (Ehsan et al., 2022).

## Conclusion:

Mobile applications need to include external APIs, unique UI, efficiency, and elaborated tests, which are all evident from the literature review. In an attempt to provide a unique wallpaper application for Android systems, Dynamic Walls will utilize the following strategies; Pexels API will be used and proper design techniques will be employed to enhance the user experience Aspect of the application, loading of images and caching will also be optimized to meet the objective of the application. Such a practice can be maintained to ensure that similar apps are compared with it to ensure that it remains distinct and competitive.

# Chapter 3

# Methodology:

## App Design and User Interface:

The first step towards the execution of our project was the approval of the design, from which the app can be easily recognized and navigated by users. The design process began with the incorporation of a splash screen featuring animated elements. The splash screen, sometimes also called the candy bar, is the first screen shown to the user when the application is started, and it usually contains logos and animations. In our app, a splash screen, which informs the user about the availability of logos and pantomimes and conveys their names in the form of an animation containing images and text, is used. All these logos are obtained from Pexels. Stock, which specifically accesses the website with a multitude of high-quality images that can help enliven the application’s interface. After the splash screen, the main screen of the app is the next model, which is created in a simple or comfortable style to use. The different fields of the primary page are categories, wallpapers, and favorite wallpapers. The categories section allows the users to scroll between different themes and styles of wallpapers, and the other favorite wallpapers section is where the users can store and easily get their favorite wallpapers. This organization focuses on the user experience in terms of a simplified connection and the availability of various kinds of wallpapers and settings.



Figure . APP Start

## Research and Development on Pexels API:

One of the most vital approaches used in this project was research and development on the Pexels API. The Pexels API is a strong resource to effectively embed into an application in order to search for and use a vast number of quality images. The goal set before us was to incorporate this API into the app to give users many options for wallpapers. The Pexels API offers two primary methods for retrieving images: However, there is a search-based one and a curated one.    In the search-based method, the keywords can be posted by the users, and the API retrieves the images that satisfy that query. This feature is very helpful for users who already know the kind of themes or subjects of the wallpapers they want to download. For this, we used Retrofit, which is a modern-type annotated HTTP client for Android and Java, to manage the API calls and returns effectively. API Retrofit helps in simplifying network requests, JSON parsing, and API management, making it suitable for integrating the Pexels API.

## Implementing Curated Images:

From the above method, we can obtain or showcase a list or gallery of images that are trendy. Top-trending and randomly suggested images are usually vast and varied, so customers can choose aesthetic photographs without making distinct requests. In our app, since the curated section is set to have 39 images, the design APK we created for demonstration is as follows: this number was chosen to allow for a rather large number of pictures while at the same time still filtering it in a way to not be overly resource-intensive or take some time to load.

For the implementation of the image curation, we defined all the respective data models and APIs for the acquisition of images in the application. This involves setting up the API specifications, managing the JSON response, and incorporating the images into the user interface of the app. As for the designated images section, our idea is to help users have a good time and not need to look for suitable and popular wallpapers manually.

## Implementing Search-Based Images with Retrofit:

As for the search-based method, R&D work on Retrofit was needed to allow users to search for specific categories and get corresponding images. With Retrofit, doing operations with the Pexels API is made easier through handling network calls and the conversion of the JSON results into high-level data types. This makes it possible for us to concentrate on giving the user the best experience without having to work on the layer that handles the network transmission.

In our app, we set up the search feature to return 10 pictures per search query. This number is chosen as it will allow for enough variety to help the users while also ensuring that the performance of the application is not substantially hindered. Users can input any keyword into the search bar, and the application will come up with a group of images that correspond to the keyword. This feature enables users to just scroll and pick wallpapers by categories of their choice, all from a single account.

To this end, it was necessary to define the set of API operations as well as the data models required for processing the received requests and constructing the corresponding response. We also incorporated the search feature into the GUI, enabling the users to view and select the images they wanted in the application. Overall, Retrofit allowed the implementation of more solid and optimized search functionality for the project.

## Comprehensive Testing and Optimization:

This case requires tests to be incorporated as a part of our methodology to evaluate the performance of the app in those scenarios on various types of devices. Compatibility testing, performance testing, and user acceptance testing were also done in an effort to point out any possible glitches.

The compatibility test was conducted whereby the mobile app was installed and operated on different Android gadgets to determine its performance on different gadgets in terms of screen sizes, resolutions, and hardware capabilities. This was useful to find out the problematic combinations when it comes to users’ interactions with the elements of the system.

Stress testing is aimed at enhancing the loading time of the app and its memory and battery consumption. Debugging and controlling the quality of an application were provided through tools like Android Profiler. These included images loading and caching, which make it easy for the app to load images very quickly and ensure that they do not use a lot of the device's working memory.

User acceptance testing was carried out as an experiment by performing the app with selected user groups and collecting their feedback about its utility and efficiency. It makes it relatively easier to identify the possible usability problems and the possible improvements to make by involving beta testers. As received by the participants, certain modifications were made to the observation that could improve the flow of features.

In order to achieve the highest results of testing as well as to make an application intuitive and engaging, we set goals to check all significant aspects and scenarios concerning usage on various devices.

|  |  |
| --- | --- |
|  |  |
| Figure . APP Desktop | Figure . APP Features |
| Figure . APP Categories | Figure . Specific Category |

## Conclusion:

The approach used in the creation of Dynamic Walls was to produce a clearly defined and exhaustive approach, which entailed the solidifying of the apps in the in the last stages, the research and development of the Pexels and Retrofit APIs, the image curation and retrieval, as well as the testing and optimization of the applications. With these methodologies, the idea was to develop a bright, combined, aesthetic and functional application that would become a worthy representative of the ‘wallpaper’ category in an increasing number of app stores. At this stage, the constant comparison of the app with similar apps will be useful for preserving the app’s differentiation and adapting it to the user’s needs.

# Chapter 4

# Result and Disscussion:

## App Design and User Interface:

The design and the interface of the application, the “Dynamic Walls,” were also essential factors for the success of the application. To start the design process, the first step was to design the splash screen, which included logos and other graphics and images sourced from Pexels. It was also important in setting the tone of the application, as it made a direct interaction with the user at the first instance. This screen, which incorporates the use of animations and visual aids, makes for a nice welcome screen to the features of the app. The use of the animations on the logos and text gave a clear impression of the app's concept as well as the rest of the interface.

The participants’ feedback showed that the use of the animated splash screen was useful in improving the attractiveness of the app. People mentioned that the dynamic elements of the splash screen made the app look interesting and prompted the users to proceed with the application’s further exploration. It was observed that the use of graphics and images was most helpful in engaging the user’s attention, especially within the first few seconds of the app launch.

Following the splash screen, the app is directed to the main screen that has been developed with a simple design to enable the user to navigate easily. The main screen is divided into three primary sections: “Categories,” “Wallpapers,” and “Favourite Wallpapers.” The design of these sections was intentionally made very basic in order to give the user a smooth experience. The “Categories” option provides the users with an option to search through different categories and styles, while the “Favourite Wallpapers” option helps users save their favourite wallpapers and access them easily.

The categorisation approach was very successful in increasing the level of user engagement. People liked the simple and easy-to-understand structure of the app that did not leave the users lost within the application. Due to this, the design of the app was made in a way that it would not overload the user’s cognition by providing information in an organised manner. This approach can be considered consistent with the principles of user-centred design, which require a clear and unambiguous interface and navigation.

In addition, the design was made with the ability to adjust to various screen sizes to give the users the same experience across different devices. This was done by paying close attention to aspects such as layout management, button placement, and text readability. From the user acceptance tests, it was gathered that users liked the interface of the app and found it to be simple and visually appealing. This can be explained by the fact that the consideration of UX design, which was vital to the success of the app design, was well incorporated.

## Integration of Pexels API:

The “Dynamic Walls” app was designed with the use of the Pexels API, which offered a vast collection of high-quality images. This was important as it made the app have a wide range of wallpapers for the users to choose from. The Pexels API supports two main methods for retrieving images: The two fundamental strategies that have been used in developing the above classification are a search-based method and a curated method. Each of the two methods was incorporated in the app to address the needs of the users.

The search-based method enables users to input certain terms in order to get specific types of wallpapers. This feature was most beneficial for the users who already had a particular set of wallpapers in mind that they wanted to find. The users who are in the search of ‘nature’ or ‘abstract’ wallpapers can type these words and get the results that are related to them. Retrofit, a new generation HTTP client for Android, was employed, which helped in effective handling of APIs through simplifying the network calls and handling of JSON. The capability of Retrofit in managing network operations with little code greatly improved the efficiency and effectiveness of the search feature as compared to the previous one.

The implementation of the Pexels API helped the app fetch content from a third-party source; the app would always provide the most recent and popular wallpapers to the users. This dynamic content delivery was important in the sense of separating Dynamic Walls from the other wallpaper applications that used static content only. This increase in the use of the app can be attributed to the search feature that is integrated with the Pexels API that enables the user to find wallpapers that suit their choice.

On the other hand, the curated method gave a set of 39 of the most popular and completely random images. This feature was implemented with the idea of users who could be willing to change the wallpaper but did not have a particular image in mind. The curated section was rich in the variety of high-quality images that corresponded to contemporary tendencies and gave the user an attractive and active interface. By limiting the filtered and curated set of photos to 39, the app was able to provide enough options without compromising the performance. The number was chosen in such a way that there is enough variation while at the same time ensuring that loading times and memory usage are kept to a minimum.

Search-based and curated methods were used to create the app, which meant that it would be able to meet the needs of a wide range of users with distinct tastes. The featured tab that contained an ever-changing set of popular images was a source of inspiration for the users. On the other hand, the search-based feature was more user-friendly since it provided the user with wallpapers that he or she would prefer. This dual approach therefore helped in increasing the satisfaction of the users by providing both the aspect of having a wide range of content as well as having detailed content.

## Implementation of Curated and Search-Based Images:

The achievement of both the curated and search-based image retrieval approaches highlighted the flexibility of the application. The curated selection, which was intended to present 39 trendy images, was selected in such a way that it would provide users with a good number of high-quality wallpapers that they could use, but not so many as to make them feel overwhelmed. Such a decision was made as it was important to find the balance between the number of images and the possibilities provided to the user. An excessive number of images will result in slow loading times and high memory usage, which may not be very encouraging to the user.

According to the users’ feedback, the curated section was appreciated by the readers. People liked the choice of wallpapers and considered it helpful to have the collection of images to choose from and discover new options. The curated section, which displayed a gallery of images that was constantly changing, played a critical role in keeping the users engaged. This helped in the design to limit the display to specific images only to avoid slowing down the application or using too much memory in less powerful devices.

The search-based method was introduced in order to give the users a more personalised experience. This feature was used to enable users to search for particular images with the help of certain key words, which would generate a list of images that were relevant to the keywords entered. This was done in an effort to offer enough choice while at the same time ensuring that the app’s performance is not greatly compromised. This approach made sure that the users were able to search through the available wallpapers and get what they wanted in the shortest time possible with no difficulty of any kind.

This study identifies some key strategies that were implemented in the app’s design and development in order to enhance its usability; these include the use of Retrofit in managing the search-based functionality of the application. This was made possible by Retrofit, which helped in managing network calls while also converting JSON responses to high-level data types, thus providing a fluid search interface to the app. The users would be able to enter certain keywords and get the results that would be relevant to them within a matter of seconds, which was quite a leap from traditional methods of handling API requests.

The search feature was also well incorporated in the app’s GUI to enable the users to be in a position to access and use the search feature with ease. The GUI design was chosen to make the search bar easily noticeable and easily accessible to the users to entice them to use this feature. From the user acceptance testing, it was noted that users were comfortable with the search function and that it was easy to use. The addition of the feature to search through the wallpapers made the app more user-friendly and versatile to use.

## Comprehensive Testing and Optimization:

The overall testing and fine tuning made it possible to provide high quality of the app, its stability, as well as its compatibility with a large number of devices. Three types of testing were conducted: such assessments can be on compatibility, performance, and user acceptance testing.

### Compatibility Testing:

Compatibility testing was performed to check the app’s functionality on different Android devices with different sizes and density of screens, as well as different hardware configurations. It was important to check the app functionality across all supported platforms, and this is why the testing was done. The app was deployed and used on various devices, such as smart phones and tablets, in order to study its performance. From the compatibility testing, it was established that the app was compatible with all the tested devices with no major screen or hardware-related problems.

However, several issues were realised during the compatibility testing; Slight off-positioning of some features on devices with unconventional screen dimensions. These problems were solved by simply manipulating the layout parameters to ensure the same display for both mobile and web applications. This was done in a highly iterative manner, due to which the compatibility level was very high and the user experience was also very much standardised.

### Performance Testing:

The performance testing aimed at assessing the application’s performance, including reaction time, loading time, memory usage, and battery consumption. Features that are available, such as Android Profiler, helped in the monitoring of the performance of the application in order to determine any areas that might be slowing it down. It is based on the test that it was observed that the app’s image loading and caching system was quite efficient in enhancing the performance. By caching the images, which are used most often, it was possible to decrease the time of their loading and decrease the network usage, thus providing the users with a better experience.

Stress testing was also carried out in order to measure the app’s behaviour when under heavy usage, When undertaking several searches at the same time or when flicking through different parts of the app. These conditions were found to have no significant adverse effect on the app’s performance, with no delays and no crashes or freezes. According to the stress testing, the app demonstrated high performance and can withstand a large number of user requests, which is critical for a good user experience.

### User Acceptance Testing:

User acceptance testing (UAT) was done to ensure that the app was tested with a group of users to obtain feedback about the usability and functionality of the app and in general the user experience. The tasks included searching for certain wallpapers, exploring the curated gallery, and saving the wallpapers that the participants liked. The feedback that was obtained from the user acceptance testing was also mostly positive, and users complimented the design of the app, its features, and the way it performed.

The testing also revealed some issues that could potentially be addressed. Some users said that the search could be improved by offering the option to filter images based on the resolution or color. These are some of the changes that were made to the app after considering the users’ feedback. Due to the cyclic nature of UAT, the process was able to be fine-tuned in the course of the testing, leading to a better product in the end.

### User Testing Feedback:

User testing feedback plays an essential role in the product development process since it gives valuable insight on how real users use a particular product or application. This feedback points out areas that may need improvement, like usability problems, design defects, or unforeseen behaviours in a system. In user testing, subjects perform given duties, and their experiences are monitored with a view to discovering possible pain points. Suggestions for enhancing navigation, increasing reactivity, or making some operations simple often feature in this feedback. By doing so, developers can improve the general experience of their customers, making sure that what they eventually come up with meets what the users were expecting from it all along. Moreover, user testing through feedback can also be utilised for prioritising features and bugs, which results in products being more elegant and easier to manipulate. Creating products that are easy to use, efficient, and fun for people is largely reliant on iterative testing cycles coupled with user feedback.

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Figure . Users Feedback

## Discussion:

The findings of the “Dynamic Walls” project point to the fact that the design, functionality, and user experience should be given due consideration when developing apps. It was found that the integration of the two methods, namely the curated and the search-based image retrieval methods, played a critical role in differentiating the wallpaper applications in the market. This work has been made possible by the integration of the Pexels API to provide a variety of images to the app and the use of Retrofit to enhance the performance of the app, especially in network-related activities.

The decisions made in the design process, such as the simple and elegant interface, ensured that the users found the app easy and pleasant to use. Due to the iterative testing and optimisation process, the applicability, compatibility, and efficiency of the app were guaranteed, which made the app even more attractive to the users.

Possible future developments for the “Dynamic Walls” app can be the enhancement of the existing functionalities, including the ability to search for specific images, edit images, and share on other social media platforms. These improvements may add more benefit to the users and may also help distinguish the app in the market. Furthermore, future enhancements and modifications shall be done often to suit the user feedback and technological changes in the market in order for the app to remain relevant in the market.

In conclusion, it can be stated that while developing the “Dynamic Walls” app, the authors successfully implemented the idea of the innovative design, high functionality, and focus on the user to create a successful and engaging product. The level of user engagement and the feedback received from this project demonstrate that a complete and cyclical model of application development is effective.

# Chapter 5

# Conclusion

## Recap of the Project Objectives and Methodology:

The Dynamic Walls project proposed the design of an Android wallpaper application that is capable of providing a plethora of high-quality images to the user and an intuitive interface. The first goal was to design an application that can incorporate both a collection of wallpapers and a search functionality in order to be able to address users with different tastes. The images used in the application were from the Pexels API, while Retrofit integration allowed for efficient communication with the API.

In order to achieve these objectives, the following methodology was used in the design and development of the app, integration of APIs, and testing phases. Due to that, the app was designed in a simple way with the aim of having a simple user interface (UI) that will make the navigation easy and improve the overall user experience (UX). The development phase involved the integration of the Pexels API through Retrofit for the purpose of dynamic content, and the testing phase involved a compatibility test, a performance test, and a user acceptance test for the purpose of determining the efficiency and satisfaction of the app’s users.

## Key Achievements and Successes:

The “Dynamic Walls” project achieved its goals since the application offers a large number of wallpapers and offers a fluid and fluent interface for various devices. The Pexels API was a great addition to the app as it provided curated as well as search-based images from a huge repository. This integration helped the users to search and navigate through the wallpapers in a more efficient way and for both targeted and general browsing needs.

The choice of Retrofit as a tool for managing network requests in the form of an HTTP client showed that it has certain performance benefits, such as faster loading and increased stability. Retrofit helped in the efficient handling of API calls and data parsing, which was very useful in enhancing the performance of the app, thus making it more responsive to the user. This way of combining the curated and the search-based image retrieval gave the users the ability to either search for the most popular images or the images that would fit their preferences.

From a design point of view, the simplicity of the app and its interface can be attributed to the success of the application. The simple and clean interface with clear grouping of content allowed the users to easily find the wallpapers they wanted and consequently improved the satisfaction of the users. It also made use of the responsive design, which adjusts the layout of the app interface to fit any screen size and display resolution, which also pointed to the developers’ focus on providing the best possible experience to the users across all the supported platforms.

## Discussion on the Integration of Pexels API and Retrofit:

The Pexels API was also an important part of the project since it helped to deliver the app with constant and variable content. Through the use of the Pexels API that provides great images, the app was able to stand out from the competition that uses no moving images. The API’s ability to work with both the search-based and the curated image retrieval methods made for a more enjoyable and more individualised experience for the user.

This decision was extended, and the use of Retrofit to manage HTTP clients for API requests gives robustness to the application. Retrofit is a very efficient library that provides easy implementation of complex network operations without much coding, and that is why it was used in this project. This was because it had a built-in mechanism for asynchronous calls whereby the application’s interface would not freeze even when it was busy processing large amounts of data or servicing many requests at the same time. This was particularly crucial because the content of the app was changing every now and then, and therefore there was a need to keep on making API calls to get new wallpapers.

The code that has been written for the integration of the Pexels API with the Retrofit has been efficient enough that it does not load the app with unwanted delays, which affected the overall performance of the app. Together with the use of these two technologies, the app was able to provide the users with a constantly changing collection of wallpapers while at the same time providing the users with easy access to the wallpapers. This has been made possible through the use of third-party services and libraries that were incorporated in the app, and this can be said to have contributed to the success of the app.

## Impact of User-Centered Design on App Success:

The user-centred design process used in the “Dynamic Walls” project played a big role in ensuring that the users’ satisfaction and interaction levels were high. The clean, non-cluttered look of the app, along with the simple interface and easy navigation, made it possible for users to navigate through the app and discover its functionalities without getting lost in the process. The application made the process as simple as possible, focussing on clarity and, as a result, reducing users’ cognitive load to find and select wallpapers.

 These features added to the app, such as the animated splash screen and the curated galleries, gave the app a lively feel that made it more interesting to use. The splash screen was an animated picture, which helped in grabbing the attention of the user immediately after the app was launched. The galleries that were selected and featured were refreshed frequently with new content, and this ensured that the users visited the app often.

 Another aspect of the user-centred design of the app that also influenced the choice to use both curated and search-based image retrieval techniques is the decision to support both. The curated section was a simple way for the users to get to the most popular and trending wallpapers, while the search-based allowed the users to find wallpapers that suited them personally. This approach made the app versatile enough to provide for both the users who would prefer to go through a set of images and those who would want to search for specific wallpapers that they like.

 In general, adherence to user-centred design principles could be considered as the main factor that contributed to the app’s success. Through consideration for the needs and preferences of the users, the project team was able to come up with an application that was useful and fun to use. This way not only improved the user satisfaction but also improved user engagement and user retention for the app, which finally leads to sustainability in the market.

## Insights from Comprehensive Testing and Optimization:

The feature and stability testing as well as the optimisation for different devices that have been undertaken in the development of “Dynamic Walls” were crucial in guaranteeing the stability of the application as well as the high performance. The testing phase included three key types of testing: compatibility testing, performance testing, and finally the user acceptance testing.

As a part of the cross-browser testing, compatibility testing was specific to ensure that the app was working properly on different Android devices with different sizes of the screen, resolution, and hardware configuration. The test results of this phase show that the app was very compatible with all tested devices with only a few minor issues that arose with the placement of the screen, which was easily fixed. This level of compatibility was important in order to provide the same level of user experience across all different devices.

To assess the usability and efficiency of the application, performance testing was conducted, including the analysis of the app’s response, loading time, memory consumption, and impact on the battery. Here, the performance of the app’s image loading and caching was tested, and it was seen that the app had excellent performance in this regard. With images cached on the app’s side, the app was able to limit its use of network resources and lessen the time it took to load images, thus enhancing the overall performance of the app for the users. Stress testing also performed during this phase also showed that the app would be able to cope with a large number of users using the app and not experience any slowdown or jam, adding more validation on the sturdiness and dependability of the app.

User acceptance testing (UAT) was useful in identifying the user friendliness and effectiveness of the app from the end user’s perception. The feedback that was collected during this phase of testing was positive to the extent that users complimented the app on its appearance, usability, and speed. But there were also some recommendations for its further development, the possibility to sort the images by their resolution or color. Some of these were considered, and the following changes were made to the app in order to make it more user-friendly. Because UAT is a cyclical process, it was possible to make gradual improvements and enhancements to the product and make it more user-friendly.

The testing and optimisation activities performed during the course of the “Dynamic Walls” project was critical to the overall stability and performance of the app as well as the level of satisfaction among the users. Through thorough experimentation of the app in different scenarios and different devices, the project team was able to pin-point and address possible problems that could arise in the future, which in turn led to the production of a high-quality app that would be able to capture the needs and wants of its consumers.

## Future Directions and Recommendations:

The process of the “Dynamic Walls” application has been effective in achieving its primary objectives, and, at the same time, there are several potential ways of its improvement to enhance the application’s value and its usage for the users. In my opinion, one possible improvement is the enhancement of search options that would include options such as image resolution, colour, and dimension. These would be filters that would allow the users to narrow down on the search results so as to get the wallpapers they need for their specific desires.

There is also a possibility of adding an edit option for images, where people can crop, change the brightness and contrast, or apply filters to the wallpapers. This new feature would be useful for users to fine-tune into detail on how they want their wallpaper to look, thus improving the usefulness of the app.

There should also be a social sharing component incorporated in the app that allows users to share their preferred wallpapers on other social media sites. It also would not only enhance the user interaction but also would be a kind of word-of-mouth advertising that could bring more users to the application.

In order to remain relevant in the market and provide better services to the users, it should also be updated on a regular basis with new features, content, and changes suggested by the user or based on the advancements in the technology. The project team can build on the iterative model of the application and improve the app so that the “Dynamic Walls” remain relevant and fun for the users.

Therefore, it is safe to state that the “Dynamic Walls” project proved the concept of successful integration of creativity, durability, and the focus on the end user. The use of Pexels API and Retrofit, along with the user-friendly design and the thorough testing of the app, allowed the developers to create an app that would cater to the various needs of its users and offer them high performance and dependability.

This project is therefore a clear demonstration of the fact that when developing an app, both technical competence and user-centredness must be given due consideration. Using the current trending technologies and following the right process of designing and developing the app, the “Dynamic Walls” app has been able to establish itself in the competition for the wallpaper app.

Even as the app develops and gets even better, it has the potential of drawing in more users, hence assuring the app of long-term success. It can be certain that the knowledge obtained from this project will be valuable for future projects, and the experience gained from the “Dynamic Walls” project will help to develop new and innovative mobile applications.

# References

Ahmad, H.W.-U.-D., 2017. *Building RESTful Web Services with PHP 7*.

Ajam, G., Rodriguez, C. and Benatallah, B. (2021) 'Scout-bot: Leveraging API Community Knowledge for Exploration and Discovery of API Learning Resources,' *CLEI Electronic Journal*, 24(2). https://doi.org/10.19153/cleiej.24.2.5.

Ali, W., Riaz, O., Mumtaz, S., Khan, A.R., Saba, T. and Bahaj, S.A., 2022. Mobile Application Usability Evaluation: A Study Based on Demography. *IEEE Access* [Online], 10, pp.41512–41524. Available from: https://doi.org/10.1109/access.2022.3166893.

*Android Developers* [Online], 2024. Android Developers. Available from: https://developer.android.com/reference/android/app/WallpaperManager [Accessed 26 July 2024].

Apple, 2023. Swift Programming Language. Apple Developer. Available from: https://developer.apple.com/swift/ [Accessed 30 July 2024].

Barlev, T., 2024. *Mobile App Personalization in Custom App Development*. *Goji Labs | Mobile App, Software, & Web Development* [Online]. Available from: <https://gojilabs.com/blog/mobile-app-personalization-in-custom-app-development/>.

Bass, M. *et al.* (2021) 'Implementing an Application Programming Interface for PROMIS Measures at Three Medical Centers,' *Applied Clinical Informatics*, 12(05), pp. 979–983. https://doi.org/10.1055/s-0041-1736464.

Chatterjee, A. and Prinz, A. (2022) 'Applying Spring Security Framework with KeyCloak-Based OAuth2 to Protect Microservice Architecture APIs: A Case Study,' *Sensors*, 22(5), p. 1703. https://doi.org/10.3390/s22051703.

Chatzoglou, E., Kambourakis, G. and Kouliaridis, V. (2021) 'A Multi-Tier Security Analysis of Official Car Management Apps for Android,' *Future Internet*, 13(3), p. 58. https://doi.org/10.3390/fi13030058.

Contributors, C. (2023) Coil. https://coil-kt.github.io/coil/ (Accessed: July 31, 2024).

Coppola, R., Ardito, L. and Torchiano, M. (2019) 'Characterizing the transition to Kotlin of Android apps: a study on F-Droid, Play Store, and GitHub.' https://doi.org/10.1145/3340496.3342759.

Dzulqarnain, F. and Tukino, T. (2023) 'RANCANG BANGUN APLIKASI BELAJAR ARAB UNTUK ANDROID MENGGUNAKAN JETPACK COMPOSE DAN KOTLIN,' *Computer Based Information System Journal*, 11(1), pp. 25–35. https://doi.org/10.33884/cbis.v11i1.6666.

Ehsan, A. *et al.* (2022) 'RESTful API Testing Methodologies: Rationale, Challenges, and Solution Directions,' *Applied Sciences*, 12(9), p. 4369. https://doi.org/10.3390/app12094369.

Elgendy, I.A., Zhang, W.-Z., Liu, C.-Y. and Hsu, C.-H., 2021. An Efficient and Secured Framework for Mobile Cloud Computing. *IEEE Transactions on Cloud Computing* [Online], 9(1), pp.79–87. Available from: https://doi.org/10.1109/tcc.2018.2847347.

Facebook, 2023. React Native. React Native Documentation. Available from: https://reactnative.dev/ [Accessed 30 July 2024].

Fehling, C., Leymann, F., Retter, R., Schupeck, W. and Arbitter, P., 2014. *Cloud Computing Patterns* [Online]. Available from: https://doi.org/10.1007/978-3-7091-1568-8.

Fresco An image management library. (2023). https://frescolib.org/ (Accessed: July 31, 2024).

Gan, C.L. and Balakrishnan, V., 2016. An Empirical Study of Factors Affecting Mobile Wireless Technology Adoption for Promoting Interactive Lectures in Higher Education. *The International Review of Research in Open and Distributed Learning* [Online], 17(1). Available from: https://doi.org/10.19173/irrodl.v17i1.2111.

Gao, J., Bai, X., Tsai, W.-T. and Uehara, T., 2014. Mobile Application Testing: A Tutorial. *Computer* [Online], 47(2), pp.46–55. Available from: https://doi.org/10.1109/mc.2013.445.

*Glide* [Online], 2024. github. Available from: https://bumptech.github.io/glide/#about-glide [Accessed 26 July 2024].

Glide v4 : Fast and efficient image loading for Android (2023). https://bumptech.github.io/glide/ (Accessed: July 31, 2024).

Google, 2023. Kotlin for Android. Android Developers. Available from: https://developer.android.com/kotlin [Accessed 30 July 2024].

Hartawan, M.S. (2022) 'PENERAPAN USER CENTERED DESIGN (UCD) PADA WIREFRAME DESAIN USER INTERFACE DAN USER EXPERIENCE APLIKASI SINOPSIS FILM,' *JEIS Jurnal Elektro Dan Informatika Swadharma*, 2(1), pp. 43–47. https://doi.org/10.56486/jeis.vol2no1.161.

Hasari, Y.I., Febriansyah, A. and Anzana, Z.S. (2022) 'PENERAPAN METODE USER CENTERED DESAIN PADA PERANCANGAN INTERFACE APLIKASI PEMESANAN DAN PEMBAYARAN TIKET BIOSKOP BERBASIS MOBILE,' *Jurnal Siliwangi Seri Sains Dan Teknologi*, 8(2). https://doi.org/10.37058/jssainstek.v8i2.6254.

Idris, M., Syarif, I. and Winarno, I. (2022) 'Web Application Security Education Platform Based on OWASP API Security Project,' *EMITTER International Journal of Engineering Technology*, pp. 246–261. https://doi.org/10.24003/emitter.v10i2.705.

Ishag, M.I.M. *et al.* (2022) 'Highlighting Current Issues in API Usage Mining to Enhance Software Reusability,' *WSEAS TRANSACTIONS ON COMPUTER RESEARCH*, 10, pp. 29–34. https://doi.org/10.37394/232018.2022.10.4.

JetBrains, 2023. Kotlin Programming Language. Kotlin Official Website. Available from: https://kotlinlang.org/ [Accessed 30 July 2024].

Karjatkar, S. *et al.* (2024) 'AN EMPIRICAL STUDY OF DEVELOPING ANDRIOD APPLICATIONS,' *International Research Journal of Modernization in Engineering Technology and Science* [Preprint]. https://doi.org/10.56726/irjmets51243.

Kaura, S. (2024) 'Redesigning Android Development: Using Reactive Programming to Retrofit REST APIs and Concurrency,' *INTERANTIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING AND MANAGEMENT*, 08(04), pp. 1–5. https://doi.org/10.55041/ijsrem30623.

Kim, M. *et al.* (2022) *Automated test generation for REST APIs: no time to rest yet*. https://www.semanticscholar.org/paper/Automated-test-generation-for-REST-APIs%3A-no-time-to-Kim-Xin/04e9b207e34031babe1309ad5ca4b688b78e2a1d.

Kopsiaftis, G. *et al.* (2021) 'Application Programming Interface for a Customer Experience Analysis Tool,' in *Frontiers in artificial intelligence and applications*. https://doi.org/10.3233/faia210092.

Kumar, B. and Falhi, O.A. (2022) 'Digital Transformation through APIs,' *2022 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COM-IT-CON)* [Preprint]. https://doi.org/10.1109/com-it-con54601.2022.9850728.

Liu, P. *et al.* (2022) *Automatically detecting API-induced compatibility issues in Android apps: a comparative analysis (replicability study)*. https://www.semanticscholar.org/paper/Automatically-detecting-API-induced-compatibility-a-Liu-Zhao/ba320ed119a550c4a20c210e1002fcdb54eff99c.

Mahfudh, A. and Saputra, W.R. (2022) 'Perancangan User Interface User Experience Aplikasi E-Ngaji Berbasis Android Menggunakan Metode User Centered Design (UCD) Pada TPQ,' *Jurnal Ilmiah Intech Information Technology Journal of UMUS*, 4(02), pp. 255–262. https://doi.org/10.46772/intech.v4i02.885.

Mousavi, Z. *et al.* (2023) *Detecting Misuse of Security APIs: A Systematic Review*. https://www.semanticscholar.org/paper/Detecting-Misuse-of-Security-APIs%3A-A-Systematic-Mousavi-Islam/09e0c677d676b44cba1a20720df1ef0dda1b52d8.

Muddalkar, N.S.S. *et al.* (2022) 'Electric Vehicle Charging Station Finding App,' *International Journal of Advanced Research in Science Communication and Technology*, pp. 607–612. https://doi.org/10.48175/ijarsct-3359.

Mulyani, A., Kurniadi, D. and Hakim, I.L. (2021) 'Web Scraping Pada Web Media Digital untuk Membangun Aplikasi Android,' *Jurnal Algoritma*, 18(1), pp. 313–322. https://doi.org/10.33364/algoritma/v.18-1.949.

Munsch, A. and Munsch, P. (2021) 'The Future of API (Application Programming Interface) Security: The Adoption of APIs for Digital Communications and the Implications for Cyber Security Vulnerabilities,' *Journal of International Technology and Information Management*, 29(3), pp. 24–45. https://doi.org/10.58729/1941-6679.1454.

Nguyen, P.T. *et al.* (2022) 'Recommending API Function Calls and Code Snippets to Support Software Development,' *IEEE Transactions on Software Engineering*, 48(7), pp. 2417–2438. https://doi.org/10.1109/tse.2021.3059907.

Norman, D., 2013. *The Design of Everyday Things*. Constellation.

Nugraheni, A. and Maryam, M. (2022) 'PENERAPAN TEKNOLOGI QUICK RESPONSE CODE DAN APPLICATION PROGRAMMING INTERFACE PADA PERANCANGAN APLIKASI PERPUSTAKAAN (STUDI KASUS : SMP NEGERI 25 SURAKARTA),' *JIPI (Jurnal Ilmiah Penelitian Dan Pembelajaran Informatika)*, 7(3), pp. 821–834. https://doi.org/10.29100/jipi.v7i3.3096.

P, N.A. *et al.* (2022) 'ANDROID APPLICATION FOR STUDENT MANAGEMENT SYSTEM USING KOTLIN,' *International Journal of Engineering Technology and Management Sciences*, 6(6), pp. 88–91. https://doi.org/10.46647/ijetms.2022.v06i06.013.

*Pexels* [Online], 2024. https://www.pexels.com/api/. Available from: https://www.pexels.com/api/ [Accessed 26 July 2024].

Philippaerts, P., Preuveneers, D. and Joosen, W. (2022) 'OAuch: Exploring Security Compliance in the OAuth 2.0 Ecosystem,' *emanticscholar.org* [Preprint]. https://doi.org/10.1145/3545948.3545955.

Picasso (2023). https://square.github.io/picasso/ (Accessed: July 31, 2024).

Qazi, F. (2023a) 'Application Programming Interface (API) Security in Cloud Applications,' *EAI Endorsed Transactions on Cloud Systems*, 7(23), p. e1. https://doi.org/10.4108/eetcs.v7i23.3011.

Qazi, F. (2023b) 'Application Programming Interface (API) Security in Cloud Applications,' *EAI Endorsed Transactions on Cloud Systems*, 7(23), p. e1. https://doi.org/10.4108/eetcs.v7i23.3011.

Quispe, K.V.C. (2022) 'Desarrollo de aplicaciones móviles usando el lenguaje Kotlin,' *Dialogos Abiertos*, 1(1), pp. 22–33. https://doi.org/10.32654/dialogosabiertos.1-1.3.

Rahat, T.A., Feng, Y. and Tian, Y. (2022) 'Cerberus,' *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security* [Preprint]. https://doi.org/10.1145/3548606.3559381.

Rgp, P.F. and Hadi, A. (2019) 'RANCANG BANGUN APLIKASI BELAJAR PEMROGRAMAN DENGAN GAME EDUCATION PADA SMARTPHONE BERBASIS ANDROID,' *Voteteknika (Vocational Teknik Elektronika Dan Informatika)*, 7(3), p. 30. https://doi.org/10.24036/voteteknika.v7i3.105086.

Rua, R. and Saraiva, J., 2023. A large-scale empirical study on mobile performance: energy, run-time and memory. *Empirical Software Engineering* [Online], 29(1). Available from: https://doi.org/10.1007/s10664-023-10391-y.

Senapartha, I.K.D. (2021) 'Implementasi Single Sign-On Menggunakan Google Identity, REST dan OAuth 2.0 Berbasis Scrum,' *Jurnal Teknik Informatika Dan Sistem Informasi*, 7(2). https://doi.org/10.28932/jutisi.v7i2.3437.

Setiawan, B. and Triase, T. (2023) 'IMPLEMENTASI DESAIN UI/UX APLIKASI OURTICLE KE DALAM APLIKASI BERBASIS ANDROID,' *SIBATIK JOURNAL Jurnal Ilmiah Bidang Sosial Ekonomi Budaya Teknologi Dan Pendidikan*, 2(3), pp. 805–818. https://doi.org/10.54443/sibatik.v2i3.665.

Spencer, T. (2022) 'Securing the API economy,' *Computer Fraud & Security*, 2022(4). https://doi.org/10.12968/s1361-3723(22)70570-2.

Sulowski, D. and Kozieł, G. (2019) 'Comparative analysis of Kotlin and Java languages used to create applications for the Android system,' *Journal of Computer Sciences Institute*, 13, pp. 354–358. https://doi.org/10.35784/jcsi.1332.

Wang, C., Zhang, Y. and Lin, Z. (2023) 'Uncovering and Exploiting Hidden APIs in Mobile Super Apps,' *Advancing Computing as a Science & Profession* [Preprint]. https://doi.org/10.1145/3576915.3616676.

Weichbroth, P., 2024. Usability Testing of Mobile Applications: A Methodological Framework. *Applied Sciences* [Online], 14(5), p.1792. Available from: <https://doi.org/10.3390/app14051792>.

'What can APIs do for you?' (2022) *Past Global Change Magazine*, 30(1), p. 63. https://doi.org/10.22498/pages.30.1.63.

Wulandari, N., Wibowo, A. and Susanto, B. (2021) 'Penerapan RESTful API untuk Membangun Program Pembayaran Piutang Menggunakan Otentikasi OAuth 2.0,' *Deleted Journal*, 5(1), pp. 1–10. https://doi.org/10.21460/jutei.2021.51.230.

Zhu, X. *et al.* (2022) 'Fuzzing: A Survey for Roadmap,' *ACM Computing Surveys*, 54(11s), pp. 1–36. https://doi.org/10.1145/3512345.