Journey Genie: Travel Planning Application

Project Progress Report submitted

to

MANIPAL ACADEMY OF HIGHER EDUCATION

For Partial Fulfilment of the Requirement for the

Award of the Degree

of

Bachelor of Technology

in

Computer and Communication Engineering

By

Joshita Bolisetty	Shubhanshu Verma	Snigdha Jha
(210953072)	(210953072)	(210953082)

Under the guidance of

Dr. Vibha

Assistant Professor – Senior Scale

Department of I & CT

March 2024

1. Introduction

The Journey Genie app endeavours to deliver a feature-rich travel planning and management solution tailored exclusively for Android users. Our primary goal is to ensure a seamless experience for travellers by offering intuitive trip planning tools, access to diverse travel-related information, and efficient itinerary management capabilities.

Journey Genie boasts a user-friendly interface and advanced accessibility features, catering to travellers of all abilities. Our focus lies in providing an inclusive experience that empowers users to plan their trips effortlessly. Through Journey Genie, users can not only plan their itineraries but also access real-time weather updates and explore destinations using interactive maps. Additionally, they can discover various activities and attractions tailored to their preferences.

Key functionalities of Journey Genie include comprehensive trip planning tools, efficient itinerary management features, and seamless integration with backend services for real-time weather updates and database management. Developed using Kotlin, the Journey Genie app ensures compatibility with the Android platform while leveraging APIs for weather and maps to deliver accurate and up-to-date information to users.

2. Literature Survey

The paper by [1]Putranto et al. , 2020 delved into the comparison between Java and Kotlin as programming languages for Android application development. Java, renowned for its platform independence and object-oriented paradigm, has been in use for developing Android applications since a long time, despite its high verbosity. In contrast, Kotlin emerges as a modern alternative with concise syntax, combining object-oriented and functional programming features while maintaining seamless interoperability with Java. The study evaluates these languages through the lens of performance metrics such as compilation time, lines of code, and ecosystem support. Additionally, it explores their respective advantages and disadvantages, providing insights into their suitability for different development priorities, from APK size optimization to expedited development cycles. Through a meticulous analysis of experimental findings and existing literature, this survey offers a comprehensive understanding of the differences between Java and Kotlin, empowering us to make informed decisions in our Android development endeavour.

To know more about use of Kotlin for Android Development, we led to the study by [2]Oliveira et al, 2020 investigating the implications of Kotlin's official integration into the Android Platform, examining Stack Overflow inquiries related to Kotlin and Android, and conducting interviews with some developers. Results indicated that developers perceived Kotlin as accessible and beneficial, and preferred to use it over Java to make potential improvements in code quality, readability, and productivity. The key benefits of Kotlin that has resulted in it being more preferred are the seamless interoperability with Java, null safety features, reduced verbosity, support for lambdas and higher-order functions, and robust IDE integration. Despite these benefits, some disadvantages still exist, including handling null variables and optionals, toolset issues, readability concerns from functional paradigm usage, and Java-Kotlin interoperability complexities. As a conclusion, the developers recognize Kotlin's advantages in modernizing development practices while maintaining compatibility

with the established Java-based environment, signalling its promising impact on Android development.

The tutorial by [3]Batool et al. , 2019 provided outlines the development process for a basic Android weather application using Kotlin and the OpenWeatherMap API. It offers a step-by-step approach, starting from project setup in Android Studio to integrating Retrofit service interface to define the API calls and implementing UI layout and data fetching logic. It emphasizes the importance of prerequisite knowledge in Kotlin programming and familiarity with Android Studio. By leveraging Retrofit and Glide libraries, the tutorial demonstrates efficient data retrieval and image loading functionalities, enhancing the app's usability and visual appeal. While the tutorial provides a solid foundation for building a weather app or adding weather related functionalities to our app. In conclusion, tutorial serves as a valuable resource for developers seeking to create Android applications using Kotlin and integrating external APIs for real-time data retrieval.

[4]This tutorial available on mediumcom provided steps outlining the process of integrating Google Maps into a Kotlin/Android project, focusing on two main functionalities: displaying a marker on the map and obtaining the current location (latitude and longitude).For showing a marker on Google Maps, developers the first step is to add dependencies in the build.gradle file and set up the Google Maps API key in the AndroidManifest.xml. Then, the UI setup is demonstrated using a Fragment with SupportMapFragment.To obtain the current location, additional dependencies are added in the build.gradle file, including play-services-location. The AndroidManifest.xml is updated with the necessary permissions and Google Maps API key.

Kabukas[5] in his paper analysed the performance of SQLite and Firebase for use as a database alongside an Android application. The two major differences that got highlighted between these two were that SQLite is used as a relational database and it serves from local, whereas Firebase is used as a NoSQL database, and it is a real-time cloud-hosted database. Besides these differences, the paper gives a thorough analysis of both their performance in various operations like Inserting, querying, fetching, or deleting data. The results indicated that SQLite provides better performance than its counterpart, Firebase for most types of data operations except the operation of deletion of data. The paper concludes with the fact that though SQLite is better in performance, the choice of using SQLite or Firebase relies on several other factors like if local storage space availability is high or multiple clients does not exists, then SQLite can be used, else it is be better to stick with Firebase.

Babic[6] in his comprehensive guide provided the step by step process to create a simple chat application for Android using Firebase as the backend. The article outlines the implementation of Firebase for user authentication, message storage, and real-time updates, while adhering to the Model-View-Presenter (MVP) architectural pattern. Through detailed code snippets and explanations, the author illustrates key concepts such as creating Firebase references, handling user registration and login, retrieving and displaying messages, and managing user sessions. By integrating Firebase with MVP, the author emphasizes modularity, scalability, and the separation of concerns, providing a solid foundation for building chat applications and similar projects.

[7]The tutorial presents a detailed walkthrough on incorporating OTP View, also known as PinView, into an Android application using the PinView library. The PinView is the kind of

dialog box we get when we are prompte dto enter a received OTP, that is One Time Password. The article underscores the significance of OTP Views in enhancing security through two-factor authentication and phone number verification processes. It provides a detailed step by step guide encompassing project setup, library integration, layout definition, and Java implementation. While the tutorial effectively achieves its goal of enabling OTP verification in Android applications, it could further enhance its educational value by providing insights into advanced customization options and best practices for handling OTP functionality.

[8]This article provides a comprehensive guide to creating notifications in Kotlin for Android applications, catering to developers seeking to implement notification functionality for developmental purposes. By explaining the significance and types of notifications, including status bar, notification drawer, heads-up, and lock-screen notifications, the article offers a clear understanding of their varied formats and designs. With a step-by-step implementation approach, it navigates developers through project setup, layout design, and code implementation, ensuring a systematic and accessible learning experience. Through its detailed code snippets, explanations, and insights, the article equips developers with the knowledge and skills necessary to effectively integrate notifications into their Kotlin-based Android applications, contributing to their proficiency in mobile app development.

3. Problem Definition

The problem addressed by Journey Genie is the lack of a comprehensive and user-friendly travel planning and management solution tailored specifically for Android users. Current options may lack intuitive tools, accessibility features, and seamless integration with real-time information services, leading to possibly unsatisfactory trip planning experiences. Journey Genie aims to bridge this gap by offering a feature-rich app that empowers users of all abilities to effortlessly plan and manage their trips while accessing accurate and up-to-date travel-related information.

4. Objectives

Some objectives of the Journey Genie app include:

1. Enhancing User Experience: We want to provide users with a seamless and intuitive travel planning and management experience, ensuring ease of use and accessibility for all travellers.
2. Comprehensive Trip Planning: Journey Genie aims to offer a wide range of tools and features for users to plan every aspect of their trip, including itinerary creation, destination exploration, activities at the destination itself, along with real-time weather updates.
3. Inclusive and User-Friendly Design: The app strives to cater to travellers of all metaphorical shapes and sizes by incorporating advanced accessibility features, ensuring that everyone can enjoy the benefits of efficient trip planning regardless of their physical or cognitive limitations.
4. Real-Time Information Access: Journey Genie seeks to integrate with backend services to provide users with up-to-date information such as weather forecasts,

geolocation, and database management, enabling them to make informed decisions during their travels.

5. Seamless Integration: The app aims to seamlessly integrate with Android devices, leveraging programming languages and APIs for weather and maps to ensure compatibility and reliability.
6. Personalized Recommendations: Journey Genie endeavours to offer personalized and popular recommendations for activities and attractions based on user preferences, enhancing the overall travel experience and encouraging exploration of new destinations.

5. Methodology

1. Requirement Gathering for Journey Genie:

Understanding the precise requirements and features necessary for Journey Genie is paramount to its success. This entails describing core functionalities such as user authentication, destination and accommodation search capabilities, itinerary creation etc. Furthermore, it involves conducting market research through existing applications and websites to prioritize features and comprehend user preferences. By collecting and analyzing this data, we could compile a comprehensive list of user stories and prioritize features based on their importance and feasibility.

2. Designing Journey Genie

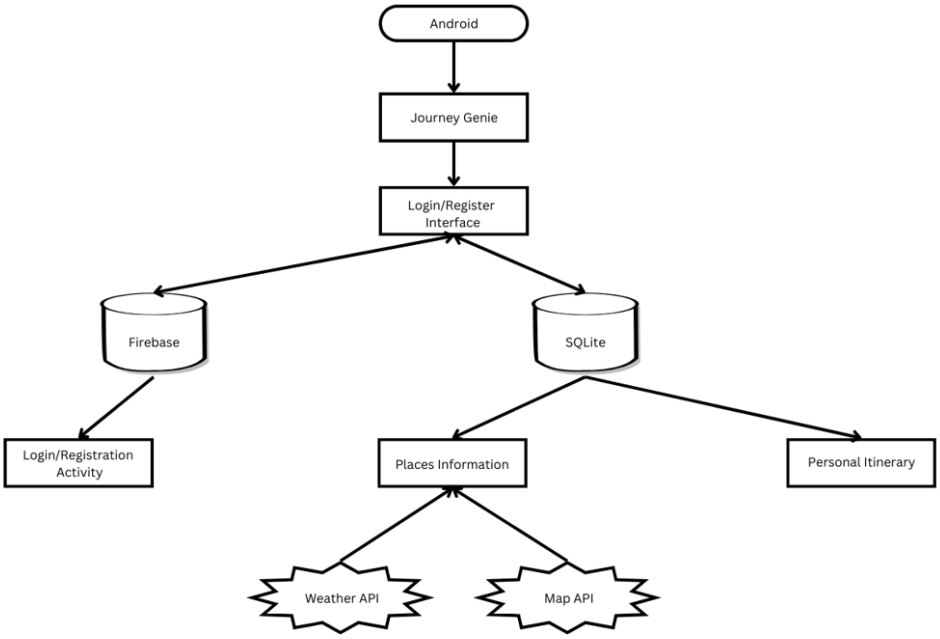
Designing the architecture and flow of Journey Genie involves crafting wireframes and mock-ups to visualize the user interface and experience. It also encompasses defining the app's architecture, including the data model, navigation structure, and interaction flow. During this phase, decisions regarding the application of Material Design principles, UI elements, and animations are made to ensure a coherent and intuitive user experience. Moreover, considerations for scalability and flexibility are factored in to seamlessly accommodate future updates and features.

3. Development of Journey Genie:

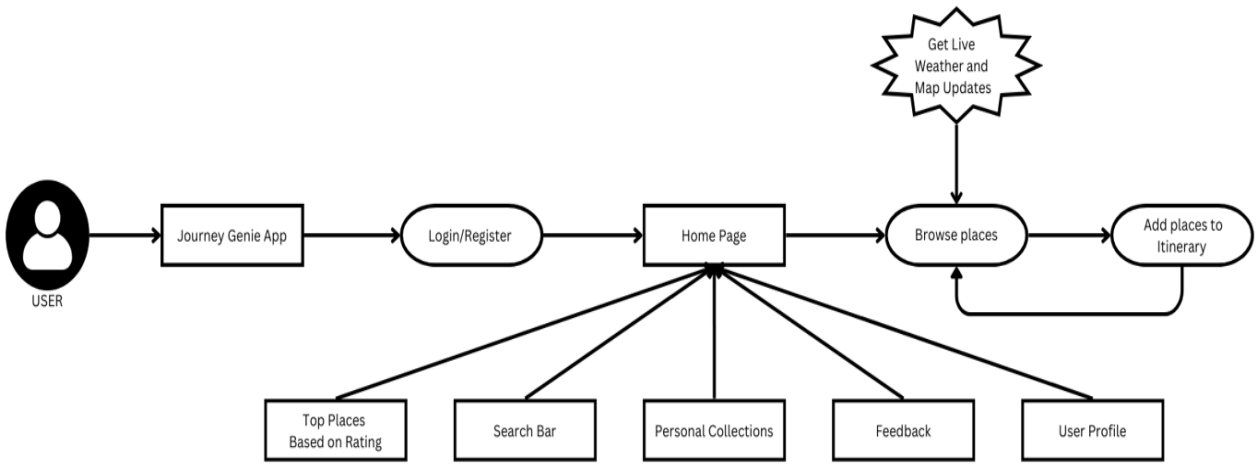
The development phase entails implementing the design by coding and integrating various modules of Journey Genie. This includes setting up the development environment with appropriate tools and frameworks, implementing user authentication systems, and developing UI components and layouts. Integration with external APIs for retrieving real-time data on weather and maps is crucial. Throughout the development process, utilizing appropriate technologies and following best practices ensures the creation of clean, efficient, and maintainable code.

4. Testing Journey Genie:

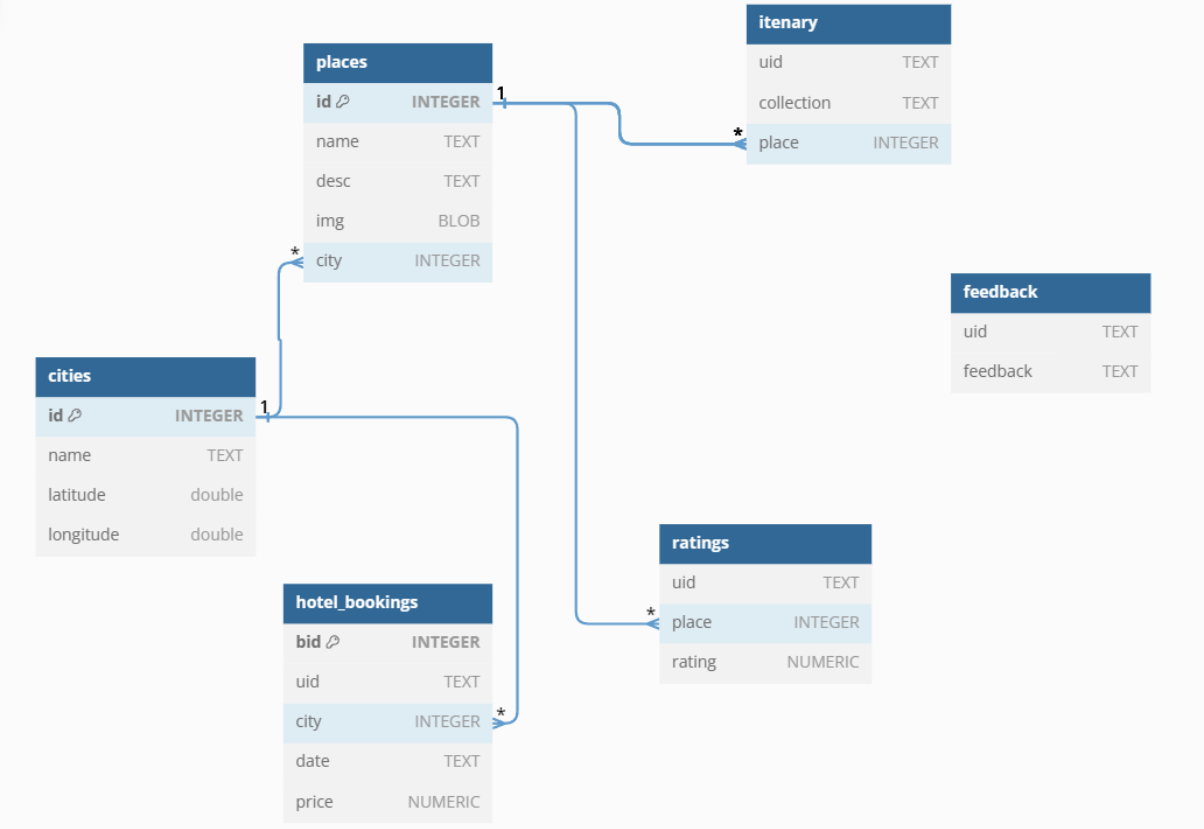
Thoroughly testing Journey Genie for functionality, performance, and usability is imperative to ensure a high-quality user experience. This involves conducting unit tests for individual components and functions, as well as integration testing to verify seamless communication between different modules and APIs. Additionally, performance optimizations and bug fixes are yet to be identified and addressed promptly, iteratively refining the development process to deliver a polished and reliable travel management app.



Application Architecture (Low Level Design)



Workflow of the application (High Level Design)



Schema Diagram of the database

6. Work done So Far

- Frontend implementation
- Backend design
- Weather API implementation

Below are the screenshots of the outputs for the same:

Let's Enjoy the Beautiful World

Explore new horizons and broaden your own

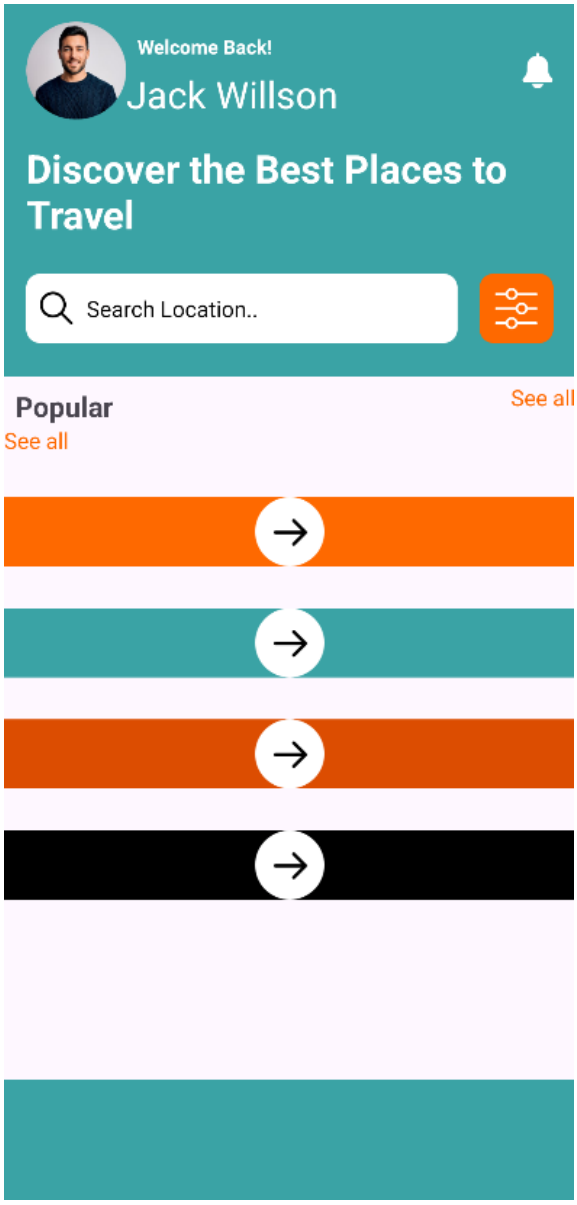
Get Started →

TravelGenie: Log In

Enter your username:

Enter your password:

Login


Welcome Back!
Jack Willson
Discover the Best Places to Travel
Search Location..
Popular
See all
See all


Snigdha
snigdha@MADLab.com
Snigdha Jha
+91 123 456 7890
MIT Manipal, Udupi, India

Feedback Form

Please Enter Your Feedback

Send Feedback

Places to Travel

Pick From these Cities for your Next Journey!

1. *Mumbai*

2. *Manipal*

Booking Details

Booking ID: ABC123

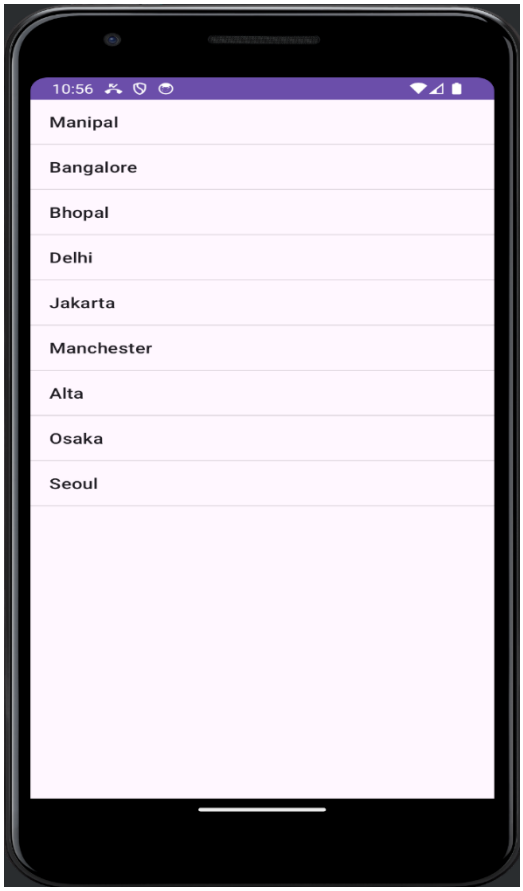
Destination: Paris

Date: 12th April, 2024

Time: 10:00 AM

Price: $500

Edit Booking


10:56
Manipal
Bangalore
Bhopal
Delhi
Jakarta
Manchester
Alta
Osaka
Seoul


10:59
Manipal
Clouds
Date/Day Temperature Humidity
09/04/24 Max: 33.3°C Avg. Humidity: 61.1%
Tuesday Min: 23.5°C
10/04/24 Max: 32.8°C Avg. Humidity: 64.8%
Wednesday Min: 23.5°C
11/04/24 Max: 32.7°C Avg. Humidity: 64.1%
Thursday Min: 24.2°C
12/04/24 Max: 32.6°C Avg. Humidity: 66.1%
Friday Min: 23.8°C
13/04/24 Max: 32.6°C Avg. Humidity: 63.9%
Saturday Min: 24.1°C


11:00
Osaka
Rain
Date/Day Temperature Humidity
09/04/24 Max: 14.8°C Avg. Humidity: 57.9%
Tuesday Min: 6.7°C
10/04/24 Max: 17.8°C Avg. Humidity: 52.3%
Wednesday Min: 10.4°C
11/04/24 Max: 20.1°C Avg. Humidity: 61.8%
Thursday Min: 13.2°C
12/04/24 Max: 21.7°C Avg. Humidity: 59.3%
Friday Min: 13.8°C
13/04/24 Max: 22.1°C Avg. Humidity: 49.8%
Saturday Min: 12.5°C


10:56
Seoul
Clear
Date/Day Temperature Humidity
09/04/24 Max: 18.7°C Avg. Humidity: 42.9%
Tuesday Min: 11.3°C
10/04/24 Max: 17.1°C Avg. Humidity: 63.3%
Wednesday Min: 12.4°C
11/04/24 Max: 18.1°C Avg. Humidity: 70.9%
Thursday Min: 10.8°C
12/04/24 Max: 21.2°C Avg. Humidity: 43.6%
Friday Min: 13.2°C
13/04/24 Max: 23.5°C Avg. Humidity: 32.3%
Saturday Min: 15.2°C

7 Remaining Work

- Integration of Frontend, Backend and API
- Implementation of the Map API

8 Scope

Journey Genie's scope spans across offering tools for planning trips, such as creating itineraries and booking accommodations, along with other features for users such as secure login with authentication, feedback-process enabled, and so on.

Additionally, it includes integrating with backend services to provide real-time updates on weather forecasts and travel advisories, enabling travellers to make well-informed decisions during their journeys.

9 References

[1]B. P. D. Putranto, R. Saptoto, O. C. Jakaria and W. Andriyani, "A Comparative Study of Java and Kotlin for Android Mobile Application Development," 2020 3rd International Seminar on Research of Information Technology and Intelligent Systems (ISRITI), Yogyakarta, Indonesia, 2020, pp. 383-388, doi: 10.1109/ISRITI51436.2020.9315483. keywords: {Java;Computer languages;Performance evaluation;Motion pictures;Tools;Testing;Application programming interfaces;Programming Language;Java;Kotlin;Android;Comparison},

[2]V. Oliveira, L. Teixeira and F. Ebert, "On the Adoption of Kotlin on Android Development: A Triangulation Study," 2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER), London, ON, Canada, 2020, pp. 206-216, doi: 10.1109/SANER48275.2020.9054859. keywords: {Productivity;Industries;Java;Codes;Virtual machining;Software;Internet;Android;Kotlin;Java},

[3]How to build a simple Android Weather App using Kotlin and OpenWeatherMap API" , 2023, Medium.com.

[4]Needone APP "How to place a location on Google Map using Kotlin Medium", 2017, Medium.com

[5]Abdullah Talha. (2019). "A Performance Comparison of SQLite and Firebase Databases from A Practical Perspective. 7. 314-325. 10.29130/dubited.441672."

[6](2016) "Using Firebase to create a “simple” Chat application in Android.", Medium.com

[7](2023) "How to Implement OTP View in Android?", GeeksforGeeks

[8](2021)"Notifications in Android with Example", GeeksforGeeks