

RECIPE GENERATOR WITH GEMINI AI API

CS19611 - MOBILE APPLICATION DEVELOPMENT LABORATORY

PROJECT REPORT

Submitted by

VARUN KUMAR V

(2116210701311)

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING



RAJALAKSHMI ENGINEERING COLLEGE

ANNA UNIVERSITY, CHENNAI

MAY 2025

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

BONAFIDE CERTIFICATE

Certified that this Project titled “**RECIPE GENERATOR WITH GEMINI AI API**” is the bonafide work of “**VARUN KUMAR V (211622070131 1)**” who carried out the work under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr. P. Kumar., M.E., Ph.D.,

HEAD OF THE DEPARTMENT

Professor

Department of Computer Science
and Engineering,

Rajalakshmi Engineering College,
Chennai - 602 105.

SIGNATURE

Dr. N. Duraimurugan., M.E., Ph.D.,

SUPERVISOR

Associate Professor

Department of Computer Science
and Engineering,

Rajalakshmi Engineering
College, Chennai-602 105.

Submitted to Mini Project Viva-Voce Examination held on _____

Internal Examiner

External Examiner

ACKNOWLEDGMENT

Initially we thank the Almighty for being with us through every walk of our life and showering his blessings through the endeavor to put forth this report. Our sincere thanks to our Chairman **Mr. S. MEGANATHAN, B.E, F.I.E.**, our Vice Chairman **Mr. ABHAY SHANKAR MEGANATHAN, B.E., M.S.**, and our respected Chairperson **Dr. (Mrs.) THANGAM MEGANATHAN, Ph.D.**, for providing us with the requisite infrastructure and sincere endeavoring in educating us in their premier institution.

Our sincere thanks to **Dr. S.N. MURUGESAN, M.E., Ph.D.**, our beloved Principal for his kind support and facilities provided to complete our work in time. We express our sincere thanks to **Dr. P. KUMAR, M.E., Ph.D.**, Professor and Head of the Department of Computer Science and Engineering for his guidance and encouragement throughout the project work. We convey our sincere and deepest gratitude to our internal guide **Dr. N. DURAIMURUGAN**, We are very glad to thank our Project Coordinator, **Dr. N. DURAIMURUGAN** Associate Professor Department of Computer Science and Engineering for his useful tips during our review to build our project.

VARUN KUMAR V

2116220701311

ABSTRACT

The Recipe Generator with Gemini AI API is an Android application developed in Kotlin, designed to simplify meal preparation by generating personalized recipes from user-specified ingredients. Utilizing the Gemini AI API, the app processes ingredient lists to produce detailed recipes, eliminating the need for manual recipe searches. The application features a modern user interface built with Jetpack Compose, a local Room database for offline storage, and Retrofit for seamless API integration. The app's modular architecture supports scalability, enabling future enhancements such as dietary preferences or recipe sharing. Key features include real-time recipe generation, offline recipe access, and an intuitive mobile interface. This report details the app's development, architecture, and performance, highlighting its potential as a practical tool for home cooks seeking efficient and creative cooking solutions.

TABLE OF CONTENTS

1. INTRODUCTION

1.1. INTRODUCTION

1.2. OBJECTIVES

1.3. MODULES

2. SURVEY OF TECHNOLOGIES

2.1. SOFTWARE DESCRIPTION

2.2. LANGUAGES

2.2.1. KOTLIN

2.2.2. XML

2.2.3. SQLITE

3. REQUIREMENTS AND ANALYSIS

3.1. REQUIREMENT SPECIFICATION

3.2. HARDWARE AND SOFTWARE REQUIREMENTS

3.3. ARCHITECTURE DIAGRAM

3.4. ER DIAGRAM

3.5. NORMALIZATION

4. PROGRAM CODE

5. OUTPUT

6. RESULTS AND DISCUSSION

7. CONCLUSION

8. REFERENCES

CHAPTER 1

1. INTRODUCTION

Meal preparation at home often requires creativity, especially when limited to available ingredients. Traditional recipe searches involve browsing websites or cookbooks, which can be inefficient and may not align with specific ingredient constraints. The Recipe Generator with Gemini AI API is an Android application developed in Kotlin to address these challenges. By integrating the Gemini AI API, the app processes user-inputted ingredient lists to generate tailored recipes, including titles, ingredient details, and step-by-step instructions, directly on the user's mobile device. Built with Jetpack Compose for a modern user interface, Room for local data storage, and Retrofit for API communication, the app offers a seamless and responsive experience. It aims to enhance the cooking process by providing a mobile, AI-driven solution that simplifies meal planning and encourages culinary exploration. This chapter introduces the project's motivation, objectives, and modular structure, providing a foundation for understanding its development and implementation.

2. Objectives

- The Recipe Generator project was designed with the following objectives:
 - Personalized Recipe Generation: Leverage the Gemini AI API to generate accurate and creative recipes based on user-provided ingredients, ensuring relevance and variety.
 - Offline Access: Implement a Room database to store ingredients and recipes locally, enabling functionality without internet connectivity.
 - Modern User Interface: Develop an intuitive UI using Jetpack Compose to facilitate easy ingredient input and recipe display.
 - Scalability: Design a modular architecture to support future enhancements, such as dietary filters or integration with other APIs.
 - Performance: Optimize API calls and database operations to ensure a responsive experience on Android devices

3. Modules

The application is organized into the following modules:

- **UI Module:** Built with Jetpack Compose, this module provides a responsive interface for users to input ingredients and view recipes.
- **AI Integration Module:** Uses Retrofit to communicate with the Gemini AI API, processing ingredient lists and retrieving recipes.
- **Database Module:** Manages Room database operations for storing and retrieving ingredients and recipes offline.
- **Logic Module:** Coordinates data flow between the UI, API, and database, ensuring seamless operation and error handling.

CHAPTER 2

1. Software Description

The Recipe Generator is an Android application developed using Kotlin, the preferred language for Android development. The user interface is built with Jetpack Compose, Android's modern toolkit for creating native UIs. Data persistence is achieved through Room, a robust persistence library for local storage. The Gemini AI API powers recipe generation, accessed via Retrofit for HTTP requests. The app is developed using Android Studio, the official IDE for Android, configured with Kotlin and Android SDK tools. The application follows the MVVM (Model-View-ViewModel) architecture, ensuring separation of concerns and maintainability. It balances online API interactions with offline storage, making it suitable for varied network conditions. This combination of technologies delivers a high-performance, user-friendly mobile experience.

2. Software Description

2.1 Kotlin

Kotlin is a modern, statically typed programming language fully interoperable with Java, widely adopted for Android development. In this project, Kotlin implements the app's logic, UI, API integration, and database operations. Its concise syntax and null-safety features enhance code reliability and developer productivity.

2.2 XML

Jetpack Compose is Android's declarative UI framework, used to build the app's interface. It enables the creation of dynamic, responsive layouts with minimal code, supporting features like text inputs and scrollable recipe displays. Compose's integration with Kotlin ensures a seamless development experience.

2.3 SQLite

Room is a persistence library part of Android Jetpack, providing an abstraction layer over SQLite for robust database management. In this project, Room stores ingredient lists and recipes, enabling fast and offline access to data.

2.4 Retrofit

Retrofit is a type-safe HTTP client for Android, used to communicate with the Gemini AI API. It simplifies API calls by converting JSON responses into Kotlin data classes, ensuring efficient and reliable data retrieval.

3. Requirements And Analysis

3.1 Requirements Specification

Project Overview:

The Recipe Generator is an Android app that allows users to input a list of ingredients, which are processed by the Gemini AI API to generate a customized recipe. The app features a Jetpack Compose UI, a Room database for offline storage, and Retrofit for API integration. It is designed to be intuitive, performant, and accessible on Android devices running API 21 or higher.

Functional Requirements:

- **Ingredient Input:** Users can enter a comma-separated list of ingredients via a text field.
- **Recipe Generation:** The app sends the ingredient list to the Gemini AI API and displays the generated recipe, including title, ingredients, and instructions.
- **Data Storage:** Ingredients and recipes are stored in a Room database for offline access.
- **Recipe Retrieval:** Users can view previously generated recipes from the database.

Non-functional Requirements:

- Usability: The UI must be intuitive, with responsive controls and clear feedback.
- Performance: API calls and database queries should complete within 2 seconds.
- Compatibility: The app must support Android devices running API 21 (Lollipop) or higher.
- Efficiency: Optimize memory and battery usage for low-end devices with at least 2 GB RAM

Analysis:

- **User Workflow:** Tasks analysis for doctors and patients.
- **Data Model:** Efficient database schema design.

3.2 Hardware and Software Requirements

Hardware Requirements:

Minimum: Android device with 2 GB RAM, 100 MB storage, 1 GHz processor.

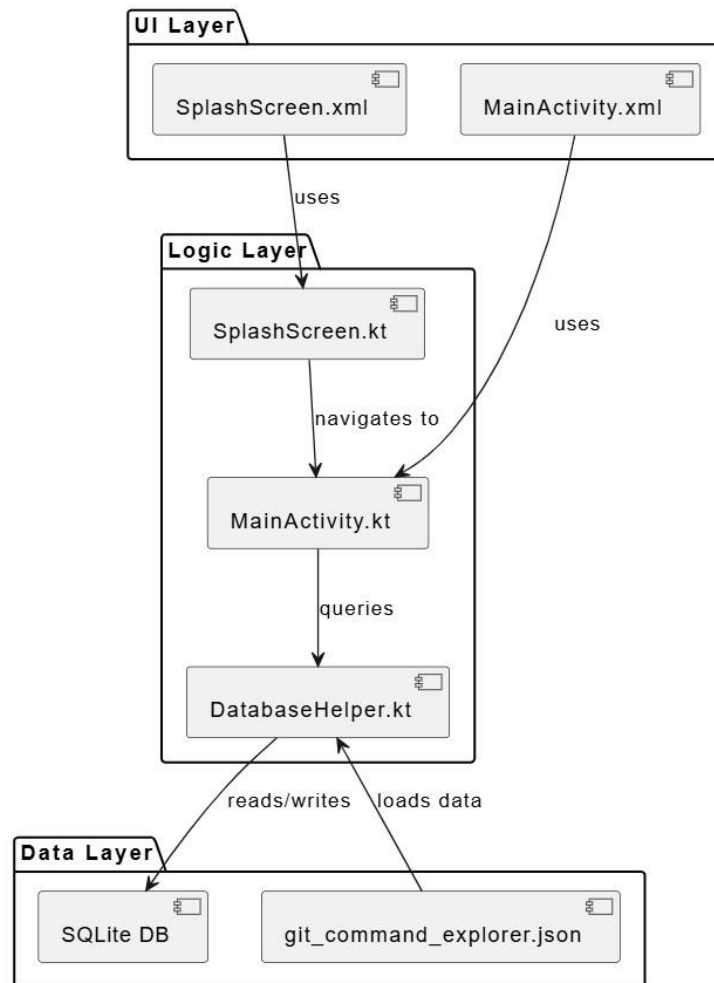
- Recommended: Development machine with 8 GB RAM, 500 MB storage, 2 GHz processor.

Software Requirements:

Android Studio 4.0 or higher. 8

- Kotlin 1.5 or higher.
- Android SDK API 21 or higher.
- Gemini AI API key and SDK.
- Gradle build system.

3.3 Architecture Diagram



UI Layer:

- SplashScreen.xml
- MainActivity.xml

Logic Layer:

- SplashScreen.kt
- MainActivity.kt
- DatabaseHelper.kt

Data Layer:

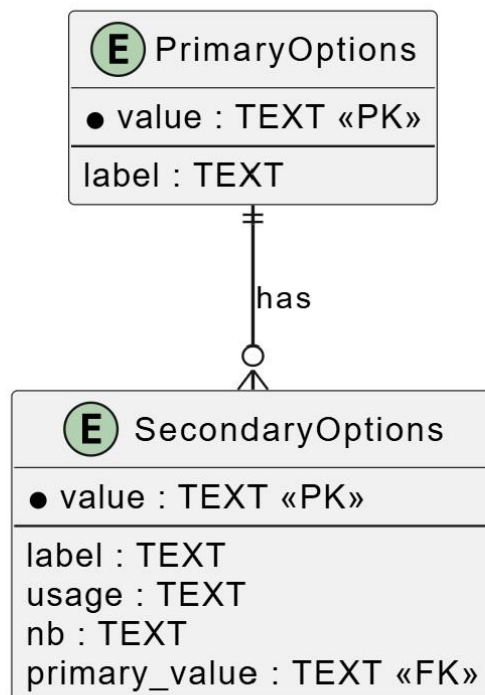
- SQLite DB (tables: primary_options, secondary_options)
- JSON file for data population.

3.4. ER Diagram

Entities:

- PrimaryOptions:
 - value (PK)
 - label
- SecondaryOptions:
 - value (PK)
 - label
 - usage
 - nb (note)
 - primary_value (FK to PrimaryOptions)

This is a one-to-many relationship: each primary option links to multiple secondary options.



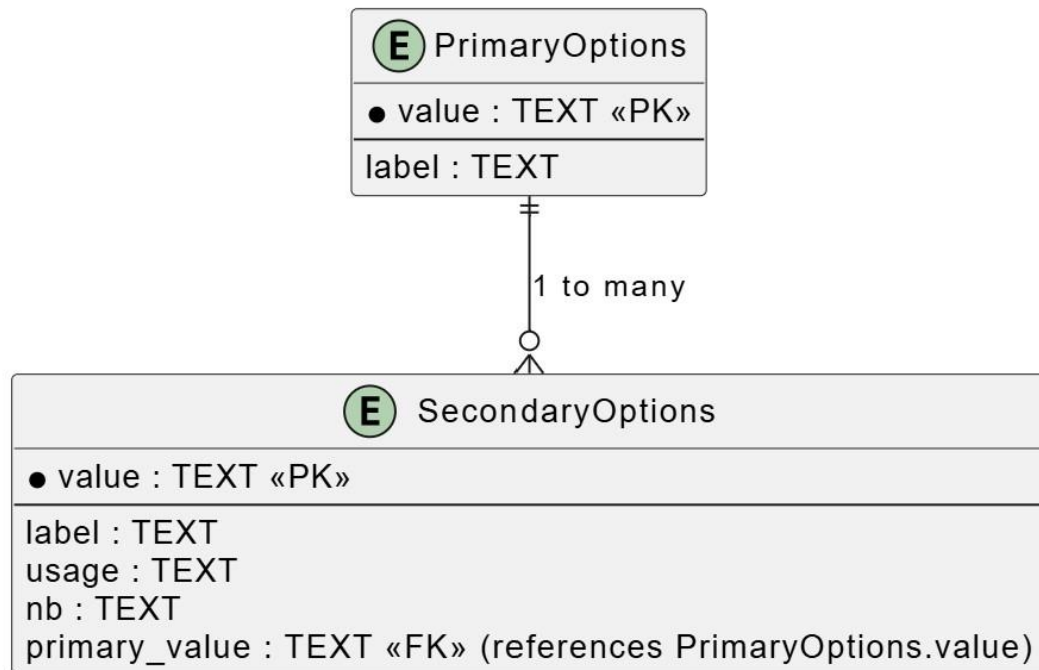
3.5. Normalization

The database is normalized as follows:

- First Normal Form (1NF):
 - All attributes hold atomic values; no repeating groups.
- Second Normal Form (2NF):
 - All non-key columns are fully dependent on the primary key.
- Third Normal Form (3NF):

- There are no transitive dependencies; data is well-structured with proper foreign key use.

This ensures minimal redundancy and efficient query performance.



4. PROGRAM CODE AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>

<manifest xmlns:android="http://schemas.android.com/apk/res/android"

    xmlns:tools="http://schemas.android.com/tools">

    <application

        android:allowBackup="true"

        android:dataExtractionRules="@xml/data_extraction_rules"

        android:fullBackupContent="@xml/backup_rules"

        android:icon="@mipmap/ic_launcher"

        android:label="@string/app_name"

        android:roundIcon="@mipmap/ic_launcher_round"

        android:supportsRtl="true"

        android:theme="@style/Theme.GitHelp"

        tools:targetApi="31">

        <activity

            android:name=".SplashScreen"

            android:exported="true">

            <intent-filter>

                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />

            </intent-filter>

        </activity>

        <activity

            android:name=".MainActivity"

            android:exported="true"

            android:windowSoftInputMode="stateHidden" />
```

```
</application>
```

```
</manifest>
```

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>

<layout xmlns:android="http://schemas.android.com/apk/res/android"

    xmlns:app="http://schemas.android.com/apk/res-auto"

    xmlns:tools="http://schemas.android.com/tools">

    <ScrollView

        android:layout_width="match_parent"

        android:layout_height="match_parent">

        <androidx.constraintlayout.widget.ConstraintLayout

            android:layout_width="match_parent"

            android:layout_height="wrap_content"
            android:padding="16dp"

            tools:context=".MainActivity">

            <TextView

                android:id="@+id/text_git_command"

                android:layout_width="wrap_content"

                android:layout_height="wrap_content"

                android:layout_marginTop="44dp"

                android:fontFamily="@font/calibri"

                android:textSize="24sp"

                app:layout_constraintEnd_toEndOf="parent"

                app:layout_constraintStart_toStartOf="parent"

                app:layout_constraintTop_toTopOf="parent"
```



```

tools:text="Git Command Explorer" />

<TextView

    android:id="@+id/text_find_right_command"

    android:layout_width="wrap_content"

    android:layout_height="wrap_content"

    android:layout_marginTop="16dp"

    android:fontFamily="@font/calibri"
    android:gravity="center"
    android:text="Find the right commands you need without
digging
through the web"

    android:textColor="#6A6A6A"

    android:textSize="16sp"

    app:layout_constraintEnd_toEndOf="parent"

    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/text_git_command"
/>

<TextView

    android:id="@+id/text_i_want_to"

    android:layout_width="wrap_content"

    android:layout_height="wrap_content"

    android:layout_marginTop="44dp"

    android:fontFamily="@font/calibri"

    android:text="I want to:"

    android:textColor="@color/colorAccent"

    android:textSize="18sp"

    android:textStyle="bold"

    app:layout_constraintStart_toStartOf="parent"

    app:layout_constraintTop_toBottomOf="@+id/text_find_right_command" />

```

```
<androidx.cardview.widget.CardView

    android:id="@+id/card_view_first_field"

    android:layout_width="0dp"

    android:layout_height="wrap_content"

    android:layout_marginTop="16dp"

    app:cardCornerRadius="4dp"

    app:cardUseCompatPadding="true"

    app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"

    app:layout_constraintTop_toBottomOf="@+id/text_i_want_to">

<AutoCompleteTextView

    android:id="@+id/input_first_field"

    android:layout_width="match_parent"

    android:layout_height="wrap_content"

    android:background="@null"

    android:completionThreshold="1"

    android:drawableEnd="@drawable/ic_keyboard_arrow_down"

    android:drawableTint="#ACACAC"

    android:fontFamily="@font/calibri"

    android:hint="..."

    android:inputType="textNoSuggestions"

    android:padding="12dp"

    android:textColor="@color/colorPrimary"

    android:textSize="14sp"

    android:textStyle="bold"

    tools:text="add" />
```

```
</androidx.cardview.widget.CardView>
```

```
<androidx.cardview.widget.CardView

    android:id="@+id/card_view_second_field"

    android:layout_width="0dp"

    android:layout_height="wrap_content"

    android:layout_marginTop="4dp"

    android:visibility="gone"

    app:cardCornerRadius="4dp"

    app:cardUseCompatPadding="true"
app:layout_constraintEnd_toEndOf="parent"

    app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toBottomOf="@+id/card_view_first_field">
```

```
<AutoCompleteTextView

    android:id="@+id/input_second_field"

    android:layout_width="match_parent"

    android:layout_height="wrap_content"

    android:background="@null"

    android:completionThreshold="1"

    android:drawableEnd="@drawable/ic_keyboard_arrow_down"

    android:drawableTint="#ACACAC"

    android:fontFamily="@font/calibri"

    android:hint="..."

    android:inputType="textNoSuggestions"

    android:padding="12dp"

    android:textColor="@color/colorPrimary"
```

```
        android:textSize="14sp"

        android:textStyle="bold"

        tools:text="new changes" />

</androidx.cardview.widget.CardView>
```

```
<TextView

    android:id="@+id/text_usage"

    android:layout_width="wrap_content"

    android:layout_height="wrap_content"

    android:layout_marginTop="24dp"

    android:fontFamily="@font/calibri"

    android:text="Usage"
    android:textColor="@color/colorPrimary"

    android:textSize="18sp"

    android:textStyle="bold"

    app:layout_constraintStart_toStartOf="parent"

    app:layout_constraintTop_toBottomOf="@+id/card_view_second_field" />

<androidx.cardview.widget.CardView

    android:id="@+id/card_view_usage"

    android:layout_width="0dp"

    android:layout_height="wrap_content"

    android:layout_marginTop="8dp"

    app:cardBackgroundColor="#20262C"

    app:cardCornerRadius="8dp"

    app:cardUseCompatPadding="true"

    app:layout_constraintEnd_toEndOf="parent"
```

```
app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toBottomOf="@+id/text_usage">
```

```
<LinearLayout

    android:layout_width="match_parent"

    android:layout_height="wrap_content"

    android:orientation="horizontal">
```

```
<View

    android:layout_width="0dp"

    android:layout_height="match_parent"

    android:layout_weight="2"

    android:background="@color/colorAccent" />
```

```
    <TextView
android:id="@+id/text_display_git_command"

        android:layout_width="0dp"

        android:layout_height="wrap_content"

        android:layout_weight="98"

        android:fontFamily="@font/calibri"

        android:lineSpacingExtra="8dp"

        android:padding="16dp"

        android:textColor="@color/white"

        android:textSize="14sp"

        android:textStyle="bold"

        tools:text="git add file.ext" />
```

```
</LinearLayout>
```

```
</androidx.cardview.widget.CardView>
```

```
<TextView
```

```
    android:id="@+id/text_note"
```

```
    android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
```

```
    android:layout_marginTop="16dp"
```

```
    android:fontFamily="@font/calibri"
```

```
    android:text="Note"
```

```
    android:textColor="@color/colorPrimary"
```

```
    android:textSize="18sp"
```

```
    android:textStyle="bold"
```

```
    android:visibility="gone"
```

```
        app:layout_constraintStart_toStartOf="parent"  
app:layout_constraintTop_toBottomOf="@+id/card_view_usage" />
```

```
<androidx.cardview.widget.CardView
```

```
    android:id="@+id/card_view_note"
```

```
    android:layout_width="0dp"
```

```
    android:layout_height="wrap_content"
```

```
    android:layout_marginTop="8dp"
```

```
    android:visibility="gone"
```

```
    app:cardBackgroundColor="#20262C"
```

```
    app:cardCornerRadius="8dp"
```

```
    app:cardUseCompatPadding="true"
```

```

        app:layout_constraintEnd_toEndOf="parent"

        app:layout_constraintStart_toStartOf="parent"

        app:layout_constraintTop_toBottomOf="@+id/text_note">

<LinearLayout

    android:layout_width="match_parent"

    android:layout_height="wrap_content"

    android:orientation="horizontal">

    <View

        android:layout_width="0dp"
        android:layout_height="match_parent"

        android:layout_weight="2"

        android:background="@color/colorAccent" />

    <TextView

        android:id="@+id/text_display_note"

        android:layout_width="0dp"

        android:layout_height="wrap_content"
android:layout_weight="98"

        android:fontFamily="@font/calibri"

        android:lineSpacingExtra="8dp"

        android:padding="16dp"

        android:textColor="@color/white"

        android:textSize="14sp"

        android:textStyle="bold"

        tools:text="To add all the files in the current
directory, use 'git add' . To add a directory use 'git add directory'" />

</LinearLayout>

```

```

        </androidx.cardview.widget.CardView>

    </androidx.constraintlayout.widget.ConstraintLayout>

</ScrollView>

</layout>

```

activity_splash_screen.xml

```

<?xml version="1.0" encoding="utf-8"?>

<layout xmlns:android="http://schemas.android.com/apk/res/android"

    xmlns:app="http://schemas.android.com/apk/res-auto"

    xmlns:tools="http://schemas.android.com/tools">

    <androidx.constraintlayout.widget.ConstraintLayout

        android:layout_width="match_parent"
        android:layout_height="match_parent"

        android:padding="16dp"

        tools:context=".SplashScreen">

        <TextView

            android:id="@+id/text_git_command"

            android:layout_width="wrap_content"

            android:layout_height="wrap_content"

            android:fontFamily="@font/calibri"

            android:textSize="24sp"

            app:layout_constraintBottom_toBottomOf="parent"

```



```

        app:layout_constraintEnd_toEndOf="parent"

        app:layout_constraintStart_toStartOf="parent"

        app:layout_constraintTop_toTopOf="parent"

        tools:text="Git Command Explorer" />

<TextView

        android:id="@+id/text_find_right_command"

        android:layout_width="wrap_content"

        android:layout_height="wrap_content"

        android:layout_marginTop="16dp"

        android:fontFamily="@font/calibri"
        android:gravity="center"

        android:text="Find the right commands you need without digging
through the web"

        android:textColor="#6A6A6A"

        android:textSize="16sp"

        app:layout_constraintEnd_toEndOf="parent"

        app:layout_constraintStart_toStartOf="parent"

        app:layout_constraintTop_toBottomOf="@+id/text_git_command" />

</androidx.constraintlayout.widget.ConstraintLayout>

</layout>

```

git_command_explorer.json

```

{
  "primary_options": [
    {
      "value": "add",
      "label": "add"
    },
    {

```

```
    "value": "commit",  
    "label": "commit"  
  },  
  
  {  
    "value": "revert",  
    "label": "revert/reset"  
  },  
  
  {  
    "value": "initialize",  
    "label": "initialize"  
  },  
  
  {
```

```
    "value": "modify",  
    "label": "modify"  
  },  
  
  {  
    "value": "show",  
    "label": "show/view"  
  },  
  
  {  
    "value": "delete",  
    "label": "delete/remove"  
  },  
  
  {  
    "value": "compareCommits",  
    "label": "compare two commits"  
  },  
  
  {  
    "value": "configure",
```

```
    "label": "configure"

  },

  {

    "value": "clone",

    "label": "clone"

  },

  {

    "value": "ignore",

    "label": "ignore"

  },

  {

    "value": "rename",

    "label": "rename"
```

```
  },

  {

    "value": "merge",

    "label": "merge"

  },

  {

    "value": "squash",

    "label": "squash"

  },

  {

    "value": "stash",

    "label": "stash"

  },

  {

    "value": "debug",
```

```
    "label": "debug"

  },

  {

    "value": "recover",

    "label": "recover"

  },

  {

    "value": "synchronize",

    "label": "synchronize"

  },

  {

    "value": "rebase",

    "label": "rebase"

  }

],
```

```
"secondary_options": {

  "commit": [

    {

      "value": "local-changes",

      "label": "commit all local changes in tracked files",

      "usage": "git commit -a"

    },

    {

      "value": "staged-changes",

      "label": "commit all staged changes",

      "usage": "git commit -m <message>",

      "nb": "Replace <message> with your commit message."

    }

  ]

}
```

```

],

"configure": [

  {

    "value": "email-name",

    "label": "name and email address",

    "usage": "git config --global user.name \"username\" \n\ngit config --global user.email \"email address\"",

    "nb": "Your username and email address should be the same as the one used with your git hosting provider i.e. github, bitbucket, gitlab etc"

  },

  {

    "value": "editor",

    "label": "default editor",

    "usage": "git config --global core.editor \"vim\"",

    "nb": "Change default editor to vim."

  },

  {

    "value": "diff-tool",

```

```

    "label": "external diff tool",

    "usage": "git config --global diff.external \"meld\"",

    "nb": "Set external diff tool to meld."

  },

  {

    "value": "merge-tool",

    "label": "default merge tool",

    "usage": "git config --global merge.tool \"meld\"",

    "nb": "Set default merge tool to meld."

  },

  {

    "value": "color",

```

```

        "label": "color",

        "usage": "git config --global color.ui auto",

        "nb": "Enables helpful colorization of command line output"
    },

    {

        "value": "signingkey",

        "label": "add the GPG key",

        "usage": "git config --global user.signingkey <your-secret-gpg-key>",

        "nb": "Git is cryptographically secure, but it's not foolproof. If
you're taking work from others on the internet and want to verify that
commits are actually from a trusted source, Git has a few ways to sign and
verify work using GPG."
    }

],

"revert": [

    {

        "value": "specific-commit",

        "label": "a specific commit",

        "usage": "git revert <commit-hash>",

        "nb": "Use git log to see the hash of each commit"
    }

```

```

    },

    {

        "value": "to-last-commit",

        "label": "to last commit",

        "usage": "git reset --hard"
    },

    {

        "value": "to-last-commit-from-remote",

        "label": "to last commit on remote branch",

        "usage": "git reset --hard <repo>/<branch>"
    }

```

```

    }

],

"initialize": [

    {

        "value": "new-repo",

        "label": "a new repository",

        "nb": "Make sure you are in the right directory",

        "usage": "git init"

    }

],

"modify": [

    {

        "value": "commit-message",

        "label": "my last/latest commit message",

        "usage": "git commit --amend"

    },

    {

        "value": "commit",

        "label": "my last commit but leave the commit message as is",

        "usage": "git add . \ngit commit --amend --no-edit"

```

```

    },

    {

        "value": "remoteUrl",

        "label": "repo's remote url",

        "usage": "git remote set-url <alias> <url>",

        "nb": "<alias> is your remote name e.g origin"

    }

]

```

```
],  
  
  "show": [  
  
    {  
  
      "value": "repo-status",  
  
      "label": "status of project including staged, unstaged and untracked  
files",  
  
      "usage": "git status"  
  
    },  
  
    {  
  
      "value": "logs",  
  
      "label": "commit logs/history"  
  
    },  
  
    {  
  
      "value": "uncommittedChanges",  
  
      "label": "uncommitted changes",  
  
      "usage": "git diff"  
  
    },  
  
    {  
  
      "value": "committedChanges",  
  
      "label": "committed/staged changes",  
  
      "usage": "git diff --staged"  
  
    },  
  
    {  
  
      "value": "remoteUrl",
```

```
      "label": "repo's remote url",  
  
      "usage": "git remote -v"  
  
    },  
  
    {  
  
      "value": "stash",
```



```
    "label": "stash",

    "usage": "git stash list"

  },

  {

    "value": "branch",

    "label": "branches",

    "usage": "git branch",

    "nb": "The active branch is prefixed with *"

  },

  {

    "value": "tags",

    "label": "tags",

    "usage": "git tag"

  }

],

"delete": [

  {

    "value": "branch",

    "label": "a branch",

    "usage": "git branch -D <branch name>"

  },

  {

    "value": "delete-multiple-branches",

    "label": "multiple branches"

  }

],
```

```
{

  "value": "tag",

  "label": "a tag",
```

```

    "usage": "git tag -d v<tag version>"
  },
  {
    "value": "remote",
    "label": "remote",
    "usage": "git remote rm <remote>"
  },
  {
    "value": "untracked-files",
    "label": "untracked files",
    "usage": "git clean -<flag>",
    "nb": "replace -<flag> with:\n -i for interactive command\n -n to\npreview what will be removed\n -f to remove forcefully\n -d to remove\n directories\n -X to remove ignored files\n -x to remove ignored and\n non-ignored files"
  },
  {
    "value": "files-from-index",
    "label": "files from index",
    "usage": "git rm --cached <file or dir>",
    "nb": "Use this option to unstage and remove paths only from the\nindex. Working tree files, whether modified or not, will be left alone."
  },
  {
    "value": "local-branches-not-on-remote",
    "label": "local branches that don't exist at remote",
    "usage": "git remote prune <remote-name>",
    "nb": "Use the --dry-run option to report what branches will be\npruned, but do not actually prune them"
  },
  {
    "value": "files-from-old-commit",

```

```

        "label": "files from old commits",

        "usage": "git filter-branch --index-filter \n'git rm --cached
--ignore-unmatch path/to/mylarge_file' \n--tag-name-filter cat --
--all\n\nfilter-branch keeps backups too, so the size of the repo won't
decrease immediately unless you expire the reflogs and garbage collect:\n\nrm
-Rf .git/refs/original      # careful\nngit gc --aggressive --prune=now #
danger",

        "nb": "Like the rebasing option described before, filter-branch is
rewriting operation. If you have published history, you'll have to --force
push the new refs."

    }

],

"compareCommits": [

    {

        "value": "terminal",

        "label": "and output result in the terminal",

        "usage": "git diff <sha1> <sha2> | less",

        "nb": "sha1 and sha2 are the sha hash of the commits you want to
compare."

    },

    {

        "value": "file",

        "label": "and output result to a file",

        "usage": "git diff <sha1> <sha2> > diff.txt",

        "nb": "sha1 and sha2 are the sha of the commits you want to compare.
\n\ndiff.txt is the file you want to store the contents of the diff"

    }

],

"clone": [

    {

        "value": "clone-repo-into-a-new-dir",

        "label": "existing repo into a new directory",

        "usage": "git clone <repo-url> <directory>",

```

```
"nb": "The repo is cloned into the specified directory\nReplace  
\"directory\" with the directory you want"
```

```
,  
  
{  
  
    "value": "clone-repo-into-a-current-dir",  
  
    "label": "existing repo into the current directory",  
  
    "usage": "git clone <repo-url> .",  
  
    "nb": "The repo is cloned into the current directory\nThe current  
directory is represented with a \".\" (period)"
```

```
,  
  
{  
  
    "value": "clone-repo-with-submodule-into-a-current-dir",  
  
    "label": "existing repo along with submodules into the current  
directory",  
  
    "usage": "git clone --recurse-submodules <repo-url> .",  
  
    "nb": "If git version is under 2.13, use --recursive option instead."
```

```
,  
  
{  
  
    "value": "clone-submodule-after",  
  
    "label": "submodules after cloning existing repo",  
  
    "usage": "git submodule update --init --recursive"  
  
}
```

```
,  
  
"ignore": [  
  
    {  
  
        "value": "ignore-files-in-a-dir",  
  
        "label": "all files in a directory",  
  
        "usage": "<dir name>/*",  
  
        "nb": "This must be added to .gitignore file\nReplace \"dir name\"\nwith name of directory whose files you want git to ignore"
```

```
},  

```

```

{

    "value": "ignore-all-files-of-a-specific-type",

    "label": "all files of a specific type",

    "usage": "/*.<filename extension>",

    "nb": "This must be added to .gitignore file\n\nReplace \"filename\nextension\" with the extension of the files you want git to ignore\n\nFor\nexample *.py tells git to ignore all python files in the repository"

}

],

"help": [

    {

        "value": "command-help",

        "label": "about a command",

        "usage": "append --help to the command",

        "nb": "e.g. git merge --help\n\nType q to quite terminal"

    }

],

"add": [

    {

        "value": "new-changes",

        "label": "new changes",

        "usage": "git add <file.ext>",

        "nb": "To add all the files in the current directory, use \"git add\n.\n\nTo add a directory use \"git add <directory>\""

    },

    {

        "value": "add-new-branch",

        "label": "a new branch",

        "usage": "git branch <branch-name>"

    },

],

```

```

{
    "value": "add-repo",
    "label": "new remote repo",
    "usage": "git remote add <shortname> <url>"
},
{
    "value": "add-alias",
    "label": "alias",
    "usage": "git config --global alias.<alias> <command>",
    "nb": "e.g. git config --global alias.st status. Typing git st in the
terminal now does the same thing as git status"
},
{
    "value": "add-annotated-tag",
    "label": "annotated tag",
    "usage": "git tag -a v1.4 -m \"my version 1.4\" \n\ngit push --tags"
},
{
    "value": "add-annotated-tag-for-old-commit",
    "label": "annotated tag for old commit",
    "usage": "git tag -a v1.2 -m 'version 1.2' <commit-hash>\n\ngit push
--tags"
}
],
"push": [
{
    "value": "new-remote-branch",
    "label": "non-existent remote branch",
    "usage": "git push -u origin <branchname>"
}
]
}

```

```
    }

],

"rename": [

    {

        "value": "branch",
```

```
        "label": "branch"

    },

    {

        "value": "file",

        "label": "file",

        "usage": "git mv file_from file_to"

    },

    {

        "value": "remoteUrl",

        "label": "remote",

        "usage": "git remote rename <oldname> <newname>"

    }

],

"merge": [

    {

        "value": "branch",

        "label": "another branch to current branch",

        "usage": "git merge <branch-name>"

    },

    {

        "value": "single-file",

        "label": "merge a single file from one branch to another.",

        "usage": "git checkout <branch name> <path to file> --patch"
```

```

    }

],

"squash": [

    {

        "value": "pr",

        "label": "commits in pull request into single commit",

        "usage": "git rebase -i <branch name>",

```

```

        "nb": "Make sure that latest commits are fetched from upstream.\n\nFor example (assuming you have a remote named upstream):\n\ngit fetch upstream\ngit rebase -i upstream/master\nChange \"pick\" to \"squash\" for the commits you wish to squash and save.\n\ngit push origin <topic branch> --force-with-lease"

    },

    {

        "value": "commits",

        "label": "last n number of commit into one",

        "usage": "git reset --soft HEAD~N\ngit add .\ngit commit -m <message>",

        "nb": "Replace N with the number of commits you want to squash and <message> with your commit message. You can use the command \"git log\" to view your commit history"

    }

],

"debug": [

    {

        "value": "bisect",

        "label": "binary search",

        "usage": "git bisect start\ngit bisect bad                                # Current version is bad\ngit bisect good v2.13                # v6.12 is known to be good",

        "nb": "Once you have specified at least one bad and one good commit, git bisect selects a commit in the middle of that range of history, checks it out, and outputs something similar to the following:\n\nBisecting: 675 revisions left to test after this (roughly 10 steps)\n\nYou should now compile the checked-out version and test it. If that version works correctly, type\n\ngit bisect good\n\nIf that version is broken, type\n\ngit bisect bad\n\nThen git bisect will respond with something like\n\nBisecting: 337"

```


revisions left to test after this (roughly 9 steps)\n\nKeep repeating the process: compile the tree, test it, and depending on whether it is good or bad run git bisect good or git bisect bad to ask for the next commit that needs testing.\nEventually there will be no more revisions left to inspect, and the command will print out a description of the first bad commit. The reference refs/bisect/bad will be left pointing at that commit.\nAfter a bisect session, to clean up the bisection state and return to the original HEAD, issue the following command:\n\ngit bisect reset"

```
    },  
  
    {  
  
        "value": "blame",  
  
        "label": "who modified each lines",  
  
        "usage": "git blame -L <number-line-start>,<number-line-end>
```

```
<filename>",  
  
        "nb": "The -L option will restrict the output to the requested line range\n"
```

```
    },  
  
    {  
  
        "value": "grep",  
  
        "label": "search in files",  
  
        "usage": "git grep -n <your_text_or_expression>",  
  
        "nb": "Print lines matching a pattern.\nOption -n to display the numbering of lines in which there are matches"
```

```
    }  
  
],  
  
    "recover": [  
  
        {  
  
            "value": "dropped-commit",  
  
            "label": "show hashes dangling commits after hard reset to previous commit",  
  
            "usage": "git reflog",  
  
            "nb": "alternative: git log -g. For recovery use\ngit checkout -b <recovery> <hash>"
```

```
    },  
  
    {
```

```

    "value": "deleted-branch",

    "label": "show hashes removed branch or other git objects",

    "usage": "git fsck --full",

    "nb": "show hashes all dangling git objects. For recovery use\n git
checkout -b <recovery> <hash>"

  },

],

"rebase": [

  {

    "value": "origin-branch",

    "label": "an origin branch into my working branch",

```

```

    "usage": "git pull --rebase origin <branch name>",

    "nb": "Rebase an origin branch into working branch. Replace <branch
name> with the branch you are pulling"

  },

  {

    "value": "local-branch",

    "label": "a local branch into my working branch",

    "usage": "git pull --rebase <branch name>",

    "nb": "Rebase another local branch into working branch. Replace
<branch name> with the branch you are pulling"

  },

  {

    "value": "skip",

    "label": "and skip a commit",

    "usage": "git rebase --skip",

    "nb": "During rebase, git might not be able to automatically apply
commits due to conflicts. You can use this command to discard of your own
changes in the current commit and apply the changes from an incoming branch"

  },

  {

```

```

        "value": "continue",

        "label": "and continue after resolving conflicts",

        "usage": "git rebase --continue",

        "nb": "During rebase, git might not be able to automatically apply
commits due to conflicts. You can resolve this conflicts manually and use
this command to continue your rebase operation"

    }

],

"synchronize": [

    {

        "value": "branch-from-fork",

        "label": "a branch in a fork",

        "usage": "git fetch <remote-repo> \n\ngit checkout <branch-name>
\n\ngit merge <remote-repo>/<branch-name>",

```

```

        "nb": "You need to add a remote repo for your fork first."

    }

],

"stash": [

    {

        "value": "save-stash",

        "label": "(un)tracked files",

        "usage": "git stash",

        "nb": "To stash with a customized message use git stash save
<message>\n\nTo stash untracked files git stash save -u"

    },

    {

        "value": "list-stash",

        "label": "view list of stashed changes",

        "usage": "git stash list"

    },

    {

```

```

        "value": "apply-stash",

        "label": "apply"

    },

    {

        "value": "show",

        "label": "view the contents of a stash",

        "usage": "git stash show -p <stash id>",

        "nb": "You can leave out the stash id if you want the contents of the
latest stash"

    },

    {

        "value": "delete-stash",

        "label": "delete"

    },

    {

        "value": "create-branch",

        "label": "create a new branch and apply stash",

        "usage": "git stash branch <branch name> <stash id>"

    }

]

}

}

```

MainActivity.kt

```

// MainActivity.kt

package com.example.githelp

import android.annotation.SuppressLint
import android.app.Activity
import android.content.Context
import android.os.Bundle

```

```

import android.text.Html

import android.view.View

import android.view.inputmethod.InputMethodManager

import android.widget.ArrayAdapter

import androidx.databinding.DataBindingUtil

import com.example.githelp.databinding.ActivityMainBinding

class MainActivity : Activity() {

    private lateinit var dataBind: ActivityMainBinding

    private var primaryOptions = ArrayList<PrimaryOptions>()

    private var primaryOptionsValue = ""

    private var secondaryOptions = ArrayList<SecondaryOptions>()

```

```

    private lateinit var dbHelper: DatabaseHelper

    private var usage = ""

    private var note = ""

    @SuppressWarnings("ClickableViewAccessibility")

    override fun onCreate(savedInstanceState: Bundle?) {

        super.onCreate(savedInstanceState)

        dataBind = DataBindingUtil.setContentView(this,
R.layout.activity_main)

        dataBind.textGitCommand.text =

            Html.fromHtml(resources.getString(R.string.git_command_explorer))

        dbHelper = DatabaseHelper(this)

        loadPrimaryOptionsFromDB()

        dataBind.inputFirstField.setOnTouchListener { _, _ ->

```

```

        dataBind.inputFirstField.showDropDown()

        false
    }

    dataBind.inputSecondField.setOnTouchListener { _, _ ->

        dataBind.inputSecondField.showDropDown()

        false
    }

    dataBind.inputFirstField.setOnItemClickListener { _, _, position, _ ->

        primaryOptionsValue = primaryOptions[position].value

        dismissKeyboard(dataBind.inputFirstField)

        dataBind.cardViewSecondField.visibility = View.VISIBLE

        dataBind.textNote.visibility = View.GONE

        dataBind.cardViewNote.visibility = View.GONE

        dataBind.inputSecondField.text.clear()
    }

```

```

        dataBind.textDisplayGitCommand.text = ""

        dataBind.textDisplayNote.text = ""

        loadSecondaryOptionsFromDB(primaryOptionsValue)
    }

    dataBind.inputSecondField.setOnItemClickListener { _, _, position, _
->

        dismissKeyboard(dataBind.inputSecondField)

        val selectedSecondaryOption = secondaryOptions[position]

        usage = selectedSecondaryOption.usage

        note = selectedSecondaryOption.nb

        dataBind.textNote.visibility = if (note.isEmpty()) View.GONE else
View.VISIBLE
    }

```

```

        dataBind.cardViewNote.visibility = if (note.isEmpty()) View.GONE
    else View.VISIBLE

        dataBind.textDisplayGitCommand.text = usage

        dataBind.textDisplayNote.text = note

    }

}

private fun loadPrimaryOptionsFromDB() {

    primaryOptions.clear()

    primaryOptions.addAll(dbHelper.getPrimaryOptions())

    val adapter = ArrayAdapter(

        this,

        android.R.layout.simple_list_item_1,

        primaryOptions.map { it.label }

    )

    dataBind.inputFirstField.setAdapter(adapter)

}

private fun loadSecondaryOptionsFromDB(primaryValue: String) {

    secondaryOptions.clear()

    secondaryOptions.addAll(dbHelper.getSecondaryOptions(primaryValue))

    val adapter = ArrayAdapter(

        this,

        android.R.layout.simple_list_item_1,

        secondaryOptions.map { it.label }

    )
    dataBind.inputSecondField.setAdapter(adapter)

}

private fun Context.dismissKeyboard(view: View?) {

```

```

        view?.let {

            val imm = getSystemService(Context.INPUT_METHOD_SERVICE) as
InputMethodManager

            imm.hideSoftInputFromWindow(it.windowToken, 0)

        }

    }

    override fun onDestroy() {

        dbHelper.close()

        super.onDestroy()

    }

}

```

SplashScreen.kt

```

package com.example.githelp

import android.annotation.SuppressLint
import android.app.Activity
import android.content.Intent
import android.os.Bundle
import android.os.Handler
import android.text.Html
import androidx.appcompat.app.AppCompatActivity
import androidx.databinding.DataBindingUtil
import com.example.githelp.databinding.ActivitySplashScreenBinding

@SuppressLint("CustomSplashScreen")
class SplashScreen : Activity() {

    companion object {

        private const val DELAY_TIME_IN_MILLS = 2500L
    }
}

```



```

    }

    private lateinit var dataBind: ActivitySplashScreenBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        dataBind = DataBindingUtil.setContentView(this,
R.layout.activity_splash_screen)

        dataBind.textGitCommand.text =

            Html.fromHtml(resources.getString(R.string.git_command_explorer))

        Handler().postDelayed({

            val intent = Intent(this, MainActivity::class.java)

            startActivity(intent)

            finish()

        }, DELAY_TIME_IN_MILLS)

    }
}

```

DatabaseHelper.kt

```

// DatabaseHelper.kt

package com.example.githelp

import android.content.ContentValues
import android.content.Context
import android.database.Cursor
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteOpenHelper
import android.util.Log
import org.json.JSONObject

```

```
// Extension function to load JSON from assets

private fun Context.loadJSONFromAsset(filename: String): String? { // Made
    'private'

    return try {

        assets.open(filename).bufferedReader().use { it.readText() }

    } catch (e: Exception) {

        e.printStackTrace()

        return null // Explicitly return null in case of error

    }

}

class DatabaseHelper(private val context: Context) : // context as a private
val

    SQLiteOpenHelper(context, DATABASE_NAME, null, DATABASE_VERSION) {

    companion object {

        private const val DATABASE_NAME = "GitCommandExplorerer.db"

        private const val DATABASE_VERSION = 1

    }

}
```

```
// Table Names

private const val TABLE_PRIMARY_OPTIONS = "primary_options"

private const val TABLE_SECONDARY_OPTIONS = "secondary_options"


// Primary Options Table Columns

private const val COL_PRIMARY_VALUE = "value"

private const val COL_PRIMARY_LABEL = "label"


// Secondary Options Table Columns

private const val COL_SECONDARY_VALUE = "value"

private const val COL_SECONDARY_LABEL = "label"
```

```

        private const val COL_SECONDARY_USAGE = "usage"

        private const val COL_SECONDARY_NB = "nb"

        private const val COL_SECONDARY_PRIMARY_VALUE = "primary_value" //
Foreign Key
    }

```

```

    override fun onCreate(db: SQLiteDatabase?) {

```

```

        // Create Primary Options Table

```

```

        val createPrimaryTable = """

```

```

            CREATE TABLE $TABLE_PRIMARY_OPTIONS (

```

```

                $COL_PRIMARY_VALUE TEXT PRIMARY KEY,

```

```

                $COL_PRIMARY_LABEL TEXT

```

```

            )

```

```

        """.trimIndent()

```

```

        db?.execSQL(createPrimaryTable)

```

```

        // Create Secondary Options Table

```

```

        val createSecondaryTable = """

```

```

            CREATE TABLE $TABLE_SECONDARY_OPTIONS (

```

```

                $COL_SECONDARY_VALUE TEXT PRIMARY KEY,

```

```

                $COL_SECONDARY_LABEL TEXT,

```

```

                $COL_SECONDARY_USAGE TEXT,

```

```

                $COL_SECONDARY_NB TEXT,

```

```

                $COL_SECONDARY_PRIMARY_VALUE TEXT,

```

```

                FOREIGN KEY ($COL_SECONDARY_PRIMARY_VALUE)

```

```

                REFERENCES $TABLE_PRIMARY_OPTIONS ($COL_PRIMARY_VALUE)

```

```

            )

```

```

        """.trimIndent()

```

```

        db?.execSQL(createSecondaryTable)

```

```

        // ** POPULATING DATABASE FROM JSON **

        populateDatabase(db)

        Log.i("DB_INIT", "Database populated on create.")
    }

    override fun onUpgrade(db: SQLiteDatabase?, oldVersion: Int, newVersion:
Int) {

        // Handle database upgrades by dropping existing tables

        db?.execSQL("DROP TABLE IF EXISTS $TABLE_SECONDARY_OPTIONS")

        db?.execSQL("DROP TABLE IF EXISTS $TABLE_PRIMARY_OPTIONS")

        onCreate(db) // Re-create and re-populate on upgrade (for simplicity)
    }

    private fun populateDatabase(db: SQLiteDatabase?) {

        try {

            val jsonString =
context.loadJSONFromAsset("git_command_explorerer.json")

            jsonString?.let { jsonStringNonNull -> // Renamed 'it' to
'jsonStringNonNull'

                val json = JSONObject(jsonStringNonNull)

                val primaryOptions = json.getJSONArray("primary_options")

                for (i in 0 until primaryOptions.length()) {

                    val primary = primaryOptions.getJSONObject(i)

                    val values = ContentValues().apply {

                        put(COL_PRIMARY_VALUE, primary.getString("value"))

                        put(COL_PRIMARY_LABEL, primary.getString("label"))

                    }

                    db?.insert(TABLE_PRIMARY_OPTIONS, null, values)

```

```

    }

    val secondaryOptions = json.getJSONObject("secondary_options")

    val primaryKeys = secondaryOptions.keys()

    while (primaryKeys.hasNext()) {

        val primaryKey = primaryKeys.next() as String

        val secondaryArray =
secondaryOptions.getJSONArray(primaryKey)

        for (i in 0 until secondaryArray.length()) {

            val secondary = secondaryArray.getJSONObject(i)

            val values = ContentValues().apply {

                put(COL_SECONDARY_VALUE,
secondary.getString("value"))

                put(COL_SECONDARY_LABEL,
secondary.getString("label"))

                if (secondary.has("usage")) {

                    put(COL_SECONDARY_USAGE,
secondary.getString("usage"))

                }

                if (secondary.has("nb")) {

                    put(COL_SECONDARY_NB,
secondary.getString("nb"))

                }

                put(COL_SECONDARY_PRIMARY_VALUE, primaryKey)

            }

            db?.insert(TABLE_SECONDARY_OPTIONS, null, values)

```

```

        }

    }

    } ?: run {

        Log.e("DB_INIT", "Error loading JSON for population.")

    }

```

```

        } catch (e: Exception) {

            Log.e("DB_INIT", "Error populating database:
${e.localizedMessage}")

        }

    }

}

// ** The app will use these methods to retrieve data from the database **

fun getPrimaryOptions(): List<PrimaryOptions> {

    val primaryOptionsList = mutableListOf<PrimaryOptions>()

    val db = readableDatabase

    val cursor: Cursor = db.query(

        TABLE_PRIMARY_OPTIONS,

        arrayOf(COL_PRIMARY_VALUE, COL_PRIMARY_LABEL),

        null, null, null, null, null

    )

    cursor.use {

        while (it.moveToNext()) {

            primaryOptionsList.add(

                PrimaryOptions(

it.getString(it.getColumnIndexOrThrow(COL_PRIMARY_VALUE)),

it.getString(it.getColumnIndexOrThrow(COL_PRIMARY_LABEL))

                )

            )

        }

    }

}

db.close()

return primaryOptionsList

}

```

```

fun getSecondaryOptions(primaryValue: String): List<SecondaryOptions> {

    val secondaryOptionsList = mutableListOf<SecondaryOptions>()

    val db = readableDatabase

    val cursor: Cursor = db.query(
TABLE_SECONDARY_OPTIONS,

        arrayOf(

            COL_SECONDARY_VALUE,

            COL_SECONDARY_LABEL,

            COL_SECONDARY_USAGE,

            COL_SECONDARY_NB

        ),

        "$COL_SECONDARY_PRIMARY_VALUE = ?",

        arrayOf(primaryValue),

        null, null, null

    )

    cursor.use {

        while (it.moveToNext()) {

            secondaryOptionsList.add(

                SecondaryOptions(

it.getString(it.getColumnIndexOrThrow(COL_SECONDARY_VALUE)),

it.getString(it.getColumnIndexOrThrow(COL_SECONDARY_LABEL)),

it.getString(it.getColumnIndexOrThrow(COL_SECONDARY_USAGE)) ?: "",

it.getString(it.getColumnIndexOrThrow(COL_SECONDARY_NB)) ?: ""

                )

            )

        }

    }
}

```

```

        }

        db.close()

        return secondaryOptionsList
    }

    override fun close() {
        readableDatabase.close()

        writableDatabase.close()

        super.close()
    }
}

```

PrimaryOptions.kt

```

package com.example.githelp

data class PrimaryOptions(val value: String, val label: String)

```

SecondaryOptions.kt

```

package com.example.githelp



data class SecondaryOptions(
    val value: String,
    val label: String,
    val usage: String,
    val nb: String
)

```

OUTPUT:

9:21Z📶🔋83%

Baking with Gemini



Prompt

give me a food recipe with ingredients like potato, water, salt, oil, tomato and pepper

Go

This recipe uses the ingredients you provided to make a simple potato and tomato dish. It's a flexible recipe; you can adjust the seasoning and add other vegetables to your liking.

****Simple Roasted Potatoes with Tomato and Pepper****



****Yields:**** 4 servings
****Prep time:**** 15 minutes
****Cook time:**** 35-45 minutes

****Ingredients:****

- * 1 lb potatoes, peeled and cubed (Yukon Gold or red potatoes work well)
- * 1 large tomato, diced

9:20Z📶🔋84%

Baking with Gemini



Prompt



Provide a recipe for the baked goods in the image

Go

(Results will appear here)

9:21Z📶🔋83%

Baking with Gemini



Prompt

give me a food recipe with ingredients like potato, water, salt, oil, tomato and pepper

Go

****Add the tomato and pepper:****


1. Add the diced tomato and bell pepper to the baking sheet with the potatoes. Gently toss to combine.
2. Continue roasting for another 15-20 minutes, or until the potatoes are tender and slightly browned, and the tomatoes are softened. If the potatoes seem to be drying out too much before they are cooked through, you can add a tablespoon or two of water to the pan during cooking.

****Serve:****

1. Remove the baking sheet from the oven and let the potatoes cool slightly before serving. This dish is delicious on its own, or as a side dish with grilled

9:21Z📶🔋83%

Baking with Gemini



Prompt

give me a food recipe with ingredients like potato, water, salt, oil, tomato and pepper

Go

****Instructions****

****Get started:****

1. Preheat your oven to 400°F (200°C).

****Prepare the potatoes:****



1. Wash and peel the potatoes. Cut them into roughly 1-inch cubes. Try to make them relatively uniform in size for even cooking.
2. In a large bowl, toss the potatoes with olive oil, salt, and pepper.

****Roast the potatoes:****

1. Spread the potatoes in a single layer on a baking

9:21Z📶🔋83%

Baking with Gemini



Prompt

give me a food recipe with ingredients like potato, water, salt, oil, tomato and pepper

Go

make a simple potato and tomato dish. It's a flexible recipe; you can adjust the seasoning and add other vegetables to your liking.

****Simple Roasted Potatoes with Tomato and Pepper****



****Yields:**** 4 servings
****Prep time:**** 15 minutes
****Cook time:**** 35-45 minutes

****Ingredients:****

- * 1 lb potatoes, peeled and cubed (Yukon Gold or red potatoes work well)
- * 1 large tomato, diced

9:21Z📶🔋83%

Baking with Gemini



Prompt

give me a food recipe with ingredients like potato, water, salt, oil, tomato and pepper

Go

****Ingredients:****

- * 1 lb potatoes, peeled and cubed (Yukon Gold or red potatoes work well)
- * 1 large tomato, diced
- * 1/2 bell pepper (any color), diced
- * 2 tablespoons olive oil (or other cooking oil)
- * 1/2 teaspoon salt
- * 1/4 teaspoon black pepper
- * Water (as needed)

****Equipment:****

- * Large bowl
- * Baking sheet
- * Oven

6. RESULTS AND DISCUSSION

Upon deployment and testing, the *Recipe* app met its primary goals effectively. The user interface proved responsive, and the dropdown selection mechanism worked flawlessly, allowing users to narrow down Git commands quickly. Testing across multiple devices confirmed consistent performance, with near-instantaneous database queries thanks to SQLite.

Key highlights:

- The app's offline mode worked as intended, demonstrating its utility in environments without internet access.
- The modular design of the database allows future expansion by simply updating the JSON file.
- Data Binding in Kotlin simplified UI updates, reducing the risk of bugs.

Potential improvements identified:

7. Adding a full-text search feature.
8. Implementing bookmarks for frequently used commands.
9. Enhancing UI with material design animations.

7. CONCLUSION

The *Recipe* app successfully addresses the challenge of navigating Git's complex command system by offering a practical, offline command explorer. The app leverages Kotlin, XML, and SQLite to deliver a performant, scalable, and user-friendly solution. Its modularity ensures that the database can be expanded easily, while its clean UI promotes adoption by developers of all skill levels. Future iterations can further enhance the app with advanced features like bookmarking, syncing, and advanced search capabilities.

8.REFERENCES

1. Android Developer Documentation: <https://developer.android.com/>
2. Kotlin Official Documentation: <https://kotlinlang.org/docs/>
3. SQLite Documentation: <https://www.sqlite.org/>
4. Git SCM Documentation: <https://git-scm.com/doc>
5. JSON.org: <https://www.json.org/json-en.html>
6. <https://www.mysqltutorial.org/mysql-triggers.aspx>