ABSTRACT

This project aims to develop a mobile application for Android devices using Android Studio, which serves as a calendar app. This mobile application development project focuses on creating an Android calendar app using Android Studio. The app will provide users with essential calendar features, including event creation, event reminders, and synchronization with the device's built-in calendar. The project report outlines the development process, key features, challenges faced, and future improvements. The app utilizes the Android operating system's calendar APIs and features a user-friendly interface to enhance the user experience.

The project report provides a comprehensive overview of the project, its objectives, and the outcomes achieved.

The app will enable users to manage their events and appointments efficiently. Key features include event creation, event reminders, calendar synchronization, and multiple views. The project report provides a concise overview of the project, its objectives, and the main features implemented. The app is designed to offer a user-friendly interface and enhance the overall calendar experience for Android users.

Table of Contents

CHAPTER 1:
INTRODUCTION

Android calendar application that allows users to manage their events, set reminders, and view their schedules in a user-friendly interface. Even the remainder is also present over there which is used to set the events and even we can create our own personal timetable in it. The app should provide essential calendar functionalities while offering a seamless and intuitive user experience.

1.1 ANDROID STUDIO:

Android Studio is a robust integrated development environment (IDE) created exclusively for the development of Android apps. It is the developer's go-to tool, offering a complete set of features and capabilities that streamline the entire development process. Android Studio's user-friendly interface and vast capabilities enable developers to create high-quality, inventive, and feature-rich Android applications.

Android Studio accelerates the development workflow by including advanced features such as intelligent code completion, real-time error checking, and robust debugging tools. Its easy connection with the Android SDK, as well as support for other programming languages like as Java and Kotlin, provides developers with freedom and allows them to create dynamic and performant Android apps.

From creating visually appealing user interfaces to optimizing app performance, Android Studio has a tool for every facet of Android app development. Android Studio facilitates the development and distribution of Android apps with its emulator for testing apps on virtual devices, APK analyzer for optimizing app size, and built-in support for version control systems.

Whether you're a seasoned developer or just starting out, Android Studio is the ideal tool for making your app ideas a reality. It enables developers to unleash their creativity, use the power of the Android platform, and provide unique user experiences to millions of Android users around the world.

1.2 JAVA:

For decades, Java, a widely used programming language, has been a cornerstone of software development. Java, is known for its versatility and platform freedom, enabling developers to create powerful, scalable, and secure applications across a wide range of fields.

Java's "write once, run anywhere" principle enables programs to be built on a single platform and operate flawlessly across multiple operating systems. It has a large ecosystem of libraries, frameworks, and tools to help in speedy development and productivity.

The object-oriented nature of Java encourages code reusability, maintainability, and modularity. Its extensive type checking, automatic garbage collection, and robust memory management all contribute to the creation of stable and trustworthy software.

Java is widely used in enterprise systems, Android app development, web apps, and other areas. It has strong APIs for networking, multithreading, database connectivity, and user interface development, making it a go-to language for a wide range of development requirements.

Java continues to expand and adapt to contemporary development trends, keeping its relevance in today's digital ecosystem, thanks to its widespread adoption, vast documentation, and thriving community. Java provides a solid basis for building sophisticated, cross-platform apps, whether you're a newbie or an experienced developer.

1.3 KOTLIN:

Kotlin is a modern, statically typed programming language that is very popular among Android developers. JetBrains' Kotlin language smoothly integrates with existing Java codebases and provides various benefits that improve productivity and code readability.

Kotlin's compact syntax reduces boilerplate code, allowing for speedier and

more efficient development. It is compatible with both object-oriented and functional programming paradigms, allowing developers to produce clean, expressive code. The null safety feature in Kotlin aids in the elimination of null pointer exceptions, making apps more robust and reliable.

Kotlin is fully compatible with Java, allowing developers to make use of existing Java libraries and frameworks. Its seamless integration with the Android Studio IDE offers advanced tools like as intelligent code completion, refactoring assistance, and real-time error feedback.

Kotlin is a versatile language that appeals to a wide range of Android development demands, from simple utility functions to complicated app structures. Kotlin has established itself as the preferred language for developing high-quality, modern Android applications due to its modern features, robust community support, and smooth learning curve.

1.4 XML:

For building user interfaces, XML (Extensible Markup Language) is a crucial technology utilized in Android development. It is a markup language that describes the organization and presentation of user interface elements in a clear and legible manner. Developers can use XML to separate the design of the user interface from the logic of the program.

XML is used in Android development to construct layout files that describe the layout and appearance of views like buttons, text fields, and images. XML has a hierarchical structure that allows developers to effectively nest and organize UI components.

Developers can define characteristics, attributes, and behaviors for each UI element using XML-based layout files. Developers can produce a consistent and visually appealing user interface across various screen sizes and orientations by utilizing XML.

The usage of XML in Android development improves code reuse, maintainability, and collaboration between designers and developers. Because of its simplicity and versatility, XML continues to be an important component in creating intuitive and visually appealing user interfaces for Android applications.

1.5 FIREBASE:

Google Firebase is a comprehensive and powerful mobile and online development platform. It offers a variety of tools and services that help to simplify and accelerate the development process. Developers may use Firebase to create high-quality apps with minimal backend infrastructure configuration.

Firebase provides a variety of services, including real-time database,

authentication, cloud storage, and hosting. These services interact seamlessly with mobile and web apps, offering capabilities such as real-time data synchronization, user authentication, cloud storage, and serverless functionalities.

The real-time database provided by Firebase allows developers to create responsive and collaborative applications, while the authentication service provides secure user management and authentication mechanisms. Cloud storage makes it simple to store and retrieve user-generated material, and hosting enables quick deployment and scaling of web applications.

With Firebase, developers can concentrate on providing amazing user experiences and creative features while Google handles backend infrastructure and upkeep. Firebase is a popular choice for startups, organizations, and developers trying to build scalable and dependable applications due to its extensive documentation, user-friendly SDKs, and robust analytics capabilities.

1.6 FIRESTORE:

Google Firestore is a robust and adaptable NoSQL document database that is part of the Firebase platform. It is intended for storing, synchronizing, and querying data for web, mobile, and server applications. Firestore is a scalable and real-time database solution that synchronizes data across devices and platforms smoothly.

Developers may simply store and retrieve data in the form of documents organized into collections using Firestore. It offers offline data persistence, allowing apps to run even when the device is turned off. Real-time data synchronization allows for instant data updates across different clients, resulting in responsive and collaborative user experiences.

Firestore also has strong querying capabilities, allowing developers to run complicated queries and retrieve data based on certain conditions. It works in tandem with other Firebase services, including Firebase Authentication, Cloud Functions, and Cloud Storage, to enable the development of comprehensive and scalable apps.

Firestore provides a flexible and dependable database solution that simplifies data administration, improves collaboration, and enables real-time synchronization in your applications, whether you're developing a tiny application or a large-scale business solution.

1.7 FEATURES OF CALENDAR APP

- Event Creation: Users can create new events by specifying details such as event title, date, time, and location. The app provides an intuitive interface for entering and saving event information.

- Event Reminders: The app allows users to set reminders for their events. Reminders can be set to notify users at specific times or intervals before the event occurs.

- Calendar Synchronization: The app synchronizes with the device's built-in calendar to ensure that events created within the app are reflected in the device's calendar and vice versa. This ensures seamless integration with other calendar apps on the device.

- Multiple Views: The app provides various views for users to visualize their events, including daily, weekly, and monthly views. Users can switch between these views to navigate through their calendar effortlessly.

- Event Categorization: Users can categorize events based on different types or themes.

CHAPTER 2:
REQUIREMENT SPECIFICATION

2.1 FUNCTIONAL REQUIREMENTS

It is necessary to define the functional requirements as the features and functionalities that the Calendar application should provide for the users. These requirements may include the following:

1. **Event Creation:**

   Users can create new events by specifying details such as event title, date, time, and location. The app provides an intuitive interface for entering and saving event information.Set event reminders and notifications

2. **Event Reminders:**

   The app allows users to set reminders for their events. Reminders can be set to notify users at specific times or intervals before the event occurs.

3. **Calendar Synchronization:**

   The app synchronizes with the device's built-in calendar to ensure that events created within the app are reflected in the device's calendar and vice versa. This ensures seamless integration with other calendar apps on the device.Search events by title or description

4. **Multiple Views:**

   The app provides various views for users to visualize their events, including daily, weekly, and monthly views. Users can switch between these views to navigate through their calendar effortlessly.

5. **Event Categorization:**

   Users can categorize events based on different types or themes, making it easier to manage and search for specific events.

2.2 SOFTWARE REQUIREMENTS

The Calendar Application software requirements outline both the software components and tools that are requiredto operate the application in the most efficient manner. For example:

1. **Development Framework and Language:**
   o The application should be developed using Kotlin as the primary programming language.
   o The development framework should be compatible with Kotlin, and should make the development of Android applications a breeze.

2. **User Interface:**
   o The application should be designed to be intuitive and user-friendly in nature.
   o The UI/UX design principles should also be followed to ensure a seamless user experience with the application.

3. **Backend:**
   o The application must use Firebase as the backend for the user authentication and the authentication of event/task data.
   o Firebase Realtime Database or Firestore should be used to store task/event and remind later.

4. **Compatibility:**
   o The application should be compatible with Android devices running a minimum of Android OS version 4.1.

2.3 HARDWARE REQUIREMENTS

The hardware requirements specify all the hardware requirements that must be met to successfully run Calendar App. These requirements include, among others:

1. **Android Devices:**
   - The application should be able to run on a wide array of Android devices, including smartphones and tablets.
   - Identify whether you will need minimum device specifications based on the target audience, such as RAM and processor speed.

2. **Internet Connectivity:**
   - To authenticate users, retrieve event/task data, it should be required that usershave an active in order to access the application and updated in application.

CHAPTER 3:
DESIGN OF THE PROJECT

3.1 SYSTEM DESIGN

The Android calendar app follows a client-server architecture. The client side is implemented as an Android application using Android Studio, while the server side consists of a backend server and a database. The client communicates with the server through API endpoints for user authentication and data synchronization.

1. **Client-Side Components:**
   o The client-side component of Android applications is composed of the user interface (UI), which is created using XML and Kotlin.
   o Interactions with users are enabled by UI controls, which are implemented through Kotlin code.

2. **Server-Side Components:**
   o For users to register and log in, Firebase Authentication provides an extremely secure authentication method, utilizing an email and password credentials that provide secure access.
   o To store and retrieve quiz data, including quizzes, questions, answers, and scores, Firebase Realtime Database is used.
   o In order to implement the server-side logic, Firebase provides Cloud Functions and Cloud Firestore, two of the backend services offered by Firebase.

3. **Communication Protocols:**
   o The communication between the Android application and Firebase services is performed over HTTPS, ensuring the data transmission between client and server components is secure.
   o The communication between the client and server components within the Android application and Firebase service is performed over JSON.

3.2 DATABASE DESIGN

In order to determine Calendar's logical design, we focus on the internal structure and behavior of the system. Based on this:

The database schema includes tables for users, events, and notifications. The users table stores user information such as username, password, and device token for push notifications. The events table stores event details like title, date, time, description, and associated user ID. The notifications table stores notification details, including the event ID, scheduled time, and notification message.

3.3 USER INTERFACE DESIGN

A physical design for the Calendar application focuses on the hardware and infrastructure aspects of it.It is important to consider the following points:

1. **Hosting and Deployment:**
   o In order to provide users with the Calendar application, Firebase Hosting is used as a hosting platform for serving the Android application to users.
   o Under this system, Firebase Realtime Database is utilized to store event/task data, which is scaled byFirebase and automatically maintained by the platform as well.

2. **Device Compatibility:**
   o Calendar was designed to be compatible with Android-based devices that are running Android OS version 5.0 (Lollipop) or higher.
   o The UI has been converted to adjust to different screen sizes so that users will enjoy a consistent experience across devices with different screen resolutions.
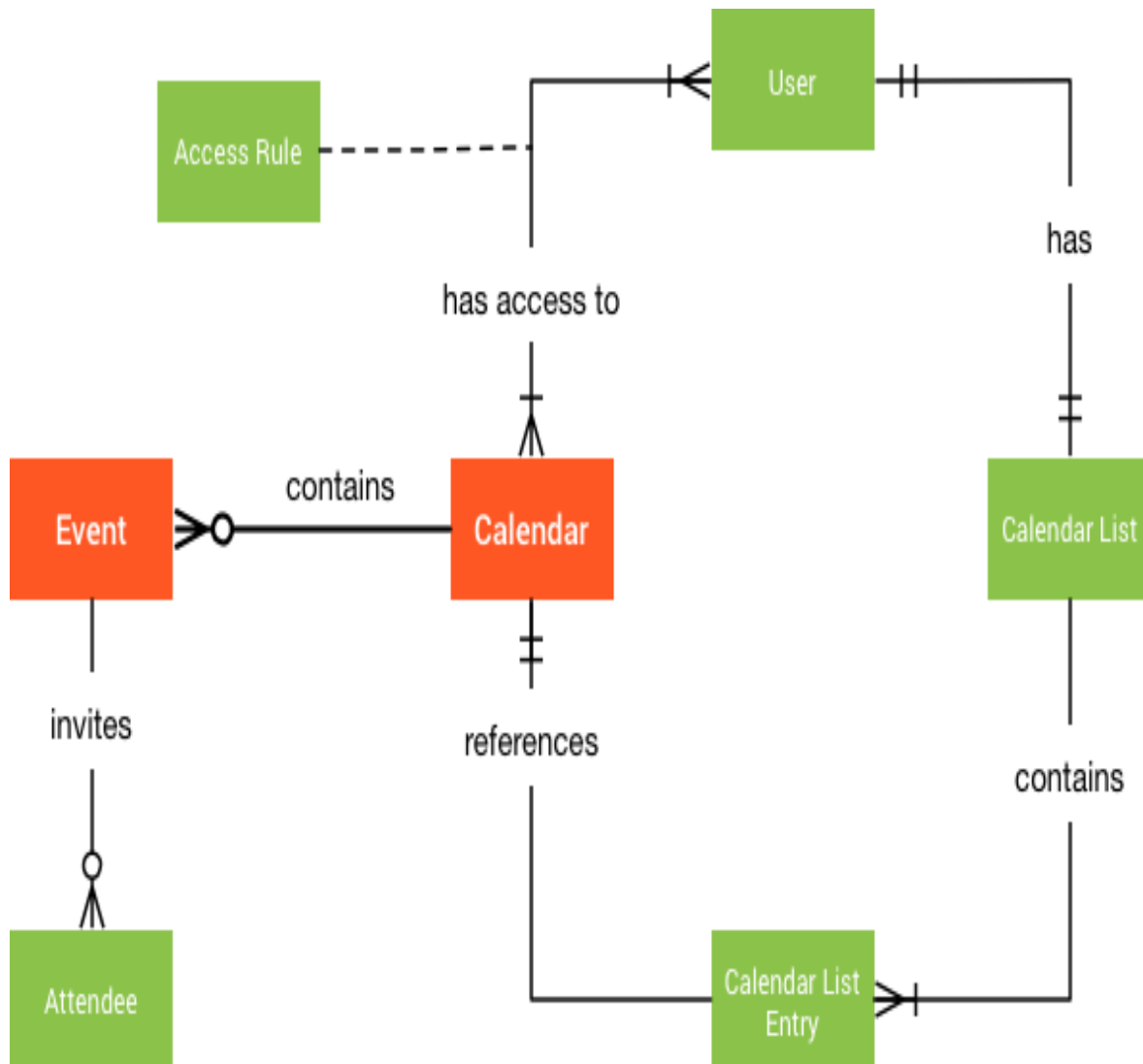
3. **Network Requirements:**
   o The Calendar application requires an offline application.
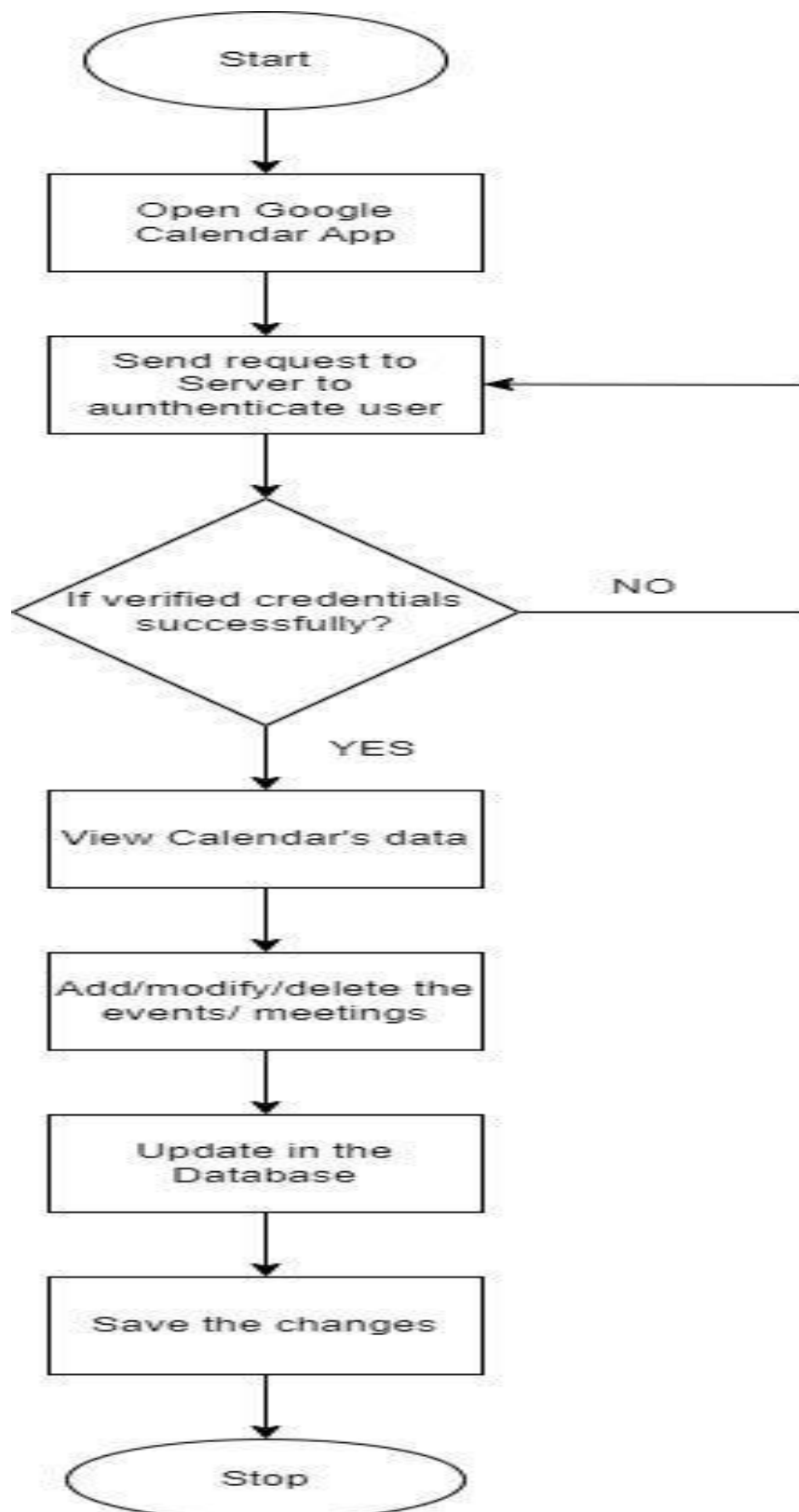   o The HTTPS protocol ensures the communication between the client and server is secure.

4. **Backup and Recovery:**
   o The Firebase service provides automatic backups and replication of Realtime Databases, protecting the data integrity and availability of the real-time database.
   o As well, Firebase offers the capability to recover your data through its backup and restore service.

3.4 DATA FLOW DIAGRAM

3.5 FLOW CHARTS

```
                    ┌─────────────┐
                    │    Start    │
                    └─────────────┘
                          │
                          ▼
                 ┌──────────────────┐
                 │   Open Google    │
                 │  Calendar App    │
                 └──────────────────┘
                          │
                          ▼
                 ┌──────────────────┐
                 │  Send request to │◄──────────┐
                 │    Server to     │           │
                 │ aunthenticate user│          │
                 └──────────────────┘           │
                          │                      │
                          ▼                      │
                      ◇ If verified ◇     NO     │
                      credentials  ─────────────┘
                      successfully?
                          │
                          │ YES
                          ▼
                 ┌──────────────────┐
                 │ View Calendar's  │
                 │      data        │
                 └──────────────────┘
                          │
                          ▼
                 ┌──────────────────┐
                 │ Add/modify/delete│
                 │  the events/     │
                 │    meetings      │
                 └──────────────────┘
                          │
                          ▼
                 ┌──────────────────┐
                 │  Update in the   │
                 │    Database      │
                 └──────────────────┘
                          │
                          ▼
                 ┌──────────────────┐
                 │ Save the changes │
                 └──────────────────┘
                          │
                          ▼
                    ┌─────────────┐
                    │    Stop     │
                    └─────────────┘
```

CHAPTER 4:
IMPLEMENTATION

4.1 activity_main.xml

```
<? xml version = "1.0" encoding = "utf-8" ?>
    < LinearLayout
        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"
        android: layout_width = "match_parent"
        android: layout_height = "match_parent"
        android: orientation = "vertical"
        tools: context = ".MainActivity" >


    < LinearLayout
    android: layout_width = "match_parent"
    android: layout_height = "wrap_content"
    android: orientation = "horizontal"
    android: layout_marginBottom = "20dp"
    android: layout_marginTop = "20dp" >
    < Button
        android: layout_width = "0dp"
        android: layout_height = "wrap_content"
        android: layout_weight = "1"
        android: text = "@string/back"
        android: textColor = "@color/blue"
        android: background = "@null"
        android: textStyle = "bold"
        android: onClick = "previousMonthAction"
        android: textSize = "20sp"  < TextView
        android: id = "@+id/monthYearTV"
```

```
    android: layout_width = "0dp"

    android: layout_weight = "2"

    android: layout_height = "wrap_content"

    android: text = "Feb 2021"

    android: textSize = "20sp"

    android: textAlignment = "center"

    android: textColor = "@color/black" />

 < Button

    android: layout_width = "0dp"

    android: layout_height = "wrap_content"

    android: layout_weight = "1"

    android: text = "@string/forward"

    android: textColor = "@color/blue"

    android: background = "@null"

    android: textStyle = "bold"

    android: onClick = "nextMonthAction"

    android: textSize = "20sp" />


</ LinearLayout >


< LinearLayout

   android: layout_width = "match_parent"

   android: layout_height = "wrap_content"

   android: orientation = "horizontal" >

   < TextView

    android: layout_width = "0dp"

    android: layout_height = "wrap_content"

    android: layout_weight = "1"

    android: text = "SUN"

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

```
android: textAlignment = "center"
android: textSize = "16sp" />


< TextView
    android: layout_width = "0dp"
    android: layout_height = "wrap_content"
    android: layout_weight = "1"
    android: text = "MON"
    android: textColor = "@color/gray"
    android: textAlignment = "center"
    android: textSize = "16sp" />


< TextView
    android: layout_width = "0dp"
    android: layout_height = "wrap_content"
    android: layout_weight = "1"
    android: text = "TUE"
    android: textColor = "@color/gray"
    android: textAlignment = "center"
    android: textSize = "16sp" />


< TextView
    android: layout_width = "0dp"
    android: layout_height = "wrap_content"
    android: layout_weight = "1"
    android: text = "WED"
    android: textColor = "@color/gray"
    android: textAlignment = "center"
    android: textSize = "16sp" />
```

```xml
< TextView
    android: layout_width = "0dp"
    android: layout_height = "wrap_content"
    android: layout_weight = "1"
    android: text = "THUR"
    android: textColor = "@color/gray"
    android: textAlignment = "center"
    android: textSize = "16sp" />


< TextView
    android: layout_width = "0dp"
    android: layout_height = "wrap_content"
    android: layout_weight = "1"
    android: text = "FRI"
    android: textColor = "@color/gray"
    android: textAlignment = "center"
    android: textSize = "16sp" />


< TextView
    android: layout_width = "0dp"
    android: layout_height = "wrap_content"
    android: layout_weight = "1"
    android: text = "SAT"
    android: textColor = "@color/gray"
    android: textAlignment = "center"
    android: textSize = "16sp" />
 < androidx.recyclerview.widget.RecyclerView
   android: id = "@+id/calendarRecyclerView"
   android: layout_width = "match_parent"
   android: layout_height = "match_parent" /


 </ LinearLayout >
```

4.2 calendar_cell.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
xmlns:app="http://schemas.android.com/apk/res-auto">

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/cellDayText"
    android:text="1"
    android:textSize="20sp"
    android:textColor="@color/black"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintVertical_bias="0.25"/>

</androidx.constraintlayout.widget.ConstraintLayout
```

4.3 ic_launcher_background.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<vector xmlns:android="http://schemas.android.com/apk/res/android"
android:width="108dp"
android:height="108dp"
android:viewportWidth="108"
```

```
    android:viewportHeight="108">
<path
    android:fillColor="#3DDC84"
    android:pathData="M0,0h108v108h-108z" />
<path
    android:fillColor="#00000000"
    android:pathData="M9,0L9,108"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M19,0L19,108"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M29,0L29,108"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M39,0L39,108"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M49,0L49,108"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M59,0L59,108"
```

```
      android:strokeWidth="0.8"

      android:strokeColor="#33FFFFFF" />

<path

      android:fillColor="#00000000"

      android:pathData="M69,0L69,108"

      android:strokeWidth="0.8"

      android:strokeColor="#33FFFFFF" />

<path

      android:fillColor="#00000000"

      android:pathData="M79,0L79,108"

      android:strokeWidth="0.8"

      android:strokeColor="#33FFFFFF" />

<path

      android:fillColor="#00000000"

      android:pathData="M89,0L89,108"

      android:strokeWidth="0.8"

      android:strokeColor="#33FFFFFF" />

<path

      android:fillColor="#00000000"

      android:pathData="M99,0L99,108"

      android:strokeWidth="0.8"

      android:strokeColor="#33FFFFFF" />

<path

      android:fillColor="#00000000"

      android:pathData="M0,9L108,9"

      android:strokeWidth="0.8"

      android:strokeColor="#33FFFFFF" />

<path

      android:fillColor="#00000000"

      android:pathData="M0,19L108,19"

      android:strokeWidth="0.8"
```

```
      android:strokeColor="#33FFFFFF" />
<path
      android:fillColor="#00000000"
      android:pathData="M0,29L108,29"
      android:strokeWidth="0.8"
      android:strokeColor="#33FFFFFF" />
<path
      android:fillColor="#00000000"
      android:pathData="M0,39L108,39"
      android:strokeWidth="0.8"
      android:strokeColor="#33FFFFFF" />
<path
      android:fillColor="#00000000"
      android:pathData="M0,49L108,49"
      android:strokeWidth="0.8"
      android:strokeColor="#33FFFFFF" />
<path
      android:fillColor="#00000000"
      android:pathData="M0,59L108,59"
      android:strokeWidth="0.8"
      android:strokeColor="#33FFFFFF" />
<path
      android:fillColor="#00000000"
      android:pathData="M0,69L108,69"
      android:strokeWidth="0.8"
      android:strokeColor="#33FFFFFF" />
<path
      android:fillColor="#00000000"
      android:pathData="M0,79L108,79"
      android:strokeWidth="0.8"
      android:strokeColor="#33FFFFFF" />
<path
```

```
    android:fillColor="#00000000"

    android:pathData="M0,89L108,89"

    android:strokeWidth="0.8"

    android:strokeColor="#33FFFFFF" />

<path

    android:fillColor="#00000000"

    android:pathData="M0,99L108,99"

    android:strokeWidth="0.8"

    android:strokeColor="#33FFFFFF" />

<path

    android:fillColor="#00000000"

    android:pathData="M19,29L89,29"

    android:strokeWidth="0.8"

    android:strokeColor="#33FFFFFF" />

<path

    android:fillColor="#00000000"

    android:pathData="M19,39L89,39"

    android:strokeWidth="0.8"

    android:strokeColor="#33FFFFFF" />

<path

    android:fillColor="#00000000"

    android:pathData="M19,49L89,49"

    android:strokeWidth="0.8"

    android:strokeColor="#33FFFFFF" />

<path

    android:fillColor="#00000000"

    android:pathData="M19,59L89,59"

    android:strokeWidth="0.8"

    android:strokeColor="#33FFFFFF" />

<path

    android:fillColor="#00000000"

    android:pathData="M19,69L89,69"
```

```
    android:strokeWidth="0.8"

    android:strokeColor="#33FFFFFF" />

<path

    android:fillColor="#00000000"

    android:pathData="M19,79L89,79"

    android:strokeWidth="0.8"

    android:strokeColor="#33FFFFFF" />

<path

    android:fillColor="#00000000"

    android:pathData="M29,19L29,89"

    android:strokeWidth="0.8"

    android:strokeColor="#33FFFFFF" />

<path

    android:fillColor="#00000000"

    android:pathData="M39,19L39,89"

    android:strokeWidth="0.8"

    android:strokeColor="#33FFFFFF" />

<path

    android:fillColor="#00000000"

    android:pathData="M49,19L49,89"

    android:strokeWidth="0.8"

    android:strokeColor="#33FFFFFF" />

<path

    android:fillColor="#00000000"

    android:pathData="M59,19L59,89"

    android:strokeWidth="0.8"

    android:strokeColor="#33FFFFFF" />

<path

    android:fillColor="#00000000"

    android:pathData="M69,19L69,89"

    android:strokeWidth="0.8"
```

```
            android:strokeColor="#33FFFFFF" />
        <path
            android:fillColor="#00000000"
            android:pathData="M79,19L79,89"
            android:strokeWidth="0.8"
            android:strokeColor="#33FFFFFF" />
    </vector>
```

4.4 ic_launcher_foreground.xml

```
    <vector xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:aapt="http://schemas.android.com/aapt"

    android:width="108dp"

    android:height="108dp"

    android:viewportWidth="108"

    android:viewportHeight="108">

    <path android:pathData="M31,63.928c0,0 6.4,-11 12.1,-13.1c7.2,-2.6 26,-1.4 26,-
    1.4l38.1,38.1L107,108.928l-32,-1L31,63.928z">

        <aapt:attr name="android:fillColor">

            <gradient

                android:endX="85.84757"

                android:endY="92.4963"

                android:startX="42.9492"

                android:startY="49.59793"
```

```
        android:type="linear">

        <item

            android:color="#44000000"

            android:offset="0.0" />  <item

 android:color="#00000000"

    android:offset="1.0" />

      </gradient>

    </aapt:attr>

  </path>

  <path

android:fillColor="#FFFFFF"

android:fillType="nonZero"

android:pathData="M65.3,45.828l3.8,-6.6c0.2,-0.4 0.1,-0.9 -0.3,-1.1c-0.4,-0.2 -0.9,-0.1 -1.1,0.3l-
3.9,6.7c-6.3,-2.8 -13.4,-2.8 -19.7,0l-3.9,-6.7c-0.2,-0.4 -0.7,-0.5 -1.1,-0.3C38.8,38.328 38.7,38.828
38.9,39.228l3.8,6.6C36.2,49.428 31.7,56.028 31,63.928h46C76.3,56.028 71.8,49.428
65.3,45.828zM43.4,57.328c-0.8,0 -1.5,-0.5 -1.8,-1.2c-0.3,-0.7 -0.1,-1.5 0.4,-2.1c0.5,-0.5 1.4,-0.7 2.1,-
0.4c0.7,0.3 1.2,1 1.2,1.8C45.3,56.528 44.5,57.328 43.4,57.328L43.4,57.328zM64.6,57.328c-0.8,0 -1.5,-
0.5 -1.8,-1.2s-0.1,-1.5 0.4,-2.1c0.5,-0.5 1.4,-0.7 2.1,-0.4c0.7,0.3 1.2,1 1.2,1.8C66.5,56.528 65.6,57.328
64.6,57.328L64.6,57.328z"

android:strokeWidth="1"

android:strokeColor="#00000000" />

</vector>
```

4.5 ic_launcher.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<adaptive-icon xmlns:android="http://schemas.android.com/apk/res/android">
 <background android:drawable="@drawable/ic_launcher_background" />
 <foreground android:drawable="@drawable/ic_launcher_foreground" />
 </adaptive-icon>
```

4.6 activity_question.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<adaptive-icon xmlns:android="http://schemas.android.com/apk/res/android">
 <background android:drawable="@drawable/ic_launcher_background" />
 <foreground android:drawable="@drawable/ic_launcher_foreground" />
</adaptive-icon>
```

4.7 color.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
 <color name="gray">#888</color>
 <color name="blue">#0000FF</color>
 <color name="black">#FF000000</color>
 <color name="white">#FFFFFFFF</color>
</resources>
```

4.8 strings.xml

```xml
<resources>
        <string name="app_name">CalendarAppExample</string>
        <string name="back"><![CDATA[<-]]></string>
        <string name="forward"><![CDATA[->]]></string>
    </resources>
```

4.9 themes.xml

```
<resources
xmlns:tools="http://schemas.androi
d.com/tools">
  <!-- Base application theme. -->
               <style
 name="Theme.CalendarAppExampl
                 e"
 parent="Theme.AppCompat.Light.
        NoActionBar">
            </style>
   </resources>t.NoActionBar">
            </style>
          </resources>
```

4.10 MainActivity.java

```
package codewithcal.au.calendarappexample;

import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.GridLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.os.Bundle;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;
import java.time.LocalDate;
import java.time.YearMonth;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
```

CALENDAR APPLICATION

```java
public class MainActivity extends AppCompatActivity implements
CalendarAdapter.OnItemListener
{
    private TextView monthYearText;
    private RecyclerView calendarRecyclerView;
    private LocalDate selectedDate;
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        initWidgets();
        selectedDate = LocalDate.now();
        setMonthView();
    }
    private void initWidgets()
    {
        calendarRecyclerView = findViewById(R.id.calendarRecyclerView);
        monthYearText = findViewById(R.id.monthYearTV);
    }
    private void setMonthView()
    {
        monthYearText.setText(monthYearFromDate(selectedDate));
        ArrayList<String> daysInMonth = daysInMonthArray(selectedDate);
        CalendarAdapter calendarAdapter = new CalendarAdapter(daysInMonth, this);
        RecyclerView.LayoutManager layoutManager = new
GridLayoutManager(getApplicationContext(), 7);
        calendarRecyclerView.setLayoutManager(layoutManager);
        calendarRecyclerView.setAdapter(calendarAdapter);
    }
    private ArrayList<String> daysInMonthArray(LocalDate date)
    {
```

```
        ArrayList<String> daysInMonthArray = new ArrayList<>();

         YearMonth yearMonth = YearMonth.from(date);

        int daysInMonth = yearMonth.lengthOfMonth();

         LocalDate firstOfMonth = selectedDate.withDayOfMonth(1);

         int dayOfWeek = firstOfMonth.getDayOfWeek().getValue();

         for(int i = 1; i <= 42; i++)

         {

            if(i <= dayOfWeek || i > daysInMonth + dayOfWeek)

            {

                daysInMonthArray.add("");

            }

            else

            {

                daysInMonthArray.add(String.valueOf(i - dayOfWeek));

            }

         }

        return  daysInMonthArray;

    }

    private String monthYearFromDate(LocalDate date)

    {

        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("MMMM yyyy");

        return date.format(formatter);

}

    public void previousMonthAction(View view)

    {

        selectedDate = selectedDate.minusMonths(1);

        setMonthView();

    }

    public void nextMonthAction(View view)

    {

selectedDate = selectedDate.plusMonths(1);

        setMonthView();
```

```
      }
      @Override
      public void onItemClick(int position, String dayText)
      {
         if(!dayText.equals(""))
         {
            String message = "Selected Date " + dayText + " " + monthYearFromDate(selectedDate);
            Toast.makeText(this, message, Toast.LENGTH_LONG).show();
         }
      }
}
```

4.11 CalendarViewHolder.java

```
package codewithcal.au.calendarappexample;

import android.view.View;
import android.widget.TextView;
import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

public class CalendarViewHolder extends RecyclerView.ViewHolder implements
View.OnClickListener
{
public final TextView dayOfMonth;
private final CalendarAdapter.OnItemListener onItemListener;
public CalendarViewHolder(@NonNull View itemView, CalendarAdapter.OnItemListener
onItemListener)
{
super(itemView);
dayOfMonth = itemView.findViewById(R.id.cellDayText);
this.onItemListener = onItemListener;
itemView.setOnClickListener(this);
}
```

```java
@Override

public void onClick(View view)

{

onItemListener.onItemClick(getAdapterPosition(), (String) dayOfMonth.getText());

}

}
```

4.12 CalendarAdapter.java

```java
package codewithcal.au.calendarappexample;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;
import java.util.ArrayList;

class CalendarAdapter extends RecyclerView.Adapter<CalendarViewHolder>
{
    private final ArrayList<String> daysOfMonth;
    private final OnItemListener onItemListener;
    public CalendarAdapter(ArrayList<String> daysOfMonth, OnItemListener onItemListener)
    {
        this.daysOfMonth = daysOfMonth;
        this.onItemListener = onItemListener;
    }
    @NonNull
    @Override
    public CalendarViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType)
    {
        LayoutInflater inflater = LayoutInflater.from(parent.getContext());
        View view = inflater.inflate(R.layout.calendar_cell, parent, false);
        ViewGroup.LayoutParams layoutParams = view.getLayoutParams();
        layoutParams.height = (int) (parent.getHeight() * 0.166666666);
        return new CalendarViewHolder(view, onItemListener);
    }
    @Override


    public void onBindViewHolder(@NonNull CalendarViewHolder holder, int position)
    {
        holder.dayOfMonth.setText(daysOfMonth.get(position));
    }
```

```
@Override
  public int getItemCount()
  {
    return daysOfMonth.size();
  }
  public interface  OnItemListener
  {
    void onItemClick(int position, String dayText);
  }
}
```

4.13 ExampleInstrumentedTest.java

```java
package codewithcal.au.calendarappexample;

import android.content.Context;

import androidx.test.platform.app.InstrumentationRegistry;

import androidx.test.ext.junit.runners.AndroidJUnit4;

import org.junit.Test;

import org.junit.runner.RunWith;

import static org.junit.Assert.*;
/**
 * Instrumented test, which will execute on an Android device.
 *
 * @see <a href="http://d.android.com/tools/testing">Testing documentation</a>
 */
@RunWith(AndroidJUnit4.class)
public class ExampleInstrumentedTest {
  @Test
  public void useAppContext() {

// Context of the app under test.
    Context appContext = InstrumentationRegistry.getInstrumentation().getTargetContext();
```

```java
        assertEquals("codewithcal.au.calendarappexample", appContext.getPackageName());
    }
}
```

4.14 ExampleUnitTest.java

```java
package codewithcal.au.calendarappexample;

import org.junit.Test;
import static org.junit.Assert.*;
/**
 * Example local unit test, which will execute on the development machine (host).
 *
 * @see <a href="http://d.android.com/tools/testing">Testing documentation</a>
 */
public class ExampleUnitTest {
    @Test
    public void addition_isCorrect() {
        assertEquals(4, 2 + 2);
    }
}
```

4.15 AndroidManifest.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="codewithcal.au.calendarappexample">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
```

```xml
        android:supportsRtl="true"
        android:theme="@style/Theme.CalendarAppExample">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```
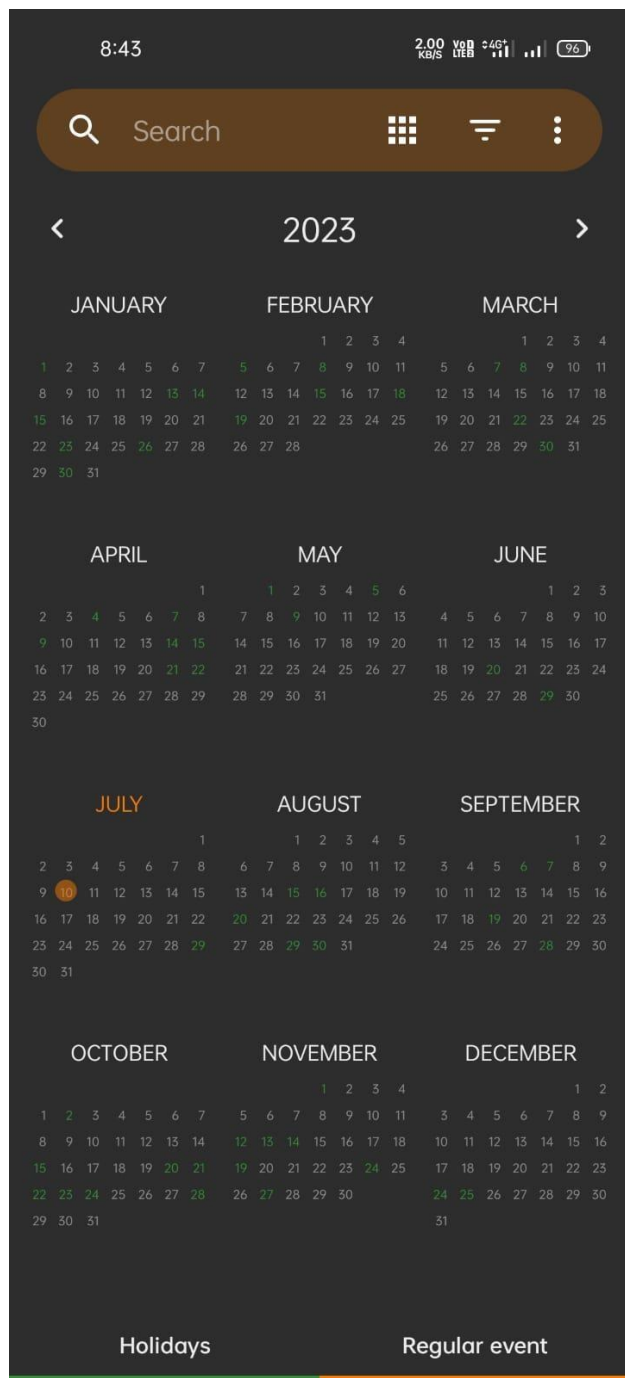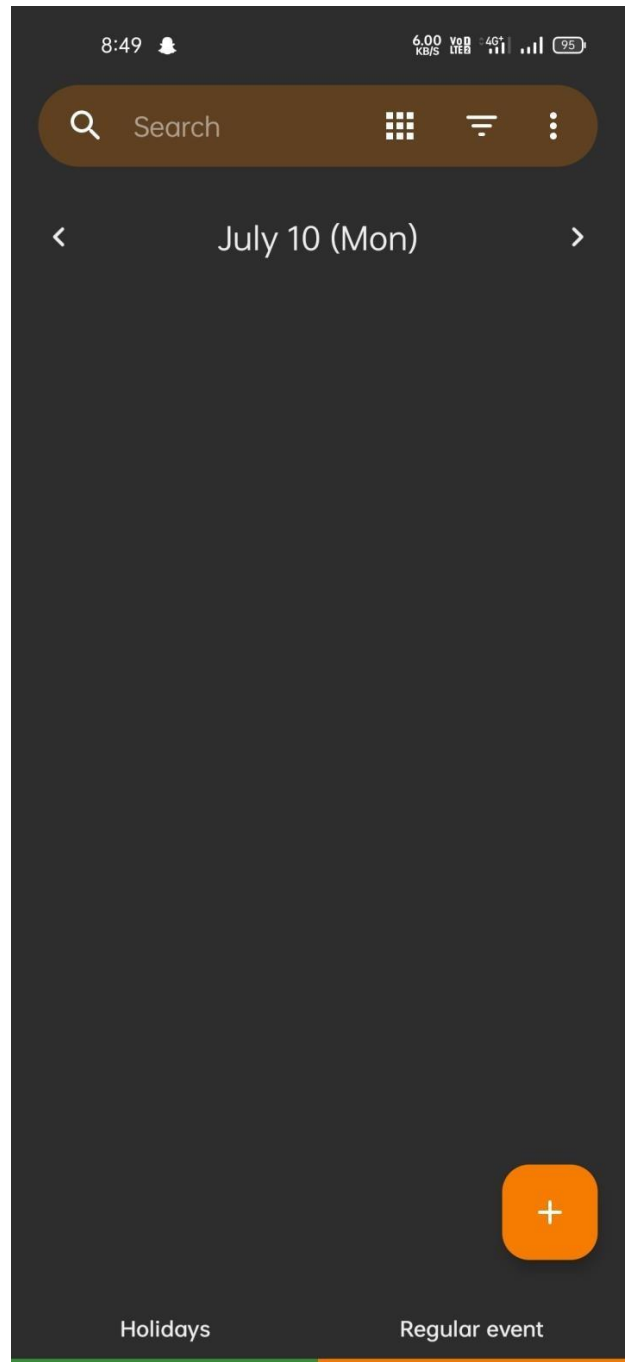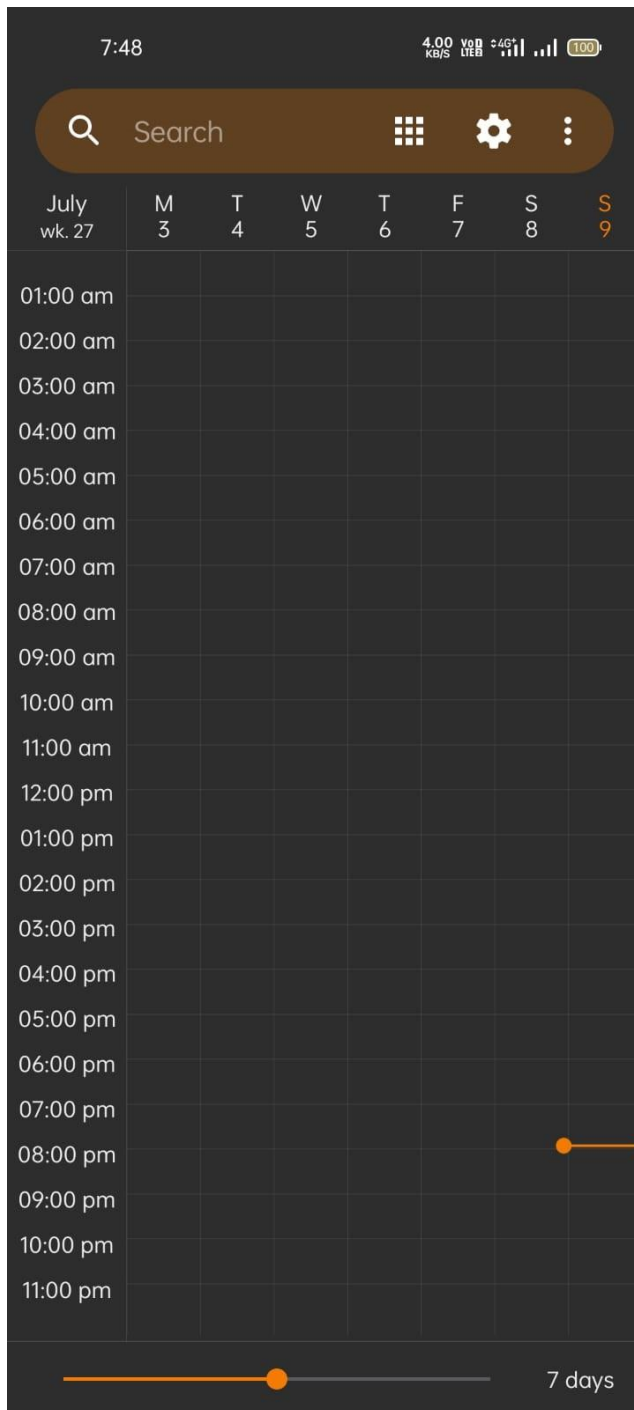
CHAPTER 5:
SNAPSHOTS
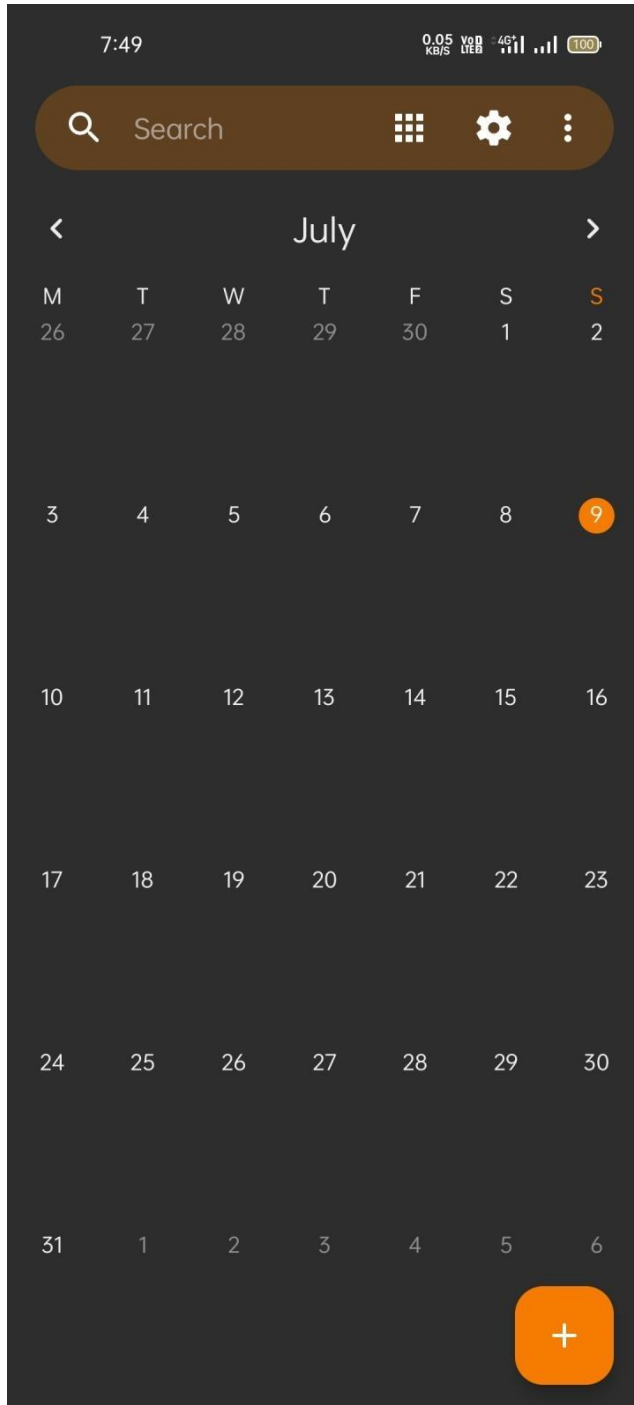
5.1 MONTHLY VIEW PAGE
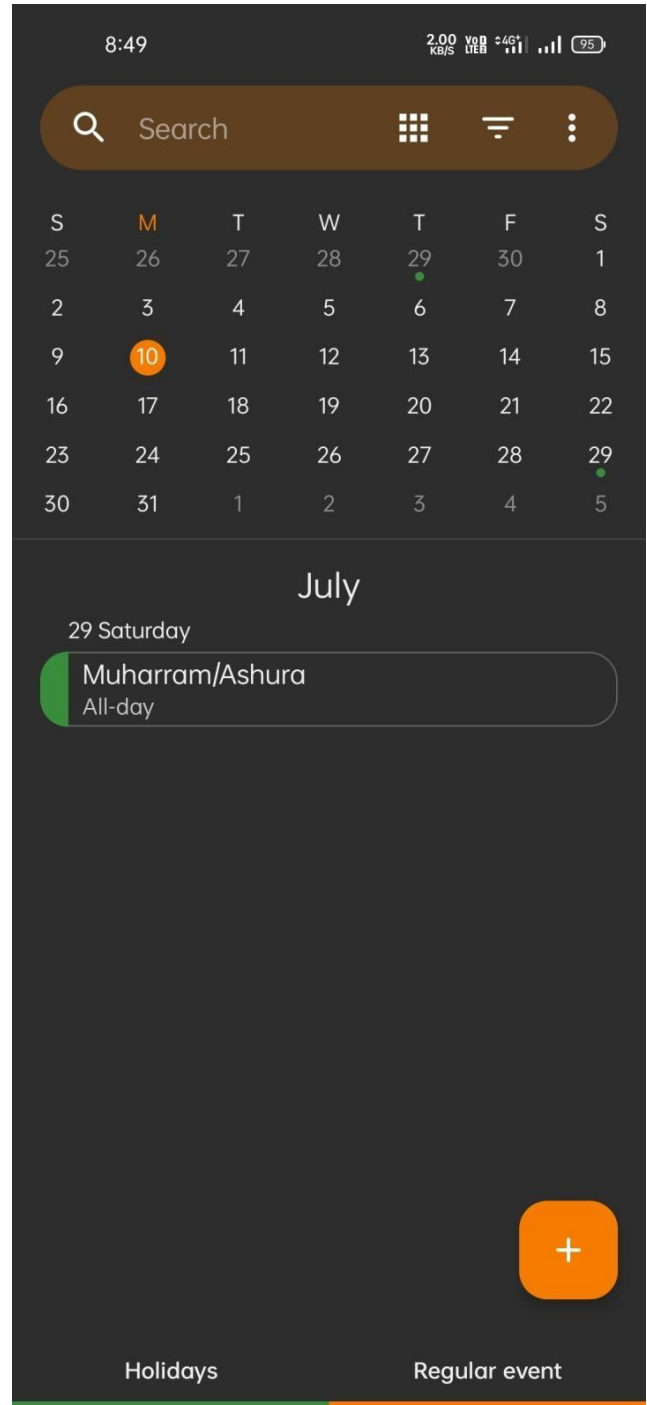
5.2 MONTHLY AND DAILY VIEW PAGE

5.3 YEARLY VIEW PAGE

5.4 DAILY VIEW PAGE
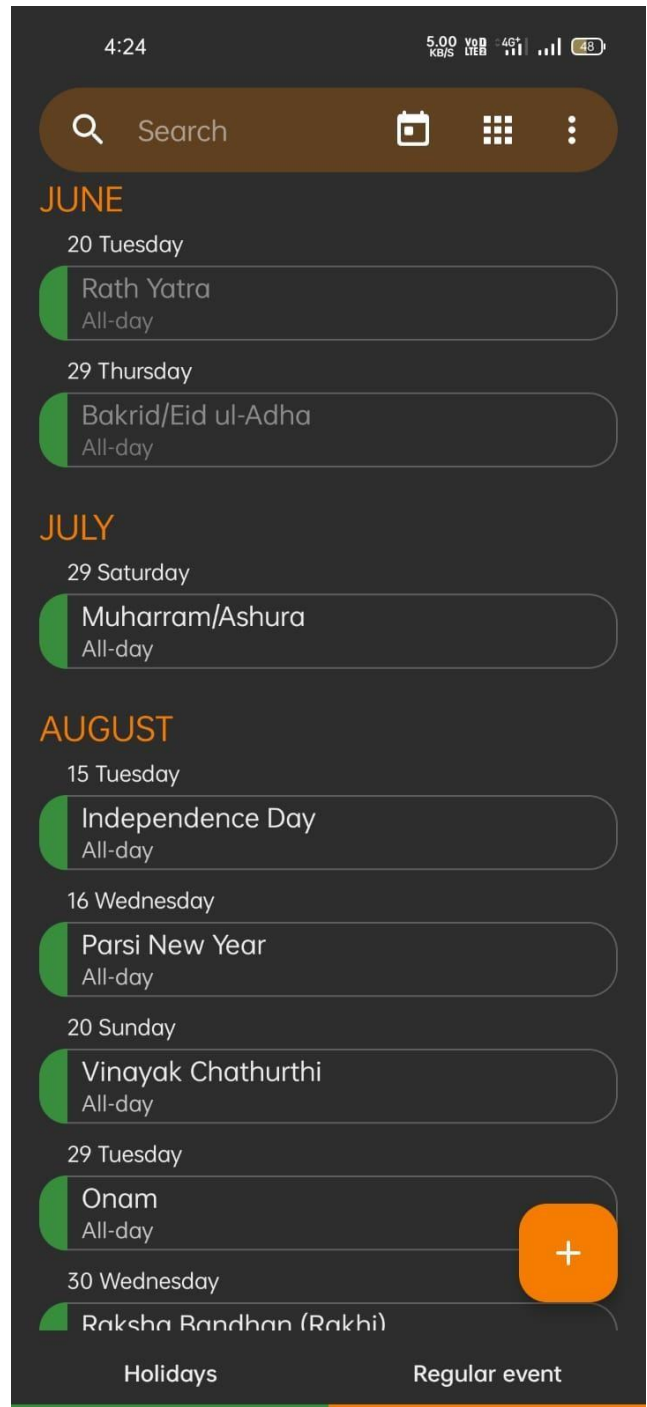
5.5 TIME TABLE PAGE
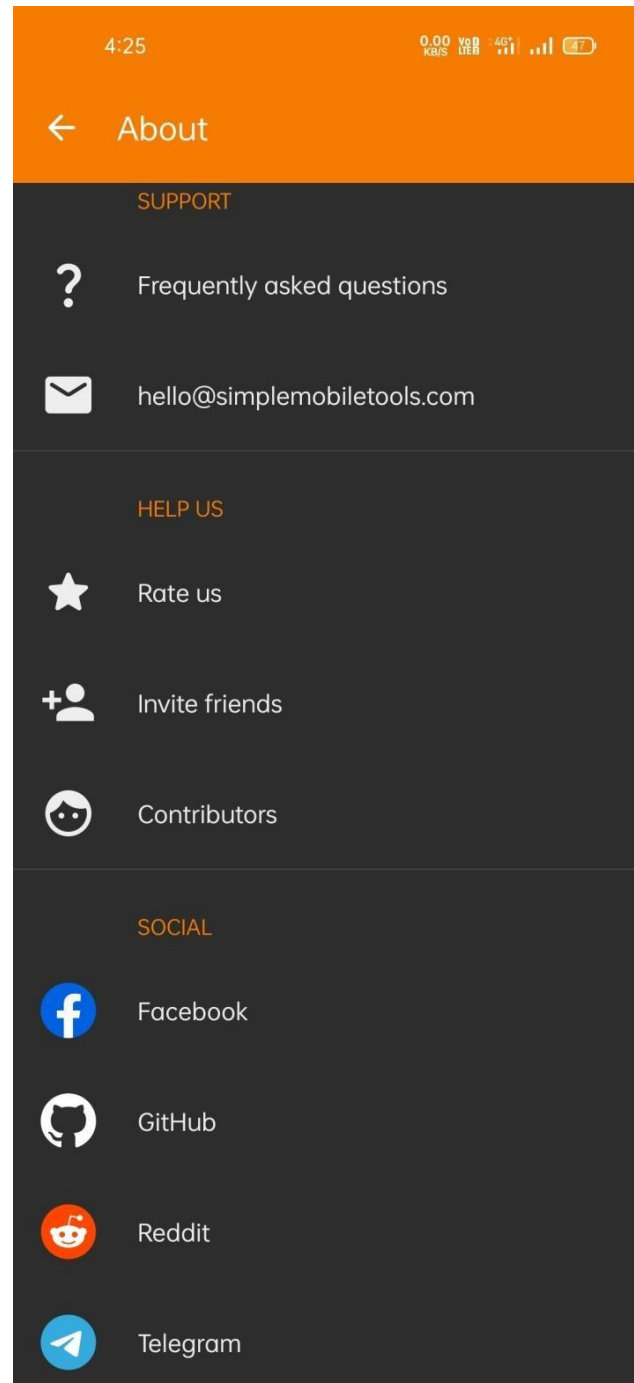

5.6 EVENTS LIST PAGE

5.7 TASK ADDING PAGE

5.8 EVENT ADDING PAGE
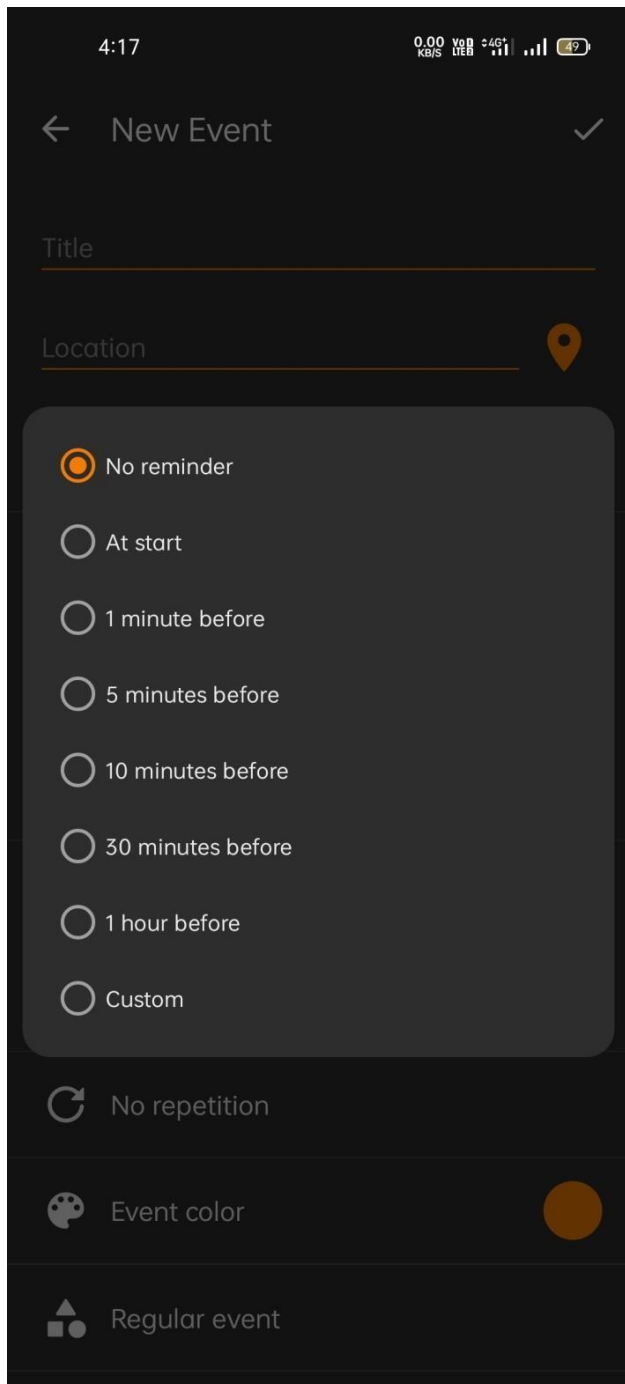
5.9 SET REMINDER TIME

5.10 ABOUT

CONCLUSION

The development of the Android calendar app using Android Studio has been successfully accomplished. The app offers users a convenient and intuitive platform to manage their events and appointments. Through features such as event creation, event reminders, calendar synchronization, and multiple views, the app enhances the productivity and organization of users' schedules.

Throughout the project, various challenges were encountered, including synchronization complexities, user interface design, and compatibility issues. However, with meticulous testing, debugging, and iterative improvements, these challenges were overcome, resulting in a stable and reliable application.

Looking ahead, there are several potential areas for future improvement. Implementing cloud synchronization capabilities would enable users to access their calendar events across multiple devices seamlessly. Additionally, offering customization options for themes, color schemes, and event display preferences would enhance the app's personalization. Collaborative features and integration with third-party services like Google Calendar or Outlook would further extend the app's functionality and user experience.

Overall, the Android calendar app project has successfully achieved its objectives of delivering a user-friendly and feature-rich application. As user feedback is gathered and technological advancements are made, the app can continue to evolve and cater to the changing needs of users in the future.