A MINI PROJECT REPORT

Submitted by

MANIKANDAN **220701159**

In partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE

(AUTONOMOUS) THANDALAM

CHENNAI-602105 MAY-2025

BONAFIDE CERTIFICATE

Certified that this project report "**QUIZ APP**" is the bonafide work of **MANIKANDAN S(220701159)** who carried out the project work under my supervision.

Submitted for the Practical Examination held on

SIGNATURE

Mr. V.Karthick
Associate professor
Computer Science and Engineering,
Rajalakshmi Engineering College,
(Autonomous),
Thandalam, Chennai - 602 105

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ABSTRACT

The **Quiz App Android application** is an interactive educational tool developed using **Android Studio with Kotlin**. It enables users to test their general knowledge through a series of multiple-choice questions. Each question is displayed with four answer options, and users can select and submit their answer to receive immediate feedback. The app dynamically moves to the next question after each submission, ensuring an engaging quiz experience. Designed with a simple and intuitive user interface using the traditional XML layout system, the application is accessible to users of all ages. The quiz content is stored locally within the app, allowing it to function smoothly without an internet connection. It promotes learning and self-assessment by offering instant validation of answers. The project demonstrates core Android development principles such as activity lifecycle management, event handling, data class usage, and UI interaction through buttons and text views. This Quiz App serves as an excellent beginner project for those looking to understand the fundamentals of mobile app development and Kotlin programming.

TABLE OF CONTENTS

1. INTRODUCTION

1.1 IMPLEMENTATION

2. SYSTEM SPECIFICATION

2.1 HARDWARE SPECIFICATION

2.2 SOFTWARE SPECIFICATION

3. SOURCE CODE

4. SNAPSHOTS

5. CONCLUSION

CHAPTER 1

1. INTRODUCTION:

Quizzes are a fun and effective way to reinforce learning and assess knowledge across various subjects. The **Quiz App Android application** is developed to fulfill this purpose by offering a simple, interactive, and accessible platform for users to test their knowledge. Built using Kotlin in Android Studio, this application presents users with multiple-choice questions, enabling them to select and submit answers to receive immediate feedback. With a clean and intuitive user interface designed using traditional XML layouts, the app emphasizes ease of use and seamless navigation. The quiz content is hardcoded into the app, eliminating the need for internet connectivity and ensuring uninterrupted usage. By tracking correct and incorrect answers, the app fosters self-assessment and continuous learning. This project provides an excellent foundation for beginners to explore key concepts in Android development, including activity lifecycle management, UI component interaction, and basic data handling using Kotlin.

1.1 IMPLEMENTATION:

The Quiz App application was implemented using the Kotlin programming language in Android Studio. The design is based on a straightforward architecture, utilizing a single activity (`MainActivity`) to control the entire quiz flow. The layout is created using XML, incorporating UI components such as `TextView` for displaying questions and `Button` widgets for presenting multiple-choice options and submitting answers.

The application begins by initializing a list of questions, each defined as an instance of a custom data class called `Question`, which includes the question text, four options, and the correct answer index. The app displays one question at a time, and users can select an answer by tapping on one of the option

buttons. The selected answer is tracked using a variable, and upon clicking the submit button, the app checks whether the answer is correct and provides immediate feedback using `Toast` messages.

After each submission, the app proceeds to the next question in the list until the quiz is completed. All UI interactions and logic are managed within the `onCreate` lifecycle method and helper functions. The application does not use persistent storage or navigation components, making it lightweight and ideal for offline use. The overall implementation focuses on clarity, modularity, and enhancing user interactivity through real-time feedback.

CHAPTER 2

SYSTEM SPECIFICATIONS

2.1 HARDWARE SPECIFICATION:

PROCESSOR - Intel® core™ i5-6006U @ 2.00 GHz

RAM - 4GB

OPERATING SYSTEM - Microsoft Windows 11

HARD DISK - 850 GB of free space

SYSTEM TYPE - 64-bit operating system , x64 based processor

2.2 SOFTWARE SPECIFICATION:

PROGRAMMING LANGUAGE:
kotlin

OPERATING SYSTEM : Microsoft

Windows 11 SOFTWARE : Android Studio

CHAPTER 3

Source Code

AndroidManifest.xml

```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="mad_lab_pro_f"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.Mad_lab_pro_f"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

MainActivity.kt

```kotlin
package com.example.mad_lab_pro_f

import android.os.Bundle
import android.widget.Button
import android.widget.TextView
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
```

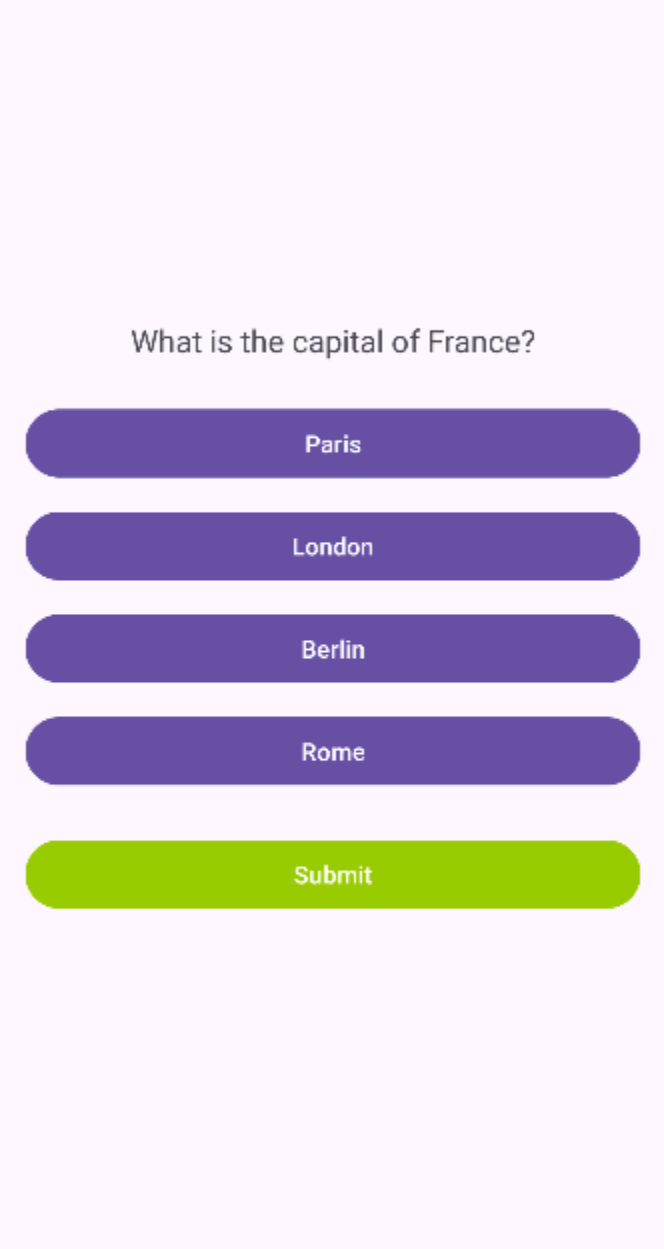
```
class MainActivity : AppCompatActivity() {

    private val questions = listOf(
        Question("What is the capital of France?", "Paris", "London", "Berlin", "Rome", 1),
        Question("What is 2 + 2?", "3", "4", "5", "6", 2),
        Question("Who wrote 'To Kill a Mockingbird'?", "Harper Lee", "J.K. Rowling", "Ernest Hemingway", "Mark Twain", 1)
    )
    private var currentQuestionIndex = 0
    private var selectedAnswer = -1 // Track the selected option (1-4)

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        // Get references to the views
        val questionText = findViewById<TextView>(R.id.questionText)
        val option1 = findViewById<Button>(R.id.option1)
        val option2 = findViewById<Button>(R.id.option2)
        val option3 = findViewById<Button>(R.id.option3)
        val option4 = findViewById<Button>(R.id.option4)
        val submitBtn = findViewById<Button>(R.id.submitBtn)

        // Display the first question
        displayQuestion(questionText, option1, option2, option3, option4)
```

```
    // Set click listeners for each option
    option1.setOnClickListener { selectedAnswer = 1 }
    option2.setOnClickListener { selectedAnswer = 2 }
    option3.setOnClickListener { selectedAnswer = 3 }
    option4.setOnClickListener { selectedAnswer = 4 }

    submitBtn.setOnClickListener {
        if (selectedAnswer != -1) {
            if (questions[currentQuestionIndex].correctOption == selectedAnswer) {
                Toast.makeText(this, "Correct!", Toast.LENGTH_SHORT).show()
            } else {
                Toast.makeText(this, "Incorrect!", Toast.LENGTH_SHORT).show()
            }
            // Move to the next question
            currentQuestionIndex++
            if (currentQuestionIndex < questions.size) {
                selectedAnswer = -1 // Reset selection for the next question
                displayQuestion(questionText, option1, option2, option3, option4)
            } else {
                Toast.makeText(this, "Quiz Completed!", Toast.LENGTH_SHORT).show()
            }
        } else {
            Toast.makeText(this, "Please select an answer", Toast.LENGTH_SHORT).show()
```

```
private fun displayQuestion(
    questionText: TextView, option1: Button, option2: Button,
    option3: Button, option4: Button
) {
    val currentQuestion = questions[currentQuestionIndex]
    questionText.text = currentQuestion.question
    option1.text = currentQuestion.option1
    option2.text = currentQuestion.option2
    option3.text = currentQuestion.option3
    option4.text = currentQuestion.option4
}


private fun getSelectedAnswer(
    option1: Button, option2: Button, option3: Button, option4: Button
): Int {
    return when {
        option1.isPressed -> 1
        option2.isPressed -> 2
        option3.isPressed -> 3
        option4.isPressed -> 4
        else -> -1
    }
```

CHAPTER 4

SNAP SHOTS



Who wrote 'To Kill a Mockingbird'?

Harper Lee

J.K. Rowling

Ernest Hemingway

Mark Twain

Submit

CHAPTER 5

CONCLUSION

The Quiz App application successfully meets its objective of providing an interactive platform for users to test their general knowledge through a series of multiple-choice questions. With a clean and intuitive interface, users can easily navigate through questions, select their answers, and receive instant feedback, making the quiz experience engaging and educational. The project effectively demonstrates core Android development concepts such as activity lifecycle management, UI handling, and event-driven programming using Kotlin. Designed as a lightweight, offline-capable app, it ensures accessibility without the need for internet connectivity. Overall, this project not only offers a functional and enjoyable quiz experience but also lays a solid foundation for future enhancements, such as integrating a score tracker, category selection, or storing quiz data using local databases or cloud services—opening doors to more advanced development practices in Android.