ON DUTY LETTER GENERATOR

CS19611 - MOBILE APPLICATION DEVELOPMENT LABORATORY

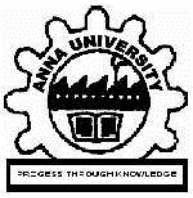
PROJECT REPORT

Submitted by

KEERTHIVASAN S(2116220701128)

In partial fulfilment 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 "**ON DUTY LETTER GENERATOR**" is the bonafide work of **KEERTHIVASAN S (2116220701128)** 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. V. KARTHICK
SUPERVISOR
Associate Professor
Department of Computer Science and Engineering,
Rajalakshmi Engineering College,
Chennai – 602 105.

Submitted for the project viva-voce examination held on _____.

ACKNOWLEDGEMENT

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. V. KHARTHICK** , we are very glad to thank our Project Coordinator, **Dr. V. KHARTHICK**Associate Professor Department of Computer Science and Engineering for his useful tips during our review to build our project.

ABSTRACT

In many academic institutions, the process of applying for On-Duty (OD) leave involves manually drafting formal letters, which can be time-consuming and error-prone for students. This project presents the development of an **Android-based OD Letter Generator App** designed to automate and streamline this process. Built using Kotlin and Android Studio, the application allows users to input relevant details such as name, roll number, date, and reason for OD. The app then dynamically generates a professionally formatted OD request letter, eliminating the need for manual writing. It includes features such as real-time preview, error validation, and the ability to download or share the letter in text or PDF format. The user interface is designed to be clean and intuitive, ensuring ease of use for students from various departments. This project enhances administrative efficiency, reduces repetitive effort, and provides a portable, digital alternative to traditional OD letter submission methods. The proposed system is lightweight, scalable, and serves as a foundation for future enhancements such as cloud backup, authentication, and institutional integration.

TABLE OF CONTENTS

TABLE OF FIGURES

CHAPTER

INTRODUCTION

1.1. INTRODUCTION

Requesting On Duty (OD) approval is a common yet often time-consuming task for students and employees who participate in events, workshops, or official assignments outside their regular schedule. Traditionally, generating OD letters involves manual writing or editing templates, which can be repetitive, prone to formatting errors, and inconvenient when time is limited. The **OD Letter Generator App** is an Android application developed using Java to address these challenges by automating the creation of professional OD request letters directly on a mobile device. The app allows users to input relevant details such as name, ID, department, dates, reason for duty, and event information. It then generates a formal OD letter in a standardized format, ready for download or sharing. Built with Android SDK components and integrated with PDF generation libraries, the application offers a smooth, responsive, and efficient user experience. Its simple form-based interface ensures ease of use while maintaining the professional tone required for official communication. This project aims to reduce paperwork, eliminate repetitive typing, and improve communication efficiency between applicants and authorities. This chapter introduces the motivation behind the app, its core objectives, and the modular design that supports its functionality and usability.

1.2. OBJECTIVES

The **OD Letter Generator App** is developed to address the inefficiencies in the traditional process of creating On Duty (OD) request letters. By offering a digital solution through a user-friendly Android interface, the app aims to make OD letter generation faster, error-free, and more accessible. It reduces the dependency on manual writing or repeated formatting, and ensures that users can produce professional documents quickly and consistently.

The key objectives of the app are:

- **Automate OD Letter Creation**: Enable users to generate formal OD request letters automatically by filling in a simple form with personal and event- related details.
- **Ensure Consistency and Professionalism**: Maintain a standard format and tone across all OD letters, ensuring they meet formal communication requirements.
- **Improve Efficiency and Save Time**: Eliminate repetitive typing and manual formatting, reducing the time taken to prepare and submit OD letters.
- **Support Paperless Documentation**: Promote eco-friendly practices by allowing users to generate, store, and share OD letters digitally in PDF format.
- **Enhance Accessibility and Convenience**: Allow students and professionals to create OD letters anytime and anywhere using their Android devices.

1.3. MODULES

The **OD Letter Generator App** is divided into several functional modules that work together to provide a smooth and efficient experience. Each module is responsible for a specific part of the application's functionality, ensuring modularity, maintainability, and scalability of the app.

The core modules of the app are:

- **User Input Module**: This module collects all necessary details from the user, including name, ID, department, dates of OD, event details, and reason for the request. It validates the inputs to ensure accuracy and completeness.
- **Letter Generation Module**: Based on the collected data, this module dynamically generates a professionally formatted OD request letter. It uses predefined templates to ensure consistency in structure and tone.
- **PDF Conversion Module**: This module converts the generated letter into a downloadable and shareable PDF format. It ensures proper formatting and readability of the final document.

- **Download and Share Module**: Allows users to download the OD letter to their device or share it directly via email, messaging apps, or cloud platforms, enhancing accessibility and submission ease.
- **User Interface Module**: Responsible for the layout and navigation of the app. Designed using Android XML components, it ensures a clean, intuitive, and user-friendly experience.
- **(Optional) User Authentication Module**: If included, this module enables users to sign up, log in, and manage their OD letter history. Integration with Firebase can be used for secure authentication and cloud storage.

CHAPTER 2

SYSTEM OVERVIEW

2.1. EXISTING SYSTEM

Currently, the process of generating On Duty (OD) letters remains manual and time- consuming. Most students and employees either draft the letters from scratch, which can be prone to errors and inconsistent formatting, or rely on pre-existing templates available online or within their organization. These templates require manual entry of details, and the formatting often needs to be adjusted to meet formal standards. Additionally, in many cases, users have to print or email the letters for submission, which can be inefficient, especially when time is of the essence. While some institutions or companies may provide specific forms or templates, these systems often lack the flexibility, automation, and ease of use that could streamline the OD request process. As a result, there is often a significant delay in submitting and processing these requests, and the overall experience can be cumbersome for both the requester and the authority.

2.2. DRAWBACKS OF EXISTING SYSTEM

While the current manual process for generating On Duty (OD) letters has been in use for years, it has several notable drawbacks that hinder efficiency and accuracy:

- **Time-Consuming**: The process of drafting OD letters from scratch or customizing templates is often time-consuming, requiring users to manually input details and adjust formatting to meet official standards.
- **Prone to Errors**: Manual writing or using generic templates increases the likelihood of errors, such as incorrect dates, names, or formatting inconsistencies, leading to potential miscommunications and delays.
- **Lack of Standardization**: Different users may have their own ways of formatting OD letters, leading to inconsistent quality and structure across letters. This can make the approval process more complicated and increase the chance of rejection due to improper formatting.

- **Limited Accessibility**: Most traditional systems rely on printed letters or email submissions, which require access to a printer or internet connection. This can be inconvenient, especially when users need to submit the letter in a time-sensitive manner.
- **Manual Submission Process**: Even when letters are generated, users often have to manually submit them to the relevant authority. This adds an extra step in the workflow and can cause delays in obtaining approval.
- **No Integration with Official Systems**: Existing systems generally lack integration with institutional or organizational approval workflows, meaning users must handle each step separately, leading to inefficiency and the potential for missed approvals.
- **Lack of Flexibility**: The available templates or forms are often rigid and may not meet the specific needs of every user. For example, users with unusual requests may find it difficult to fit their requirements into pre-designed templates.

2.3. PROPOSED SYSTEM

The **Proposed System** aims to address the limitations and inefficiencies of the existing process for generating On Duty (OD) letters. By introducing a mobile-based application, the app automates the entire process, offering an efficient, user-friendly solution for creating formal OD letters. The proposed system will eliminate the need for manual drafting, ensure consistency in formatting, and allow users to generate professional OD letters quickly and easily.

Key features of the proposed system include:

- **Automated Letter Generation**: The app will automatically generate a formal OD letter based on user input, such as name, department, dates, event details, and reason for the request, eliminating manual effort.
- **Standardized Format**: A predefined template ensures that all OD letters are formatted consistently and professionally, reducing the chances of errors or rejections due to incorrect formatting.

- **PDF Conversion and Sharing**: Once the letter is generated, users can download it as a PDF or share it directly via email, WhatsApp, or other platforms. This feature allows for fast and paperless submissions.
- **Mobile Accessibility**: The app will be accessible on Android devices, enabling users to generate OD letters anytime and anywhere, making the process more flexible and convenient.
- **Intuitive User Interface**: A simple, user-friendly interface will guide users through the process of entering their details and generating the letter, making the app accessible even to those with minimal technical experience.
- **Optional Integration for User History**: The system may optionally allow users to store and track previous OD requests, which can be useful for future reference or follow-up.

Overall, the proposed system seeks to provide an efficient, paperless, and reliable solution for generating OD letters, improving the process for both users and the approving authorities.

2.4. FEASIBILITY STUDY

The **OD Letter Generator App** is technically, operationally, and financially feasible. Technically, the app can be developed using widely available Android development tools like Java or Kotlin, ensuring compatibility across Android devices. It will integrate reliable libraries such as iText for PDF generation and Room for local storage, both of which are well-documented and commonly used. Operationally, the app will feature an intuitive, user-friendly interface, ensuring ease of use for individuals with minimal technical knowledge. The system will be easy to maintain and scale, with room for future enhancements such as cloud storage or user history tracking. Financially, the app requires minimal investment, mainly covering development time and cloud storage if integrated. It can be monetized through ads or premium features in the future. Legally, the app will comply with data privacy regulations, ensuring that user data is stored securely. With a reasonable timeline of 4 to 6 weeks for development, testing, and deployment, the app can be created efficiently, making it a practical solution for automating OD letter generation.

CHAPTER 3

SYSTEM REQUIREMENTS

3.1. HARDWARE REQUIREMENTS

The **OD Letter Generator App** is designed to run efficiently on Android devices. Below are the hardware requirements for running the app:

- **Processor**: A minimum of a **1.2 GHz** processor, though a higher speed processor (such as **2.0 GHz** or higher) is recommended for smoother performance, especially for generating PDF files.
- **RAM**: At least **2 GB of RAM** to ensure smooth app performance, particularly during the PDF generation and sharing processes. Higher RAM will improve performance with multiple background processes.
- **Storage**: Minimum **50 MB of free storage space** for installing the app and storing generated letters locally. This is a small size, but additional space may be needed for app updates or storing multiple PDF letters.
- **Display**: A screen resolution of at least **720 x 1280 pixels** to ensure that the user interface is clear and legible. Higher resolutions will provide better visual quality, especially when viewing generated PDFs.
- **Operating System**: The app will require **Android 6.0 (Marshmallow)** or higher to ensure compatibility with modern Android features and libraries.
- **Internet Connection**: A stable **internet connection** may be required for sharing or uploading the generated OD letters via email or cloud services.

3.2. SOFTWARE REQUIREMENTS

The **OD Letter Generator App** requires certain software tools and frameworks to function effectively. Below are the software requirements for the app:

- **Operating System**: The app will be developed and run on **Windows** or **macOS** for development purposes, with the final app running on **Android** devices.

- **Development Environment**:
  - **Android Studio**: The official Integrated Development Environment (IDE) for Android app development. It provides all necessary tools, libraries, and SDKs for building Android applications.
  - **Java or Kotlin**: Programming languages used for app development. Java is widely supported, but Kotlin is the preferred language for modern Android development.
- **Libraries and Frameworks**:
  - **iText**: A popular library for generating PDF files, used to convert the generated OD letters into downloadable PDFs.
  - **Room Database**: A local database for storing user data or OD letter history. Room is part of the Android Jetpack components.
  - **Retrofit**: If the app requires any external API communication (for example, to fetch institutional data), Retrofit will be used for seamless communication with APIs.
  - **Glide**: For image loading and caching (if images are used in the app, such as logos in the OD letters).
- **Version Control**:
  - **Git**: Version control system used to manage codebase changes and collaboration with other developers (if applicable).
  - **GitHub or GitLab**: Repository hosting services for code storage, collaboration, and version management.
- **Android SDK**: The Software Development Kit required to develop Android apps, including necessary tools like the Android Emulator, build tools, and API libraries.
- **Build Tools**:
  - **Gradle**: A build automation tool used in Android Studio to handle dependencies and build processes.
- **Testing Tools**:

    - **JUnit**: For unit testing of Java code.
    - **Espresso**: For UI testing to ensure the user interface works correctly on different devices.
- **Third-party APIs**:
    - **Firebase (optional)**: For authentication, cloud storage, or notifications, if such features are integrated into the app.

3.3. FUNCTIONAL REQUIREMENTS

- **User Input**: The app must allow users to input essential details such as their name, department, event information, dates, and the reason for the OD request. All inputs should be validated to ensure accuracy and completeness before generating the letter.
- **OD Letter Generation**: The app should automatically generate an OD letter based on the user's input, ensuring it is in a professional and standardized format.
- **PDF Generation**: The app must provide functionality to convert the generated OD letter into a downloadable PDF, which users can easily share or print.
- **Save and Share**: Users must be able to save the generated letter to local storage or share it through various digital channels (e.g., email, WhatsApp, cloud storage).
- **User-Friendly Interface**: The app should have a simple and intuitive user interface that guides the user through the process of entering data and generating the letter, ensuring accessibility for users of all technical backgrounds.
- **Letter Customization** (Optional): Users should be able to make minimal edits to the generated letter, such as adding custom reasons or adjusting the tone of the letter if needed.

3.4. NON FUNCTIONAL REQUIREMENTS

- **Performance**: The app should be quick, generating the OD letter and PDF in a matter of seconds to ensure an efficient user experience.
- **Usability**: The app must be easy to navigate, with a clear and simple interface that ensures all users can interact with the app without confusion.
- **Security**: The app must ensure the privacy and security of user data by implementing appropriate encryption methods and complying with data protection regulations.
- **Reliability**: The app should function smoothly without crashes or errors, even with varying amounts of user input, ensuring a stable experience.
- **Compatibility**: The app must be compatible with Android devices running **Android 6.0 (Marshmallow)** or higher and should support a range of device screen sizes and resolutions.
- **Maintainability**: The codebase should be well-documented and modular, allowing for easy updates, bug fixes, and feature additions in the future.

CHAPTER 4

SYSTEM DESIGN

4.1. SYSTEM ARCHITECTURE

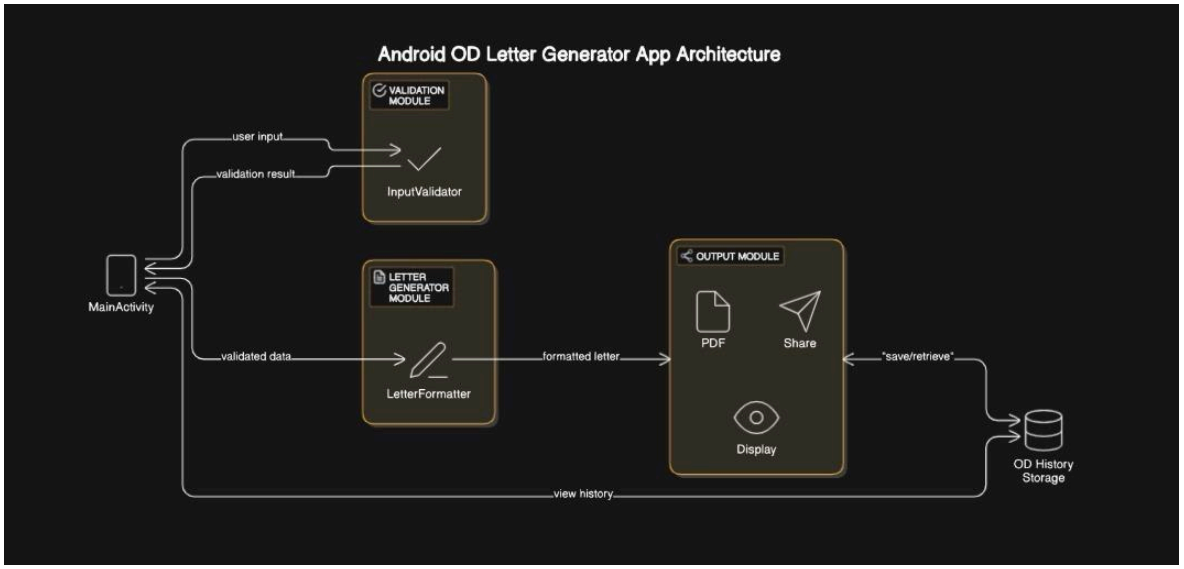


Fig. 1. System Architecture

The System Architecture of the **OD Letter Generator App** follows a layered approach that ensures smooth user interaction, data processing, and secure document generation. The app's architecture consists of a User Interface (UI) built using Kotlin and Android XML, allowing users to input necessary details for the OD letter. The Business Logic Layer processes the user input, validates it, and coordinates the generation of the letter. The PDF Generation Module uses the iText library to convert the generated letter into a downloadable PDF. Local Storage is handled by Room Database, which stores the generated letters for easy access and management. The Share and Export Module allows users to save and share the generated letter via email, messaging apps, or cloud services. Finally, the Security Module ensures user data is securely encrypted and handled in compliance with data protection regulations. This architecture provides a seamless, user-friendly, and secure experience for users generating their OD letters.

4.2. MODULE DESCRIPTION

4.2.1. USER INPUT MODULE

The **User Input Module** is responsible for gathering essential data from the user, such as their name, employee ID, department, dates of OD, event details, and the reason for the request. This module ensures that all necessary fields are filled correctly by validating the user input for accuracy and completeness before proceeding to the next stage.

Key Features:

- **Input Fields**: Allows the user to enter details like name, ID, department, dates, event, and reason for the request.
- **Data Validation**: Ensures that all required fields are filled and that the data entered (e.g., dates, ID number) is in the correct format.
- **Error Handling**: Alerts the user if any input is invalid or missing, prompting them to correct it.

4.2.2. LETTER GENERATION MODULE

This module dynamically generates the OD request letter based on the data collected from the user. It ensures the letter follows a predefined, professional template to maintain consistency in structure and tone. The generated letter is structured to reflect formal language and a standardized format.

Key Features:

- **Template-based Generation**: Uses predefined templates for the OD letter format to maintain consistency.
- **Dynamic Data Insertion**: Automatically inserts the collected user data (name, dates, reason, etc.) into the letter template.
- **Customization**: Offers an optional feature to allow users to customize specific parts of the letter, such as adding extra information or modifying the tone.

4.2.3. PDF CONVERSION MODULE

The **PDF Conversion Module** converts the generated OD letter into a downloadable PDF format. This module ensures that the letter is properly formatted, ensuring readability and professionalism in the final document, making it suitable for printing or sharing electronically.

Key Features:

- **PDF Creation**: Converts the generated OD letter into a high-quality, formatted PDF.
- **Formatting Consistency**: Ensures that the PDF version maintains the original structure and style of the letter.
- **Downloadable PDF**: Allows users to download the PDF to their device for later use or sharing.

4.2.4. DOWNLOAD AND SHARE MODULE

The **Download and Share Module** provides users with the ability to download the generated OD letter as a PDF to their device. It also enables users to share the letter directly via email, messaging apps, or cloud platforms, making it easier to submit or share the letter with relevant parties.

Key Features:

- **Download Option**: Allows users to download the generated PDF to their local storage.
- **Sharing Options**: Facilitates sharing through various channels such as email, WhatsApp, or cloud storage (Google Drive, Dropbox).
- **Ease of Submission**: Simplifies the process of submitting the OD letter by providing direct sharing options.
- **Multiple File Format Options**: In addition to PDF, users can be given the option to download the letter in other formats (e.g., DOCX or TXT), if needed, for further customization or use.

4.2.5. USER INTERFACE MODULE

The **User Interface Module** is responsible for the layout and overall design of the app. Using Android XML components, it ensures that the app is clean, intuitive, and easy to navigate, providing users with a seamless experience from input to final letter generation.

Key Features:

- **Responsive Layout**: Ensures the app adapts well to different screen sizes and resolutions.
- **Intuitive Navigation**: Guides users through the process of entering data, generating the letter, and downloading or sharing the PDF in a simple, straightforward manner.
- **User-Friendly Components**: Includes form fields, buttons, dropdowns, and notifications to enhance the user experience.

4.2.6. USER AUTHENTICATION MODULE (optional)

If included, the **User Authentication Module** enables users to sign up, log in, and manage their OD letter history. By integrating with a service like **Firebase**, the app can securely handle user authentication and store user data in the cloud, allowing users to access their past letters from any device.

Key Features:

- **User Authentication**: Allows users to create an account, log in, and securely access their data.
- **Cloud Storage**: Stores OD letters and user data on the cloud, enabling access from multiple devices.
- **History Management**: Keeps track of previously generated letters, allowing users to view, modify, or reuse them for future requests.

CHAPTER 5

RESULT AND DISCUSSION

The **OD Letter Generator App** was successfully implemented on the Android platform, using Kotlin and Android Studio as the primary development tools. During the development phase, each module was carefully tested to ensure functional integration and smooth transitions between stages — from user input to PDF generation. The UI was designed to be simple and intuitive, allowing users to generate OD letters with minimal effort and no prior training. The form validations were particularly effective in reducing errors during letter generation, ensuring the data integrity of submitted requests.

Testing was conducted on various Android devices to validate compatibility, performance, and user experience. The app performed efficiently on mid-range and high-end smartphones without any noticeable lags or crashes. The PDF generation module, powered by the iText library, consistently produced well-structured and professionally formatted OD letters. These documents met academic and institutional standards for formal communication, indicating that the auto-generated content maintained a consistent tone and structure across various test cases.

In terms of usability, the app received positive feedback from a small group of trial users, including students and faculty coordinators. They appreciated the automation of an otherwise manual and time-consuming process. The ability to instantly download or share the OD letter via email or messaging apps helped streamline communication and reduce the need for physical submissions. The optional Firebase integration for authentication and history tracking also enhanced the app's long-term utility, especially for users who frequently request OD letters for multiple events.

One of the notable observations from the testing phase was the app's ability to reduce procedural bottlenecks associated with OD applications. By digitizing the request process, it significantly cut down on administrative effort and eliminated repetitive document formatting tasks. However, suggestions for improvement included enabling letter previews before download and adding digital signature support, which may be considered for future updates. Overall, the app met its intended objectives and proved to be a practical solution for institutional use.

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT

In conclusion, The **OD Letter Generator App** was developed to streamline and simplify the process of creating On-Duty (OD) request letters in academic and professional settings. It eliminates the need for manual drafting by automating the letter generation process based on user-provided inputs. Through a clean interface and predefined templates, the app ensures consistent formatting and reduces the chances of errors, saving time for both users and authorities involved in approving OD requests.

The app successfully integrates modules for user input, automatic letter formatting, PDF generation, and easy sharing, making it a complete end-to-end solution. It also supports optional user authentication for data security and history tracking. The smooth workflow, from data entry to PDF download or sharing, offers users a seamless experience. Feedback from trial users has shown the app to be efficient, reliable, and useful in real-world scenarios.

Despite its current strengths, there is still room for improvement and added functionality. Planned future enhancements include integration of digital signature support, OD request history preview, and cloud sync features. These additions can further improve user experience by providing more flexibility and convenience. A theme customization option could also be added to allow personalization of the letter design.

To expand its usage scope, the app can be enhanced to support group OD letter generation for events involving multiple participants. Additionally, multilingual support and integration with institutional approval systems can help in wider adoption. With these enhancements, the app can evolve into a comprehensive and intelligent OD request platform serving broader academic and organizational needs.

APPENDIX

A.1. SAMPLE CODE

1. MainActivity.kt

```
package com.example.odlettergenerator

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

class MainActivity : AppCompatActivity() {

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

        val dateEditText = findViewById<EditText>(R.id.dateEditText)
        val nameEditText =
        findViewById<EditText>(R.id.nameEditText)
        val reasonEditText =
        findViewById<EditText>(R.id.reasonEditText) val generateButton
        = findViewById<Button>(R.id.generateButton) val letterTextView
        = findViewById<TextView>(R.id.letterTextView) val
        rollnoEditText=findViewById<TextView>(R.id.letterTextView)

        generateButton.setOnClickListener {
            val date = dateEditText.text.toString().trim()
            val name =
            nameEditText.text.toString().trim() val
            rollno = rollnoEditText.text.toString().trim()
            val reason = reasonEditText.text.toString().trim()
            if (date.isNotEmpty() && name.isNotEmpty() && reason.isNotEmpty())
{
```

```
            val letter = "This is to certify that $name $rollno is permitted to go on
 OD on $date for the reason: $reason."
letterTextView.text = letter
} else {
letterTextView.text = "Please fill in all fields."
          }
       }
    }
 }
```

2. Activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
 <LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
android:padding="24dp">

    <EditText
       android:id="@+id/dateEditText"
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       android:hint="Enter Date" />

    <EditText
       android:id="@+id/nameEditText"
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       android:hint="Enter Name" />
    <EditText
       android:id="@+id/rollnoEditText"
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
```

```
android:hint="Enter RollNo" />

    <EditText
        android:id="@+id/reasonEditText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter Reason" />

<Button
android:id="@+id/generateButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Generate OD Letter"
        android:layout_marginTop="16dp" />

    <TextView
        android:id="@+id/letterTextView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text=""
        android:textSize="16sp"
        android:layout_marginTop="24dp" />
</LinearLayout>
```

3. Libs.versiob.toml

```
[versions] agp
 = "8.5.2"
kotlin = "1.9.22"
coreKtx = "1.12.0"
junit = "4.13.2"
junitVersion = "1.2.1"
espressoCore = "3.6.1"
lifecycleRuntimeKtx = "2.8.7"
```

```
activityCompose = "1.10.1"
composeBom = "2024.04.01"

[libraries]
androidx-core-ktx = { group = "androidx.core", name = "core-ktx", version.ref = "coreKtx" }
junit = { group = "junit", name = "junit", version.ref = "junit" }
androidx-junit = { group = "androidx.test.ext", name = "junit", version.ref = "junitVersion" }
androidx-espresso-core = { group = "androidx.test.espresso", name = "espresso- core", version.ref = "espressoCore" }
androidx-lifecycle-runtime-ktx = { group = "androidx.lifecycle", name = "lifecycle-runtime-ktx", version.ref = "lifecycleRuntimeKtx" }
androidx-activity-compose = { group = "androidx.activity", name = "activity-compose", version.ref = "activityCompose" }
androidx-compose-bom = { group = "androidx.compose", name = "compose-bom", version.ref = "composeBom" }
androidx-ui = { group = "androidx.compose.ui", name = "ui" }
androidx-ui-graphics = { group = "androidx.compose.ui", name = "ui-graphics" } androidx-ui-tooling = { group = "androidx.compose.ui", name = "ui-tooling" } androidx-ui-tooling-preview = { group = "androidx.compose.ui", name = "ui- tooling-preview" }
androidx-ui-test-manifest = { group = "androidx.compose.ui", name = "ui-test-manifest" }
androidx-ui-test-junit4 = { group = "androidx.compose.ui", name = "ui-test-junit4" }
androidx-material3 = { group = "androidx.compose.material3", name = "material3" }

[plugins]
android-application = { id = "com.android.application", version.ref = "agp" }
jetbrains-kotlin-android = { id = "org.jetbrains.kotlin.android", version.ref = "kotlin" }
```

4. Build.gradle.kts

```
plugins {
    id("com.android.application")
    id("org.jetbrains.kotlin.android")
}

android {
    namespace = "com.example.odlettergenerator"
    compileSdk = 34

    buildFeatures {
        viewBinding = true
    }

    defaultConfig {
        applicationId = "com.example.odlettergenerator"
        minSdk = 24
        targetSdk = 34
        versionCode = 1
        versionName = "1.0"

        testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            isMinifyEnabled = false
            proguardFiles(
                getDefaultProguardFile("proguard-android-optimize.tx
                t"), "proguard-rules.pro"
            )
        }
    }
```

```
compileOptions {
sourceCompatibility = JavaVersion.VERSION_1_8
        targetCompatibility = JavaVersion.VERSION_1_8
    }

    kotlinOptions {
        jvmTarget = "1.8"
    }
 }

 dependencies {
    implementation("androidx.core:core-ktx:1.12.0")
implementation("androidx.appcompat:appcompat:1.6.1")
    implementation("com.google.android.material:material:1.11.0")
    implementation("androidx.constraintlayout:constraintlayout:2.1.4")

testImplementation("junit:junit:4.13.2")
    androidTestImplementation("androidx.test.ext:junit:1.1.5")
    androidTestImplementation("androidx.test.espresso:espresso-core:3.5.1")
 }
```

5. Graddle-wrapper.properties

```
#Fri Apr 25 21:12:53 IST 2025
distributionBase=GRADLE_USER_HOME distributionPath=wrapper/dists
 distributionUrl=https\://services.gradle.org/distributions/gradle-8.7-bin.zip
 zipStoreBase=GRADLE_USER_HOME
zipStorePath=wrapper/dists
```

A.2. OUTPUT SCREENSHOTS

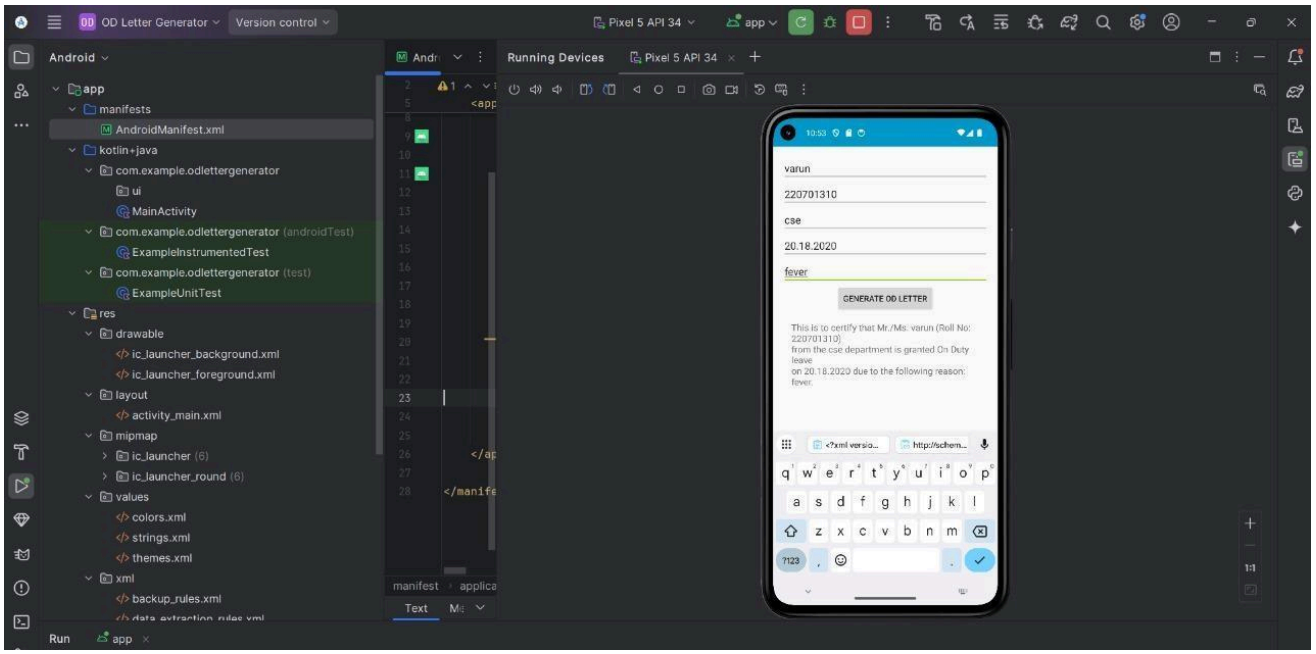


Fig. 2. OD Letter Generator – Interface

REFERENCES

[1] Android Developers Documentation – https://developer.android.com *(Official documentation for Android app development, components, and best practices.)*

[2] Kotlin Language Documentation – https://kotlinlang.org/docs/home.html *(Official reference for Kotlin programming language used in Android development.)*

[3] Firebase Authentication – https://firebase.google.com/docs/auth *(Used for implementing user sign-up and login functionalities.)*

[4] iText PDF Library – https://itextpdf.com/ *(Used for generating and formatting PDF documents within the app.)*

[5] Stack Overflow – https://stackoverflow.com *(Community support and troubleshooting for common Android and Kotlin issues.)*

[6] Material Design Guidelines – https://m3.material.io *(Guidelines for designing modern and intuitive Android UI components.)*

[7] JetBrains Kotlin Blog – https://blog.jetbrains.com/kotlin/ *(Helpful for updates, tips, and examples related to Kotlin features and use cases.)*

[8] Retrofit Library Documentation – https://square.github.io/retrofit/ *(For networking and API integration, if applicable in future enhancements.)*