VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belagavi-590018, Karnataka

A Mini Project Report on

"HOSTEL CHECK-IN CHECK-OUT"

Submitted in partial fulfilment of the requirement for the award of degree of

Bachelor of Engineering

In

Computer Science and Engineering

Submitted by

NAVYASHREE S (4NN20CS030)

PRATHEEK RAJ URS C P (4NN20CS036)

Under the Guidance of

Mr. Deepak P

Assistant Professor

Dept. of CSE

ESTD-2008

Department of Computer Science and Engineering

NIE Institute of Technology

Mysuru -570018

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NIE Institute of Technology, Mysore

ESTD-2008

CERTIFICATE

This is to certify that the mini project work entitled ***"HOSTEL CHECK-IN CHECK-OUT"*** is carried out by ***NAVYASHREE S*** bearing ***USN:*4NN20CS030** and ***PRATHEEK RAJ URS C P*** bearing ***USN:4NN20CS036*** in the partial fulfillment for the sixth semester of **Bachelor of Engineering degree** in **Computer Science and Engineering** of the **Visvesvaraya Technological University, Belagavi** during the academic year **2022-23**. The project report has been approved as it satisfies the academic requirements with respect to project work prescribed for the Bachelor of Engineering.

Signature of the Guide

Mr. Deepak P
Asst. Professor,
Dept. of CS & Engineering
NIEIT, Mysuru

Signature of the HOD

Dr. Usha M.S
Associate Professor and Head,
Dept. of CS & Engineering
NIEIT, Mysuru

External Viva

Name of the examiners	**Signature with Date**
1……....	**1**…….
2…….	**2**…….

ACKNOWLEDGEMENT

We sincerely owe our gratitude to all people who helped and guided us in completing this project work.

We are thankful to **Dr. Rohini Nagapadma**, Principal, NIEIT, Mysuru, for having supported us in our academic endeavors.

We are thankful to **Dr. Usha M S** Associate Professor and Head**,** Department of Computer Science and Engineering, NIEIT for providing us timely suggestion, encouragement and support to complete this mini-project.

We would like to sincerely thank our project guide**, Mr. Deepak P , Ms. Priya H P,** Asst. Professor in Dept. of Computer Science and Engineering for providing relevant information, valuable guidance and encouragement to complete this mini-project.

We would also like to thank all our **teaching and non-teaching staff members** of **Dept. of Computer Science and Engineering**. We are grateful to the college for keeping labs open whenever required and providing us Systems and Required software.

We are always thankful to our Parents for their valuable support and guidance in every step. Also thank all our friends for their support and guidance throughout the project.

We express our deepest gratitude and indebted thanks to NIEIT which has provided us an opportunity in fulfilling our most cherished desire of reaching our goal.

Yours Sincerely,

NAVYASHREE S (4NN20CS030)

PRATHEEK RAJ URS C P (4NN20CS036)

ABSTRACT

Our mobile application project titled **"Hostel Check-In/Check-Out"** or application named **“HOSTEL HIVE”** aims to revolutionize the process of managing hostel accommodations specifically designed for college campuses. This project focuses on developing a user-friendly mobile application tailored to the specific needs of college hostels.

This application aims to streamline and digitize the check-in and check-out procedures, providing a seamless experience for both hostel administrators and college students. With this application, students can easily complete the check-in process using their smartphones, eliminating the need for lengthy paperwork and manual verification. The app allows students to provide their personal information, such as identification details and contact information in a secure and convenient manner. Hostel administrators can efficiently validate and manage the information provided, reducing administrative burden and improving data accuracy. By digitizing the check-in and check-out process, our project aims to simplify hostel management, enhance student satisfaction, and optimize the overall experience within college hostels.

Through the **"Hostel Check-In/Check-Out"** mobile application, we aim to modernize and optimize the management of college hostels. By leveraging the power of mobile technology, our project strives to streamline administrative tasks, improve student convenience, and elevate the overall hostel experience within college campuses.

TABLE OF CONTENTS

LIST OF FIGURES PAGE NO

CHAPTER 1

INTRODUCTION

1.1 Introduction to Kotlin

JetBrains developed IntelliJ IDEA, which is the basis for Android Studio. In 2011, the company introduced the Kotlin language. Kotlin is a statically-typed, modern programming language that runs on a Java Virtual Machine (JVM) by compiling Kotlin code into Java byte-code. It can also be compiled to JavaScript source code and to native executables. Kotlin is flexible. Kotlin is object-oriented language, and a "better language" than Java, but still be fully interoperable with Java code. Though Kotlin was production ready, the language wasn't stable. When important changes in the language happened, developers had to change their codebase. Five years later, Kotlin released 1.0. At Google I/O 2017, Google announced that Android will support Kotlin as a first-class programming language from now on. For this to happen, the 3.0 release of Android Studio (AS) integrated Kotlin support out of the box! The following three minor releases of AS continued to improve the Kotlin support and the tools available. Kotlin support for JavaScript (i.e., classic back-end) is considered stable in Kotlin 1.3 by its developers, while Kotlin/JS (IR-based) in version 1.4, is considered alpha. Kotlin/Native Runtime (for e.g., Apple support) is considered beta.

1.2 Features of Kotlin

- Statically typed
- Data Classes
- Concise and Safe
- Interoperable with Java
- Functional and Object-Oriented Capabilities
- Less Compilation time
- User-Friendly

1.3 How the idea got into existence

The idea for the hostel app likely originated from recognizing the challenges and issues faced in the traditional manual check-in/check-out processes in hostels. Here are some common issues and their corresponding solutions that may have led to the development of the hostel app:

1. **Manual Paperwork and Inefficiency**:
Issue: Traditional check-in/check-out processes in hostels often involve a significant amount of manual paperwork, leading to inefficiencies, errors, and time-consuming procedures.
Solution: The hostel app aims to automate the check-in/check-out processes, replacing manual paperwork with a digital system. This reduces errors, streamlines procedures, and improves overall efficiency.

2. **Difficulty in Communication and Support**:
Issue: Students may encounter challenges in communicating with hostel staff or seeking assistance during their stay.
Solution: The app incorporates a messaging or support feature, enabling students to communicate directly with hostel staff. This facilitates prompt assistance, addresses queries, and improves overall communication channels.

3. **Lack of Check-In/Check-Out Notifications**:
Issue: Warden may forget the check-in/check-out dates, leading to confusion and potential penalties.
Solution: The app sends notifications to warden regarding the student check-in/check-out dates, ensuring they are well-informed and can avoid complications.

4. **Limited Data Management and Reporting**:
Issue: Hostel management may struggle with organizing and analyzing data related to check-ins, check-outs.
Solution: The app stores and manages check-in/check-out records and provides reporting and analytics capabilities. Hostel management can access data insights, generate reports on occupancy rates, and make informed decisions based on the data collected.

These issues and corresponding solutions demonstrate the need for a digital solution like the hostel app, which can improve efficiency, communication, data management, and overall student experience in the hostel environment. By addressing these challenges, the app aims to enhance the check-in/check-out processes and optimize hostel management operations.

CHAPTER 2

SYSTEM REQUIREMENT SPECIFICATIONS

Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, macOS and Linux based operating systems The first stable build was released in December 2014. **Android Studio** provides many excellent features that enhance productivity when building Android apps, such as a blended environment where one can develop for all Android devices, apply Changes to push code and resource changes to the running app without restarting the app, a flexible Gradle-based build system, a fast and feature-rich emulator, GitHub and Code template integration to assist you to develop common app features and import sample code, extensive testing tools and frameworks, C++ and NDK support, and many more.

2.1 Hardware Requirements

- Processor - Intel® Core i5 CPU M 370
- Processor Speed - 500 MHz or above
- RAM – Minimum of 8GB.
- Monitor resolution - A color monitor with a minimum resolution of 1000*700

2.2 Software Requirements

- Java Development Kit (JDK)
- Android Studio
- Kotlin access
- Firebase connection
- Android Emulator or a Physical device

CHAPTER 3

ABOUT THE PROJECT

3.1 Overview

The Hostel Student Check-In/Check-Out System is a digital solution designed to streamline and automate the check-in and check-out processes in a student hostel or dormitory. This system leverages technology to enhance operational efficiency, improve student experience, and provide accurate tracking student data. It replaces manual paperwork with self-service kiosks or a mobile application, allowing students to easily check in and out. The system generates digital access credentials, and maintains real-time occupancy records. Additionally, it provides hostel management with valuable insights through reporting and analytics, enabling effective resource utilizing and better overall management of the student accommodation facility. This system ultimately aims to create a seamless and efficient experience for both students and hostel staff, optimizing the management of student check-ins and check-outs.

3.2 Functionalities provided by **HOSTEL HIVE** are as follows:

1. **Student Registration**: Students can register themselves in the app through QR code and provide their personal information, contact details, and any necessary identification documents. This information is securely stored for future reference.

2. **Check-In Process**: The app simplifies the check-in process by allowing students to confirm their arrival by just clicking a toggle bar no need of re- entering the information. This helps the hostel administration track the student's arrival and also make the process easy and time saving ,which also increases the efficiency of app.

3. **Check-Out Process**: Students can initiate the check-out process through the app by indicating their departure date and personal information with contact details. This helps the hostel administration keep track of students' departure and plan their resources accordingly.

4. **Notifications**: The app sends notifications to warden regarding the student check-in and check-out informations, providing them with important information and ensuring they are aware of the required procedures.

5. **Communication and Support**: The app may include a communication feature that allows students to contact the hostel staff or administration for any inquiries, requests, or assistance during their stay.

6. **Digital Records**: The app maintains digital records of student check-ins and check-outs, providing an easily accessible and organized database for the hostel administration to refer to when needed.

7. **Reporting**: The app generates reports on check-in and check-out data, such as the number of students checked in and checked out on specific dates or within a certain time period. This information can be used for administrative purposes and to monitor occupancy trends.

3.3 Tech stack

1. **Android Development**:

- Programming Language: Kotlin
- Integrated Development Environment (IDE): Android Studio
- Jetpack: Utilize various Jetpack components such as ViewModel, LiveData, and Navigation for efficient development.

2. **Firebase Integration**:

- Firebase Authentication: Utilized Firebase Authentication for user signup, login, and OTP verification.
- Firebase Realtime Database or Firestore: Store user data, such as student information and check-in/check-out details.
- Firebase Cloud Functions: Implement server-side logic using Firebase Cloud Functions to handle Android additional functionalities or data manipulation.

3. **UI/UX**:

- XML: Design user interfaces using XML layouts in Android Studio.
- Material Design: Implement Material Design guidelines to ensure a consistent and visually appealing UI.

4. **Networking and API Integration**:

- Retrofit: Communicate with backend APIs, if required, using Retrofit library for network requests.
- Firebase Cloud Functions: Connect to custom Firebase Cloud Functions for additional backend functionality.

5. **Storage**:

- Firebase Cloud Storage: Store and retrieve files such as user information or any other relevant data.

6. **Dependency Management**:

- Gradle: Utilized Gradle build system for managing dependencies and building the Android app.

7. **Version Control**:

- Git: Git for version control and collaboration.

CHAPTER 4

SYSTEM DESIGN

4.1 Benefits of the application

1. **Time Efficiency**: The app reduces the time required for check-in and check-out processes, as students can provide their information and complete the necessary procedures digitally. This saves time for both students and hostel staff.

2. **Improved Accuracy**: With digital data entry and automated processes, the app reduces the chances of errors and inconsistencies that can occur with manual paperwork, ensuring accurate student information and check-in/check-out records.

3. **Enhanced Communication**: The app's messaging or support feature facilitates seamless communication between students and hostel staff. Students can easily seek assistance, ask questions, or report any issues, leading to faster response times and efficient problem resolution.

4. **Convenience and Flexibility**: Students can access the app anytime and anywhere, allowing them to complete check-in and check-out procedures conveniently, even outside regular working hours. This flexibility accommodates different schedules and minimizes unnecessary waiting times.

5. **Transparent Updates**: The app provides real-time notifications and updates to students and warden regarding the check-in/check-out status, or any changes in the hostel policies or procedures. This ensures transparency and keeps students informed at all times.

6. **Secure Access**: The app incorporates secure authentication mechanisms, such as Firebase Authentication, to protect student data and ensure secure access to the app's features. This enhances the privacy and security of student information.

7. **Streamlined Administrative Tasks**: The app automates various administrative tasks, such as record-keeping, reporting, and analytics. This reduces manual effort, improves data management, and allows hostel staff to focus on other important responsibilities.

8. **Historical Data Analysis**: By storing and organizing check-in/check-out records, the app enables hostel administration to analyze historical data and identify trends. This can inform future decision-making, such as forecasting demand, adjusting operational processes, or optimizing resource allocation.

9. **Positive User Experience**: Overall, the app provides a more convenient and user-friendly experience for students, offering a modern and efficient way to handle check-in and check-out processes. This contributes to increased student satisfaction and enhances the reputation of the hostel.

4.2 List of kotlin components used for application

1. **Activities and Fragments:**

- Use Kotlin classes to implement activities and fragments, representing the different screens and UI components of the app.

2. **Intents:**

- Utilize Kotlin's intent mechanism to navigate between activities and pass data between different screens.

3. **RecyclerView:**

- Implement RecyclerView in Kotlin to display lists of items, such as notifications, check-in/check-out history, or available rooms.

4. **Adapters:**

- Create custom Kotlin adapters to bind data to RecyclerViews and handle item interactions.

5. **ViewModels and LiveData:**

- Use Kotlin's ViewModel and LiveData components from the Android Architecture Components to manage UI-related data and ensure data updates are reflected in real time.

6. **Coroutines:**

- Utilize Kotlin Coroutines to handle asynchronous tasks, such as network requests or database operations, in a more concise and efficient manner.

7. **Firebase Authentication:**

- Integrate Firebase Authentication in Kotlin to handle user registration, login, and OTP verification.

8. **Firebase Realtime Database or Firestore:**

- Use Kotlin code to interact with Firebase Realtime Database or Firestore for storing and retrieving data related to user information, check-in/check-out records, and notifications.

9. **Push Notifications:**

- Implement Firebase Cloud Messaging in Kotlin to send push notifications to users, delivering check-in/check-out reminders, updates, or important information.

10. **Material Design Components**:

- Utilize Kotlin code to implement various Material Design components provided by the Material Components for Android library, ensuring a consistent and visually appealing UI.

11. **Navigation Components:**

- Use Kotlin code to implement the Navigation Component from the Android Jetpack library, facilitating navigation between different screens and handling back stack management.

12. **Retrofit or OkHttp:**

- Integrate Retrofit or OkHttp in Kotlin to handle network requests and communicate with backend APIs, if required for additional functionalities.

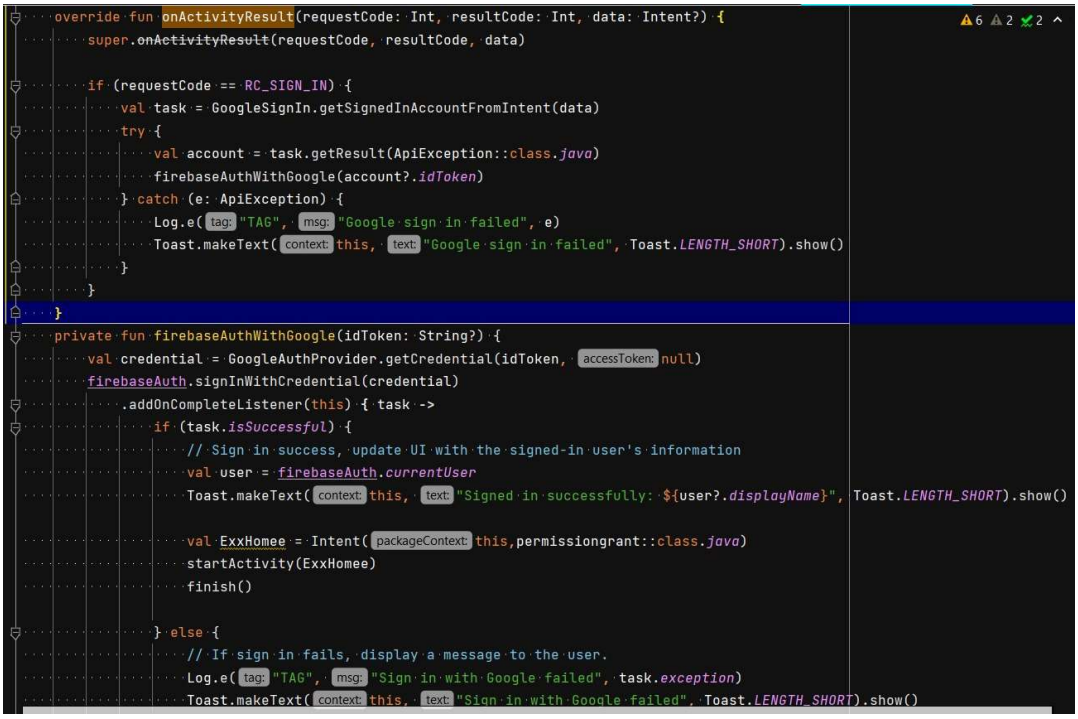
13. **Dependency Injection:**

- Implement dependency injection frameworks like Dagger or Koin in Kotlin to manage and inject dependencies into various components of the app.

CHAPTER 5

IMPLEMEANTATION

Projectadminsignin.kt



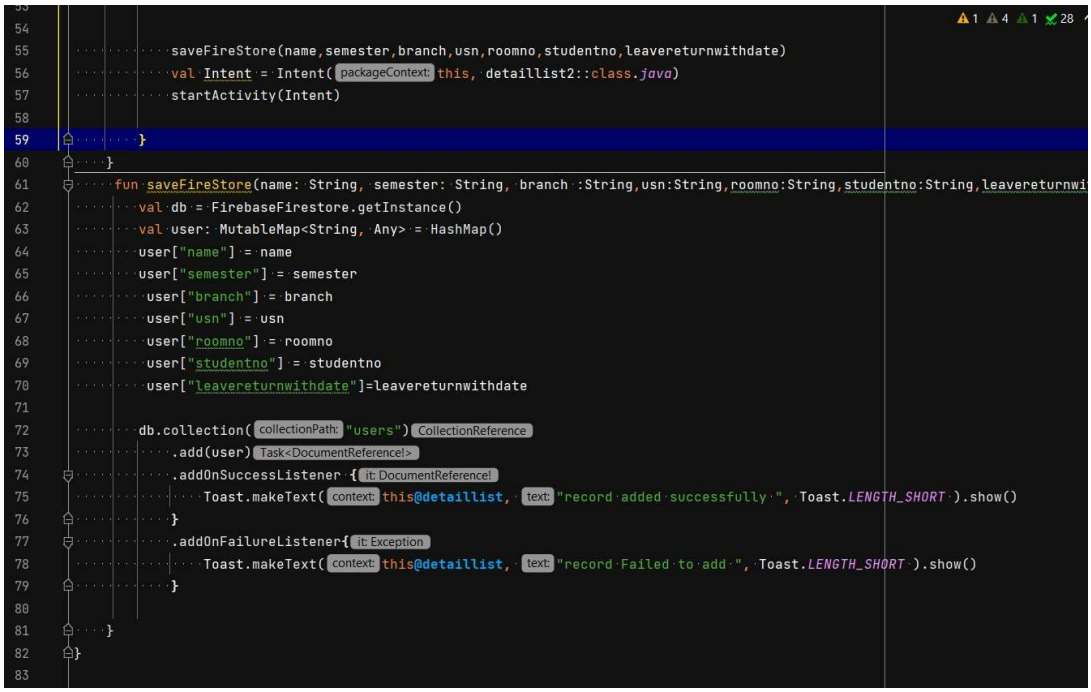
```
override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    super.onActivityResult(requestCode, resultCode, data)

    if (requestCode == RC_SIGN_IN) {
        val task = GoogleSignIn.getSignedInAccountFromIntent(data)
        try {
            val account = task.getResult(ApiException::class.java)
            firebaseAuthWithGoogle(account?.idToken)
        } catch (e: ApiException) {
            Log.e( tag: "TAG", msg: "Google sign in failed", e)
            Toast.makeText( context: this, text: "Google sign in failed", Toast.LENGTH_SHORT).show()
        }
    }
}
private fun firebaseAuthWithGoogle(idToken: String?) {
    val credential = GoogleAuthProvider.getCredential(idToken, accessToken: null)
    firebaseAuth.signInWithCredential(credential)
        .addOnCompleteListener(this) { task ->
            if (task.isSuccessful) {
                // Sign in success, update UI with the signed-in user's information
                val user = firebaseAuth.currentUser
                Toast.makeText( context: this, text: "Signed in successfully: ${user?.displayName}", Toast.LENGTH_SHORT).show()

                val ExxHomee = Intent( packageContext: this,permissiongrant::class.java)
                startActivity(ExxHomee)
                finish()

            } else {
                // If sign in fails, display a message to the user.
                Log.e( tag: "TAG", msg: "Sign in with Google failed",task.exception)
                Toast.makeText( context: this, text: "Sign in with Google failed", Toast.LENGTH_SHORT).show()
```

detaillist.kt



```
54
55          saveFireStore(name,semester,branch,usn,roomno,studentno,leavereturnwithdate)
56          val Intent = Intent( packageContext: this, detaillist2::class.java)
57          startActivity(Intent)
58
59      }
60  }
61  fun saveFireStore(name: String, semester: String, branch :String,usn:String,roomno:String,studentno:String,leavereturnwi
62      val db = FirebaseFirestore.getInstance()
63      val user: MutableMap<String, Any> = HashMap()
64      user["name"] = name
65      user["semester"] = semester
66      user["branch"] = branch
67      user["usn"] = usn
68      user["roomno"] = roomno
69      user["studentno"] = studentno
70      user["leavereturnwithdate"]=leavereturnwithdate
71
72      db.collection( collectionPath: "users") CollectionReference
73          .add(user) Task<DocumentReference!>
74          .addOnSuccessListener { it: DocumentReference!
75              Toast.makeText( context: this@detaillist, text: "record added successfully ", Toast.LENGTH_SHORT ).show()
76          }
77          .addOnFailureListener{ it: Exception
78              Toast.makeText( context: this@detaillist, text: "record Failed to add ", Toast.LENGTH_SHORT ).show()
79          }
80
81  }
82  }
83
```

otpverify.kt

```
        verifyBtn = findViewById(R.id.verifyOTPBtn)
        resendTV = findViewById(R.id.resendTextView)
        inputOTP1 = findViewById(R.id.otpEditText1)
        inputOTP2 = findViewById(R.id.otpEditText2)
        inputOTP3 = findViewById(R.id.otpEditText3)
        inputOTP4 = findViewById(R.id.otpEditText4)
        inputOTP5 = findViewById(R.id.otpEditText5)
        inputOTP6 = findViewById(R.id.otpEditText6)
    }


    inner class EditTextWatcher(private val view: View) : TextWatcher {
        override fun beforeTextChanged(p0: CharSequence?, p1: Int, p2: Int, p3: Int) {

        }

        override fun onTextChanged(p0: CharSequence?, p1: Int, p2: Int, p3: Int) {

        }

        override fun afterTextChanged(p0: Editable?) {

            val text = p0.toString()
            when (view.id) {
                R.id.otpEditText1 -> if (text.length == 1) inputOTP2.requestFocus()
                R.id.otpEditText2 -> if (text.length == 1) inputOTP3.requestFocus() else if (text.isEmpty()) inputOTP1.requestFo
                R.id.otpEditText3 -> if (text.length == 1) inputOTP4.requestFocus() else if (text.isEmpty()) inputOTP2.requestFo
                R.id.otpEditText4 -> if (text.length == 1) inputOTP5.requestFocus() else if (text.isEmpty()) inputOTP3.requestFo
                R.id.otpEditText5 -> if (text.length == 1) inputOTP6.requestFocus() else if (text.isEmpty()) inputOTP4.requestFo
                R.id.otpEditText6 -> if (text.isEmpty()) inputOTP5.requestFocus()
```

datastore.kt

```
                    LinearLayout.LayoutParams.WRAP_CONTENT,
                    LinearLayout.LayoutParams.WRAP_CONTENT
                )
                textViewParams.gravity = Gravity.RIGHT // Set the desired gravity
                textView1.layoutParams = textViewParams
                textView2.layoutParams = textViewParams

                textt.addView(linearLayout)
                linearLayout.addView(textView1)
                linearLayout.addView(textView2)
                linearLayout.addView(textView3)
                linearLayout.addView(textView4)
                linearLayout.addView(textView5)
                linearLayout.addView(textView6)
                linearLayout.addView(textView7)

                textView1.text=name1
                textView2.text=semester1
                textView3.text=branch1
                textView4.text=usn1
                textView5.text=roomno1
                textView6.text=studentno1
                textView7.text=leavereturnwithdate1

                Toast.makeText( context: this, text: "name :$name", Toast.LENGTH_SHORT).show()
                Log.d(datastore.TAG, msg: "Name: $name")
            }
        }
        .addOnFailureListener { e ->
            Log.d(datastore.TAG, msg: "Name")
        }
```

INTRODUCTION TO TESTING

TESTING:

Verification and validation are a generic name given to checking processes, which ensures that the software conforms to its specifications and meets the demands of users.

- Validation: Are we building the right product? Validation involves checking that the program has implanted meets the requirement of the users.
- Verification: Verification involves checking that the program confirms to its specification.

Results:

Several errors were detected and rectified and the whole project is working as it should have to work with proper output and high efficiency.

Test case ID	Test case	Steps to execute the test case	Expected result	Actual result	Pass/ Fail
1	Warden Sign-in	Click the button	Go to login page	Go to login page	Pass
2	Student Sign-in	Click the button	Go to PHONE NO authentication page with OTP	Go to PHONE NO authentication page with OTP	Pass
3	View	Click the button	Go to datastore page that stores and retrives data	Go to datastore page that stores and retrives data	Pass
4	Done	Click the button	Add the record to Firestore	Add the record to Firestore	Pass
5	EXIT	Click the button	Go to home screen	Go to home screen	Pass

Table 5.1 Testing for the “HOSTEL HIVE” Project

CHAPTER 6

SNAPSHOTS

Figure 6.1 home page

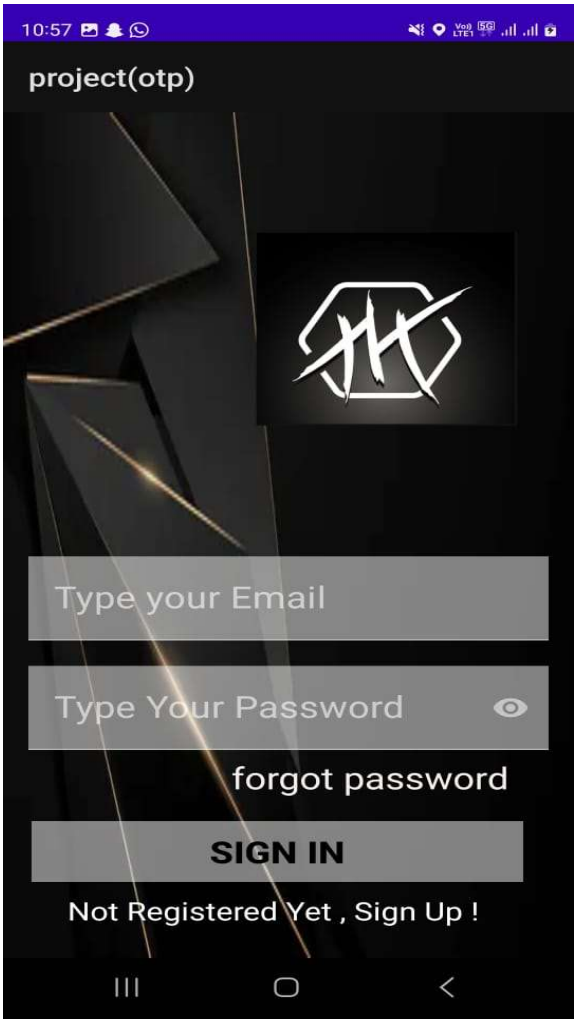


Figure 6.2 sign in page

Figure 6.3 sign up page

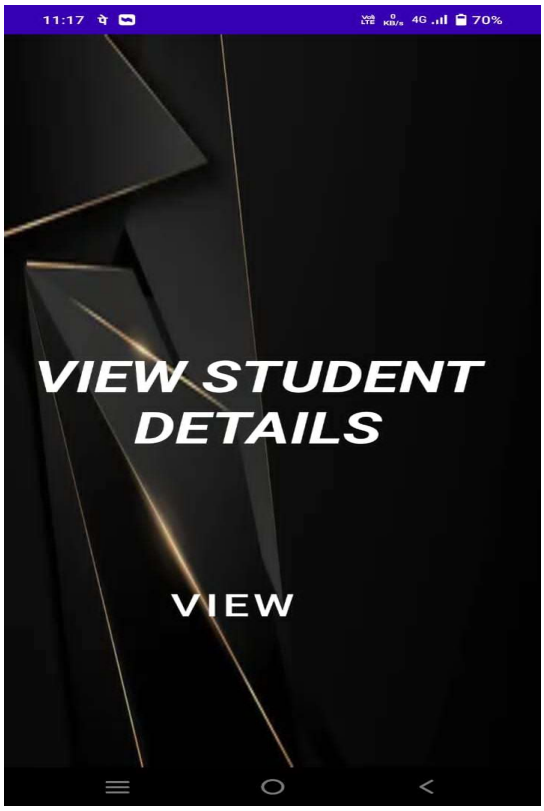


Figure6.4 database view page

Figure6.5 information page

Figure6.6 verification page

Figure6.7 OTP page

Figure6.8 detail page 1

Figure6.9 detail page 2

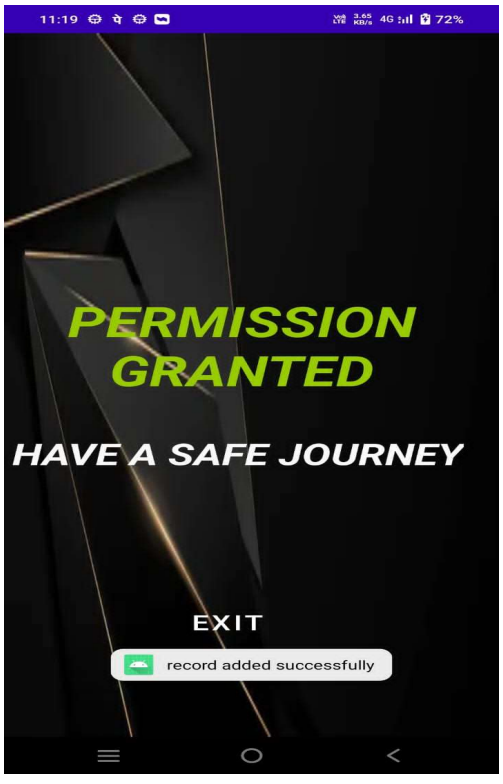


Figure6.10 status page

CHAPTER 7

CONCLUSION

In conclusion, the HOSTEL HIVE app designed for check-in and check-out processes in a college hostel offers several significant benefits for both students and hostel administration. By streamlining manual procedures, digitizing information, and leveraging the power of technology, the app greatly enhances the efficiency, convenience, and overall experience in managing student information.

The app provides a user-friendly interface for students to easily initiate and complete the check-in and check-out processes. With features such as user registration, secure authentication, and real-time notifications, students are empowered with a seamless and hassle-free experience. They can conveniently access the app, update their information, communicate with hostel staff, and receive important reminders and updates regarding their stay.

For hostel administration, the app revolutionizes their operations and simplifies administrative tasks. By automating processes, such as recording check-ins and check-outs, generating reports, and managing communication, the app saves time, minimizes errors, and improves overall data management. Hostel staff can focus on providing better support and addressing student needs, rather than being burdened with paperwork and manual record-keeping.

Overall, the hostel app transforms the traditional check-in and check-out processes, elevating the student experience, and streamlining administrative tasks. It fosters better communication, improves data accuracy, and enhances operational efficiency. By embracing digital innovation, the app drives a more seamless and convenient stay for students while empowering hostel administration with valuable tools for effective management.

With its numerous benefits, the hostel app sets a new standard in hostel management, ensuring a smooth and efficient experience for students and optimizing operations for the hostel administration. It stands as a testament to the power of technology in enhancing student accommodations and improving the overall hostel experience.

FUTURE ENHANCEMENT

1. **Payment Integration**: Integrate a secure payment gateway to allow students to make online payments for their hostel stay, additional charges, or amenities. This feature can streamline the payment process, offer transparency, and provide students with convenient payment options.

2. **Room Preference Selection**: Enhance the room selection process by allowing students to specify their room preferences, such as floor level, roommates, or specific amenities. This customization can enhance the student experience and improve satisfaction with room assignments.

3. **Feedback and Ratings**: Implement a feedback and ratings system where students can provide feedback on their stay, rate the hostel facilities, and share their experiences. This feature can help hostel administration understand student preferences, identify areas for improvement, and maintain quality standards.

4. **Digital Check-In/Check-Out Forms**: Replace physical paperwork with digital check-in and check-out forms that students can complete within the app. This streamlines the process, reduces paper waste, and improves data accuracy.

5. **Smart Room Management**: Integrate Internet of Things (IoT) devices and sensors to enable smart room management. This can include automated lighting and temperature control, energy-saving features, and real-time monitoring of room occupancy for efficient resource allocation.

6. **Social Features**: Incorporate social features within the app, such as a community forum or chat functionality, where students can connect, interact, and share information with each other. This can foster a sense of community and provide a platform for students to engage with each other.

7. **Multi-Language Support**: Extend language support to cater to international students or students from diverse backgrounds. This can enhance accessibility and ensure that the app can be used by a wider range of students.

8. **Integration with Campus Services**: Integrate the app with other campus services, such as event calendars, transportation information, or food services, to provide students with a comprehensive platform for accessing various campus resources and information.

9. **Smart Notifications**: Implement intelligent notification systems that use machine learning algorithms to provide personalized notifications and recommendations to students based on their preferences, interests, or past activities.

10. **Data Analytics and Insights**: Enhance the analytics and reporting capabilities of the app to provide deeper insights into student behavior, trends, and preferences. This can help hostel administration make data-driven decisions, optimize operations, and enhance the overall student experience.

BIBLIOGRAPHY

Websites

www.wikipedia.com

https://firebase.google.com/

www.Youtube.com

www.chatgpt.com

www.Github.com