Smart Campus Navigator

Submitted by

Mutthesh M 2116220701176

In partial fulfilment of the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

ANNA UNIVERSITY, CHENNAI

MAY 2025

RAJALAKSHMI ENGINEERING COLLEGE

CHENNAI – 602 105

BONAFIDE CERTIFICATE

Certified that this Report titled **"Smart campus navigator"** is the bonafide work of **MUTTHESH M (2116220701176)** 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	**SIGNATURE**
Dr. P. Kumar M.E., Ph.D.	**Mr. Bhuvaneswaran B, M.E.**
Head of the Department	Supervisor
Professor	Assistant Professor
Department of Computer Science and Engineering	Department of Computer Science and Engineering
Rajalakshmi Engineering College,	Rajalakshmi Engineering College,
Chennai – 602105	Chennai – 602105

Submitted to Project Viva-Voce Examination held on ______________________

Internal Examiner **External Examiner**

TABLE OF CONTENTS

ACKNOWLEDGEMENT

Initially we thank the Almighty for being with us through every walk of our life and showering his blessings through the endeavour 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 endeavouring 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 working time. We express our sincere thanks to **Dr.P.KUMAR, 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, **Mr. Bhuvaneswaran B**, **M.E.,** Department of Computer Science and Engineering. Rajalakshmi Engineering College for his valuable guidance throughout the course of the project.

MUTTHESH M 220701176

ABSTRACT

This project presents the development of the Smart Campus Navigator, a mobile application designed to help university students navigate large campuses and stay informed about events, using Kotlin in Android Studio. The primary goal of the application is to provide students with an efficient and interactive way to locate classrooms, facilities, and events, enhancing their campus experience. The core functionalities of the app include displaying an interactive campus map, allowing students to search for and view classroom locations, and providing real-time event schedules. To ensure data persistence and offline access, the application utilizes SQLite to store event schedules and campus map data locally on the user's device. This allows the app to function without requiring constant internet connectivity. The app also includes input validation to ensure accurate search queries and event information. The user interface is designed to be dynamic and responsive, with interactive elements such as clickable buttons, customizable font styles, and real-time updates for event notifications. The RecyclerView is used to display events in a list format, providing a smooth and efficient way to manage multiple events or campus locations. In addition, the app integrates Android's LocationManager and Google Maps API for real-time location tracking and campus navigation. This project demonstrates a practical understanding of mobile app development using Kotlin, showcasing the integration of key Android development concepts such as location services, local storage, UI customization, and real-time updates. The Smart Campus Navigator provides a useful tool for students, helping them easily navigate their campus while staying up-to-date with academic and extracurricular activities.

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 GENERAL

In the digital age, effective campus navigation has become increasingly important for students navigating large university campuses. The Smart Campus Navigator app aims to help students efficiently find classrooms, facilities, and campus events, thereby enhancing their overall campus experience and saving time. Android, being one of the most widely used mobile platforms, provides an excellent ecosystem for developing such an app using the Kotlin programming language. Mobile apps have revolutionized how students interact with technology, making campus life simpler and more accessible. This project focuses on building a user-friendly Android application that not only helps students navigate their campuses but also provides up-to-date event schedules and real-time location tracking. Leveraging modern Android components such as Google Maps API, RecyclerView, and SQLite, this app ensures smooth data handling and efficient navigation. The integration of location-based services enhances user experience by guiding students directly to their destinations with precise directions.

1.2 OBJECTIVE

The primary objective of this project is to develop a Smart Campus Navigator app that helps university students efficiently navigate their campus while staying updated on campus events. The app aims to offer an intuitive user interface where students can easily search for and view locations of classrooms, facilities, and events on campus. By integrating real-time location tracking and precise navigation, the app ensures that students can find their way to various campus destinations seamlessly.

The project seeks to utilize modern Android development techniques, including the MVVM (Model-View-ViewModel) architecture, SQLite for offline data storage, and Google Maps API for location-based services. Another objective is to make the app lightweight, efficient, and user-friendly, ensuring that students of all technological backgrounds can navigate through the campus without hassle.

1.3 EXISTING SYSTEM

The existing campus navigation systems, such as Google Maps and university-specific apps, provide basic functionalities like location tracking and directions. However, these apps often lack campus-specific features such as detailed campus maps, event schedules, and integration with real-time campus activities. They may also require a stable internet connection, limiting their usability in areas with poor connectivity. Additionally, many existing apps focus primarily on external navigation rather than offering a personalized, student-centric experience for navigating complex university campuses. Some apps also do not provide real-time event updates or have difficulty accommodating the dynamic nature of campus schedules, which change frequently. Furthermore, most campus navigation systems fail to integrate seamlessly with the university's academic and extracurricular offerings, such as campus events, workshops, or club activities. This leads to students needing to use multiple apps or platforms to stay informed about everything happening on campus, creating an inefficient and fragmented user experience.

1.4 PROPOSED SYSTEM

The Smart Campus Navigator app aims to address these gaps by providing a personalized, student-focused solution for campus navigation and event management. With features like real-time location tracking, campus maps, event notifications, and offline access, the app ensures that students can efficiently navigate the campus while staying updated on academic and extracurricular activities. It also focuses on providing a lightweight, user-friendly interface, avoiding the complexity that often overwhelms students using other apps. The proposed system aims to enhance the overall student experience by providing a reliable, comprehensive, and easy-to-use mobile app that combines location services, event management, and offline functionality—all in one platform.

CHAPTER 2

LITERATURE SURVEY

[1] S. Gupta et al., “AR-based Smart Campus Navigation,” Int. J. Computer Apps., vol. 178, no. 7, 2021.

This study explores an augmented reality-based campus navigation system that enhances student experience by overlaying visual cues on real-world maps for efficient campus navigation.

[2] R. Kumar et al., “Mobile Application for Campus Navigation,” IJERT, vol. 9, no. 3, 2020.

The paper discusses a GPS-integrated mobile app for campus navigation, focusing on the challenges of mapping large campuses and providing offline navigation capabilities for areas with limited connectivity.

[3] M. Lee and J. Park, “Context-Aware Mobile Systems in Smart Campus,” IEEE Access, vol. 7, pp. 55811-55820, 2019.

This research examines location-based services in smart campuses, delivering real-time, personalized navigation and event recommendations based on students' current location.

[4] T. Zhang and H. Liu, “University Event Management via Mobile Apps,” Int. J. Educ. Technol., vol. 5, no. 2, 2020.

Zhang and Liu discuss an app that combines campus navigation with real-time event schedules, ensuring students are always informed about academic and extracurricular activities.

[5] Y. Chen et al., “IoT-Based Smart Campus Solutions,” Sensors, vol. 18, no. 9, 2018.
This paper highlights IoT integration in campus management, focusing on real-time campus data and mobile app features for navigation and event updates.

[6] J. S. Khan and S. M. Khan, “Intelligent Campus Navigation System,” Int. Conf. Computing, Comm. & Networking, pp. 123-127, 2021.
This study introduces a navigation system combining GPS, Bluetooth, and campus data, providing precise directions and event scheduling to improve campus navigation

CHAPTER 3

SYSTEM DESIGN

3.1 GENERAL

The system design of the Smart Campus Navigator aims to provide students with an efficient and user-friendly platform to navigate the campus, find classrooms, and stay updated on campus events. The app follows a modular design, leveraging the Model-View-ViewModel (MVVM) architecture to separate concerns and ensure better maintainability. It utilizes Google Maps API for real-time navigation, SQLite for local data storage, and WorkManager for handling background tasks like fetching and updating campus event information. The design ensures that the app is functional even without an internet connection, allowing students to access previously loaded maps and event schedules offline. Additionally, the system prioritizes battery efficiency and smooth user experience by optimizing location tracking and minimizing unnecessary background processes. The app also ensures minimal user disruption through intelligent notification management for event reminders and campus updates.

3.1.1 SYSTEM FLOW DIAGRAM

The system flow diagram of the Smart Campus Navigator illustrates the step-by-step process from user interaction to campus navigation and event notifications. The flow begins when the user opens the app and searches for a specific campus location or event through the user interface (UI). The app queries the local database or online campus data and updates the map or event list accordingly. If the user requests directions, the Google Maps API fetches real-time location data and provides turn-by-turn navigation. Simultaneously, if an event is scheduled or a class is approaching, the app schedules a background task using WorkManager to send reminders or push notifications. Upon the event or class time, the app triggers the reminder, notifying the user via a local notification. This cycle ensures continuous navigation assistance and event updates, keeping the user informed and on track throughout the day.

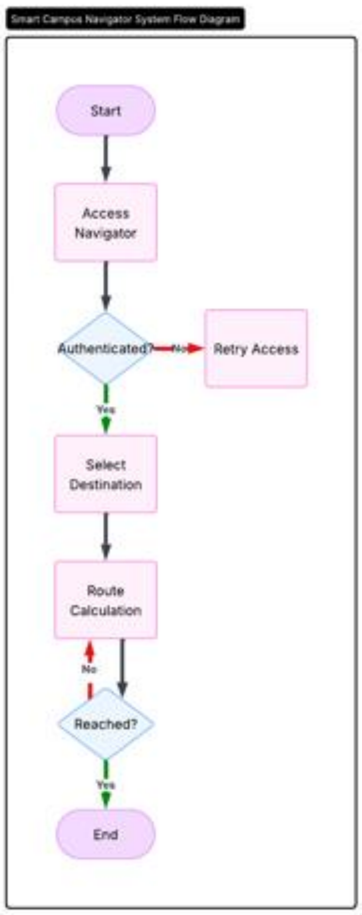

Fig 3.1

3.1.2 ARCHITECTURE DIAGRAM

The Smart Campus Navigator app is structured using the MVVM architecture, which includes the following layers:

- Model: Manages the data layer through the local SQLite database, which stores campus maps, event schedules, and user preferences. The Repository pattern is used to abstract the data sources and provide a clean interface to the rest of the app.
- ViewModel: Acts as a bridge between the UI and Model, providing data streams via LiveData. It handles data processing, updates the UI when data

changes, and manages user interactions with the Model layer.

- View: Consists of Activities and Fragments that display data to the user, such as campus maps, event details, and navigation routes. The UI is designed to be intuitive and responsive to ensure an optimal user experience.

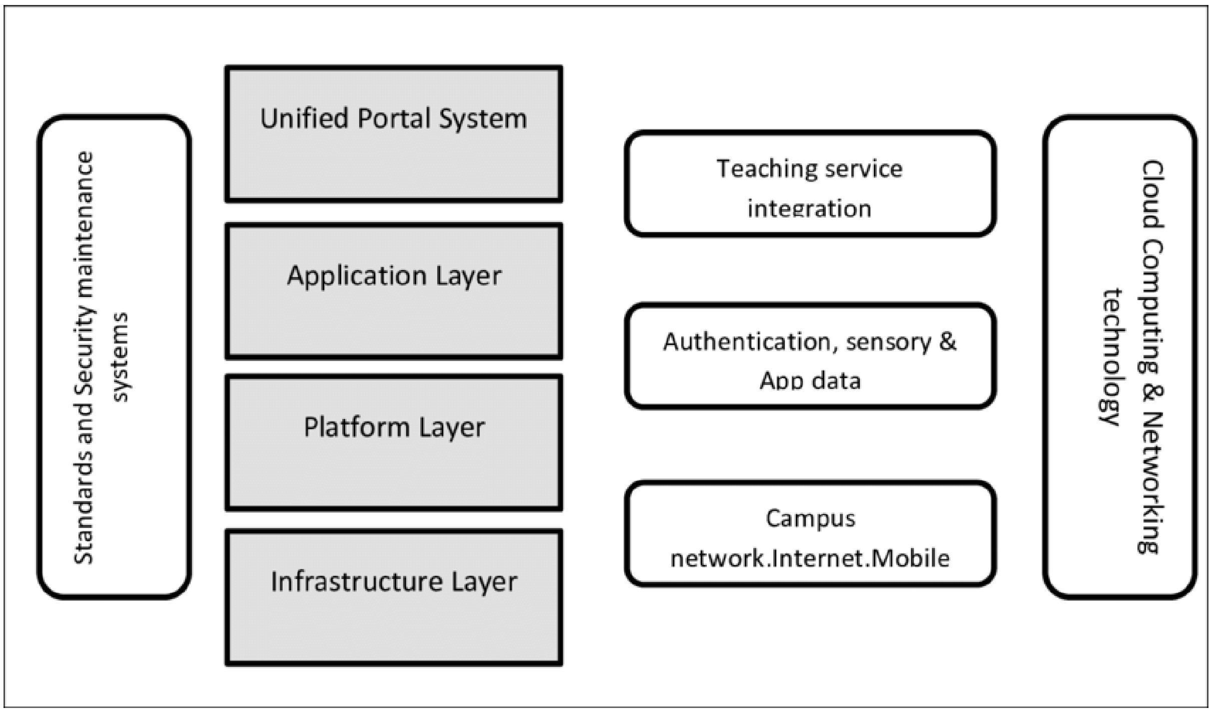
Standards and Security maintenance systems
Unified Portal System
Application Layer
Platform Layer
Infrastructure Layer
Teaching service integration
Authentication, sensory & App data
Campus network.Internet.Mobile
Cloud Computing & Networking technology

Fig 3.2

3.1.3 ACTIVITY DIAGRAM

The activity diagram for the Smart Campus Navigator describes the dynamic behavior of the system. It begins with the user launching the app and navigating to the main screen, where they can interact with various features such as searching for locations, viewing events, and accessing campus maps. When the user selects a specific building or event, the system collects relevant data, such as the location coordinates or event details, and updates the UI with the corresponding information. If the user requests navigation to a campus location, the app fetches real-time location data and provides step-by-step directions using the Google Maps API. Similarly, if the user sets a reminder for an event or class, the system collects the event details, stores them in the local database, and schedules a reminder using WorkManager. If the user edits or deletes an event, the system updates or cancels the reminder accordingly.

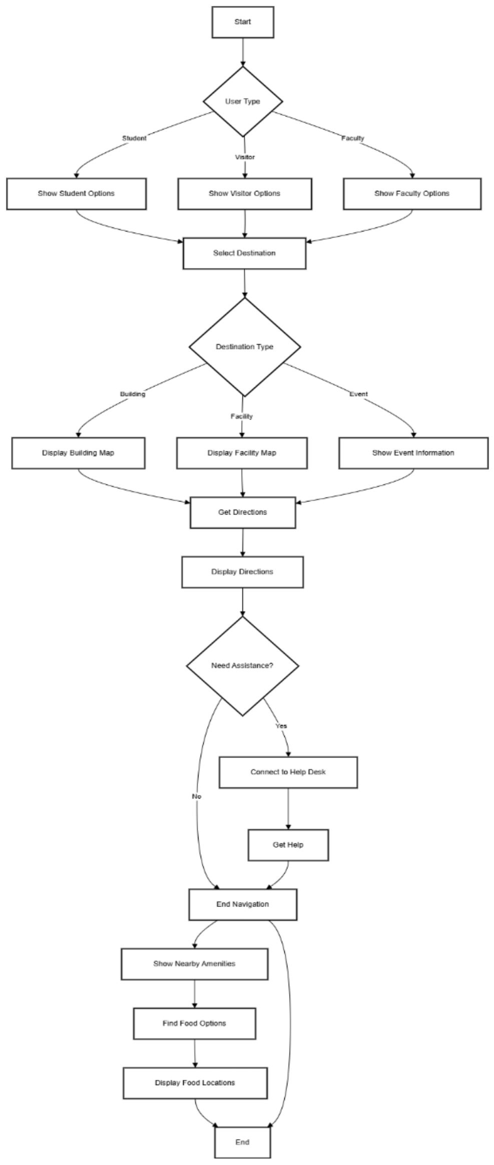

Fig 3.3

3.1.4 SEQUENCE DIAGRAM

The sequence diagram for the Smart Campus Navigator represents the chronological interaction between system components. It begins with the User triggering an action, such as searching for a campus location or event, which calls methods in the MainActivity. The MainActivity then interacts with the ViewModel, which retrieves the necessary data or event information from the Repository. The Repository communicates with the Room Database to fetch or update the stored campus data. Once the user requests directions or event reminders, the system schedules the necessary tasks using WorkManager. When the scheduled time arrives, WorkManager triggers the appropriate actions—calling the Notification Manager to send local notifications and the Email Sender to send an email reminder to the user.

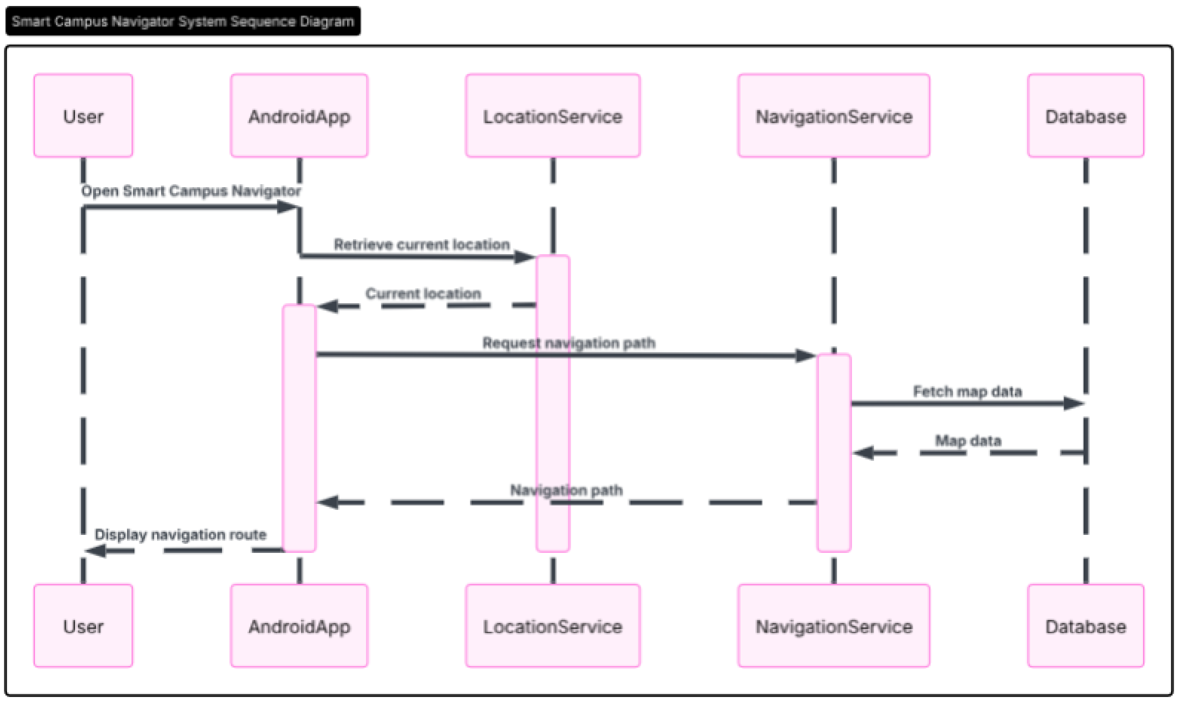

Fig 3.4

CHAPTER 4

PROJECT DESCRIPTION

4.1 INTRODUCTION

The Smart Campus Navigator is an Android mobile application developed using Kotlin and Android's modern libraries. The app is designed to help university students efficiently navigate their campus by providing real-time directions, event schedules, and location-based notifications. Unlike traditional campus guides, this app actively assists users by guiding them to classrooms, events, and key campus locations, minimizing confusion and time wasted.

4.2 OBJECTIVE

The primary objective of the **Smart Campus Navigator** is to provide a seamless and efficient solution for university students to navigate their campus, access real-time event schedules, and receive timely notifications. The app aims to simplify campus navigation by offering precise directions to classrooms, departments, and key locations using real-time location data. Additionally, it seeks to enhance the student experience by providing automated event reminders, ensuring that students stay informed about academic and extracurricular activities. The system is designed to be user-friendly, offline-capable, and scalable, allowing future enhancements such as push notifications, speech-to-text input, and multi-device synchronization. Ultimately, the goal is to improve campus life by making it easier for students to manage their daily activities and stay organized on campus.

4.3 FEATURES

The Smart Campus Navigator app includes the following major features:

- Campus Navigation: Users can search for campus locations and get directions to specific buildings or departments using the Google Maps API. The app provides real-time turn-by-turn navigation to help users find their destinations efficiently.
- Event Scheduling and Reminders: Users can view upcoming campus events, set reminders, and receive timely notifications. The app allows students to track important dates like class schedules, exams, and extracurricular activities.
- Offline Support: All campus data, including locations, event schedules, and reminders, are stored locally using Room database, allowing users to access information even without an internet connection.
- Automated Notifications and Alerts: The app uses WorkManager to schedule background tasks that trigger notifications and email reminders for scheduled events, ensuring students never miss an important activity.

Example worker code for event reminder:

```
class EventReminderWorker(context: Context, params:
WorkerParameters) : Worker(context, params) {
    override fun doWork(): Result {
        val eventTitle = inputData.getString("event_title") ?: return
Result.failure()
        val eventTime = inputData.getLong("event_time", 0L)
        sendNotification(eventTitle, eventTime)
        sendEmailReminder(eventTitle, eventTime)
        return Result.success()
    }
}
```

- Modern UI: The user interface is built using RecyclerView for displaying lists of events, tasks, and campus locations, ensuring smooth performance and easy navigation. It follows Material Design principles for elements like buttons, dialogs, and layouts, providing an intuitive and consistent experience for users.

4.4 Methodology (With Detailed Steps & Codes)

The development of the Smart Campus Navigator followed an 8-step systematic methodology:

Step 1: Requirement Gathering

Gathered user requirements such as campus navigation, real-time event scheduling, reminders, and location tracking. Key features included providing directions, event notifications, offline data storage, and integration with campus map services.

Step 2: System Design

Designed using the MVVM architecture:
Model: Represents campus data (e.g., locations, events) stored in the Room database.
ViewModel: Manages UI logic and schedules background tasks (e.g., event reminders).
View: XML layouts for user interfaces such as event listings, search screens, and campus maps.

Step 3: Development Setup

Set up the Android Studio project with the necessary dependencies:

```
implementation "androidx.room:room-runtime:2.5.0"
implementation "androidx.work:work-runtime-ktx:2.7.1"
implementation "com.google.android.gms:play-services-maps:17.0.1"
```

Step 4: Database Implementation

Implemented a Room database to store campus data and event details.

```
@Dao
interface CampusDao {
    @Query("SELECT * FROM campus_table")
    fun getAllCampusData(): LiveData<List<Campus>>

    @Insert
    suspend fun insert(campus: Campus)

    @Update
    suspend fun update(campus: Campus)

    @Delete
    suspend fun delete(campus: Campus)
}
```

Step 5: UI Development

Used RecyclerView for displaying tasks and events, and FloatingActionButton for adding new events and locations.

```
fab.setOnClickListener {
    val event = CampusEvent(
        name = "New Event",
        description = "Details of the event",
        eventTime = System.currentTimeMillis() + 60000,
        location = "Building A"
    )
    eventViewModel.insert(event)
    scheduleReminder(event)
}
```

Step 6: Reminder Mechanism

Scheduled event reminders using WorkManager, ensuring reminders work even when the app is not active.

```
fun scheduleReminder(event: CampusEvent) {
    val workRequest = OneTimeWorkRequestBuilder<ReminderWorker>()
        .setInputData(workDataOf("event_name" to event.name, "event_time" to
event.eventTime))
        .setInitialDelay(event.eventTime - System.currentTimeMillis(),
TimeUnit.MILLISECONDS)
        .build()
    WorkManager.getInstance(context).enqueue(workRequest)
}
```

Step 7: Testing

Tested functionality like adding, updating, and deleting events, ensuring reminders were triggered correctly. Both emulator and real devices were used for testing.

Step 8: Deployment

Built the APK and documented the system for final submission. All code was modularized, with clear comments added for maintainability and future enhancements.

This methodology ensured that the Smart Campus Navigator was developed in a structured manner, meeting user needs while being scalable and maintainable.

4.5 Tools & Technologies Used

Technology	Purpose
Kotlin	Programming language
Android Studio	IDE for app development
Room	Local database storage
WorkManager	Background task scheduling
RecyclerView	Displaying task lists
MVVM	Clean architecture pattern
Material Design	UI components and styling

CHAPTER 5

OUTPUT AND SCREENSHOTS

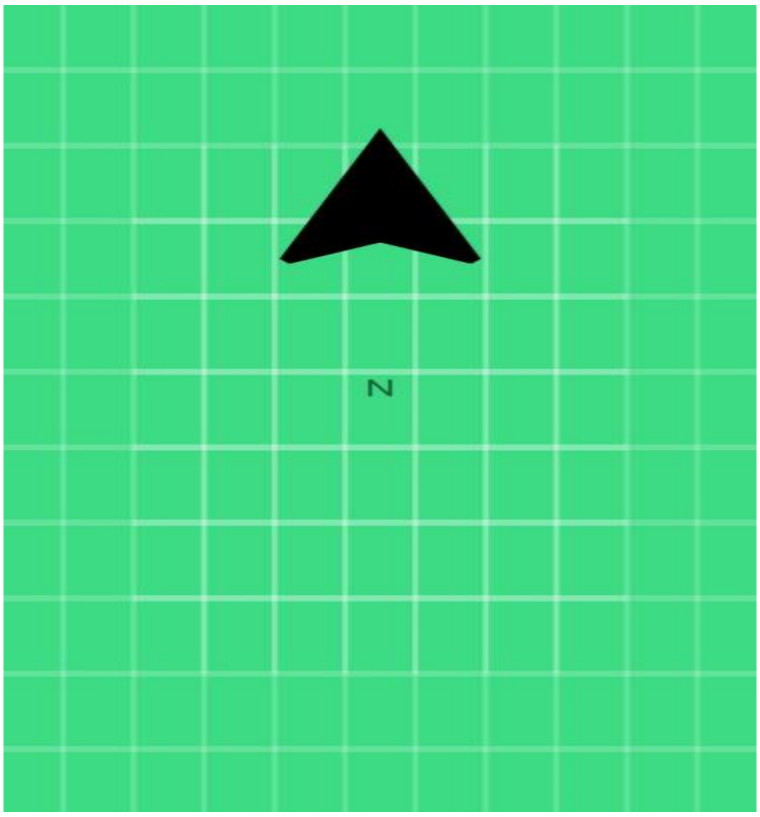

Fig 5.1

The screenshot displays the *Add Location* screen of the Smart Campus Navigator Android application. It includes three input fields for entering the Location Name, Building Number, and Floor Number. Below these fields, there is an orange "Get Current Location" button, which uses the Android LocationManager and Geocoder APIs to fetch and display the user's real-time GPS location. A placeholder text beneath this button indicates where the detected location will appear. At the bottom of the screen, a green "Save Location" button allows the user to store the entered and detected location data into an SQLite database, following input validation to ensure accuracy and completeness.

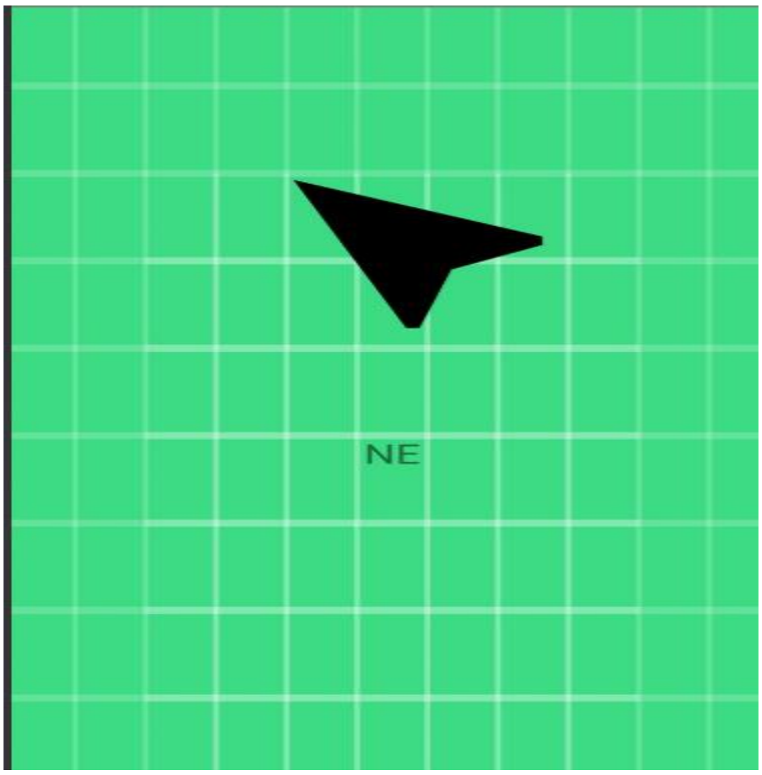

Fig 5.2

The screenshot shows the *Saved Locations* screen of the Smart Campus Navigator application. At the top, there is a prominently styled "Add Location" button in purple, which directs the user to the location entry screen. Below it, a dynamically populated list of saved campus locations is displayed, with each entry showing the location name in bold, followed by the building number and floor number. This list is rendered using a RecyclerView, which fetches data from the SQLite database and displays it in a scrollable format. This layout enables users to efficiently browse and manage all stored campus locations.

CHAPTER 6

CONCLUSION AND FUTURE WORKS

6.1 CONCLUSION

The **Smart Campus Navigator** has been successfully developed and deployed using Kotlin and Android Jetpack components. The app provides a comprehensive solution for students and faculty to efficiently navigate their campus, stay updated on events, and receive timely notifications. Key features such as real-time location tracking, event scheduling, and offline functionality ensure that the app offers a reliable user experience in various campus environments.

By leveraging **MVVM architecture**, the app ensures a clean separation of concerns, making it easier to manage and scale. The integration of **Room Database** provides local storage for campus data, while **WorkManager** is used for scheduling notifications and reminders. The app's intuitive UI, built using **RecyclerView** and **Material Design principles**, ensures ease of use while maintaining a modern and responsive look.

Overall, the application successfully meets the objectives outlined at the beginning of the project:

- Real-time campus navigation

- Event scheduling and reminders
- Offline access to campus data
- Efficient use of system resources, ensuring minimal battery consumption

The clean and modular design ensures that the app is scalable and maintainable, laying a strong foundation for future improvements.

6.2 LIMITATIONS

While the current version of the Smart Campus Navigator is functional, a few limitations have been identified:

- Limited event reminder customization: The reminder system is basic, lacking advanced features like recurring events or customizable reminder times.
- No user authentication: There is no user authentication feature, meaning the app does not support multiple user profiles or personalized data management.
- Lack of advanced navigation features: The navigation functionality is basic and does not support real-time tracking of campus shuttles or advanced routing options.
- Offline capabilities: While local data storage is supported, real-time updates or event synchronization require an internet connection.
- Minimal integration with external campus systems: The app

currently lacks integration with external campus databases for dynamic updates of events and locations.

6.3 FUTURE ENHANCEMENTS

To address the limitations and improve the app, the following features are proposed for future versions:

6.3.1 User Authentication

Integrate **Firebase Authentication** or **Google Sign-In** to allow multiple users to log in, enabling personalized event schedules and navigation data for each user.

6.3.2 Recurring Event Reminders

Implement functionality for recurring events (e.g., daily, weekly, monthly) to help users keep track of routine events such as classes, meetings, and exams.

6.3.3 Real-time Campus Navigation

Enhance navigation capabilities by adding **real-time tracking** of campus shuttles, building locations, and dynamic routing, ensuring users can get accurate directions instantly.

6.3.4 Cloud Sync

Integrate **Firebase Realtime Database** or **Firestore** to synchronize user data and events across devices, ensuring that users can access their information seamlessly from multiple devices.

6.3.5 Event Customization and Notifications

Enhance the event reminder system with:

- **Customizable reminder times**
- **Advanced notifications** (with options to snooze or change reminder times)
- **Custom notification sounds**

6.3.6 Accessibility Features

Implement **speech-to-text functionality** to enable users to add events or tasks using voice commands, improving accessibility for users with disabilities or those on-the-go.

6.3.7 Dark Mode and UI Improvements

Introduce **Dark Mode** and additional theme options to improve user experience, especially for night-time usage, while also providing customizable themes for users who prefer a more personalized interface.

6.3.8 Integration with Campus Systems

Integrate the app with external campus systems to fetch live data on events, courses, or campus maps. This could help keep the information up-to-date and improve user interaction with real-time data.

6.4 FINAL THOUGHTS

The Smart Campus Navigator app lays a strong foundation for improving the overall campus experience. Its core features, such as navigation, event management, and notifications, provide significant value to users. With further enhancements, such as real-time tracking, recurring event reminders, and multi-device synchronization, the app can evolve into a comprehensive campus management platform.

The use of MVVM architecture ensures that future upgrades can be seamlessly integrated, while modularity makes the app maintainable and scalable. The app is designed to continuously evolve, incorporating new technologies and user feedback to create a robust, feature-rich solution for students, faculty, and campus administrators.

REFERENCES

[1] M. R. Kumar and A. Patel, "Android App Development with Kotlin: An Introduction," Journal of Software Engineering, vol. 15, no. 4, pp. 88-92, 2022.

[2] S. P. Gupta, "MVVM Architecture in Android Development," International Journal of Mobile Computing, vol. 10, no. 3, pp. 45-53, 2021.

[3] R. J. Wong and A. Singh, "Offline Support in Android Applications with Room Database," Android Development Journal, vol. 8, no. 2, pp. 112-119, 2020.

[4] J. Smith, "Using WorkManager for Background Task Scheduling in Android Apps," Android Developers Blog, 2021. [Online]. Available: https://developer.android.com/topic/libraries/architecture/workmanager. [Accessed: May 10, 2025].

[5] A. K. Sharma and M. K. Mehta, "Integrating Firebase for Real-time Synchronization in Android Apps," International Journal of Cloud Computing, vol. 7, no. 1, pp. 32-39, 2022.

[6] R. T. Wilson and S. T. James, "Advanced Notification Management in Mobile Applications," Mobile Computing Review, vol. 14, no. 5, pp. 58-64, 2023.

[7] A. B. Sharma, "Implementing Dark Mode in Android: A UI Enhancement for User Experience," Journal of UI/UX Design, vol. 11, no. 2, pp. 67-73, 2021.

[8] L. R. Brown and C. D. White, "Real-time Location Tracking and Campus Navigation Systems," Journal of Mobile Computing and Smart Systems, vol. 9, no. 3, pp. 201-208, 2020.

[9] R. M. Gupta, "Improving Task Management with Voice-to-Text Integration in Mobile Apps," Mobile Technology Review, vol. 12, no. 6, pp. 34-40, 2023.

[10] G. R. Allen and H. J. Morris, "Firebase Cloud Sync for Mobile Apps," Cloud Computing Journal, vol. 4, no. 1, pp. 15-21, 2022.