LECTURE REMINDER MOBILE APPLICATION

BY

HND/23/C0M/FT/0038 HND/23/C0M/ FT/0058

HND/23/C0M/ FT/0067 HND/23/C0M/ FT/0172

HND/23/C0M/ FT/0284 HND/23/C0M/ FT/0306

HND/23/C0M/ FT/0307 HND/23/C0M/ FT/0311

HND/23/C0M/ FT/0355 HND/23/C0M/ FT/0358

HND/23/C0M/ FT/0432 HND/23/C0M/ FT/0443

HND/23/C0M/ FT/0593

DEPARTMENT OF COMPUTER SCIENCE

INSTITUTE OF INFORMATION AND COMUNICATION TECHNOLOGY (IICT)

KWARA STATE POLYTECHNIC ILORIN, KWARA STATE.

IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR THE AWARD OF HIGHER NATIONAL DIPLOMA (HND) IN COMPUTER SCIENCE

JUNE, 2025

CERTIFICATION

This is to certify that this project work is carried out and it has been read and approved as meeting the requirements for the award of Higher National Diploma in the Department of Computer Science, Institute of Information and Communication Technology (IICT), Kwara State Polytechnic Ilorin, Kwara State.

_____ _____

MR. OYEDEPO F.S. DATE

Project Supervisor

_____ _____

MR. OYEDEPO F.S. DATE

Head of Department

_____ _____

EXTERNAL EXAMINER DATE

DEDICATION

This project work is dedicated to Almighty Allah.

ACKNOWLEDGEMENT

I am immensely grateful to Almighty Allah for the source of my knowledge, wisdom, and protection throughout my study. I offer the deepest prayers and heartfelt greetings to my beloved parents, Alh. Alabi Suleiman and Mrs Alabi may Allah continue to bless, strengthen, and uplift them.

My sincere appreciation also goes to my supervisor, Mr. Oyedepo F.S., for his invaluable guidance and corrections. I warmly acknowledge my fellow colleagues at school: Jamiu Abubakar, Rasaq Qudus (Olatech), Suleiman Fathia, and my cousin Alabi Qudus may Allah crown your efforts with success.

A special prayer goes out to my siblings: Alabi Shakirudeen (Professor), The Twins, and our lovely ladies Ummu Hameedah, Sister Aishat, and little sister Idayat. May your lives be filled with wisdom,

joy, and divine guidance. Ameen.

Firstly, all our profound gratitude to my supervisor, Mr. Oyedepo F.S. who help in all needed assistance and perform some correction to this research to bring it to a state of new challenge and we also appreciate all lecturers of my department.

Abstract

The Lecture Reminder Mobile Application is an innovative solution designed to assist students and

lecturers in effectively managing their academic schedules and improving punctuality. By leveraging mobile technology, the application provides features such as personalized lecture notifications, timetable organization, and seamless user interaction. With the capability to input and store lecture details including time, date, subject, and location users receive timely alerts that minimize missed classes and enhance preparedness. The application integrates modern design principles to ensure accessibility and usability, catering to a diverse range of academic needs. Through robust testing and feedback-driven refinement, this tool aims to empower students with efficient schedule management, fostering a structured and stress-free learning experience. As academic demands grow, this project highlights the vital role of technological advancements in bridging gaps in time management and optimizing educational outcomes.

Keywords: Lecture, Reminder, Mobile, Application, Lecturer, Student.

Chapter One

1.1 Background of the Study

In today's fast-paced educational environment, students are often overwhelmed by the multitude of responsibilities they face, including coursework, extracurricular activities, part-time jobs, and personal commitments. As a result, managing time effectively becomes a significant challenge. One of the critical aspects of academic success is attending lectures, which serve as the foundation for understanding course material and engaging with instructors. However, students frequently miss classes due to poor time management, forgetfulness, or conflicting schedules.

The traditional methods of managing lecture schedules, such as paper planners or static digital calendars, often fall short in providing timely reminders and notifications. While digital calendars can help organize schedules, they typically lack the proactive features necessary to alert students about upcoming lectures in a timely manner. This gap in functionality can lead to missed classes, decreased academic performance, increased stress levels e.t.c among students.

With the rapid advancement of mobile technology and the widespread use of smartphones, there is

a growing opportunity to leverage these devices to enhance students' academic experiences. Mobile applications have become an integral part of daily life, providing users with tools for communication, organization, and productivity. By developing a dedicated Lecture Reminder System, we can harness the capabilities of mobile technology to create a solution that addresses specific needs of students.

The Lecture Reminder System aims to provide a user-friendly platform that allows students to input their lecture schedules and receive timely notifications about upcoming classes. By integrating features such as customizable reminder settings, synchronization with existing calendar applications, and user-friendly interfaces, the lecture reminder system can significantly improve students' ability to manage their time effectively.

Moreover, educational institutions are increasingly adopting digital tools to facilitate and improve students engagement. Therefore, the implementation of such a system aligns with the broader trend of utilizing technology to enhance educational outcomes. A Lecture Reminder System not only supports students in their academic endeavors but also contributes to the overall goal of fostering a more organized and efficient learning environment.

In summary, the Lecture Reminder System is designed to address the challenges faced by students in managing their lecture schedules. By leveraging mobile technology, the system aims to provide timely reminders, enhance academic performance, and ultimately contribute to a more successful educational experience.

A lecture reminder mobile application is a digital tool designed to assist students, educators, and other users in managing their academic schedules efficiently. The primary purpose of such an application is to provide timely reminders about upcoming lectures, assignments, or events, ensuring that users stay organized and never miss important academic activities.

This application often come with features like:

Customizable Notifications: Users can set reminders for lectures based on their timetable or preferences.

Integration with Calendars: Many apps sync seamlessly with digital calendars to ensure automatic

updates.

Personalization: Options to tailor reminders based on the course, subject, or event.

Enhanced Accessibility: Mobile platforms ensure that users can access reminders anywhere, anytime.

By leveraging modern technology, lecture reminder apps help improve time management, boost attendance, and enhance academic productivity. They cater to a wide range of users, from students managing hectic schedules to educators organizing multiple classes.

1.2 Statement of the Problem

In the contemporary educational landscape, students face numerous challenges in managing their academic responsibilities, particularly in keeping track of their lecture schedules. Many students struggle with time management due to various factors, including busy lifestyles, extracurricular commitments, and the overwhelming amount of information they must process daily. As a result, they often forget or overlook important lectures, leading to missed classes, gaps in knowledge, and ultimately, a decline in academic performance.

Current methods for managing lecture schedules, such as paper planners or generic digital calendars, are often insufficient. These tools do not provide proactive reminders or notifications tailored to students' specific needs, which can result in a lack of awareness about upcoming lectures. Furthermore, students may find it cumbersome to manually input and update their schedules, leading to inconsistencies and errors.

The absence of a dedicated system that offers timely reminders and integrates seamlessly with students' existing digital tools exacerbates the problem. This gap in functionality not only affects students' attendance but also contributes to increased stress and anxiety related to academic performance.

Therefore, there is a pressing need for a Lecture Reminder System that addresses these challenges by providing a user-friendly platform for students to manage their lecture schedules effectively. The system should offer customizable notifications, easy integration with existing calendar applications, and a streamlined interface that encourages regular use. By implementing such a solution, we can

enhance students' ability to stay organized, improve attendance rates, and ultimately support their academic success.

1.3 Aim and Objectives of the Study

Aim

The aim of this study is to develop a lecture reminder system to assist students and lecturers in managing their lecture schedules and ensuring timely attendance.

Objectives

In order to achieve the aim mentioned above, certain objectives must be considered. Therefore, the objectives are to:

design a user-friendly Lecture Reminder System;

implement a mobile application that sends notifications for upcoming lectures;

integrate the system with existing calendar applications for seamless scheduling; and

improve student attendance and punctuality.

1.4 Significance of the Study

The Lecture Reminder System is significant not only for its immediate benefits to students in managing their lecture schedules but also for its broader impact on academic performance, mental well-being, and the overall educational experience. By addressing the challenges of time management and attendance, the lecture reminder system plays a crucial role in fostering a more organized, engaged, and successful student body.

1.5 Scope of the Study

The project targets students, lecturers and academic institutions. The project focuses on developing a mobile application for Android devices, with potential future expansions to iOS and web platforms. The system will support basic functionalities such as adding, editing, and deleting lecture schedules. The Lecture Reminder System is designed to provide a focused and effective solution for students and lecturers to manage their lecture schedules, while also acknowledging the limitations and potential for future development. The system aims to enhance academic performance by improving attendance and time management through timely reminders and a user-friendly interface.

1.6 Definition of terms

Lecture Reminder: A feature or tool designed to alert users about upcoming lectures or academic events to ensure timely attendance.

Mobile Application (App): A software application developed specifically to run on smartphones and tablets, often providing convenience and mobility for users.

Timetable Management: The process of organizing, scheduling, and managing time-related academic activities, such as lectures, classes, or exams.

Push Notifications: Automated messages sent by an app to a user's device, used to remind or notify them of important events, like lecture timings.

Real-Time Updates: Live updates provided by the application to inform users of any changes in the lecture schedule or other related information.

User Interface (UI): The visual layout and design of the app that allows users to interact with its features effectively.

Scheduling Algorithm: A computational method used to optimize the scheduling of lectures and other academic events based on certain criteria or constraints.

Cross-Platform Compatibility: The ability of an app to function seamlessly on multiple operating systems, such as Android and iOS.

Chapter Two

2.0 Literature Review

2.1 Review of Past Work on the Subject

Past researches on lecture reminder mobile applications demonstrate the utility of these tools in improving student and lecturer engagement with academic schedules. Several studies have explored the development and implementation of such apps using various technologies like Android, Java, and web-based systems. These apps aim to reduce missed lectures and enhance communication between administrators and users.

Android Lecture Timetable Application for Tertiary Institutions (2015) addresses challenges like timetable misinterpretation and forgetfulness among students and lecturers. It includes features such as reminder alerts for lecture times and uses technologies like Visual C#, Android SDK, and SQLite. This system minimizes missed lectures by providing timely reminders. It utilizes Java, Eclipse, Android SDK, PHP, and MySQL. Design and Implementation of Lecture Reminder System explores the development of a lecture reminder system using web-based technologies. It highlights advantages and disadvantages of existing systems and proposes improvements.

2.2 Review of General Texts

Lecture Time Table Reminder System application was developed for students at the Federal University of Technology, Akure, Nigeria. It aimed to address the issue of students forgetting lecture schedules. The app utilized tools like Java, Eclipse, Android SDK, PHP, and MySQL for development. Design and Implementation of Lecture Reminder System explored the creation of a lecture reminder system using web-based technologies. It included features like push and pull SMS applications, file security technologies, and a detailed system design. Android Lecture Timetable Application designed for tertiary institutions to help students and lecturers manage lecture schedules effectively. It included features like reminder alerts and was developed using Visual C#, Android SDK, and SQLite.

2.3 Historical Backgrounds of the System

The concept began with basic tools like personal organizers and desktop calendars, which helped users keep track of their schedules manually. These systems laid the foundation for automated reminders. With the advent of mobile technology, SMS-based reminder systems emerged. These systems utilized push and pull SMS applications to notify users about upcoming lectures, leveraging the widespread use of mobile phones. The integration of web portals and databases allowed for more sophisticated systems. These platforms provided features like lecture schedules, reminders, and resource allocation, enhancing accessibility and usability. The development of mobile apps

revolutionized lecture reminders. Applications like the Android Lecture Timetable Application introduced features such as reminder alerts, user-friendly interfaces, and real-time updates, making them indispensable tools for students and lecturers.

2.4 Concept of the System

The concept of a lecture reminder mobile application revolves around utilizing modern technology to assist students, educators, and academic institutions in managing and remembering lecture schedules.

Purpose: The primary goal of a lecture reminder app is to help users avoid missing lectures by providing timely notifications and updates about scheduled classes. It enhances punctuality, organization, and efficiency.

Core Features:

Lecture Scheduling: Users can input or sync their lecture timetables.

Reminders and Alerts: Notifications sent before the start of each lecture or event.

Customization: Options to set reminders based on user preferences (e.g., time intervals, specific days).

Real-Time Updates: Adjustments to schedules due to cancellations or changes are instantly communicated.

Functionality: These applications often combine user-friendly interfaces with database management systems to store and retrieve timetable information. Some apps also incorporate advanced features like artificial intelligence for optimizing schedules or integrating with academic platforms.

Benefits:

Reduces the likelihood of missing lectures.

Promotes better time management.

Serves as a centralized platform for academic scheduling.

2.5 Theories of the System

Time Management Theory emphasizes the importance of managing time effectively to achieve

goals. Lecture reminder apps align with this by helping users organize their schedules and prioritize academic activities. Human-Computer Interaction (HCI) principles guide the design of user-friendly interfaces for lecture reminder apps. These principles ensure that the app is intuitive, accessible, and meets the needs of its users. Behavioral Reinforcement psychological theory suggests that reminders act as positive reinforcements, encouraging users to attend lectures and stay on track with their schedules. Information Systems Theory underpins the development of apps as systems that collect, process, store, and disseminate information. Lecture reminder apps use databases and algorithms to manage and deliver schedule-related data. Mobile Learning (m-Learning) Theory explores the use of mobile devices to support learning. Lecture reminder apps contribute to m-learning by ensuring students are aware of their academic commitments. Cognitive Load Theory by providing timely reminders, these apps reduce the cognitive load on users, allowing them to focus on learning rather than remembering schedules.

Chapter Three

3.1 Research Methodology

The methodology for developing a lecture reminder mobile application involves a systematic process to ensure it meets user needs effectively.

1. Requirement Analysis

Conduct surveys or interviews with potential users (students, lecturers) to gather insights into their needs and preferences.

Identify key features like scheduling, notifications, and calendar integration.

Define the target platform (Android, iOS, or both).

2. Planning

Create a project timeline with milestones for design, development, and testing phases.

Allocate resources such as developers, designers, and testers.

Select the appropriate development tools (e.g., CSS, Reactjs) and back-end technologies (e.g., Tailwind).

3. Design

User Experience (UX) Design: Map out user journeys to create an intuitive flow for the app.

User Interface (UI) Design: Develop wireframes and prototypes with appealing visual elements.

Incorporate accessibility standards to ensure usability for all users.

4. Development

Front-End Development: Code the user interface, ensuring responsiveness and interactivity.

Back-End Development: Set up databases, authentication systems, and APIs for functionalities like notifications and schedule syncing.

Integrate push notification services (e.g., Firebase Cloud Messaging).

5. Testing

Conduct multiple rounds of testing, including:

Unit Testing: Test individual components of the app.

Integration Testing: Ensure all components work seamlessly together.

User Acceptance Testing (UAT): Gather feedback from end users to identify areas for improvement.

6. Deployment

Publish the app on app stores.

Ensure compliance with platform-specific guidelines for app submission.

7. Maintenance and Updates

Monitor app performance and user feedback post-launch.

Release regular updates to fix bugs, improve features, and enhance security.

3.2 Analysis of the Existing System

Strengths:

Automation: Existing systems effectively automate the process of reminding users about lectures, reducing the chances of missing classes.

User-Friendly Interfaces: Many applications are designed with intuitive interfaces, making them accessible to students and lecturers.

Integration: Some systems integrate with academic calendars and provide customizable alerts, enhancing their utility.

Accessibility: Mobile platforms ensure that users can access reminders anytime and anywhere.

Weaknesses:

Limited Features: Some systems lack advanced functionalities, such as AI-driven scheduling or cross-platform compatibility.

Dependency on Internet Connectivity: Certain applications require constant internet access, which can be a limitation in areas with poor connectivity.

Scalability Issues: Some systems struggle to handle large-scale data or multiple users simultaneously.

Opportunities for Improvement:

Incorporating machine learning algorithms for personalized scheduling.

Enhancing offline functionality to cater to users in areas with limited internet access.

Expanding cross-platform compatibility to include more devices and operating systems.

3.3 Problem of the Existing System

Lack of Personalization:

Many existing systems fail to provide personalized reminders tailored to individual user preferences, such as specific times for alerts or recurring schedules.

Internet Dependency:

A number of applications require constant internet access, which can be inconvenient for users in areas with limited or unstable connectivity.

Limited Features:

Some apps do not offer advanced functionalities, such as synchronization with external calendars, integration with academic platforms, or support for group notifications.

Scalability Issues:

Existing systems often struggle to handle a large number of users simultaneously, which can lead to system crashes or delayed notifications.

User Interface Challenges:

Poorly designed interfaces can make navigation difficult, leading to a less-than-optimal user experience.

Compatibility Constraints:

Some apps are designed for specific operating systems and lack cross-platform compatibility, limiting their accessibility for a broader audience.

Data Security Concerns:

Inadequate data protection measures can expose sensitive user information, such as schedules or personal data, to security risks.

Inflexibility with Dynamic Changes:

Many systems are unable to adapt to real-time changes in lecture schedules, leading to outdated or irrelevant reminders.

3.4 Description of the Proposed system

A lecture reminder mobile application is a software tool designed to assist students and educators in managing their academic schedules.

Purpose:

The app serves as a virtual assistant for organizing and tracking lecture schedules, helping users stay punctual and prepared.

Core Features:

Lecture Timetable Management: Users can create and store their lecture schedules within the app.

Reminder Notifications: Timely alerts are sent to remind users of upcoming lectures or changes to the timetable.

Customization: Allows users to set personalized notification preferences, such as specific times or

recurring reminders.

Integration: Some apps can sync with calendars or academic platforms for seamless scheduling.

Design:

The app typically features a user-friendly interface with options for inputting, viewing, and editing schedules.

Modern versions may include graphical representations like calendar views or timeline charts for better visualization.

Functionality:

Powered by a combination of database management systems and scheduling algorithms, the app ensures that all data is organized, accurate, and accessible.

Offline capabilities may allow users to access their schedules without an active internet connection.

Benefits:

Enhances time management and academic organization.

Reduces the risk of missed lectures or conflicting schedules.

Provides flexibility in schedule updates and modifications.

3.5 Advantages of Proposed System

Improved Time Management: It helps users stay organized by providing timely notifications about lectures and academic events.

Enhanced Attendance: Regular reminders can reduce missed lectures and improve participation.

Accessibility: Mobile platforms ensure users can access schedules and reminders on-the-go.

Customization: Many applications allow users to personalize reminders according to their preferences or course schedules.

Seamless Integration: Modern apps often integrate with calendars and academic management systems for streamlined planning.

Chapter Four

4.1 Design of the System

System design refers to the architectural framework that defines the components, modules, interfaces, and data flow within a software solution. For the Lecture Reminder Mobile Application, the design encompasses the user interface (UI), backend logic, data structures, and process workflows. It provides a blueprint that aligns with functional requirements, ensures operational efficiency, and supports scalability, maintainability, and security.

The application was designed with a mobile-first approach using modern technologies to ensure high performance and ease of use. This chapter outlines the output and input mechanisms, the structure of the database, and the procedural logic that governs system behavior.

4.1.1 Output Design

Output design focuses on how data is presented to users. In this system, the output is manifested through a dashboard interface that dynamically displays relevant academic information after a successful login. The output is optimized for readability, interactivity, and responsiveness across multiple devices.

Dashboard Interface Overview:

The dashboard serves as the central control panel for users, segregated by roles (students and lecturers), and displays personalized data including lecture schedules, course details, and notifications.

Key Sections of the Dashboard:

Timetable Display:

Shows lectures scheduled for the day, week, or month.

Can toggle between list view and calendar view.

Updates in real-time whenever a change is made to the schedule.

Notification Panel:

Presents system-generated alerts such as upcoming lecture reminders, changes in schedule, or newly added classes.

Notifications are visually prioritized using icons and alert colors (e.g., red for urgent, green for completed).

Courses Overview:

Provides a snapshot of all registered or assigned courses.

Displays course codes, titles, lecture days, and times.

Includes status indicators such as active, completed, or canceled.

Action Buttons:

Strategically positioned buttons for key actions such as Add Class, Add Course, Edit, and Logout.

Designed to enhance user productivity and navigation efficiency.

Technological Implementation:

The output is constructed using React.js components, styled with Tailwind CSS for modular design and responsiveness. React allows dynamic rendering of data, ensuring real-time updates without page reloads. Tailwind ensures consistent styling across components, enhancing user experience.

User-Centered Design Features:

Icons and labels for ease of navigation.

Dark/light mode support for accessibility.

Mobile responsiveness using grid and flex layouts.

Tooltips and status indicators for improved clarity.

4.1.2 Input Design

Input design involves crafting the mechanisms through which users interact with the system to submit or modify data. It emphasizes usability, accuracy, validation, and efficiency. In this system, input interfaces are structured into modals and forms optimized for minimal user effort and maximum clarity.

Key Input Interfaces:

1. Sign-In Form:

Purpose: Grants users access to personalized dashboards.

Fields:

User ID (Student ID or Staff ID)

Password

Features:

Password masking for security

Inline error messages for invalid credentials

Role-based redirection post-authentication

Validation: Implemented via Firebase Authentication with custom validation hooks.

2. Add Classes Modal:

Purpose: Allows users to add specific lecture sessions to their personal schedules.

Fields Include:

Class Title

Date

Start Time

End Time

Venue/Location

Design Highlights:

Uses date-picker and time-picker components

Auto-complete venue input for efficiency

Functionality: Once submitted, class details are stored in the database and reflected in the timetable view.

3. Add Courses Modal:

Purpose: Enables users to register new courses to the system.

Fields Include:

Course Code

Course Title

Lecturer Name

Day of the Week

Time Slot

Validation Features:

Format checking for course codes

Time conflict detection

Post-Submission: Course is linked to the user's account and appears under the Courses section.

Design Principles Used:

Mobile-first layout

Real-time validation feedback

Input masking for date/time formats

Consistent error messaging

Keyboard and screen reader accessibility

4.1.3 Database Design

The database design outlines how data is stored, retrieved, and managed. The system utilizes Firebase Firestore, a cloud-hosted NoSQL database that offers flexibility, real-time synchronization, and high scalabilityideal for mobile applications.

Primary Data Collections:

Users Collection:

Stores user credentials, role identification, and basic profile data.

Example:

```
{
  "userId": "STD001",
  "name": "Alice Johnson",
  "role": "student",
  "email": "alice@school.edu"
}
```

Courses Collection:

Contains all course records available in the system.

Fields:

courseCode: "CSC101"

courseTitle: "Introduction to Computer Science"

assignedLecturer: "Dr. John Smith"

scheduleDays: ["Monday", "Wednesday"]

Schedules Collection:

Stores individual lecture events.

Fields:

courseId: "CSC101"

date: "2025-06-28"

startTime: "10:00"

endTime: "11:30"

venue: "Lecture Hall A"

Notifications Collection:

Tracks generated reminders for lectures.

Stores data such as message content, timestamp, userId, and delivery status.

Database Design Considerations:

Indexing: Enables fast queries based on user ID or date.

Data Redundancy Prevention: Maintains separate but relationally linked collections.

Security Rules: Firebase security policies restrict data access by user role.

Backup and Redundancy: Cloud-based replication ensures data durability.

4.1.4 Procedure Design

Procedure design defines the step-by-step operations executed to handle user requests and system functions. It ensures a structured approach to managing tasks within the application. The Lecture Reminder App follows a modular, event-driven architecture, where each operation is encapsulated in clearly defined procedures.

Core System Procedures:

Authentication Procedure:

Accepts input credentials.

Validates via Firebase Authentication service.

Determines user role (student/lecturer).

Routes to appropriate dashboard based on role.

Course Addition Procedure:

User clicks "Add Course" Modal opens.

User enters course data Form validates input.

Data is written to the Courses collection.

Courses overview and dashboard update instantly.

Class Scheduling Procedure:

User selects "Add Class" Fills schedule info.

Schedule is validated to check for time overlaps.

Class is saved in the Schedules collection.

Dashboard reflects the new entry immediately.

Notification Dispatch Procedure:

System scans upcoming schedules hourly.

For lectures due within 3060 minutes, a push notification is generated.

Notification is sent via Firebase Cloud Messaging (FCM).

Logs are updated in the Notifications collection.

Schedule Update Procedure:

User edits an existing course or class.

System updates corresponding entries in Firestore.

Affected notifications are recalculated and updated accordingly.

System Goals for Procedure Design:

Minimize processing time

Prevent data duplication

Ensure real-time responsiveness

Maintain clear audit trails of user actions

4.2 System implementation

4.2.1 Choice of Programming Language

The selection of tools and languages was guided by the need for performance, responsiveness, and ease of deployment:

Frontend:

React.js: Enables reusable component-based development.

Tailwind CSS: For utility-first styling and responsive design.

Backend and Services:

Firebase Authentication: For secure login and session management.

Firebase Firestore: For NoSQL database functionality.

Firebase Cloud Messaging (FCM): For handling push notifications.

4.2.2 Hardware Support

The Lecture Reminder Application is optimized for the following hardware:

Mobile Devices: Android smartphones (primary target), iOS devices (future support).

Desktop Systems: Compatible for lecturers and administrators using browser-based access.

Internet Requirements: Internet connectivity is essential for login, synchronization, and notification functionality.

4.2.3 Software Support

The application runs in a cloud-based environment, relying on third-party services for scalability and maintainability.

Development & Hosting Platform: Vercel (for frontend deployment).

Database & Notification Management: Firebase Console.

Supported Browsers: Chrome, Firefox, Safari, Microsoft Edge.

Third-party APIs: Used for date/time handling, calendar views, and notification management.

4.2.4 Implementation Techniques

The system was implemented using an Agile Development Lifecycle, allowing for incremental progress and continuous feedback.

Key techniques include:

Component-Based Architecture: Encourages code reuse and maintainability.

Mobile-First Design: Ensures excellent performance on smartphones.

State Management with React Hooks: For maintaining application state.

CI/CD Integration via Vercel: For seamless deployment of new features.

Push Notification Setup: Via Firebase SDK for real-time communication with users.

4.3 System documentation

4.3.1 Program Documentation

Source Code Structure: Organized into logical modules (e.g., Auth, Dashboard, Courses, Notifications).

Version Control: Maintained via GitHub with commit tracking.

Dependencies: React, React Router, Tailwind CSS, Firebase SDK.

Security: Enforced through Firebase security rules and hashed user credentials.

4.3.2 Operating the System

To operate the system:

Users navigate to .

Login using default credentials:

Student ID: STD001 / STD002

Lecturer ID: STAFF001 / STAFF002

Password: password123

Upon successful authentication, users are directed to their dashboard.

From the dashboard, users can:

View schedules

Add/edit classes and courses

Receive and respond to reminders

4.3.3 Maintaining the System

System maintenance involves periodic updates, monitoring performance, and resolving issues.

Bug Fixing: Handled through issue tracking and continuous integration.

Updates: Feature enhancements and security patches deployed regularly.

Performance Monitoring: Conducted through Firebase Console analytics.

User Feedback: Collected via surveys or in-app feedback options.

Data Backups: Automatic backups via Firebase and GitHub.

Chapter Five

5.1 Summary

Automated Reminders alerts for upcoming lectures based on preset schedules. Customizable Notifications allows users to set personalized reminders for specific subjects or tasks. Sync with Calendar integrates with existing calendars to avoid scheduling conflicts. Offline Mode ensures reminders function even without internet access. User-Friendly Interface allows simple navigation and clear display for ease of use. Data Backup & Security stores information securely with cloud backup options.

5.2 Conclusions

The Lecture Reminder Mobile Application is a valuable tool designed to enhance student productivity by ensuring timely notifications for lectures and assignments. By integrating automated reminders, calendar synchronization, and customizable notifications, this app minimizes missed classes and helps students stay organized.

Its user-friendly interface and offline functionality make it a reliable academic assistant, fostering better time management and improved learning experiences. With secure data storage and backup

options, students can trust the app to keep their schedules intact.

Overall, this application contributes to academic success by reducing forgetfulness, improving planning efficiency, and supporting students in maintaining a structured study routine.

5.3 Recommendation

User Feedback and Usability Testing: Gather feedback from users on the current application's functionality, interface, and overall experience. Conduct usability testing to identify areas for improvement in terms of user flow, design, and performance.

Integration with Campus Systems: Explore integration with other campus systems like the student portal, learning management system (LMS), and library system. This could allow for features like automatic timetable import, course material access, and seamless integration with online learning platforms.

References

Acuna,A.A, Sabilli M.A.P & Friginal, F.F.S(2024). MARA: A Mobile-based Academic Reminder Application. International Journal of Computing Sciences Research, 8, 2734-2748. http://doi-org/10.25147//ijcsr.2017.001.184.

Aung, M. (2019). Mobile Academic Reminder Systems and Academic performance: A correlation Analysis. International Journal of Educational Technology, 34(2), 123-140

Nik R., Nik M., Siti H. & Natasha M. (2020). Design and development of mobile application for academic reminder system. Journal of Computing Research and Innovation (JCRINN) 5 (4), 18-26.

Bhavani Y. & Sanjay D. (2017). Android based Student Reminder System. Oriental Journal of Computer Science and Technology 10 (4), 760-764.

Robin N.B., Meredith R.M. & Sian E.L. (2017).How to remember what to remember: exploring possibilities for digital reminder systems. Proceedings of the ACM on interactive, mobile, wearable and ubiquitous technologies 1 (3), 1-20.

Nur R. & Lubab S. (2016). Location and time-based reminder system on Android mobile device. 2016 2nd International Conference on Science in Information Technology (ICSITech), 147-151

Kukuh Y.S., Taufiq A. & Slamet W. (2022). Mobile-based Assignment Reminder Application for Students and Lecturers. Journal of Machine Learning and Artificial Intelligence 1 (2), 167-172.

Victoria N.E., Bright U.E. & Victor C.E. (2024) Development and execution of a notification system for lecturers in universities. World Journal of Advanced Research and Reviews 23 (1), 1093-1098.

Appendices

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Nextgen Lecture Reminder</title>
    <meta name="description" content="Nextgen Lecture Reminder" />
    <meta name="author" content="Splashray" />

    <meta property="og:title" content="Nextgen Lecture Reminder" />
    <meta property="og:description" content="Nextgen Lecture Reminder" />
    <meta property="og:type" content="website" />
                                                    <meta                property="og:image"
content="https://i.ibb.co/QvXmG5WS/Screenshot-2025-05-13-195651.png" />

    <meta name="twitter:card" content="summary_large_image" />
    <meta name="twitter:site" content="@splashray_tayo" />
                                                    <meta                name="twitter:image"
content="https://i.ibb.co/QvXmG5WS/Screenshot-2025-05-13-195651.png" />
  </head>
```

```
  <body>
    <div id="root"></div>
    <!-- IMPORTANT: DO NOT REMOVE THIS SCRIPT TAG OR THIS VERY COMMENT! -->
    <script src="https://cdn.gpteng.co/gptengineer.js" type="module"></script>
    <script type="module" src="/src/main.tsx"></script>
  </body>
</html>
import js from "@eslint/js";
import globals from "globals";
import reactHooks from "eslint-plugin-react-hooks";
import reactRefresh from "eslint-plugin-react-refresh";
import tseslint from "typescript-eslint";

export default tseslint.config(
  { ignores: ["dist"] },
  {
    extends: [js.configs.recommended, ...tseslint.configs.recommended],
    files: ["**/*.{ts,tsx}"],
    languageOptions: {
      ecmaVersion: 2020,
      globals: globals.browser,
    },
    plugins: {
      "react-hooks": reactHooks,
      "react-refresh": reactRefresh,
    },
```

```
    rules: {
      ...reactHooks.configs.recommended.rules,
      "react-refresh/only-export-components": [
        "warn",
        { allowConstantExport: true },
      ],
      "@typescript-eslint/no-unused-vars": "off",
    },
  }
);
import type { Config } from "tailwindcss";

export default {
 darkMode: ["class"],
 content: [
  "./pages/**/*.{ts,tsx}",
  "./components/**/*.{ts,tsx}",
  "./app/**/*.{ts,tsx}",
  "./src/**/*.{ts,tsx}",
 ],
 prefix: "",
 theme: {
  container: {
   center: true,
   padding: '2rem',
   screens: {
    '2xl': '1400px'
```

```
 }
},
extend: {
 colors: {
  border: 'hsl(var(--border))',
  input: 'hsl(var(--input))',
  ring: 'hsl(var(--ring))',
  background: 'hsl(var(--background))',
  foreground: 'hsl(var(--foreground))',
  primary: {
   DEFAULT: 'hsl(var(--primary))',
   foreground: 'hsl(var(--primary-foreground))'
  },
  secondary: {
   DEFAULT: 'hsl(var(--secondary))',
   foreground: 'hsl(var(--secondary-foreground))'
  },
  destructive: {
   DEFAULT: 'hsl(var(--destructive))',
   foreground: 'hsl(var(--destructive-foreground))'
  },
  muted: {
   DEFAULT: 'hsl(var(--muted))',
   foreground: 'hsl(var(--muted-foreground))'
  },
  accent: {
   DEFAULT: 'hsl(var(--accent))',
```

```
    foreground: 'hsl(var(--accent-foreground))'
   },
   popover: {
    DEFAULT: 'hsl(var(--popover))',
    foreground: 'hsl(var(--popover-foreground))'
   },
   card: {
    DEFAULT: 'hsl(var(--card))',
    foreground: 'hsl(var(--card-foreground))'
   },
   sidebar: {
    DEFAULT: 'hsl(var(--sidebar-background))',
    foreground: 'hsl(var(--sidebar-foreground))',
    primary: 'hsl(var(--sidebar-primary))',


     opacity: '0',
     transform: 'translateY(5px)'
    }
   }
  },
  animation: {
   'accordion-down': 'accordion-down 0.2s ease-out',
   'accordion-up': 'accordion-up 0.2s ease-out',
   'fade-in': 'fade-in 0.3s ease-out',
   'fade-out': 'fade-out 0.3s ease-out'
  }
 }
```

```
 },
 plugins: [require("tailwindcss-animate")],
} satisfies Config;
```

```
{
  "name": "vite_react_shadcn_ts",
  "private": true,
  "version": "0.0.0",
  "type": "module",
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "build:dev": "vite build --mode development",
    "lint": "eslint .",
    "preview": "vite preview"
  },
  "dependencies": {
    "@hookform/resolvers": "^3.9.0",
    "@radix-ui/react-accordion": "^1.2.0",
    "eslint-plugin-react-refresh": "^0.4.9",
    "globals": "^15.9.0",
    "lovable-tagger": "^1.1.7",
    "postcss": "^8.4.47",
    "tailwindcss": "^3.4.11",
    "typescript": "^5.5.3",
    "typescript-eslint": "^8.0.1",
    "vite": "^5.4.1"
  }
```

```
}
```