TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PURWANCHAL CAMPUS

A MINOR PROJECT PROPOSAL ON PROJECT TITLE

BY

SANGYOG PURI (PUR078BCT078)

SANSKAR RIJAL(PUR078BCT079)

SUJAN NAINAWASTI (PUR078BCT090)

VISION PEER CHAUDHARY (PUR078BCT0)

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

PURWANCHAL CAMPUS

DHARAN, NEPAL

MARCH, 2025

ACKNOWLEDGEMENT

We are very thankful to the Department of Electronics and Computer Engineering for providing this opportunity. It has been a greatly endearing experience.

We would like to express our deepest gratitude to our esteemed supervisor, Asst. Prof. Prasanga Regmi, for his invaluable guidance and support throughout the project.

Additionally, we would like to express our appreciation to Deputy Head of Department (HOD) Pukar Karki for routinely checking on us and ensuring the progress of the project.

We would also like to acknowledge the support and mentorship provided by our seniors, for their insightful suggestions, constructive feedback, and generous allocation of time and resources toward the successful completion of our project.

Sincerely,
Sangyoug Puri
Sanskar Rijal
Sujan Nainawasti
Vision Peer Chaudhary

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

API	: Application Programming Interface
Colab	: Colaboratory
REST	: Representational State Transfer
IOE	: Institute Of Engineering
IOEPC	: Institute Of Engineering Purwanchal Campus
MVVM	: Model-View-ViewModel

CHAPTER 1

INTRODUCTION

1.1 Background

The Institute of Engineering Purwachal Campus (IOEPC) is an education institution in Nepal, offering a range of undergraduate and graduate programs in various engineering disciplines. Despite advancements in technology, many administrative and academic processes at campus remain manual and offline, leading to inefficiencies and delays. Technical processes are still being handled through paper-based methods, which are prone to errors and time-consuming. There is no direct connection between teachers and students, leading to delayed communication due to the dependency on class representatives. To address these issues, we propose the development of a mobile application that integrates all IOE services, providing a centralized and efficient platform for users. This application aims to streamline processes such as attendance management and assessment marks, making it easier for teachers to conduct daily activities and ensuring easy integration of existing offline systems and flexibility for future enhancements.

1.2 Gap Identification

Despite the availability of various educational management systems, there is a lack of tailored solutions that cater specifically to the needs of our college. Our applications address this gap by providing customized functionalities that align with the requirements of students, teachers, and administrators.

1.3 Motivation

The motivation behind this project is to simplify and automate the routine tasks of students, teachers, and administrators. By leveraging technology, we aim to reduce manual efforts, minimize errors, and improve the overall user experience. Our team, comprising students of IOEPC, saw an opportunity to contribute to the institution by developing a solution that benefits both students and the institute.

1.4 Objectives

- Develop a user-friendly application for teachers to manage attendance, internal marks, notes, and notices.
- Create a student application that allows students to view their attendance, internal marks, notes, and notices.
- Implement an admin interface to manage user accounts for students and teachers.
- Improve online access to the educational system by using a monolithic architecture.

1.5 Scope

The proposed applications can be adopted by educational institutions to streamline their academic and administrative processes. The teacher application allows teachers to manage attendance, internal marks, notes, and notices efficiently. The student application provides students with easy access to their academic information. The admin interface enables administrators to manage user accounts for students and teachers, ensuring a smooth and secure user experience. The application will serve as an external interface, facilitating activities such as result publication and online attendance, and integrating existing systems to better serve stakeholders.

1.5.1 Academic Study/Research

Academic research is a crucial scope of our project. By sharing our application implementation with other researchers, they can benefit from the study of educational management systems, optimization techniques, and user experience improvements. Our customizable applications with an easy-to-use interface have an important scope in academia.

1.5.2 Comparison of Scope with Existing Systems

There are various educational management systems available, but there are several key differences between these technologies and our applications. Existing systems may not be tailored to the specific needs of our college, whereas our applications provide customized functionalities that align with the requirements of students, teachers, and administrators. Our applications support real-time data access and management.

CHAPTER 2

RELATED THEORY

2.1 Monolithic Architecture

A monolithic architecture is a software design pattern where all components of an application are integrated into a single code base. This approach simplifies deployment and management, as all functionalities are contained within a single codebase. It can be faster in handling requests when the number of requests is fixed and the load is less [1].

2.2 MVVM Architecture

Model-View-ViewModel (MVVM) is a software architectural pattern that facilitates the separation of the development of the graphical user interface (the view) from the development of the business logic or back-end logic (the model). The view model of MVVM is a value converter, meaning the view model is responsible for exposing (converting) the data objects from the model in such a way that objects are easily managed and presented. This pattern is particularly useful in mobile application development for maintaining a clean separation of concerns and enhancing testability [2].

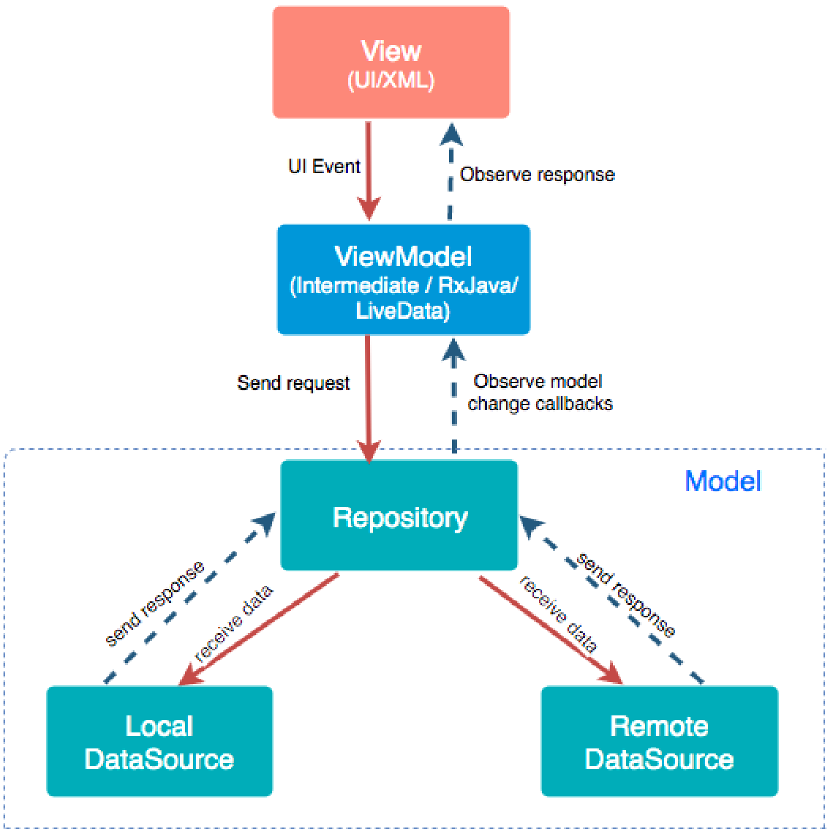

Figure 2.1: MVVM Architecture

2.3 REST API

REST (Representational State Transfer) is an architectural approach for developing web services. A RESTful API (Application Programming Interface) is built using REST concepts. RESTful APIs transfer a representational state of the resource to the endpoint or requester. The information can be of any format like JSON, HTML, PHP, XML, etc.

REST APIs are used for data exchange between client and server. They communicate with the server via HTTP (Hypertext Transfer Protocol) techniques. The most often used HTTP methods in RESTful APIs are GET, POST, PUT, PATCH, and DELETE.

2.3.1 GET

The GET method is used when data is needed to retrieve from the server. It reads the data from the server. It is used to retrieve data like user profiles, product information, or other resources.

2.3.2 POST

The POST method is used to send data to the server. It is useful for creating new resources to be retrieved later on the server.

2.3.3 PUT

The PUT method is used to update an existing resource on the server. It replaces the existing resource with the new one that is sent in the request.

2.3.4 PATCH

The PATCH method is similar to the PUT method, but it only updates the specified fields of an existing resource. It is used when only a portion of a resource needs to be updated, rather than replacing the entire resource.

2.3.5 DELETE

The DELETE method is used to delete a resource from the server. It removes the resource from the server and makes it unavailable for further use.

CHAPTER 3

LITERATURE REVIEW

In 2010, JetBrains created Kotlin to enhance the Java Virtual Machine experience for the user. It supports both object-oriented and functional programming paradigms. With the support of non-nullable types, the applications are immune to null pointer exceptions. Furthermore, Kotlin appeared as the most loved language in the Stack Overflow developer survey for the year 2018 and 2019 [3].

The monolithic architecture has every service of the application interconnected and has a very modular architecture and good decoupling between their internal components. In terms of Node.js, all the services are part of the same code base and run in a single process [4].

3.1 Related Work

An Android App for Kathmandu University displays information about the university, university calendar, course registration, applying for transcripts, news, notices, and student activities. It provides information about student welfare, the central library, publication, and research of the university. This app brings all the major components of their website into a single dashboard with added benefits of personalization [5].

An application for The British College (Student Portal) displays the courses' schedule and exams schedule for students from anywhere and anytime, also notifying the students of student lectures' schedule and exams automatically, viewing the academic information and grades report (marks transcript) for the students. This system depends on the university website directly.

CHAPTER 4

METHODOLOGY

4.1 Overview

The development of this project follows an Agile Development Model, ensuring continuous feedback and iterative improvements. The system is built using a Monolithic Architecture, integrating all components into a single unified application. This structure allows seamless communication between different system modules, improving efficiency and maintainability.

The project is developed using:

- **Front-end:** React (for web) and Kotlin (for mobile).
- **Back-end:** Node.js with PostgreSQL as the database.
- **Version Control:** GitHub and Git tools were used for tracking changes and collaboration.
- **Deployment:** The system is hosted on Azure and Vercel to ensure scalability and high availability.

4.2 Requirement Analysis

To ensure the system meets the needs of students, teachers, and administrators, a requirement analysis phase was conducted. This involved:

- **User Interviews:** Discussions with students and faculty members to identify key functionalities.
- **Comparative Analysis:** Reviewing existing educational management systems to understand their strengths and weaknesses.

Key requirements derived from this phase include:

- A student dashboard for accessing attendance, marks, and notices.
- A teacher dashboard for managing attendance and internal assessments.

- An admin panel for user account management.

4.3 System Architecture

The system is developed using a Monolithic Architecture, where all functionalities are integrated into a single codebase. This approach simplifies development, deployment, and maintenance.

4.3.1 Architecture Components

- **Front-end (React & Kotlin)**
  - React is used for the web interface.
  - Kotlin is used for Android-based mobile applications.
- **Back-end (Node.js & PostgreSQL)**
  - Node.js handles API requests and server-side processing.
  - PostgreSQL is used as the database to store structured data.
- **Deployment**
  - Azure for cloud hosting.
  - Vercel for front-end deployment.

4.4 Development Methodology

The project follows an Agile Development Model, explicitly incorporating:

- **Sprint-Based Development:** The development cycle is divided into multiple sprints, each lasting two weeks.
- **Daily Stand-up Meetings:** To discuss progress, challenges, and next steps.
- **Iterative Feedback:** Testing and improvements after each sprint cycle.

4.4.1 Sprint Breakdown Example

- **Sprint 1:** Set up project repository, database schema, and authentication module.

- **Sprint 2:** Develop student dashboard and implement attendance tracking.
- **Sprint 3:** Complete teacher and admin functionalities.
- **Sprint 4:** Testing, debugging, and final deployment.

4.5 Implementation Strategy

The system is implemented in three key phases:

4.5.1 Front-End Development

- React is used to build reusable UI components for web applications.
- Kotlin is used for mobile development to ensure a smooth user experience.
- State Management is handled using Redux for React applications.

4.5.2 Back-End Development

- Node.js with Express manages API requests and business logic.
- REST API Architecture is used for communication between the front-end and back-end.

4.5.3 Database Management

- PostgreSQL stores user data, academic records, and attendance.

4.6 Testing and Validation

Testing was conducted to identify usability issues and ensure system reliability. The testing approach included:

- **Unit Testing:** Each component was individually tested using Jest (for Node.js).
- **Integration Testing:** API endpoints were tested using Postman to ensure smooth communication.
- **User Acceptance Testing (UAT):** A group of students and teachers used the sys-

tem and provided feedback.

4.7 Deployment and Maintenance

- **Hosting Platforms:**
  - Azure is used for cloud hosting of the back-end.
  - Vercel is used for front-end deployment.
- **Version Control & CI/CD:**
  - GitHub & Git tools are used for source code management.
  - Continuous Integration (CI) is implemented to automate testing and deployment.
- **Future Maintenance:**
  - Regular updates based on user feedback.
  - Bug fixes and feature enhancements will be tracked using GitHub Issues.

CHAPTER 5

SYSTEM DESIGN

5.1 Overview

5.2 Other section goes here

CHAPTER 6

TECHNOLOGY USED

6.1 Front-End

6.1.1 React

React, an open-source project developed by Meta, is a JavaScript library that permits developers to create an interactive user interface for web applications. It helps in simplifying the building of complex user interfaces. The major work of React involves breaking down the UI into reusable components that are then utilized to construct the more complex UI elements. It supports code reusability and makes the use of state changes easier within the application. In React, there is the virtual DOM whose implementation helps to improve the time and performance for rendering the content in DOM [6].

6.1.2 TypeScript

To add the optional static typing to the language, a superset of JavaScript is used which is known as TypeScript. It is better known to improve code quality and maintainability by catching the bugs at the time of compiling. It comprises several features like auto-completion, code navigation, and helping to prevent common errors during programming. With the help of TypeScript, the developers write more maintainable and scalable code creating a better user experience for their applications [7].

6.1.3 Kotlin

Kotlin, developed by JetBrains, is a statically typed programming language that is fully interoperable with Java. It features a concise syntax, reducing boilerplate code, and introduces null safety to prevent common null pointer exceptions. With smart casts, extension functions, and built-in support for asynchronous programming via coroutines, Kotlin provides a modern and efficient coding experience. It is widely adopted for Android development due to its compatibility with Java and its advanced features [8].

6.2 Back-End

6.2.1 Node.js

Node.js is a popular and powerful backend framework that enables developers to build fast, scalable, and event-driven applications using JavaScript. It's known for its efficient I/O operations, non-blocking I/O model, and a vast library of modules and packages via its package manager, npm. Node.js is also flexible and easy to learn, thanks to its JavaScript syntax and supportive community. With Node.js, developers can build robust and high-performance backend systems for web and mobile applications alike [4].

6.2.2 PostgreSQL

PostgreSQL is widely used by organizations of all sizes for storing and managing their data. It is a powerful, free, and open-source database management system known for its ability to handle complex transactions and robust security features. Developers can extend and customize PostgreSQL to match their specific needs. It is a reliable database solution that is beneficial for any organization intending to store and manage their data efficiently.

CHAPTER 7

EXPECTED RESULTS

REFERENCES

[1] O. Al-Debagy and P. Martinek, “A comparative review of microservices and monolithic architectures,” in *2018 IEEE 18th International Symposium on Computational Intelligence and Informatics (CINTI)*, 2018, pp. 000 149–000 154.

[2] M. Fowler, “Presentation model,” https://martinfowler.com/eaaDev/PresentationModel.html, 2025, accessed: Mar. 1, 2025.

[3] V. Oliveira, L. Teixeira, and F. Ebert, “On the adoption of kotlin on android development: A triangulation study,” in *2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, London, ON, Canada, 2020, pp. 206–216.

[4] M. Casciaro and L. Mammino, *Node.js Design Patterns*, 2nd ed. Packt Publishing, 2020, referenced pages: 418-421.

[5] “Kathmandu university app,” ”[Online]. Available: https://play.google.com/store/apps/details?id=com.gatewaysoft.kathmanduuniversity&hl=en-US&pli=1, 2025, accessed: Mar. 1, 2025.

[6] “React,” [Online]. Available: https://react.dev/, 2025, accessed: Mar. 1, 2025.

[7] “Typescript,” [Online]. Available: https://www.typescriptlang.org/, 2025, accessed: Mar. 1, 2025.

[8] “Kotlin,” [Online]. Available: https://kotlinlang.org/docs/home.html, 2025, accessed: Mar. 1, 2025.

APPENDIX A

APPENDIX B