**DEPARTMENT NOTICE BOARD APP**

A PROJECT REPORT

submitted

*in the partial fulfillment of the requirements for the award of the degree of*

**BACHELOR OF TECHNOLOGY**

in

**COMPUTER SCIENCE AND ENGINEERING**

By

**M Rishwika 21B81A05G2**

**K Shanmukha Tharaka Ram 21B81A05H5**

**P Varshith Sai 21B81A05J7**

Under the guidance of

**Dr. A. Vani Vathsala**

**Head of the Department**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

CVR COLLEGE OF ENGINEERING

**(*An Autonomous institution, NBA, NAAC Accredited and Affiliated to JNTUH, Hyderabad*)**

Vastunagar, Mangalpalli (V), Ibrahimpatnam (M), Rangareddy (D), Telangana- 501 510

**2025**



**CVR College of Engineering**

ACCREDITED BY NATIONAL BOARD OF ACCREDITATION, NAAC

(Approved by AICTE & Govt. of Telangana and Affiliated by JNT University)

### DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

**CERTIFICATE**

This is to certify that the project entitled “**Department Notice Board App**” being submitted **M. RISHWIKA (21B81A05G2), K. SHANMUKHA THARAKA RAM (21B81A05H5) and P. VARSHITH SAI (21B81A05J7)** in partial fulfillment for the award of Bachelor of Technology in Computer Science and Engineering to the CVR College of Engineering, is a record of bona fide work carried out by them under my guidance and supervision during the year 2024-2025.

The results embodied in this project work have not been submitted to any other University or Institute for the award of any degree or diploma.

Signature of the project guide, Signature of the HOD

#### Dr. A. Vani Vathsala Dr. A Vani Vathsala

Head of the Department

Department of CSE Department of CSE

External Examiner

# ACKNOWLEDGEMENTS

We sincerely thank **Dr. K. Ramamohan Reddy,** Principal, CVR College of Engineering, for his cooperation and encouragement throughout the Major Project.

We earnestly thank **Dr. A. Vani Vathsala,** Head of Department, Department of Computer Science and Engineering, CVR College of Engineering, for giving timely cooperation and taking necessary action throughout the course of our Major Project.

We express our sincere thanks and gratitude to our Major Project professor in-charge **Dr. V Dattatreya**, Department of Computer Science and Engineering, CVR College of Engineering, for his valuable help and encouragement throughout the Major Project.

We express our sincere thanks and gratitude to our guide **Dr. A. Vani Vathsala,** Head of Department, Department of Computer Science and Engineering, CVR College of Engineering, for her valuable help and encouragement throughout the Major Project.

We express our sincere thanks and gratitude to our Section Coordinator **Ms. K Sravani** Department of Computer Science and Engineering, CVR College of Engineering, for her valuable help and encouragement throughout the Major Project.

Finally, we thank all those whose guidance helped us in this regard. We record our sincere appreciation and indebtedness to our parents and all the lecturers for their understanding and cooperation, without whose encouragement and blessing it would not have been possible to complete this work.

**With Regards**

# M. Rishwika 21B81A05G2

# K. Shanmukha Tharaka Ram 21B81A05H5

# P. Varshith Sai 21B81A05J7

# ABSTRACT

The **Department Notice Board System** is a web and mobile-based application designed to enhance communication within an academic institution. It enables **administrators, faculty, and students** to manage notices and complaints efficiently while ensuring **role-based access control** for secure and structured operations.

The backend is built using **Node.js, Express, and MongoDB**, providing a robust REST API. **JWT authentication** is used for secure access control, and Firebase authentication is integrated for user registration. The frontend is developed using **React for the web and React Native (Expo) for mobile**, ensuring a dynamic and interactive user experience. **Lottie animations (JSON files)** are used to enhance the UI with smooth, engaging transitions.

Key features include **creating, deleting, and viewing notices**, along with a **complaint management system** where students can submit complaints and faculty/admins can respond. Admins have **full control** over notices, user roles, and permissions. Faculty members can **manage notices and respond to complaints**, while students can **view notices and file complaints**.

Testing was conducted iteratively using **Postman for API validation** and the **Expo Go app** for real-time mobile testing and debugging. Role-based access and security measures were thoroughly verified to ensure a secure and efficient user experience.

This system provides a **centralized, accessible, and structured** communication platform for academic institutions. Future enhancements may include **real-time push notifications, analytics on notices and complaints, and an improved role-based dashboard** to further enhance usability and effectiveness

**Table of contents**

|  |  |  |  |
| --- | --- | --- | --- |
|  | | | Page No |
|  |  | List of Tables |  |
|  |  | List of Figures |  |
|  |  | Abbreviations |  |
|  |  | Symbols |  |
| 1 |  | **INTRODUCTION** |  |
|  | 1.1 | Motivation |  |
|  | 1.2 | Problem Statement |  |
|  | 1.3 | Project Objectives |  |
|  | 1.4 | Project Report Organization |  |
| 2 |  | **LITERATURE REVIEW** |  |
|  | 2.1 | Existing Work |  |
|  | 2.2 | Limitations of Existing work |  |
| 3 |  | **REQUIREMENT ANALYSIS** |  |
|  | 3.1 | Software requirements |  |
|  | 3.2 | Hardware requirements |  |
|  | 3.3 | User requirements |  |
| 4 |  | **SYSTEM DESIGN** |  |
|  | 4.0 | Proposed System architecture |  |
|  | 4.1 | Proposed methods/ Algorithms |  |
|  | 4.2 | Class / Use case / Activity/ Sequence Diagrams |  |
|  | 4.3 | Datasets and Technology stack |  |
| 5 |  | **IMPLEMENTATION** |  |
|  | 5.1 | Front page Screenshot |  |
|  | 5.2 | Results and Discussions |  |
|  | 5.3 | Testing |  |
|  | 5.4 | Validation |  |
| 6 |  | **CONCLUSION** |  |
|  | 6.1 | Conclusion |  |
|  | 6.2 | Future scope |  |
|  |  | **REFERENCES** |  |
|  |  | **APPENDIX: (If any like Published paper / source code)** |  |

# 1. INTRODUCTION

The Department Notice Board System is a digital platform designed to streamline communication between faculty, students, and administrators. Traditional notice boards often lead to missed updates and inefficiencies, which this system resolves through a web and mobile-based solution.

Built with Node.js, Express, and MongoDB for the backend and React Native with Expo for the frontend, it ensures secure and scalable access. The system features role-based accesscontrol (RBAC), where admins, faculty, and students have distinct permissions. Firebase authentication and JWT-based authorization enhance security.

Lottie animations provide a dynamic user experience, while testing was conducted iteratively using Postman and Expo Go to ensure smooth performance. This system effectively digitizes announcements and complaint management, improving accessibility, engagement, and communication within educational institutions.

**1.1 MOTIVATION:**

In educational institutions, traditional notice boards often lead to inefficiencies, missed announcements, and lack of accessibility. Students and faculty may struggle to stay updated with important information, especially in large departments. To address these issues, a digital notice board system provides a streamlined solution for real-time communication. This system ensures accessibility from anywhere, reducing dependency on physical notices. Additionally, it allows students to submit complaints and receive timely responses. By leveraging modern technologies, the project enhances engagement, improves efficiency, and fosters a more organized and interactive communication channel within academic institutions.

**1.2 PROBLEM STATEMENT:**

In many educational institutions, communication between administration, faculty, and students is often fragmented, leading to inefficiencies in information dissemination. Traditional notice boards require manual updates, limiting accessibility and often resulting in students missing important announcements. Additionally, physical notices may get removed, damaged, or overlooked, making it difficult to ensure that every student is informed in a timely manner.

Another major issue is the lack of an organized platform for students to submit complaints or concerns. In many cases, students struggle to report their grievances due to unclear processes or lack of anonymity. Without a structured system, complaints may go unaddressed, reducing students’ trust in the institution’s administration. Faculty members also face challenges in managing notices and responding to student concerns efficiently. A manual or paper-based system often leads to delays, miscommunication, and unnecessary administrative burdens.

The absence of a centralized platform also prevents institutions from ensuring proper role-based access control. Notices should be managed by faculty, while students should be limited to viewing notices and submitting complaints. However, existing systems often lack a structured user role management mechanism, leading to potential misuse or inefficiencies in notice distribution.

To overcome these challenges, the project proposes a digital Department Notice Board System. This system will enable real-time notice updates, ensuring that students and faculty receive instant notifications. It will also incorporate a complaint management module where students can submit issues, and faculty can respond promptly. The platform will implement role-based authentication, granting different levels of access to students, faculty, and administrators.

With this solution, institutions can streamline their communication channels, enhance transparency, and improve responsiveness to student concerns. The digital approach eliminates the limitations of traditional notice boards while fostering a more connected and informed academic environment. The system will be tested iteratively to ensure usability, accessibility, and security, providing a robust and scalable solution for departmental communication.

**1.3 PROJECT OBJECTIVES:**

1. **Develop a Centralized Digital Notice Board**

The primary objective of this project is to create a centralized platform for academic and administrative notices. Traditional notice boards are inefficient, requiring manual updates and physical presence to access information. By developing a digital system, students and faculty can access notices in real-time from anywhere. The system ensures that important announcements, deadlines, and events are always available, reducing miscommunication and increasing accessibility.

**2. Role-Based Access Control for Different User Types**

To maintain a structured and secure platform, role-based access control (RBAC) is implemented. Different types of users have varying permissions:

* **Admin:** Has full control over notices, permissions, and complaint management. Admins can oversee system functionality and manage user access.
* **Faculty:** Can create, edit, and delete notices related to academics, events, or university regulations. They also have access to handle student complaints.
* **Students:** Can only view notices and lodge complaints regarding academic or administrative issues.  
  By enforcing role-based access, the system prevents unauthorized modifications and ensures **data integrity**.

**3. Implement a Complaint Management System**

Another major objective is to provide a structured complaint system where students can report issues related to academics, administration, or facilities. This feature ensures that students can voice their concerns transparently, while faculty and admins can track, respond, and resolve complaints efficiently. The system allows real-time tracking of complaint status and fosters a more accountable academic environment.

**4. Enhance User Experience with a Modern UI/UX**

The project aims to deliver a seamless user experience through a well-designed and intuitive interface. The frontend is developed using React Native with Expo, ensuring compatibility across Android and iOS devices. Unlike traditional web-based portals, a mobile-friendly approach ensures better accessibility for students and faculty who prefer using smartphones.

* Lottie animation JSON files are integrated to enhance visual feedback and provide a more interactive experience.
* The design follows a simple and effective navigation structure, making it easy for users to find relevant notices, submit complaints, and interact with the system.

**5. Secure Authentication and Authorization Mechanisms**

Security is a critical aspect of the system. To ensure secure user authentication and data protection, the platform uses:

* **JWT (JSON Web Token):** Secure login and session management to prevent unauthorized access.
* **Encryption of User Data:** Passwords are hashed before storing them in the database to prevent security breaches.
* **Role-based restrictions:** Prevents unauthorized users from modifying sensitive data.  
  These measures ensure that **only authenticated users** can access and interact with the system according to their roles, safeguarding the integrity of notices and complaints.

**6. Optimize Performance, Scalability, and Maintainability**

The system is designed to handle multiple users concurrently without affecting performance. Efficient backend optimization ensures that notices and complaints are processed quickly, even under high load. The system architecture is built for scalability, allowing future expansion and integration of additional features. Code maintainability is also a priority, ensuring that future updates and bug fixes can be implemented smoothly.

**7. Iterative Testing and Deployment Strategy**

To deliver a stable and bug-free system, testing was conducted iteratively using Expo Go, allowing real-time debugging and refinements. Continuous testing ensures that:

* UI/UX elements function as intended.
* The authentication and authorization mechanisms work securely.
* Notice creation, complaint management, and role-based access are tested for reliability.
* Performance is optimized for different devices and user loads.  
  This iterative approach allows for faster issue resolution and ensures a seamless user experience.

By fulfilling these objectives, the project successfully delivers an efficient, secure, and scalable digital notice board that enhances communication, transparency, and accessibility for students, faculty, and administrators.

**1.4 PROJECT REPORT ORGANISATION**

This project report details the development of a Department Notice Board system, built as a web and mobile application using React Native and Expo for the frontend, and Node.js with MongoDB for the backend. The report is structured to provide a clear understanding of the project’s objectives, implementation, and outcomes.

The Introduction presents the background, motivation, problem statement, and objectives of the project. It outlines the need for an efficient, digital notice board system for academic institutions, ensuring easy access to notices and structured complaint management.

The Literature Review discusses existing systems, highlighting limitations such as poor user experience, lack of role-based access, and inefficient complaint resolution. The Methodology section explains the technologies used, including Express.js for the backend, Firebase for authentication, and JWT for security. It describes the role-based access control for admin, faculty, and students, along with the iterative testing process using Expo Go.

The Implementation section covers the integration of notice creation, complaint submission, and role management. It also discusses the usage of Lottie animations for enhanced user experience. The Testing and Results section presents the iterative improvements and verification processes conducted using Postman and Expo Go.

The Conclusion and Future Work section summarizes the achievements, challenges, and potential enhancements, such as improved UI/UX, additional role-based permissions, and further security measures. This structured organization ensures clarity and documentation of all aspects of development, allowing future enhancements and scalability.

**2. LITERATURE SURVEY**

**2.1 EXISTING WORK:**

Over the years, various digital notice board systems have been developed to improve communication within organizations, particularly in academic institutions. These systems range from simple web-based platforms to complex mobile applications with real-time notification features. While these solutions have contributed to streamlining information dissemination, they often come with limitations in terms of usability, scalability, and role-based access control.

**1. Traditional Notice Board Systems**

Historically, physical notice boards have been the primary medium for information sharing in institutions. These boards, often placed in common areas, display printed or handwritten notices related to academic schedules, events, and general announcements. While widely used, this approach has significant drawbacks:

* **Limited Accessibility:** Notices are only visible within the premises, making it difficult for students or faculty members to stay updated remotely.
* **Manual Updates:** Admins must manually update notices, leading to inefficiencies and delays.
* **Lack of Security and Role Control:** Anyone can post or remove notices, increasing the risk of unauthorized changes.

**2. Web-Based Notice Board Systems**

With the advent of web technologies, institutions began implementing digital notice boards as web applications. These platforms allow administrators to upload notices online, making them accessible to students and faculty via a web portal. Common features of web-based notice boards include:

* **User Authentication:** Basic authentication mechanisms such as username-password combinations to restrict access.
* **Text-Based Notices:** Ability to post and view text-based announcements.
* **Limited Interactivity:** Most systems lack features for user engagement, such as commenting or responding to notices.

Despite these improvements, web-based solutions still face several challenges:

* **Lack of Mobile Support:** Many web platforms are not optimized for mobile devices, reducing accessibility.
* **Inefficient Complaint Management:** Most web-based notice boards do not support complaint handling or structured communication between students and faculty.

**3. Mobile-Based Notice Board Applications**

To address the limitations of web-based systems, mobile notice board applications were developed using technologies like Android, iOS, and cross-platform frameworks such as React Native. These applications offer enhanced accessibility and real-time updates, making them more convenient for students and faculty.

Some notable features of mobile-based notice board systems include:

* **Push Notifications:** Users receive instant alerts for new notices or announcements.
* **Role-Based Access:** Admins can manage notices, while students can only view or interact with them.
* **Better UI/UX:** Modern frameworks provide improved user experience with intuitive interfaces.

However, mobile applications also come with drawbacks:

* **Platform-Specific Development:** Native apps require separate development for Android and iOS, increasing costs and maintenance efforts.
* **Limited Security Measures:** Many existing applications do not implement strong authentication and authorization mechanisms, making them vulnerable to unauthorized access.
* **No Proper Complaint System:** Most notice board apps focus only on announcements and lack features for structured complaint management.

**4. Firebase-Based Notice Board Systems**

Some recent solutions leverage Firebase for real-time notice updates and authentication. Firebase-powered applications provide:

* **Cloud Storage:** Notices are stored in Firebase Realtime Database or Firestore.
* **Authentication Services:** Firebase Authentication is used for user sign-in via Google, email, or social media.
* **Real-Time Updates:** Users get instant notice updates without refreshing the application.

While Firebase-based solutions improve performance and accessibility, they also have limitations:

* **Limited Customization:** Firebase imposes certain constraints, making it difficult to implement advanced role-based access control.
* **Scalability Issues:** Managing large datasets and handling high traffic can become costly.

**5. Existing Complaint Management Systems**

Most complaint management systems available today are standalone platforms designed for customer support or HR-related grievances. These systems use ticket-based tracking where users raise complaints, and designated personnel resolve them. However, these platforms are often:

* **Not Integrated with Notice Boards:** Separate from notice board applications, requiring users to switch between systems.
* **Complex for Educational Institutions:** Many solutions are tailored for corporate environments rather than academic use.
* **Lack of Role-Specific Permissions:** Generic complaint platforms do not offer role-based complaint handling where faculty and admins have different levels of control.

**6. Advancements in Integrated Systems**

Many modern solutions aim to combine digital notice boards with complaint management features, offering a comprehensive communication platform. These advancements include:

* **Seamless Notice and Complaint Handling:** A unified system that allows institutions to manage notices and complaints in one place.
* **Improved Role-Based Access Control:** Assigning different permissions for admins, faculty, and students ensures better management of notices and complaints.
* **Enhanced User Experience:** The adoption of modern UI/UX design, animations (like Lottie), and real-time interactions has made these systems more intuitive and engaging.
  1. **LIMITATIONS OF EXISTING WORK:**

Despite the advancements in various digital systems, many existing solutions still suffer from several limitations that impact their effectiveness, efficiency, and scalability. The limitations of current systems can be categorized into different areas, such as accessibility, security, scalability, usability, and integration. Understanding these shortcomings is crucial for developing improved solutions that address existing challenges.

### ****1. Accessibility and Availability Issues****

One of the primary limitations of existing digital systems is their dependence on internet connectivity. Many applications and platforms require a stable and high-speed internet connection to function properly. This poses a challenge in regions with poor network infrastructure, limiting accessibility for users in remote or underdeveloped areas.

Moreover, platform dependency remains a significant issue. Some systems are designed exclusively for specific platforms, such as Windows, macOS, Android, or iOS, restricting access for users who do not have compatible devices. Lack of cross-platform support creates accessibility barriers and limits the system’s reach.

Another common limitation is language and localization constraints. Many existing systems are developed in a single language, making it difficult for non-native speakers to use them efficiently. Without proper localization features, users may struggle with navigation, leading to a poor user experience.

### ****2. Scalability and Performance Limitations****

Many systems, particularly those built on older technologies, struggle to scale efficiently as the number of users grows. Performance bottlenecks, such as slow load times, crashes, and system downtimes, often occur when a system is not designed to handle high traffic or large amounts of data.

Limited cloud support further exacerbates scalability issues. Systems that rely on local storage or on-premise servers may face significant difficulties when trying to accommodate a growing user base. Cloud-based solutions offer better scalability, but many existing systems lack proper cloud integration.

Data management inefficiencies also pose challenges. Poorly optimized databases, redundant data storage, and inefficient indexing methods contribute to slow queries and increased processing time. As a result, users may experience delays in retrieving or updating information.

### ****3. Security and Privacy Concerns****

Security vulnerabilities are one of the most critical limitations of existing systems. Many platforms lack robust authentication mechanisms, making them susceptible to unauthorized access and data breaches. Weak password policies, lack of multi-factor authentication (MFA), and improper session management contribute to security risks.

Data privacy concerns are another major issue. Many systems collect and store user data without proper encryption or anonymization techniques, increasing the risk of data leaks. Compliance with data protection regulations such as GDPR, HIPAA, or CCPA is often overlooked, leaving users' personal information exposed.

Another security limitation is the vulnerability to cyber threats, including phishing attacks, malware, and denial-of-service (DoS) attacks. Systems with inadequate security protocols are at risk of being compromised, leading to financial and reputational damage.

### ****4. Usability and User Experience Challenges****

Many existing systems suffer from poor UI/UX design, making them difficult to use. Complex navigation, cluttered interfaces, and a lack of intuitive workflows lead to frustration among users. If a system is not user-friendly, it reduces engagement and adoption rates.

Limited customization options also hinder user experience. Many applications have rigid interfaces that do not allow users to personalize their experience based on their preferences. Lack of dark mode, font resizing, and accessibility features for differently-abled users further contribute to usability issues.

Additionally, inefficient notification systems in many applications result in missed updates or excessive alerts that overwhelm users. A balanced notification mechanism is necessary to enhance user engagement without causing notification fatigue.

### ****5. Integration and Interoperability Issues****

Many existing systems operate in isolation and do not integrate well with other platforms. Lack of API support or limited compatibility with third-party services makes it difficult for users to synchronize data across multiple applications.

For example, many enterprise solutions do not offer seamless integration with productivity tools like Microsoft Office, Google Workspace, or Slack. This lack of interoperability forces users to switch between platforms manually, leading to inefficiencies and loss of productivity.

Another integration challenge is the absence of standardized data formats. Systems using proprietary file formats often require conversion before being used with other platforms, resulting in data loss or corruption. Implementing universally accepted formats, such as JSON, XML, or CSV, can enhance data interoperability.

### ****6. Maintenance and Upgradability Challenges****

Many existing systems face challenges related to maintenance and updates. Legacy software, which has been in use for several years, often lacks support for modern features and security updates. Organizations using outdated systems struggle to keep up with technological advancements, making their infrastructure vulnerable to security threats and performance issues.

Frequent software updates can also be a problem if they introduce bugs or compatibility issues. Poorly tested updates may lead to crashes, data loss, or reduced performance, causing inconvenience to users.

Furthermore, dependency on proprietary technologies limits the ability to upgrade or customize systems. If a platform is built using a closed-source framework, making modifications or adding new features can be difficult and expensive.

### ****7. Cost-Related Limitations****

The financial aspect of implementing and maintaining a system is another limitation. Many existing solutions come with high initial costs, including licensing fees, hardware requirements, and infrastructure setup. Small businesses, educational institutions, and non-profit organizations may struggle to afford these expenses.

Additionally, ongoing maintenance costs, such as cloud storage fees, subscription charges, and technical support expenses, add to the overall financial burden. Some systems also require expensive add-ons or premium subscriptions to access essential features, making them less accessible for budget-conscious users.

### ****8. Lack of Offline Functionality****

Many modern systems rely heavily on an internet connection, making them inaccessible during network downtime. Users who need to access information or perform critical tasks offline are often left without options.

Offline functionality is especially crucial for mobile applications, where users may not always have reliable internet access. Implementing offline data caching and synchronization features can help address this limitation, but many existing solutions still lack these capabilities.

# 3. SOFTWARE & HARDWARE SPECIFICATIONS

**3.1 SOFTWARE REQUIREMENTS:**

To develop a robust and efficient system, it is essential to define the software requirements that support the application's functionality, performance, and security. These requirements are categorized into system software, development tools, backend technologies, frontend technologies, and additional libraries or frameworks used in the project.

1. System Software Requirements

These are the fundamental software components required to run the application on a computer or server.

* Operating System: Windows 10/11, macOS, or Linux (Ubuntu)
* Node.js: Required for running the backend and frontend development environment
* Database Management System (DBMS): MongoDB (NoSQL database)
* Web Server: Express.js (Node.js framework)

2. Backend Development Requirements

The backend is responsible for handling business logic, authentication, database operations, and API communications.

* Programming Language: JavaScript (Node.js)
* Framework: Express.js (for handling API requests)
* Database: MongoDB (for storing users, notices, and complaints)
* Authentication: JSON Web Token (JWT) for secure authentication and authorization
* Environment Variables Management: .env for managing configuration settings
* Middleware: Custom authentication and role-based access control (RBAC)

3. Frontend Development Requirements

The frontend is responsible for user interaction, displaying notices, and complaint management.

* Framework: React Native (for mobile application development)
* UI Library: Native components with additional styling
* Animations: Lottie JSON files for interactive animations
* State Management: React Context API (for managing global state)
* HTTP Requests: Axios (for API communication)

4. Development Tools

* Code Editor: Visual Studio Code (VS Code)
* Package Manager: npm (Node Package Manager)
* Version Control: Git and GitHub for code management and collaboration
* Testing Platform: Expo Go for iterative testing on mobile devices

5. Additional Libraries and Dependencies

* CORS: Handles cross-origin requests
* bcrypt.js: Hashing passwords securely
* Mongoose: MongoDB object modelling for Node.js
* Express Validator: Validates and sanitizes input data

These software requirements ensure the system is secure, scalable, and user-friendly, providing a seamless experience for administrators, faculty, and students.

* 1. **HARDWARE REQUIREMENTS**

To ensure smooth development, deployment, and execution of the **Department Notice Board System**, the following hardware requirements are recommended:

**1. Development System Requirements**

These specifications are required for developers working on the **frontend and backend** of the application:

* **Processor:** Intel Core i5 (10th Gen or higher) / AMD Ryzen 5 or higher
* **RAM:** Minimum 8 GB (16 GB recommended for optimal performance)
* **Storage:** SSD with at least 256 GB free space (512 GB recommended)
* **Graphics Card:** Integrated GPU (Dedicated GPU optional for better performance)
* **Operating System:** Windows 10/11, macOS, or Linux (Ubuntu 20.04 or later)
* **Internet Connection:** Stable broadband connection for testing APIs and using **Expo Go**

**2. Server Requirements (for Hosting Backend and Database)**

These specifications apply if the backend is deployed on a cloud server:

* **Processor:** Intel Xeon or equivalent (2.5 GHz, Quad-Core or higher)
* **RAM:** 16 GB or more for handling concurrent user requests
* **Storage:** Minimum 512 GB SSD for MongoDB database and logs
* **Network:** High-speed internet with at least 100 Mbps bandwidth
* **Security:** SSL/TLS encryption support for secure API communication

**3. Mobile Testing Devices**

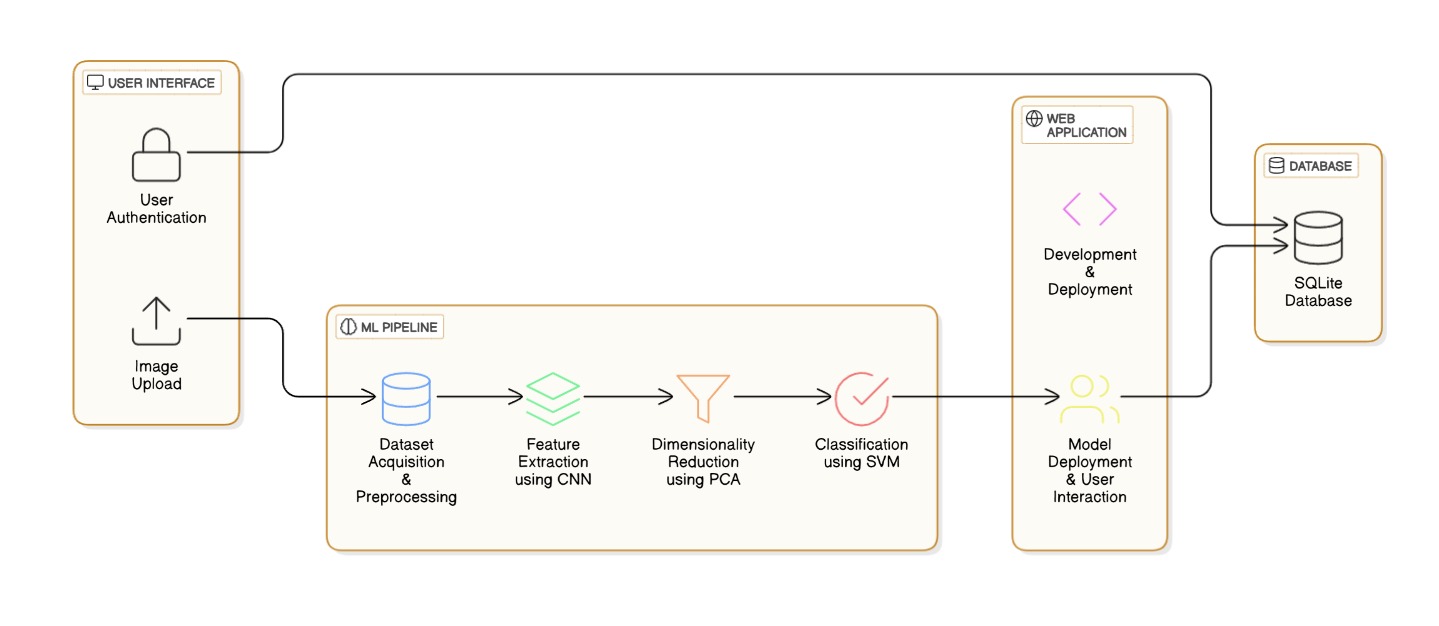
Since the project is built using **React Native (Expo)**, testing should be performed on real devices:

* **Android:** Minimum Android 9 (Pie) with 4 GB RAM
* **iOS:** iPhone 8 or later with iOS 14+

## 

## 4. PROPOSED SYSTEM DESIGN

* 1. **PROPOSED SYSTEM ARCHITECTURE**



This architecture represents a web-based Alzheimer's disease detection system that integrates machine learning (ML) and a user-friendly interface for real-world application. The system follows a structured pipeline, ensuring efficient processing, classification, and deployment. Below is a detailed breakdown of each component:

**1. User Interface (Frontend)**

The user interface (UI) serves as the entry point for the application. Built using React.js, it provides a seamless experience for users, allowing them to:

* Authenticate: Users must log in before accessing the system. This ensures secure access and enables personalized tracking of uploaded MRI scans.
* Upload MRI Images: The UI allows users to upload brain MRI scans, which are then passed to the backend for processing.

This frontend interacts with the Flask-based backend through API calls, ensuring a smooth and dynamic experience.

**2. Machine Learning Pipeline**

Once an MRI image is uploaded, it enters the ML pipeline, which consists of four key stages:

**a) Dataset Acquisition & Preprocessing**

* The system loads and preprocesses MRI images.
* Preprocessing includes skull stripping, noise removal, and intensity normalization, ensuring that the images are clean and suitable for analysis.
* The dataset comprises real and synthetic MRI scans categorized into four groups:
  + No Impairment
  + Very Mild Impairment
  + Mild Impairment
  + Moderate Impairment

**b) Feature Extraction using CNN**

* A pre-trained Convolutional Neural Network (CNN) extracts meaningful features from MRI scans.
* Instead of using CNN for direct classification, it acts as a feature extractor, converting images into a compact numerical representation.
* This extracted data is then used as input for further processing.

**c) Dimensionality Reduction using PCA**

* Principal Component Analysis (PCA) reduces the high-dimensional feature set obtained from the CNN.
* This step helps in:
  + Eliminating redundant information.
  + Reducing computation costs.
  + Improving the efficiency of classification.

d) **Classification using SVM**

* A Support Vector Machine (SVM) classifier processes the reduced features and assigns them to one of the four Alzheimer's impairment categories.
* SVM is chosen for its high accuracy and robustness in handling medical image classification tasks.
* The final classification output is then sent back to the web application for display.

**3. Web Application (Backend)**

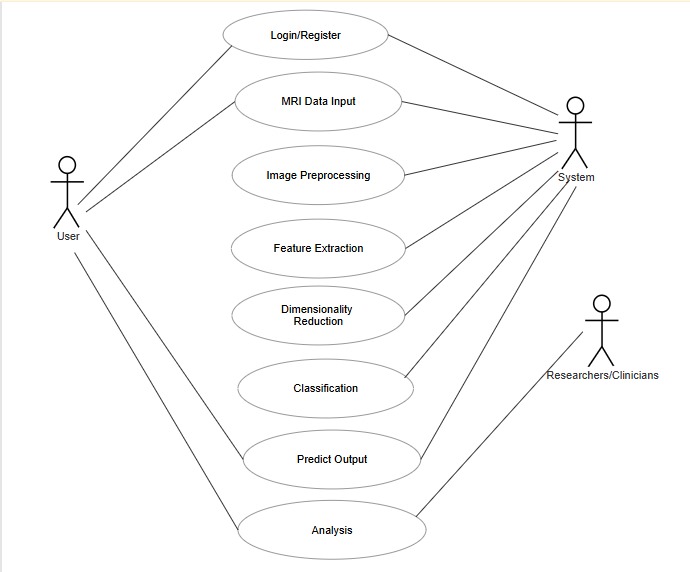
* The Flask-based backend serves as the bridge between the frontend, ML pipeline, and database.
* It handles model deployment, API requests, and user interactions.
* The backend:
  + Receives uploaded images.
  + Passes them through the ML pipeline.
  + Stores and retrieves results for users.

**4. Database (SQLite)**

* The system uses an SQLite database for user authentication and data storage.
* It stores:
  + User login credentials (hashed for security).
  + Uploaded MRI scan metadata (filename, upload date, classification result).
* This database allows users to track their past scans and access classification history.5. Detailed Venue View

**4.2 UML DIAGRAMS**

**4.2.1 USE CASE DIAGRAM**

****

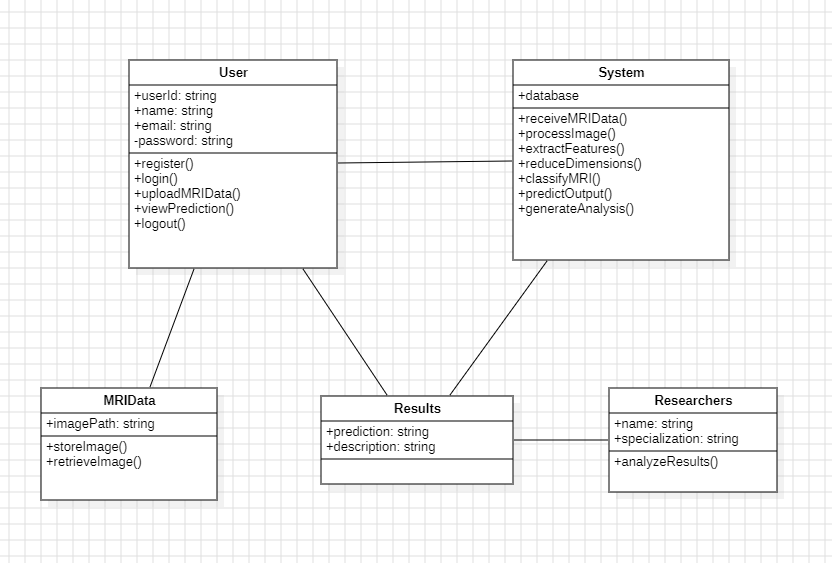
**Fig 4.2.1 Use Case Diagram**

This use case diagram depicts the interactions between users and a system designed for MRI data analysis, specifically for tasks like disease prediction or diagnosis. The system is represented by the central ellipse, and the actors interacting with it are the User and Researchers/Clinicians, shown as stick figures. Each oval shape represents a specific function or use case within the system.

The diagram illustrates a sequential workflow, starting with the "Login/Register" use case, suggesting that users must authenticate themselves to access the system. Following login, the "MRI Data Input" use case allows users to upload or input MRI data into the system for analysis. The system then performs a series of automated steps: "Image Preprocessing" prepares the data for further analysis, "Feature Extraction" identifies relevant features within the images, and "Dimensionality Reduction" reduces the complexity of the data while retaining important information.

After these automated steps, the "Classification" use case likely employs machine learning models to categorize or diagnose the MRI data. The system then "Predicts Output," which could be a diagnosis, risk assessment, or other relevant information derived from the analysis. Finally, the "Analysis" use case allows Researchers/Clinicians to review and interpret the results, potentially providing insights into disease progression, treatment response, etc. The lines connecting the actors to the use cases represent the interactions; for instance, the User initiates the Login/Register and MRI Data Input use cases, while both the User and Researchers/Clinicians can access the Analysis use case to review results. This diagram effectively communicates the system's functionality and the roles of different users within the MRI data analysis process.

**4.2.2 CLASS DIAGRAM**

**Fig 4.2.2 Class Diagram**

This class diagram represents the design of a system for MRI data analysis, outlining the structure and relationships between different entities. The diagram is organized into four classes: User, System, MRIData, and Results, with an additional "Researchers" class representing a specialized user role. Each class encapsulates data (attributes) and actions (methods) relevant to its function within the system.

The **User** class stores information about a user, including their unique ID (userld), name, email, and password (the latter marked as private with a "-" symbol). It defines methods for user registration (register()), login (login()), MRI data upload (uploadMRIData()), viewing prediction results (viewPrediction()), and logout (logout()). This class interacts with both the System and the MRIData classes, uploading MRI data for processing and retrieving results.

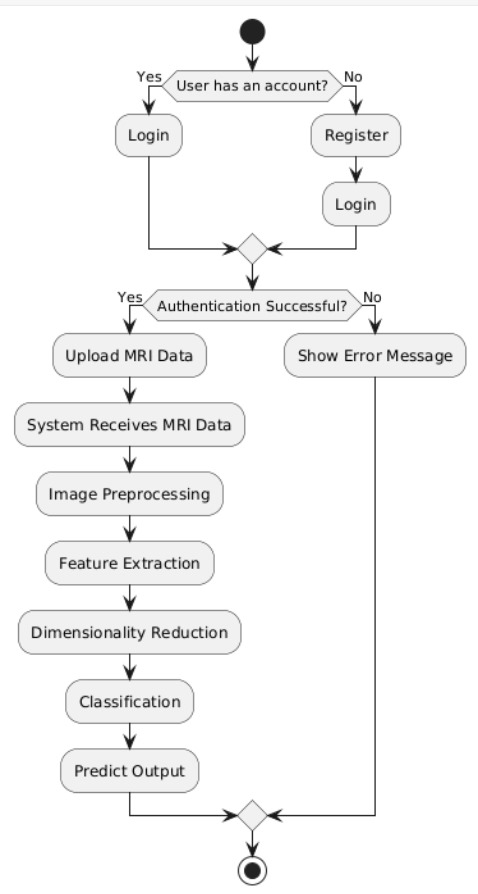
The **System** class is the core of the application, containing the logic for processing MRI data. It has a connection to a database and defines methods for receiving MRI data (receiveMRIData()), processing the image (processImage()), extracting features (extractFeatures()), reducing dimensions (reduceDimensions()), classifying the MRI (classifyMRI()), predicting output (predictOutput()), and generating analysis (generateAnalysis()). This class orchestrates the entire analysis workflow.

The **MRIData** class represents the MRI image data, storing the imagePath and providing methods to storeImage() and retrieveImage(). It interacts with both the User (for upload) and the System (for processing).

The **Results** class stores the outcome of the MRI analysis, including the prediction and a textual description. It is related to the System (which generates the results) and the User (who views them).

Finally, the **Researchers** class, a specialized type of user, extends the basic User class (indicated by the line with a triangle). It inherits user attributes and adds a specialization attribute and an analyzeResults() method, suggesting that researchers have additional capabilities for in-depth analysis of the results.

**4.2.3 ACTIVITY DIAGRAM**



**Fig 4.2.3 Activity Diagram**

This activity diagram visually represents the workflow of an MRI data analysis system, detailing the sequence of operations and decision points involved. It starts with a solid black circle, indicating the initial activity.

The first step is a decision, represented by a diamond, asking "User has an account?". If the answer is "Yes," the user proceeds to the "Login" activity. If "No," the user goes to "Register" and then logs in. After either login or registration, the diagram merges back to a single "Login" activity, suggesting that registration ultimately leads to a login process.

Following the "Login" activity, another decision diamond checks "Authentication Successful?". If "No," the process branches to "Show Error Message" and then loops back to the "Login" activity, allowing the user to retry. If authentication is "Yes," the user proceeds to "Upload MRI Data."

Once the MRI data is uploaded, the system performs a series of automated activities: "System Receives MRI Data," "Image Preprocessing," "Feature Extraction," "Dimensionality Reduction," "Classification," and finally, "Predict Output." These activities are depicted as rounded rectangles and occur sequentially, showing the flow of data through the system.

After the "Predict Output" activity, the workflow merges back to a final activity, represented by a diamond with a circle inside. This symbol likely signifies a decision point where the process might branch based on the output, but the diagram doesn't show the possible branches. The workflow concludes with a bullseye symbol (a circle within a circle), indicating the end of the activity.

In summary, this activity diagram illustrates the user login and registration process, including error handling for failed authentication. It then outlines the automated steps the system takes to process MRI data and generate predictions. The diagram effectively communicates the procedural flow and decision logic of the MRI analysis system.