School Management System

A modern managing system

Technical Design Documentation

Jitendra Kumar Yadav(2210991750) & Manasvi(2210991876)

December, 2024

# Contents

1. Introduction and System Overview
2. Architecture
   * High-Level Architecture
   * Component Interactions
3. Backend Design
   * Technologies Used
   * Project Structure
   * API Design
   * Database Schema
   * Middleware
   * Authentication Flowchart
4. Frontend Design
   * Technologies Used
   * Project Structure
   * Routing
   * State Management
5. Security Considerations
6. Deployment Plan
7. Future Plans
8. Conclusion

# Introduction and System Overview

## Introduction

A School Management System (SMS) is an advanced online platform designed to streamline and automate various administrative and academic operations in schools. It provides a centralized solution for managing student records, attendance, grades, fees, and communication between teachers, students, and parents. With user-friendly interfaces, the system ensures efficiency in daily tasks like scheduling classes, tracking performance, and generating reports. Accessible from anywhere, it enhances transparency and collaboration while reducing manual errors. By integrating cutting-edge technology, the system empowers schools to focus more on delivering quality education.

This design document provides a comprehensive overview of the project’s architecture, components, technology stack, and design decisions. It serves as a guide for developers, stakeholders, and contributors who are involved in the development and maintenance of the SMS platform.

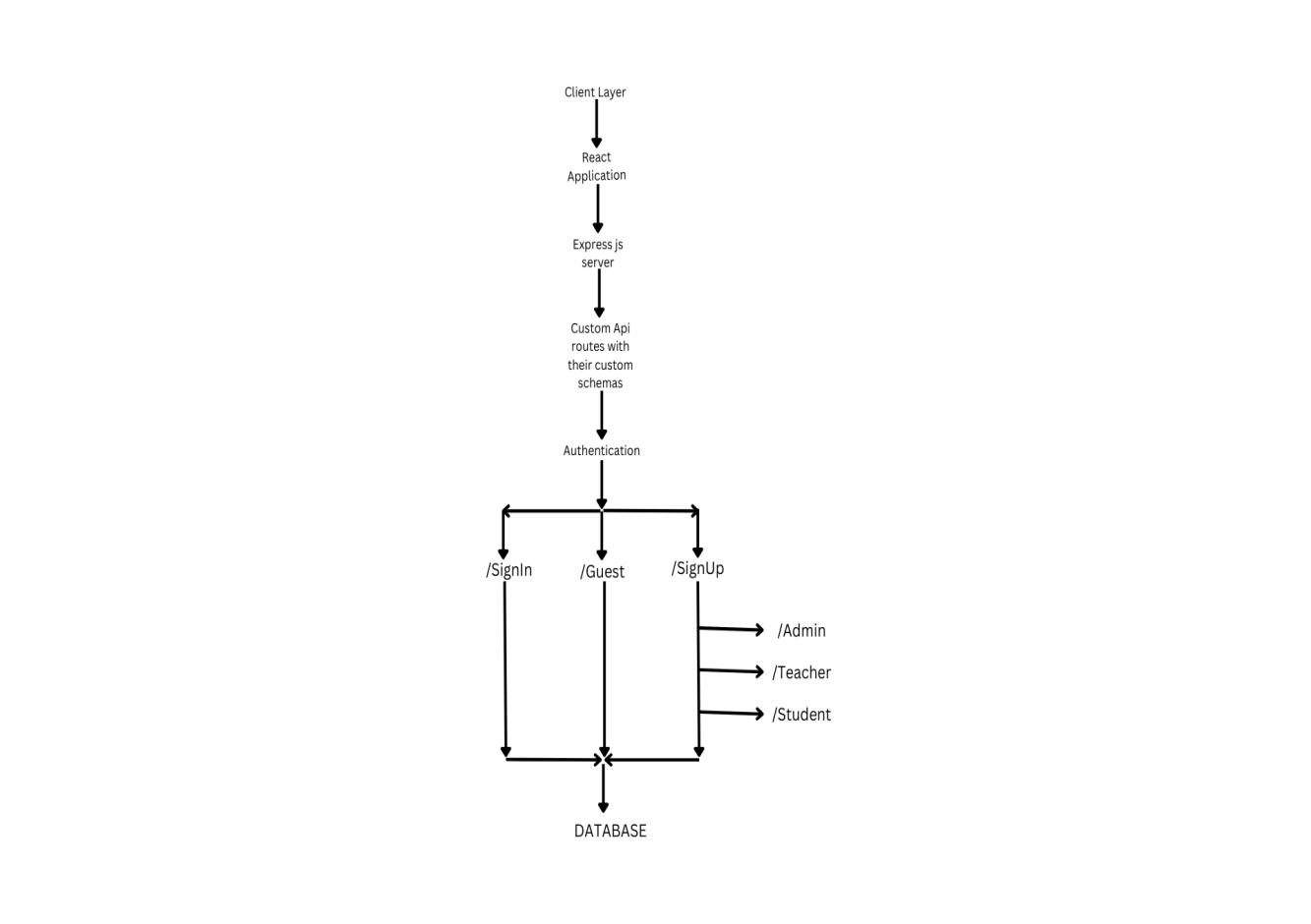
## System Overview

SMS is built using the MERN stack (MongoDB, Express.js, React, Node.js), implementing modern web development practices with a focus on responsive design. The application integrates with the custom API for data and maintains its own database for user information and interactions. The system is designed to provide scalability, maintainability, and responsive user experience across desktop computers, tablets, and mobile devices.

# Architecture

## High-Level Architecture

1. **Frontend**: Developed with React.js, responsible for the client-side user interface and interactions. Features responsive design principles using Tailwind CSS for optimal display across all devices.Equipped with use of Material UI for better and smooth usage by clients.
2. **Backend API**: Built with Express.js and Node.js, handling server-side logic, API endpoints, and authentication. Optimized for handling requests from various devices and screen sizes.
3. **Database**: Utilizes MongoDB for storing user data, saved books, ratings, and comments.



## Component Interactions

### Client-Server Communication:

* + JWT-based authentication
  + JSON data format for requests and responses

### Database Operations:

* + MongoDB Atlas cloud database
  + Mongoose custom schemas for data modeling
  + CRUD operations for user data and client interactions

### Custom Access:

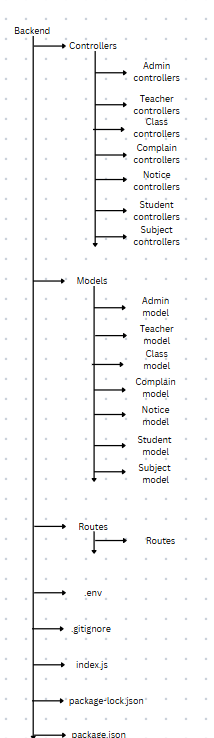
* + Admin gets compelete access (principal,groups)
  + Teacher get acces upto only changing students , subjects , attendance
  + Students get only read access for their information

# Backend Design

## Technologies Used

* **Node.js**: JavaScript runtime environment
* **Express.js**: Web application framework
* **MongoDB**: NoSQL database for data persistence
* **Mongoose**: ODM for MongoDB
* **bcrypt**: Password hashing
* **cors**: Cross-Origin resource sharing

## Project Structure

****

**API Design**

Authentication (/auth)

* POST /signup: User registration
* POST /login: User authentication
* POST /save: Updates user’s saved books

## Database Schema

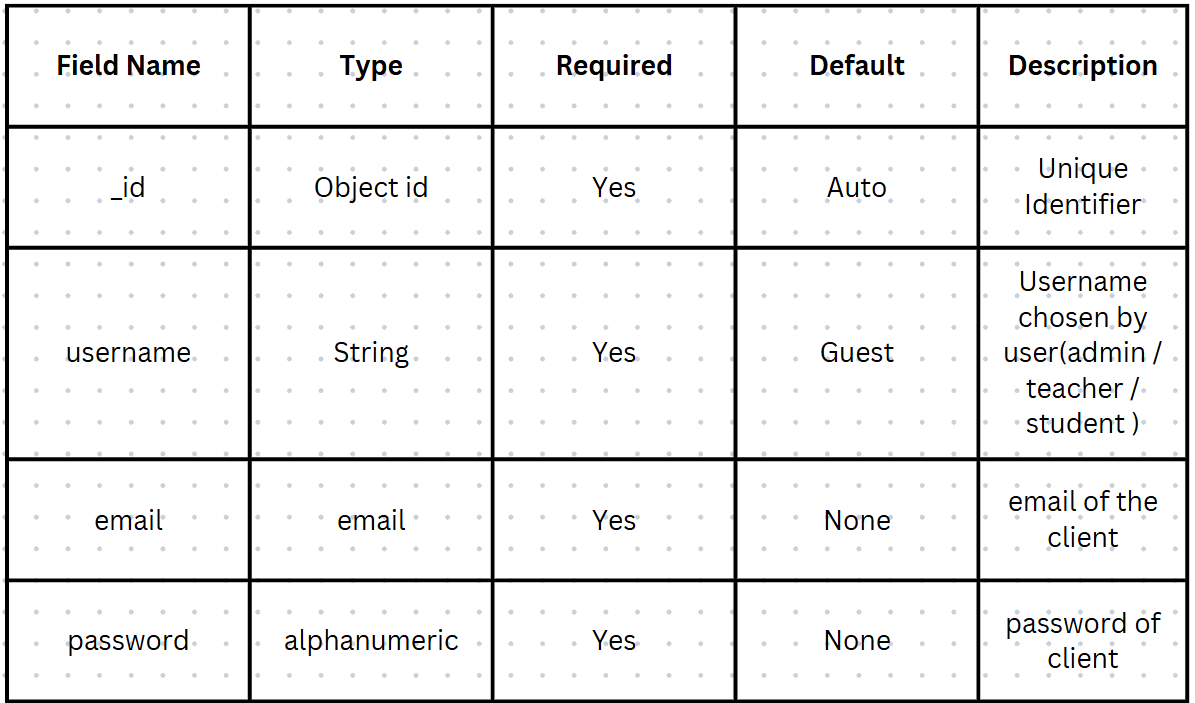
### User Model

Collection name: admin , teacher ,student

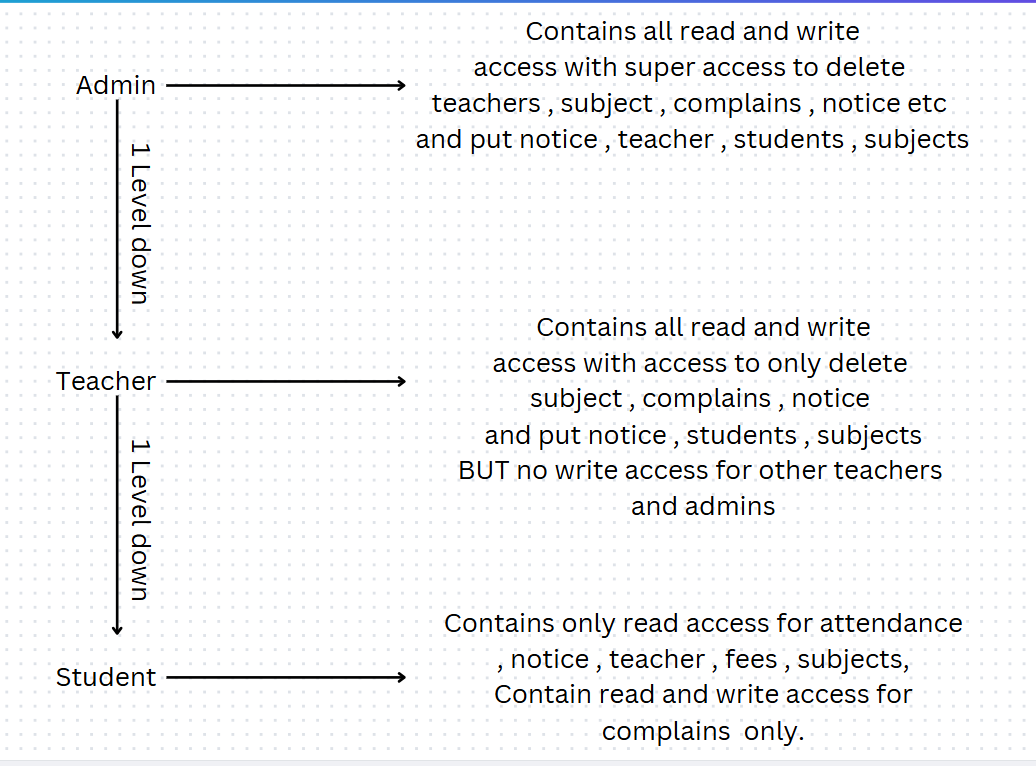
Purpose: Stores user information for authentication and authorization.

All the three have same type of table with same elements but have different access hiererchy as given below : -

**Table :-**



**Heirerchy :-**



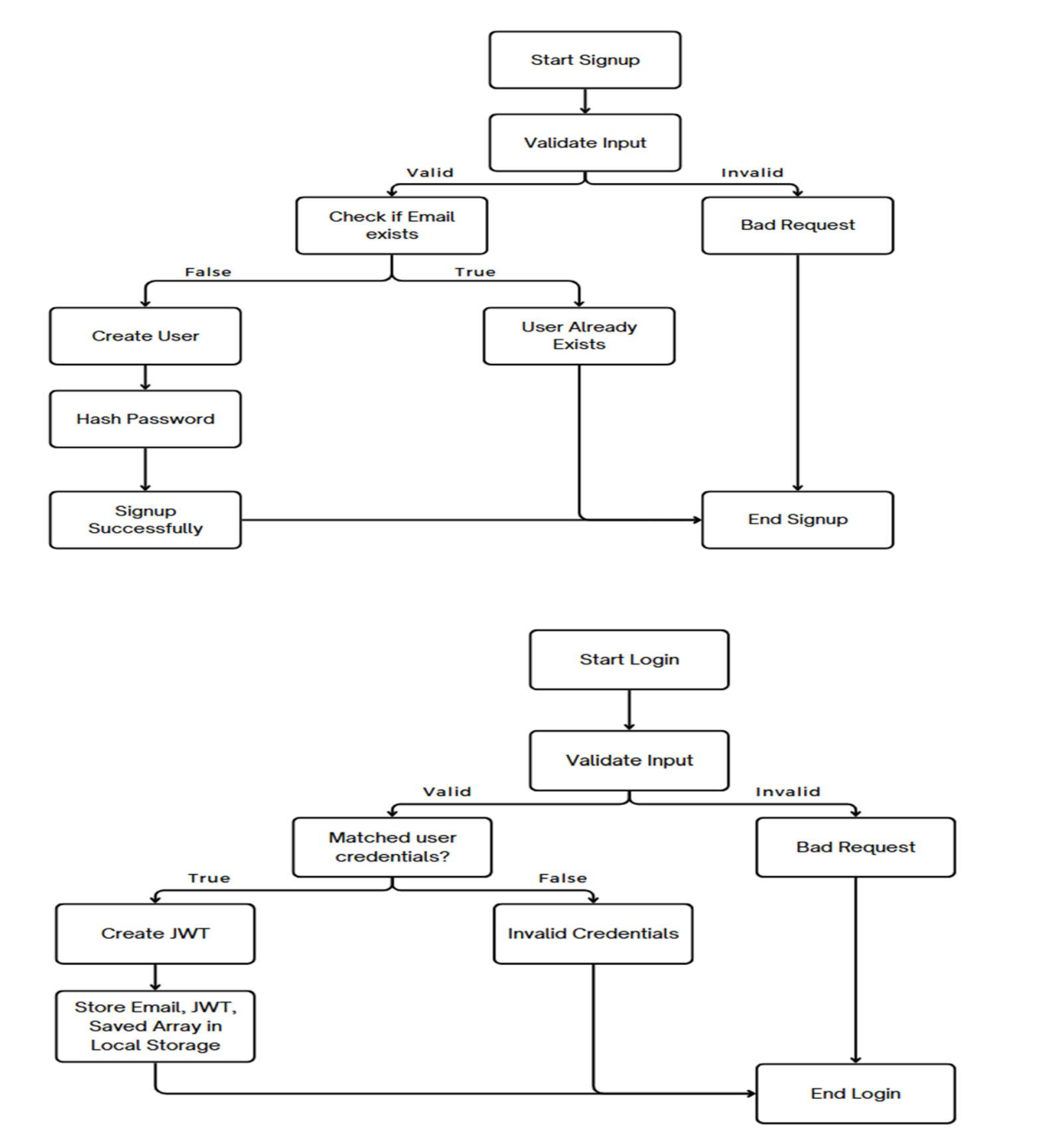
## Middleware

### Authentication Middleware

* + Validates inputs through joi
  + Protects routes requiring authentication

### Error Handling Middleware

* + Formats error responses
  + Provides device-specific error messages

**Authentication Flowchart**

# Frontend Design

## Technologies Used

* + **React.js**: UI library
  + **React Router**: Client-side routing
  + **React Toastify**: Toast notifications
  + **Tailwind CSS**: Utility-first CSS framework for responsive styling

## Responsive Design Implementation

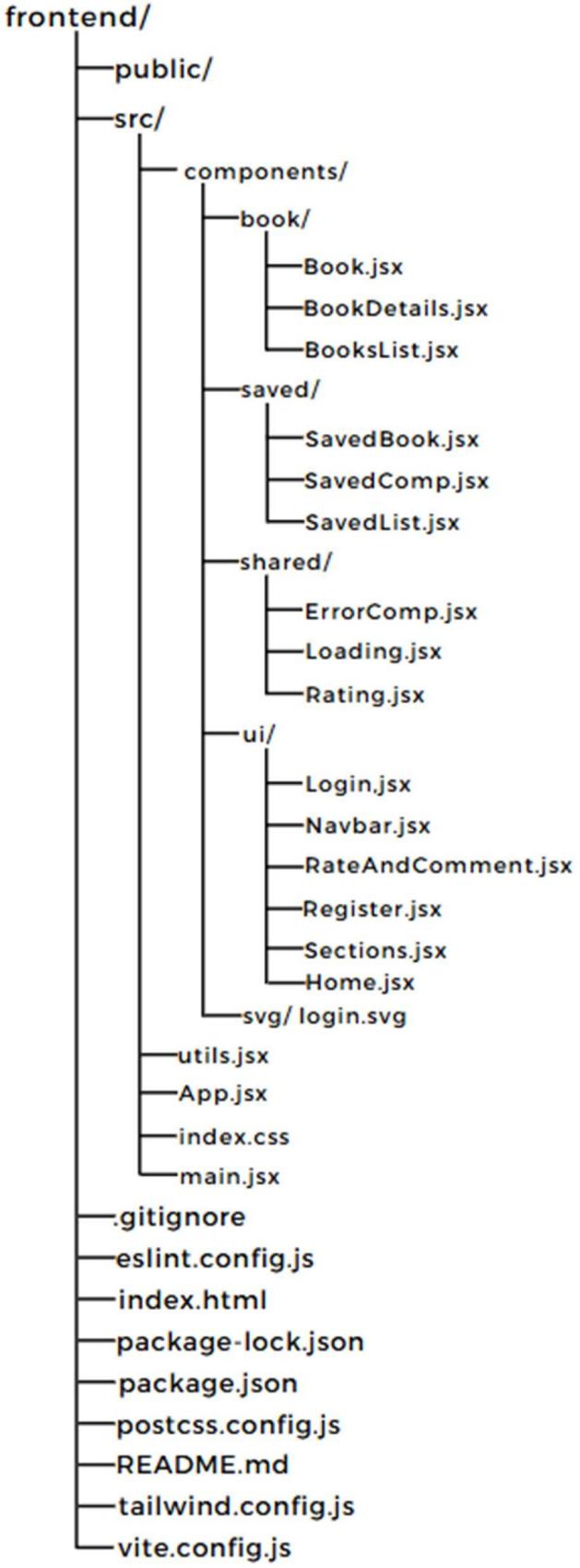
### Breakpoints

* + Desktop Computers: 1440px
  + Laptops : 1024px
  + Tablets: 768px
  + Mobiles: 426px

## Routing

* + /home: Responsive home page with recent books and saved books
  + /login: Responsive login page
  + /register: Responsive register page

**Project Structure**

****

# Security Considerations

## Authentication

* JWT implementation
* Secure password hashing with bcrypt
* Private Routing

## Data Protection

* + **Encryption at Rest:** MongoDB's AES-256 encryption to secure data stored on disk.
  + **Encryption in Transit:** TLS/SSL encryption to protect data during transmission.
  + **Authentication:** SCRAM, LDAP, or x.509 certificates for secure authentication.
  + **Backup and Recovery:** Encrypt and securely store backups, and ensure a reliable recovery process.

## API Security

* Secure headers implementation
* Request validation
* Error-handling

# Deployment

## Database Deployment

* + MongoDB Atlas cluster setup
  + Data migration strategy

## Backend Deployment

* + AWS EC2 instance
  + Environment configuration
  + Monitoring setup

## Frontend Deployment

* + AWS EC2 instance
  + Build optimization for different devices
  + CDN configuration

## Mobile Testing

* + Performance validation
  + Responsive design verification
  + Touch interaction testing

# Future Plans

## Short-term Enhancements

* Social sharing features
* Enhanced mobile user interface
* User Feedback

## Medium-term Goals

* User reading statistics
* User search history
* Light mode UI
* User book addition

## Long-term Vision

* AI-powered recommendations
* Different downloadable formats
* Additional API integrations
* Consumer support chatbot

# Conclusion

BookNXT represents a modern approach to digital book access and management, built on the MERN stack with comprehensive responsive design implementation. The platform’s architecture ensures seamless operation across all devices while maintaining high performance and user engagement. It provides an optimal reading experience regardless of the user’s chosen device.

The system’s modular design while maintaining security and performance standards. As BookNXT continues to evolve, this technical foundation will support new features and capabilities across an expanding range of devices and use cases.