**A Project Report on**

**“EduAdmin”**

**Submitted in partial fulfilment of requirement for the Award of degree**

**Bachelor of Computer Application**

**Of**

**Mangalore University**



**By**

**Bhakthishri S Sharma I Poornima U Varshitha**

**9071126184 192292714 192292744**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Internal Guide** |  |  |  |  | **External Guide** |
| **Ms. Hema Subhashini** |  |  |  |  | **Ms. Swathi Devadiga** |
| **Assistant professor,** |  |  |  |  | **Qtech Solutions,** |
| **Dept of Computer Science,** |  |  |  |  | **Mangalore** |

**Vivekananda college, Puttur**

**VIVEKANANDA DEGREE COLLEGE,**

**PUTTUR, D.K**



**VIVEKANANDA DEGREE COLLEGE, PUTTUR, D.K**

**2021-2022**

**Phone: 08251230455** [**https://vcputtur.ac.in/**](https://vcputtur.ac.in/)

**Declarations**

We hereby declare that the project entitled **“EduAdmin”** has been proposed by us during the year 2021-2022 under guidance and supervision of our internal guide Ms. Hema Subhashini, Assistant professor, Computer Science department, Vivekananda College, Puttur and external guide Ms.Swathi Devadiga, Qtech solutions, Mangalore Submitted to the Mangalore University in the partial fulfilment of the requirement of Bachelor Computer Application (BCA).

We also declare that this project is the result of our own efforts.

Place: Puttur

Date:15-06-2024

Bhakthishri S Sharma I

Poornima U

Varshitha

# **ACKNOWLEDGEMENT**

We express our sincere thanks to **Ms. Hema Subhashini**, Assistant Professor, Vivekananda College, Puttur whose guidance has helped us to complete this project in time.

We are indebted to **Mr.Vishnuganapathi Bha**t, Principal and **Mr.Prakash Kuma**r, H.O.D Computer Science Department, for their advice and suggestions at various stages of the work.

We express our deepest gratitude and respect to our internal guide **Ms. Hema Subhashini**, our Lecturer, Department of Computer Science, for her valuable guidance and encouragement while doing this project work.

Lastly, we take this opportunity to express our regards to all those who have supported us directly or indirectly in completing this project work.

Bhakthishri S Sharma I

Poornima U

Varshitha

**INDEX**

## 1.Introduction -------------------------------------------------- 1-4

1.1 System Introduction

1.1.1 Title

1.1.2 Project Category

1.1.3 Overview

1.2 Objectives

1.3 Scope of the system

1.4 Structure of the system

1.5 System Architecture

1.6 End user

1.7 Software / Hardware requirement Specification

1.7.1 Hardware Requirements

1.7.2 Software Requirements

## 2.Software Requirement Specification ------------------- 5-10

2.1 Introduction

2.2. Overall Description

2.2.1 Product Perspective

2.2.2 Product Functions

2.2.3 User Classes and Characteristics

2.2.4 General Constraints

2.2.5 Assumptions

2.3 Special Requirements

2.3.1 Software Requirements 2.3.2 Hardware Requirements

2.4 Functional requirements:

2.4.1 Admin

2.4.2 Bookstore

2.4.3 User

2.5 Design Constraints

2.6 System Attributes

2.7 Other Requirements

2.7.1 Safety Requirements

2.7.2 Security Requirements

## 3.System design -----------------------------------------------11-16

3.1 Introduction

3.2 Assumptions and constraints

3.2.1 Assumptions

3.2.2 General Constraints

3.3 Description of Programs

3.3.1 Context Flow Diagrams (CFD)

3.4.2 Data Flow Diagram (DED)

3.4 ER Diagram

## 4.Database design --------------------------------------------17-22

4.1 Introduction

4.3 Normalized Tables

4.3.1 Table Name: Student

## 5.System design -----------------------------------------------23-30

5.1. Introduction

5.2 Structure of the Software Package

5.3 Modular Decomposition Components

5.3.1 Admin

5.3.2 Student Management

5.3.3 Student

## 6.User Interface ----------------------------------------------31-43

6.1 Home Page

6.1.1 Home - Add

6.1.2 Home - Delete

6.1.3 Home - Edit

6.2 Profile Page

6.3 Student operations

6.3.1 Add Student

6.3.2 Edit Student

6.3.3 Delete Student

## Conclusion ------------------------------------------------------ 82

## Limitations ------------------------------------------------------ 83

## Scope for Enhancement -------------------------------------- 84

## Abbreviations and Acronyms--------------------------------85

## Bibliography / References ----------------------------------- 86

# **1. INTRODUCTION**

## 1.1 Introduction

EduAdmin is a comprehensive student database application designed to streamline the management of student information for educational administrators. This project provides a robust platform where administrators can easily enter, view, edit, and delete student records. Each student has a separate profile page that includes their registration date. Developed using the MERN stack (MongoDB, Express.js, React, Node.js), EduAdmin ensures efficient and secure data handling, hosted on Vercel for reliable and scalable deployment.

**1.1.2 Project Title**

EduAdmin

**1.1.3 Project Category**  Web Application

**1.1.3 Overview**

**EduAdmin** streamlines student information management for educational institutions. Administrators can add, view, edit, and delete student records. Each student profile shows detailed information, including registration dates, with data validation to ensure accuracy.

The User can perform the following activities:

* Add student
* View Students
* Edit Student Information
* Delete Student Records
* View Individual Student Profiles

The System can perform the following activities:

* Data Validation to Ensure Correct Formats
* Display of Registration Date in Student Profiles
* Efficient Data Management and Storage using MongoDB
* Seamless User Interface and Navigation using React
* Secure API Interaction using Express.js
* Reliable Hosting and Deployment on Vercel

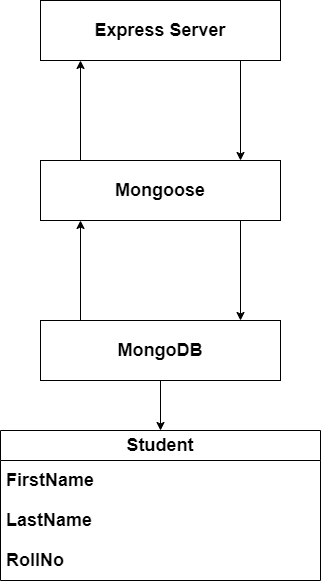
## 1.2 Objectives

* The main objective of the EduAdmin system is to manage all student information efficiently under one platform.
* Provide administrators with tools to add, view, edit, and delete student records.
* Ensure data integrity and accuracy through robust validation mechanisms.
* Display comprehensive student profiles, including registration dates.
* Facilitate easy navigation and user interaction with a seamless user interface.

## 1.3 Scope of the system

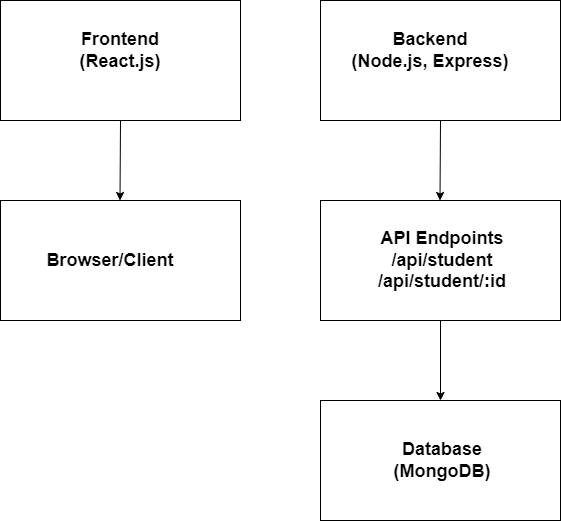
EduAdmin offers a comprehensive platform for educational institutions to manage student data effectively. The role of the administrator includes adding new students, viewing existing records, editing student information, and deleting records as necessary. Each student's profile page displays detailed information, including their registration date, ensuring thorough and accurate record-keeping.

**1.4 Structure of the System**



## 1.5 Software Architecture

Architectural design involves decomposing a large, complex system into smaller subsystems that provide related services. This layout framework controls and communicates with each subsystem. The "EduAdmin" student database application uses the MERN stack to create a responsive, secure, and scalable platform for managing student information.



### 1.6 End User

* **Administrator:**
  + Responsible for managing student records, including adding, viewing, editing, and deleting student information.
  + Has access to view detailed student profiles and perform administrative functions.

### 1.7 Software / Hardware Requirements Specification

#### **1.7.1 Hardware Requirements**

* **Device:** Desktop or Laptop
* **RAM:** 4GB and above

#### **1.7.2 Software Requirements**

* **Frontend:** React (React-router-dom for routing)
* **Backend:** Node.js, Express.js
* **Database:** MongoDB Atlas
* **Editor:** Visual Studio Code
* **OS:** Any modern OS

These requirements ensure that EduAdmin runs smoothly and efficiently, providing a reliable platform for managing student information.

**2. SOFTWARE REQUIREMENT SPECIFICATION**

## 2.1 Introduction

A Software Requirements Specification (SRS) is a document that describes what the software will do and how it will be expected to perform. It also describes the functionality the product needs to fulfill all user needs. This Software Requirements Specification document provides a complete description of all the functionalities and the specifications of the "EduAdmin" system. The following section provides an overview of the derived Software Requirements Specification (SRS) for the application. To begin with, the purpose of the document is presented, and its intended audience is outlined. Subsequently, the scope of the project specified by the document is given with a particular focus on what the resultant software will do and the relevant benefits associated with it.

**2.2 Overall Description**

This section gives an overview of the whole system. The system will be explained in its context to show how it interacts with other systems and introduce its basic functionality. It will also describe the types of users who will use the system and what functionality is available for each type. Finally, the constraints and assumptions for the system will be presented.

**2.2.1 Product Perspective**

The Bookify software is a fully independent product and not a part of any other system. The users of the system are categorized as admin, user and book store. The system gives online access to each separate user. This system provides an easy way for the users, so they can make use of the relevant books. This Project is self-contained. It provides a simple database rather than complex ones for high requirements and it provides a good and easy graphical user interface (GUI) to both new as well as experienced users of the system. This project mainly focuses on the solution regarding the difficulty to maintain books and profit margins manually.

The EduAdmin system includes the following functionalities:

* **Add Student:** Allows administrators to input new student details into the database.
* **View Students:** Administrators can view a comprehensive list of all students, including essential details.
* **Edit Student Information:** Administrators can update existing student records as needed.
* **Delete Student Records:** Administrators can remove student records from the database.
* **View Student Profiles:** Each student has a detailed profile page that includes their registration date.

##### **2.2.3 User Characteristics**

* **Administrators:** Users responsible for managing student records. They should be familiar with basic computer operations and have a moderate understanding of database management.

##### **2.2.4 Constraints**

* **Data Validation:** The system must ensure all entered data adheres to the correct format to maintain data integrity.
* **Security:** Only authorized administrators should have access to the system.
* **Scalability:** The system should be able to handle an increasing number of student records without performance degradation.

##### **2.2.5 Assumptions**

* The administrators have basic computer literacy.
* The educational institution has a reliable internet connection to access the online system.
* The system will be accessed primarily through desktop or laptop computers.

These sections outline the overall description and functionality of the EduAdmin system, providing a clear understanding of what the software aims to achieve and the requirements needed to fulfill its purpose.

## 2.3 SPECIFIC REQUIREMENTS

**2.3.1 Software Requirements**

* Operating system: Windows
* Text editor/IDE: Visual Studio Code
* Language: Javascript
* Frameworks: React (for frontend), Node.js with Express.js (for backend)
* Database: MongoDB Atlas
* Browser: Chrome, Mozilla Firefox, or any other modern browsing application

**2.3.2 Hardware Requirements**

* Processor: Intel Pentium dual-core or above
* Processor Speed: 2GHz
* **RAM:** 4GB and above
* Hard Disk: Minimum 40 GB

## 2.4 Functional Requirements

**2.4.1 Functional Requirements**

**Admin:**

* **Login**: Admin will enter the system using an email and password.
* **Add Student:** Admin can add new student details to the database, including first name, last name, and roll number.
* **View Students:** Admin can view a comprehensive list of all students with essential details.
* **Edit Student Information:** Admin can update existing student records.
* **Delete Student Records:** Admin can remove student records from the database.
* **View Student Profiles:** Admin can view detailed profiles of each student, including registration dates.

**System Functions:**

* **Data Validation:** Ensures that all entered data adheres to the correct format.
* **Profile Management:** Displays detailed profile information for each student.
* **Routing:** Seamless navigation using React-router-dom.
* **API Requests:** Efficient communication between frontend and backend using the Axios library.

This section outlines the detailed functional requirements for the EduAdmin system, specifying the necessary software and hardware components as well as the key functionalities expected from the system.

## 2.5 Design Constraints:

* **Mandatory Field Validation:** While administrators are adding or editing student records, mandatory fields such as first name, last name, and roll number must be validated to ensure appropriate data is entered. If not, a proper error message should be displayed, and the data should not be stored until all mandatory fields are correctly filled.
* **Profile Details Validation:** All mandatory fields must be completed by administrators while updating student profile details.
* **User-Friendly Design:** The system must be designed to be easy to use and visible on most modern web browsers.
* **Data Integrity:** Ensure that all data entered is validated and stored correctly to maintain data integrity and avoid inconsistencies.

## 2.6 System Attributes

The Quality of the database is maintained in such a way that it can be very user-friendly to users of the database.

* **Performance:** The system should be able to handle multiple administrators accessing and updating student records simultaneously using any modern web browser without significant performance degradation.
* **Reliability:** The system must include robust validation of user inputs to avoid incorrect storage of records. Regular backups should be maintained to prevent data loss.
* **User-Friendly Interfaces:** The system should have intuitive and easy-to-use screen layouts, making it straightforward for administrators to manage student information.
* **Maintainability:** The system should be designed in such a way that modifications and updates can be made with minimal effort. This includes a clean and modular codebase.
* **Portability:** The system shall be portable, allowing for easy switching of servers if needed. This includes using cloud-based solutions like MongoDB Atlas and Vercel for deployment.
* **Flexibility:** The system should be flexible, allowing for updates and changes to student data and system features as required.
* **Timeliness:** The system should perform all operations efficiently, minimizing the time required for tasks such as adding, updating, and viewing student records.

These sections provide a clear understanding of the constraints and attributes required for the EduAdmin system, ensuring it meets the necessary standards for functionality, usability, and performance.

## 2.7 Other Requirements

**2.7.1 Safety Requirements:**

* **User Levels:** There is one user level, which is the administrator. Access to the system will be protected by a login screen requiring a username and password. This ensures only authorized personnel can access and manage student records.
* **Immutable Email ID:** Once registered, the email ID associated with an admin account cannot be changed. This ensures each admin is unique and easily identifiable.
* **Data Backup:** Regular backups must be maintained to ensure the security of the system database. This will allow the system to be restored in case of any emergency, preventing data loss.

**2.7.2 Security Requirements:**

* **Server Security:** The server hosting the EduAdmin application will have security measures in place to prevent unauthorized write or delete access. Reading access will be restricted to authenticated users.
* **Secure Application:** The EduAdmin system will be secure, with different access rights defined for various functionalities. Admins will have maximum privileges to access all parts of the system.
* **Authentication and Authorization:** Strong authentication mechanisms will be implemented to ensure only authorized admins can log in. Access control measures will ensure that only those with the right privileges can perform specific actions like adding, editing, or deleting student records.

These additional requirements ensure the EduAdmin system is safe and secure, protecting both user data and system integrity.

**Creating a 40-page project report for your "EduAdmin" project involves detailed documentation across several sections, including introduction, system requirements specification (SRS), design, implementation, testing, conclusion, and appendices. Below is an outline and some detailed sections to guide you. Each section will need to be expanded with detailed descriptions, diagrams, and other content to reach the desired length.**

**### EduAdmin Project Report**

**---**

**#### Title Page**

**- Project Title: EduAdmin**

**- Your Name**

**- Your Institution**

**- Date**

**#### Table of Contents**

**1. Introduction**

**2. System Requirements Specification (SRS)**

**- Functional Requirements**

**- Non-Functional Requirements**

**- Use Case Diagrams**

**3. System Design**

**- Architectural Design**

**- ER Diagram**

**- Data Flow Diagrams (DFD)**

**4. Implementation**

**- Technology Stack**

**- Code Structure**

**- Key Features**

**5. Testing**

**- Test Plan**

**- Test Cases**

**- Results**

**6. Deployment**

**- Hosting Platform**

**- Deployment Steps**

**7. Conclusion**

**8. References**

**9. Appendices**

**- User Manual**

**- Installation Guide**

**---**

**### 1. Introduction**

**EduAdmin is a comprehensive student database application designed to simplify student information management for educational administrators. Developed using the MERN stack (MongoDB, Express.js, React, Node.js), EduAdmin provides a robust platform for managing student records, including adding, viewing, editing, and deleting student information. This report details the project's specifications, design, implementation, testing, and deployment processes.**

**### 2. System Requirements Specification (SRS)**

**#### 2.1 Functional Requirements**

**1. \*\*User Management:\*\***

**- Admin can add new students.**

**- Admin can view a list of all students.**

**- Admin can edit student details.**

**- Admin can delete student records.**

**2. \*\*Data Validation:\*\***

**- Ensure that student information is entered in the correct format (e.g., name contains only letters, roll number is unique and numeric).**

**3. \*\*Student Profile:\*\***

**- Each student has a separate profile page displaying detailed information, including registration date.**

**#### 2.2 Non-Functional Requirements**

**1. \*\*Performance:\*\***

**- The application should handle a large number of student records efficiently.**

**2. \*\*Usability:\*\***

**- The user interface should be intuitive and easy to navigate.**

**3. \*\*Reliability:\*\***

**- The application should be reliable and maintain data integrity.**

**4. \*\*Security:\*\***

**- Data should be securely stored and transmitted.**

**5. \*\*Scalability:\*\***

**- The system should be able to scale to accommodate a growing number of users and records.**

**#### 2.3 Use Case Diagrams**

**- Include detailed use case diagrams showing the interactions between admin users and the system.**

**### 3. System Design**

**#### 3.1 Architectural Design**

**- Explain the overall architecture of the system, including client-server interactions, and the role of each component (MongoDB, Express.js, React, Node.js).**

**#### 3.2 ER Diagram**

**- Include the Entity-Relationship (ER) diagram showing the relationships between different data entities such as Student, Admin, and other relevant entities.**

**#### 3.3 Data Flow Diagrams (DFD)**

**- Provide detailed DFDs (Level 0, Level 1, etc.) illustrating the flow of information within the system.**

**### 4. Implementation**

**#### 4.1 Technology Stack**

**- Detail the technologies used: MongoDB for database, Express.js for backend API, React for frontend, Node.js for server-side operations, Axios for HTTP requests, and Vercel for deployment.**

**#### 4.2 Code Structure**

**- Describe the structure of the codebase, including key directories and files.**

**#### 4.3 Key Features**

**- Explain the implementation of key features such as adding, viewing, editing, and deleting student records, data validation, and individual student profiles.**

**### 5. Testing**

**#### 5.1 Test Plan**

**- Outline the overall testing strategy, including unit testing, integration testing, and system testing.**

**#### 5.2 Test Cases**

**- Provide detailed test cases for each feature, including input data, expected outcomes, and actual results.**

**#### 5.3 Results**

**- Summarize the results of testing, including any defects found and resolved.**

**### 6. Deployment**

**#### 6.1 Hosting Platform**

**- Discuss the choice of Vercel for hosting and its benefits.**

**#### 6.2 Deployment Steps**

**- Provide a step-by-step guide on how to deploy the application on Vercel.**

**### 7. Conclusion**

**Summarize the project, its achievements, and potential future enhancements. Reflect on the learning experience and the practical application of the technologies used.**

**### 8. References**

**List all the references used in the project report, including books, articles, and online resources.**

**### 9. Appendices**

**#### 9.1 User Manual**

**- Provide a detailed user manual, including how to use each feature of the application.**

**#### 9.2 Installation Guide**

**- Include an installation guide for setting up the development environment and deploying the application.**

**---**

**### Detailed Content Examples**

**Here are examples of detailed content for specific sections to help you expand your report:**

**#### ER Diagram**

**Create an ER diagram using a tool like draw.io or any other diagramming tool, showing the relationships between entities such as:**

**- \*\*Student\*\*: student\_id (PK), first\_name, last\_name, roll\_no, registration\_date**

**- \*\*Admin\*\*: admin\_id (PK), username, password**

**- \*\*Course\*\*: course\_id (PK), course\_name, description**

**- \*\*Enrollment\*\*: enrollment\_id (PK), student\_id (FK), course\_id (FK), enrollment\_date**

**#### Use Case Diagram**

**Create use case diagrams using a diagramming tool, showing interactions such as:**

**- \*\*Admin\*\*: Login, Add Student, View Students, Edit Student, Delete Student**

**- \*\*Student\*\*: View Profile**

**#### Implementation Details**

**Provide code snippets and explanations for key components. For example:**

**- \*\*Backend API (Express.js)\*\*:**

**```javascript**

**const express = require('express');**

**const router = express.Router();**

**const Student = require('../models/Student');**

**// Add Student**

**router.post('/add', async (req, res) => {**

**try {**

**const { firstName, lastName, rollNo } = req.body;**

**const newStudent = new Student({ firstName, lastName, rollNo });**

**await newStudent.save();**

**res.status(201).json(newStudent);**

**} catch (error) {**

**res.status(400).json({ message: error.message });**

**}**

**});**

**// Other routes (view, edit, delete)**

**```**

**- \*\*Frontend (React)\*\*:**

**```javascript**

**import React, { useState, useEffect } from 'react';**

**import axios from 'axios';**

**const StudentList = () => {**

**const [students, setStudents] = useState([]);**

**useEffect(() => {**

**const fetchStudents = async () => {**

**try {**

**const response = await axios.get('/api/students');**

**setStudents(response.data);**

**} catch (error) {**

**console.error('Error fetching students:', error);**

**}**

**};**

**fetchStudents();**

**}, []);**

**return (**

**<div>**

**<h2>Student List</h2>**

**<ul>**

**{students.map(student => (**

**<li key={student.\_id}>{student.firstName} {student.lastName}</li>**

**))}**

**</ul>**

**</div>**

**);**

**};**

**export default StudentList;**

**```**

**By providing detailed descriptions, diagrams, code snippets, and thorough explanations for each section, you can expand this outline into a comprehensive 40-page report.**