WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

"Online Id Card Booking And Status Tracking Application "

Ms.N.Swathi[1] , SaiKiran Panugothu [2], K RAMU[3], Guguloth Naveen[4]

[1]Associate. Professor, CSE Dept, ACE Engineering College, Hyderabad, India
swathi.neeloju@gmail.com
[2]Student, CSE Dept, ACE Engineering College, Hyderabad, India
psainayak96@gmail.com
[3] Student, CSE Dept, ACE Engineering College, Hyderabad, India
rambharath820@gmail.com
[4] Student, CSE Dept, ACE Engineering College, Hyderabad, India
itsnaveen6@gmail.com

ABSTRACT :

In today's educational landscape, the student ID card stands as a pivotal element, granting Access to college facilities, exam halls, and libraries. It serves as an emblem of student identity within the institution. However, the regular methods of obtaining and replacing ID cards are time-consuming manual processes. Acquiring an ID card during the initial days of college or applying for loss or damaged one involves tasks such as writing letters to the HOD, roaming admin blocks, and monitoring status updates through repeated visits. To address the challenges of manual and time-consuming ID card management, we propose a dedicated mobile application for college students. This user-friendly platform empowers students to effortlessly request and manage their ID cards. By logging in, students can initiate ID card requests, make secure online payments, and track their request's progress in real-time. The application's integration with React Native technology ensures a seamless experience, and enhancing overall efficiency. This transformative solution replaces outdated card management process, significantly reducing time and effort while aligning with modern digital standards. This innovation reflects our commitment to enhancing the college experience through efficient and user centric solutions

1. INTRODUCTION :

Introducing the next evolution in college administration: a cutting-edge mobile application designed to revolutionize the management of student ID cards. In today's educational realm, where efficiency and convenience are paramount, the traditional methods of obtaining and replacing ID cards have long been a source of frustration for students. Cumbersome manual processes, involving letters to department heads and multiple visits to administrative offices, have been the norm, causing unnecessary delays and inconvenience.

In response to these challenges, we proudly present our innovative solution: a dedicated mobile app tailored specifically for college students. This intuitive platform empowers students to seamlessly request and manage their ID cards with just a few taps on their smartphones. Gone are the days of tedious paperwork and endless waiting – with our app, students can initiate ID card requests, securely make payments online, and track their request's progress in real-time, all from the palm of their hand.

Built on React Native technology, our app ensures a smooth and seamless user experience across all devices, enhancing overall efficiency and usability. By replacing outdated manual processes with a modern, digital solution, we are not only streamlining ID card management but also elevating the college experience for students.

At the heart of our innovation lies a commitment to efficiency, convenience, and user-centric design. Join us as we usher in a new era of college administration, where technology meets tradition to create a seamless and empowering experience for students.

2. OBJECTIVES

The project aims to streamline and modernize the student ID card management process within the college environment. The current manual procedures for requesting and replacing ID cards are time consuming and involve multiple administrative steps. Our primary objective is to develop a dedicated mobile application for college students, offering a user-friendly platform that simplifies ID card-related tasks. Through the application, students can log in, initiate ID card requests, make secure online payments, and track the real-time progress of their requests. Leveraging React Native technology ensures a smooth and responsive interface. By replacing outdated processes with this digital solution, the project aims to significantly reduce the time and effort required for ID card management. The integration of modern technology aligns with current digital standards, emphasizing our commitment

to enhancing the efficiency and user experience within the educational institution. In essence, the project seeks to set a new standard for streamlined, user-friendly, and digitally empowered student ID card management

.3. PROBLEM STATEMENT

The current manual processes associated with student ID card management in the college setting present significant challenges, leading to inefficiencies and inconveniences for students. Obtaining or replacing ID cards involves navigating through complex administrative procedures, submitting physical paperwork, and enduring multiple visits to administrative offices. These outdated methods not only consume considerable time and effort but also contribute to a lack of user satisfaction and hinder the overall efficiency of the college's administrative processes. Additionally, the absence of a streamlined digital solution further exacerbates these challenges, as students lack a user-friendly platform to initiate and manage ID card-related tasks seamlessly. The absence of real-time tracking mechanisms for ID card requests contributes to uncertainty and frustration among students, impacting their overall experience within the academic institution. In light of these issues, there is a pressing need for a modernized and efficient system that can replace the current manual ID card management processes. This project addresses these challenges by proposing a dedicated mobile application that streamlines the entire ID card management workflow, providing students with a convenient, secure, and digitally empowered solution to enhance their overall experience within the college environment

4. LITERATURE SURVEY

The literature survey for the proposed student ID card management mobile application reveals a growing acknowledgment of the challenges associated with manual processes in educational institutions and a concurrent trend towards digital solutions. The following literature highlights key insights into the existing landscape and explores the potential benefits and innovations that can be derived from implementing a dedicated mobile application for student ID card management.

1. Digital Transformation in Education: - Numerous studies emphasize the ongoing digital transformation within educational institutions. Digital solutions are increasingly being explored to streamline administrative processes, enhance user experience, and improve overall efficiency.
2. Challenges of Manual ID Card Management: - Research indicates that the manual procedures involved in issuing and replacing student ID cards pose significant challenges. Cumbersome administrative steps and the lack of real-time tracking contribute to time wastage, inefficiency, and frustration among students.
3. Mobile Applications in Education: - A growing body of literature discusses the benefits of mobile applications in educational settings. Mobile apps have been successfully implemented to improve communication, streamline processes, and provide students with convenient access to essential services. 4. User-Centric Design in Educational Apps: - Studies on user-centric design principles underscore the importance of creating applications that are intuitive, easy to navigate, and responsive. User satisfaction plays a pivotal role in the success and adoption of mobile applications within educational contexts. 5. Integration of React Native Technology: - Literature highlights the significance of choosing appropriate technologies for mobile application development. The integration of React Native technology is explored as a means to ensure a seamless, cross-

platform experience, contributing to enhanced usability and accessibility. 6. Security and Privacy Considerations: - Discussions around digital solutions in educational institutions emphasize the importance of robust security measures to safeguard sensitive student information. Research emphasizes the need for secure online payment processes and data protection in mobile applications. 7. Case Studies on Successful Implementations: - Case studies of educational institutions that have successfully implemented digital solutions for ID card management shed light on best practices, challenges faced, and outcomes achieved. These real-world examples provide valuable insights for the proposed project.

In conclusion, the literature survey underscores the urgency and relevance of transitioning from manual to digital solutions in the realm of student ID card management. The proposed mobile application aligns with the broader trends in educational technology, offering a user-centric, efficient, and secure approach to address the identified challenges within the context of college administrative processes.

5. PROPOSED SYSYTEM

The proposed system for student ID card management introduces a comprehensive and user-centric digital solution designed to replace the existing manual processes. Central to this innovation is a dedicated mobile application tailored for college students, offering an intuitive platform for initiating and managing ID card-related tasks. By eliminating the need for physical paperwork, the online request initiation feature simplifies the process, reducing administrative complexities. Secure online payment functionalities within the application enhance convenience, eliminating the necessity for in person transactions.

A key advancement is the real-time progress tracking feature, providing students with immediate updates on their ID card requests and minimizing the need for repeated visits to administrative offices. The integration of React Native technology ensures a responsive and cross-platform experience, addressing the diverse technological preferences of students. Digital verification and authentication streamline the verification process, expediting ID card issuance while maintaining security and accuracy. Built on user-centric design principles, the proposed system prioritizes an intuitive, easy-to-navigate interface to enhance the overall user experience. Immediate accessibility of ID cards is facilitated, whether during the initial days of college or in cases of loss or damage. The system significantly reduces administrative dependency by automating key processes, contributing to a more streamlined workflow and enhanced administrative efficiency during peak periods. In essence, the proposed system represents a transformative shift towards a modernized, efficient, and digitally empowered approach to student ID card management within educational institutions.

6. HARDWARE AND SOFTWARE REQUIREMENTS

6.1 HARDWARE REQUIREMENTS:

- Processor – Pentium, mac M1
- RAM – 8 GB (min)
- Hard Disk – 20 GB
- Key Board – Standard Windows Keyboard
- Mouse – Two or Three Button Mouse
- Monitor – SVGA

6.2 SOFTWARE REQUIREMENTS:

- Operating system – Windows 7, 8, 10, 11, Mac OS
- Frontend: React Native, React Js, React-Router-DOM
- Backend: NodeJS, Express Js, PostgreSQL
- State-Management: Redux-toolkit
- Authentication and Security: JWT Auth
- API: RESTful API's
- Push Notifications: Firebase Push Notifications , Notifee (Local Notifications).

7. PACKAGES USED

"dependencies": {

"@kichiyaki/react-native-barcode-generator": "^0.6.7",
"@notifee/react-native": "^7.8.2",
"@react-native-async-storage/async-storage": "^1.15.5",
"@react-native-community/blur": "^3.6.0",
"@react-native-community/datetimepicker": "^3.5.2",
"@react-native-community/netinfo": "^5.9.10",
"@react-native-masked-view/masked-view": "^0.2.8",
"@react-native-picker/picker": "^1.16.1",
"@react-navigation/bottom-tabs": "^6.2.0",
"@react-navigation/native": "^6.0.8",
"@react-navigation/stack": "^6.1.1",
"@reduxjs/toolkit": "^1.9.1",
"@vm-lib/patch-node-modules": "^1.0.0",
"axios": "^0.21.1",
"deprecated-react-native-prop-types": "^2.3.0",
"jwt-decode": "^3.1.2",
"lottie-ios": "3.2.3",
"lottie-react-native": "5.0.1",
"react": "18.1.0",
"react-native": "0.70.6",
"react-native-app-auth": "^6.4.3",
"react-native-app-intro-slider": "^4.0.4",
"react-native-base64": "^0.1.1",
"react-native-camera": "^4.2.1",
"react-native-device-info": "^10.12.0",
"react-native-document-picker": "^9.1.0",
"react-native-element-dropdown": "^2.10.4",
"react-native-fbsdk-next": "^7.3.3",
"react-native-file-viewer": "^2.1.5",
"react-native-fs": "^2.20.0",
"react-native-gesture-handler": "^1.10.3",
"react-native-haptic-feedback": "^2.2.0",
"react-native-image-crop-picker": "^0.36.3",
"react-native-image-picker": "^7.1.0",

```
    "react-native-keychain": "^6.2.0",
    "react-native-linear-gradient": "^2.8.3",
    "react-native-notifications": "4.3.3",
    "react-native-pager-view": "^6.2.3",
    "react-native-permissions": "^4.1.4",
    "react-native-popup-menu": "^0.15.10",
    "react-native-reanimated": "^2.13.0",
    "react-native-safe-area-context": "^3.3.2",
    "react-native-screens": "3.20.0",
    "react-native-size-matters": "^0.4.0",
    "react-native-skeleton-placeholder": "^5.2.4",
    "react-native-snap-carousel-v2-maintained": "^3.9.4",
    "react-native-splash-screen": "^3.2.0",
    "react-native-step-indicator": "^1.0.3",
    "react-native-svg": "^12.1.1",
    "react-native-svg-transformer": "0.14.3",
    "react-native-tab-view": "^3.5.2",
    "react-native-toast-message": "^2.1.6",
    "react-native-ui-lib": "^6.29.0",
    "react-native-video": "^5.2.1",
    "react-redux": "^7.2.4",
    "redux": "^4.1.0",
    "redux-persist": "^6.0.0",
    "redux-thunk": "^2.3.0",
    "simple-react-validator": "^1.6.0",
    "tailwind-rn": "^4.2.0",
    "tailwind-safelist-generator": "^0.1.3"
},
```


9. SOURCE CODE:

BACKEND:

index.js

```
import express from "express";
import dotenv from "dotenv";
import cors from "cors";
import adminlogin from "./routes/adminroutes/adminlogin.js";
import admin from "./routes/adminroutes/admin.js";
import student from "./routes/studentroutes/student.js";
import studentlogin from "./routes/studentroutes/studentlogin.js";
import studentAuth from "./middlewares/student/studentAuth.js";
import studentorders from "./routes/studentroutes/studentorders.js";

dotenv.config();
const app = express();
const port = process.env.PORT;
console.log(port);

//middlewares
app.use(cors());
app.use(express.urlencoded({ extended: false }));
app.use(express.json());

//adminroutes
```

```
app.use("/login", adminlogin);
app.use("/admin", admin);

//studentAPI's
app.use("/api/student", studentlogin);
app.use("/api/student", studentAuth, student);
app.use("/api/students/orders",studentAuth,studentorders)

app.listen(port, () => {
  console.log(`server connected to port ${port}`);
});
```

admin.js

```
import express from "express";
import students from "../../controllers/admincontroller/students.js";
import addstudent from "../../controllers/admincontroller/addstudent.js";
import updatestudent from "../../controllers/admincontroller/updatestudent.js";
import deletestudent from "../../controllers/admincontroller/deletestudent.js";
import getstudent from "../../controllers/admincontroller/getstudent.js";
import getorders from "../../controllers/admincontroller/getorders.js";
import { upload } from "../../middlewares/multer/multer.js";
import pricedetails from "../../controllers/admincontroller/pricedetails.js";
import updateorder from "../../controllers/admincontroller/updateorder.js";
import pastorders from "../../controllers/admincontroller/pastorders.js";

const admin = express.Router();

admin.get("/students", students);
admin.post(
  "/addstudent",
  upload.fields([{ name: "stuImg", maxCount: 1 }]),
  addstudent
);
admin.put(
  "/updatestudent/:stuid/:imgupdated",
  upload.fields([{ name: "stuimg", maxCount: 1 }]),
  updatestudent
);
admin.delete("/deletestudent/:id", deletestudent);
admin.get("/getstudent/:id", getstudent);
admin.get("/getorders", getorders);
admin.get("/pastorders", pastorders);
admin.put("/updateorder/:orderid", updateorder);
admin.put("/pricedetails", pricedetails);
```

```
export default admin;
```

Student.js

```
import express from "express";
import changepassword from "../../controllers/studentcontroller/changepassword.js";
import getstudentdetails from
"../../controllers/studentcontroller/getstudentdetails.js";
import getaccountdetails from
"../../controllers/studentcontroller/getaccountdetails.js";
import editdetails from "../../controllers/studentcontroller/editdetails.js";
import pricedetails from "../../controllers/studentcontroller/pricedetails.js";
editdetails;


const student = express.Router();

student.get("/getstudentdetails", getstudentdetails);
student.get("/getaccountdetails", getaccountdetails);
student.patch("/changepassword", changepassword);
student.patch("/editdetails/:editId", editdetails);
student.get('/price/details',pricedetails)

export default student;
```

Studentorders.js

```
import express from "express";
import editdetails from "../../controllers/studentcontroller/editdetails.js";
editdetails;
import { upload } from '../../middlewares/multer/multer.js';
import getorderdetails from
"../../controllers/studentcontroller/getorderdetails.js";
import getactiveorder from "../../controllers/studentcontroller/getactiveorder.js";
import getpastorders from "../../controllers/studentcontroller/getpastorders.js";
import ordercancel from "../../controllers/studentcontroller/ordercancel.js";
import placeorder from "../../controllers/studentcontroller/placeorder.js";

const studentorders = express.Router();

studentorders.post(
  "/placeorder",
  upload.fields([{ name: "stuImg", maxCount: 1 }]),
  placeorder
);
studentorders.get('/getactiveorder',getactiveorder)
studentorders.get('/getorderdetails',getorderdetails)
studentorders.get('/getpastorders',getpastorders)
studentorders.patch('/cancelorder',ordercancel)
```

```
export default studentorders;
```

Multer.js

```
import multer from "multer";


const storage = multer.diskStorage({
    destination: function (req, file, cb) {
      cb(null, './storage/temp')
    },
    filename: function (req, file, cb) {
      cb(null, file.originalname)
    }
  })

  export const upload = multer({ storage })
```

studentsAuth.js

```
import jwt from "jsonwebtoken";

const studentAuth = async (req, res, next) => {
  const { authorization } = req.headers;
  const token = authorization?.split(" ")[1];
  console.log(token);
  if (!token) return res.status(401).json({ message: "Token is mandatory" });

  try {
    jwt.verify(token, process.env.SECRET_ACCESS_KEY, (err, user) => {
      if (err) return res.status(403).json({ message: "Invalid token" });
      req.student = user.stuId
      console.log(req.student,"student");
      next();
    });
  } catch (error) {
    console.error("Error:", error);
    return res.status(500).json({ message: "Internal Server Error" });
  }
};

export default studentAuth;
```

DB.sql

```
create database idcard;

create table admin (
    id serial primary key,
    username varchar(30) not null,
```

```
    passkey varchar(100) not null
);

-- remove this query in prod
insert into admin (username,passkey) values
("aceecadmin","$2b$10$H.oRtt.WodaFAGRQoY5G3u/MfCvf5JmkrkafkhYCbmORSTiujQiG2");

create table students(
    stuId varchar(20) primary key,
    firstName varchar(30) not null,
    lastName varchar(30) not null,
    course varchar(20) not null,
    branch varchar(20) not null,
    batch varchar(10) not null,
    stuImg varchar(300) not null,
    dob varchar(30) not null,
    fatherName varchar(60) not null,
    address varchar(100) not null,
    phno varchar(12) not null,
    bloodGroup varchar(10),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

create table studentcredentials(
     stuId varchar(20) primary key ,
      passkey varchar(100)
      );

CREATE TABLE orders (
    oid SERIAL PRIMARY KEY,
    stuId VARCHAR(20) not null,
    order_items varchar(4) not null,
    order_reason varchar(300),
    verification_img varchar(300) not null,
    stuimg varchar(300) not null,
    order_amount varchar(4),
    payment_mode VARCHAR(50),
    order_status VARCHAR(2),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE price_details (
    tag_price NUMERIC(3,0) NOT NULL,
    holder_price NUMERIC(3,0) NOT NULL,
    card_price NUMERIC(3,0) NOT NULL
);
```

dbconfig.js

```
import pg from 'pg'
const {Pool} = pg;
import dotenv from 'dotenv'
dotenv.config();

const pool = new Pool({
    // user:process.env.DB_USER,
    // host:process.env.DB_HOST,
    // database:process.env.DB_DATABASE,
    // password:process.env.DB_PASSWORD,
    // port:process.env.DB_PORT,
    connectionString: process.env.DBConfigLink,
    ssl: {
        rejectUnauthorized: false
    }
})

export default pool;
```

AddStudent.js

```
 import pool from "../../config/dbconfig.js";
import { uploadToCloudinary } from "../../utils/cloudinary.js";
import bcrypt from 'bcrypt'

const addstudent = async (req, res) => {
  const {
    stuId,
    firstName,
    lastName,
    course,
    branch,
    batch,
    dob,
    fatherName,
    address,
    phno,
    bloodGroup,
  } = req.body;

  if (
    [
      stuId,
      firstName,
      lastName,
      course,
```

```
      branch,
      batch,
      dob,
      fatherName,
      address,
      phno,
    ].some((field) => !field || field.trim() === "")
  )
    return res.status(400).json({ message: "All details are mandatory" });

  try {
    const client = await pool.connect();

    // Check if student with given stuId already exists
    const userExists = await client.query(
      "SELECT * FROM students WHERE stuId = $1",
      [stuId]
    );

    if (userExists.rows.length > 0) {
      client.release();
      return res.status(409).json({ message: "Student already exists" });
    }

    // Upload student image to Cloudinary
    const localStuImg = req.files?.stuImg[0]?.path;
    if (!localStuImg) {
      client.release();
      return res.status(400).json({ message: "Image not provided" });
    }

    const uploadedStuImg = await uploadToCloudinary(localStuImg);
    if (!uploadedStuImg.url) {
      client.release();
      return res.status(400).json({ message: "Failed to upload image" });
    }

    // Insert student data into the database
    const uploadedQueryResult = await client.query(
      "INSERT INTO students (stuId, firstName, lastName, course, branch, batch,
stuImg, dob, fatherName, address, phno, bloodGroup) VALUES ($1, $2, $3, $4, $5, $6,
$7, $8, $9, $10, $11, $12)",
      [
        stuId,
        firstName,
        lastName,
        course,
        branch,
```

```
        batch,
        uploadedStuImg.url,
        dob,
        fatherName,
        address,
        phno,
        bloodGroup,
      ]
    );
    if (uploadedQueryResult.rowCount !== 1) {
      client.release();
      return res.status(500).json({ message: "Failed to add student" });
    }
    // Insert student credentials into the database
    const password = stuId + "@aceec";
    const hashedPassword = await bcrypt.hash(password,10);
    console.log(hashedPassword);
    const credentialsQueryResult = await client.query(
      "INSERT INTO studentcredentials (stuId, passkey) VALUES ($1, $2)",
      [stuId, hashedPassword]
    );
    if (credentialsQueryResult.rowCount !== 1) {
      // If the insertion failed
      client.release();
      return res
        .status(500)
        .json({ message: "Failed to add student credentials" });
    }
    client.release();
    return res.status(201).json({ message: `${stuId} added successfully` });
  } catch (error) {
    console.error("Error:", error);
    return res.status(500).json({ message: "Internal Server Error" });
  }
};

export default addstudent;
```

deletestudent.js

```
import pool from "../../config/dbconfig.js";

const deletestudent = async (req, res) => {
  const { id } = req.params;
  console.log(id);
  if (!id) res.status(400).json({ message: "require id" });
  try {
    const client = await pool.connect();
```

```
    const result = await client.query("delete from students where stuid = $1", [
      id,
    ]);
    console.log(result);
    const credentials = await client.query(
      "delete from studentcredentials where stuid = $1",
      [id]
    );
    const orders = await client.query("delete from orders where stuid = $1", [
      id,
    ]);
    client.release();
    res.status(200).json({ message: `${id} deleted successfully` });
  } catch (err) {
    res.status(500);
  }
};

export default deletestudent;
```

getorders.js

```
import pool from "../../config/dbconfig.js";

const getorders = async (req, res) => {
  try {
    const client = await pool.connect();
    const result = await client.query(
      "SELECT * FROM orders WHERE order_status IN ('1', '2', '3', '4')"
    );
    client.release();
    res.status(200).json({ data: result.rows });
  } catch (err) {
    console.log(err.code);
    res.status(500);
  }
};

export default getorders;
```

getsudents.js

```
import pool from "../../config/dbconfig.js";

const getstudent = async (req, res) => {
    const { id } = req.params;
    console.log(id);
    if(!id) res.status(400).json({"message":"id is mandatory"})
  try {
```

```
    const client = await pool.connect();
    const result = await client.query(
        "select * from students where id = $1",
        [id]
    );
    client.release();
    res.status(200).json({"data":result.rows[0]})
  } catch (err) {
    console.log(err.code);
    res.status(500)
  }
};


export default getstudent;
```

ADMIN CODE

Index.js

```
import React from "react";
import ReactDOM from "react-dom/client";
import "./index.css";
import App from "./App";
import {
  Route,
  RouterProvider,
  createBrowserRouter,
  createRoutesFromElements,
} from "react-router-dom";
import Login from "./components/login/Login";
import Dashboard from "./components/dashboard/Dashboard";
import { ToastContainer } from "react-toastify";
import "react-toastify/dist/ReactToastify.css";

const router = createBrowserRouter(
  createRoutesFromElements(
    <Route path="/admin" element={<App />}>
      <Route path="login" element={<Login />} />
      <Route path="dashboard" element={<Dashboard />} />
    </Route>
  )
);

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(
  <React.StrictMode>
    <RouterProvider router={router} />
    <ToastContainer
```

```
      position="top-right"
      autoClose={3000}
      hideProgressBar={false}
      newestOnTop={false}
      closeOnClick
      rtl={false}
      pauseOnFocusLoss
      draggable
      pauseOnHover
      theme="light"
      // transition:Bounce
    />
  </React.StrictMode>
);
```

addStudent.js

```
import React, { useState } from "react";
import "./style.css";
import baseUrl from "../../constants";
import { toast } from "react-toastify";
import Loader from "../../common/Loader";

function AddStudent() {
  const [loader, setLoader] = useState(false);
  const [studentData, setStudentData] = useState({
    stuId: "",
    firstName: "",
    lastName: "",
    course: "",
    branch: "",
    batch: "",
    phno: "",
    address: "",
    bloodGroup: "",
    dob: "",
    fatherName: "",
    stuImg: null,
  });

  const handleInputChange = (e) => {
    const { name, value } = e.target;
    setStudentData({
      ...studentData,
      [name]: value,
    });
  };
```

```
const handleImageChange = (e) => {
  const file = e.target.files[0];
  console.log(file, "studentimage");
  setStudentData({
    ...studentData,
    stuImg: file,
  });
};

const handleSubmit = async (e) => {
  e.preventDefault();
  console.log(studentData);
  if (studentData.stuImg == "" || studentData.stuImg == null) {
    toast.error("Please upload student image");
    return;
  }
  const idRegex = /^[a-zA-Z0-9]+$/;
  if (!idRegex.test(studentData.stuId)) {
    console.log("entered");
    toast.error("Enter valid studentId");
    return;
  }
  const nameRegex = /\d/;
  if (nameRegex.test(studentData.firstName)) {
    toast.error("Enter valid first name");
    return;
  }
  if (nameRegex.test(studentData.lastName)) {
    toast.error("Enter valid last name");
    return;
  }
  if (nameRegex.test(studentData.fatherName)) {
    toast.error("Enter valid father name");
    return;
  }
  const phoneNumberRegex = /^[6-9]\d{9}$/;
  if (!phoneNumberRegex.test(studentData.phno)) {
    toast.error("Enter valid phone number");
    return;
  }
  setLoader(true);
  try {
    const formData = new FormData();
    for (const key in studentData) {
      if (studentData.hasOwnProperty(key)) {
        formData.append(key, studentData[key]);
      }
    }
```

```
    const url = `${baseUrl}/admin/addstudent`;
    const response = await fetch(url, {
      method: "POST",
      body: formData,
    });
    const data = await response.json();
    if (response.ok) {
      setStudentData({
        ...studentData,
        stuId: "",
        firstName: "",
        lastName: "",
        course: "",
        branch: "",
        batch: "",
        phno: "",
        address: "",
        bloodGroup: "",
        dob: "",
        fatherName: "",
        stuImg: null, // Reset the image after successful submission
      });
      setLoader(false);
      toast.success("Student Added Successfully");
      console.log("Student Data Submitted:", data);
    }
  } catch (error) {
    console.log("Error:", error);
    setLoader(false);
    toast.error(error);
  }
};

return (
  <div className="add-student-container">
    <h2>Add Student</h2>
    <form onSubmit={handleSubmit} className="student-form">
      <div className="form-row">
        <div className="form-group">
          <label htmlFor="stuId">Student ID:</label>
          <input
            type="text"
            id="stuId"
            name="stuId"
            value={studentData.stuId}
            onChange={handleInputChange}
            required
          />
```

```
        </div>
        <div className="form-group">
          <label htmlFor="firstName">First Name:</label>
          <input
            type="text"
            id="firstName"
            name="firstName"
            value={studentData.firstName}
            onChange={handleInputChange}
            required
          />
        </div>
        <div className="form-group">
          <label htmlFor="lastName">Last Name:</label>
          <input
            type="text"
            id="lastName"
            name="lastName"
            value={studentData.lastName}
            onChange={handleInputChange}
            required
          />
        </div>
      </div>
      <div className="form-row">
        <div className="form-group ">
          <label htmlFor="course">Course :</label>
          <select
            id="course"
            name="course"
            value={studentData.course}
            onChange={handleInputChange}
            required
            className="blood-group"
          >
            <option value="">Select Student Course</option>
            <option value="B.TECH">B.TECH</option>
          </select>
        </div>
        <div className="form-group ">
          <label htmlFor="branch">Branch :</label>
          <select
            id="branch"
            name="branch"
            value={studentData.branch}
            onChange={handleInputChange}
            required
            className="blood-group"
```

```
          >
            <option value="">Select Student Branch</option>
            <option value="CSE">CSE</option>
            <option value="MECH">MECH</option>
            <option value="EEE">EEE</option>
            <option value="ECE">ECE</option>
            <option value="CIVIL">CIVIL</option>
            <option value="CSE-DS">CSE-DS</option>
            <option value="CSE-AIML">CSE-AIML</option>
            <option value="CSE-CS">CSE-CS</option>
            <option value="CSE-IT">CSE-IT</option>
            <option value="CSE-IOT">CSE-IOT</option>
          </select>
        </div>
        <div className="form-group ">
          <label htmlFor="batch">Batch :</label>
          <select
            id="batch"
            name="batch"
            value={studentData.batch}
            onChange={handleInputChange}
            required
            className="blood-group"
          >
            <option value="">Select Student Batch</option>
            <option value="2020-2021">2020-2021</option>
            <option value="2021-2022">2021-2022</option>
            <option value="2022-2023">2022-2023</option>
            <option value="2023-2024">2023-2024</option>
          </select>
        </div>
        <div className="form-group ">
          <label htmlFor="bloodGroup">Blood Group:</label>
          <select
            id="bloodGroup"
            name="bloodGroup"
            value={studentData.bloodGroup}
            onChange={handleInputChange}
            required
            className="blood-group"
          >
            <option value="">Select Blood Group</option>
            <option value="A+">A+</option>
            <option value="A-">A-</option>
            <option value="B+">B+</option>
            <option value="B-">B-</option>
            <option value="AB+">AB+</option>
            <option value="AB-">AB-</option>
```

```
            <option value="O+">O+</option>
            <option value="O-">O-</option>
          </select>
        </div>
      </div>
      <div className="form-group address">
        <label htmlFor="address">Address:</label>
        <input
          type="text"
          id="address"
          name="address"
          value={studentData.address}
          onChange={handleInputChange}
          required
        />
      </div>
      <div className="form-row last-row">
        <div className="form-group">
          <label htmlFor="fatherName">Father Name:</label>
          <input
            type="text"
            id="fatherName"
            name="fatherName"
            value={studentData.fatherName}
            onChange={handleInputChange}
            required
          />
        </div>
        <div className="form-group">
          <label htmlFor="dob">Date of Birth:</label>
          <input
            type="date"
            id="dob"
            name="dob"
            value={studentData.dob}
            onChange={handleInputChange}
            required
          />
        </div>
        <div className="form-group">
          <label htmlFor="phno">Phone number:</label>
          <input
            type="text"
            id="phno"
            name="phno"
            value={studentData.phno}
            onChange={handleInputChange}
            required
```

```
            />
          </div>
        </div>
        <div className="form-group">
          <label htmlFor="stuImg">Student Image:</label>
          <div>
            <input
              type="file"
              id="stuImg"
              name="stuImg"
              onChange={handleImageChange}
              accept="image/jpeg"
              style={{ display: "none" }}
            />
            <label htmlFor="stuImg" className="image-upload-label">
              <img
                src={
                  studentData.stuImg
                    ? URL.createObjectURL(studentData.stuImg)
                    : "https://www.pngfind.com/pngs/m/610-6104451_image-placeholder-
png-user-profile-placeholder-image-png.png"
                }
                alt="Student Image"
                className="uploaded-image"
              />
            </label>
          </div>
        </div>
        <button type="submit" className="submit-button">
          Add Student
        </button>
      </form>
      {loader && <Loader />}
    </div>
  );
}

export default AddStudent;
```

dashboard.js

```
import React from "react";
import "./style.css";
import { useDispatch, useSelector } from "react-redux";
import { changeCompo } from "../../redux/slices/SetCompo";
import Students from "../students/Students";
import AddStudent from "../addstudent/AddStudent";
import Orders from "../orders/Orders";
import PastOrders from "../pastorders/PastOrders";
```

```
function Dashboard() {
  const dispatch = useDispatch();
  const component = useSelector((state) => state.component);
  const handleButtonClick = (button) => {
    dispatch(changeCompo(button));
  };

  return (
    <div className="dashboard-container">
      <div className="left-column">
        <button
          className={component === "students" ? "active" : ""}
          onClick={() => handleButtonClick("students")}
        >
          Students
        </button>
        <button
          className={component === "add" ? "active" : ""}
          onClick={() => handleButtonClick("add")}
        >
          Add Student
        </button>
        <button
          className={component === "orders" ? "active" : ""}
          onClick={() => handleButtonClick("orders")}
        >
          Orders
        </button>
        <button
          className={component === "past" ? "active" : ""}
          onClick={() => handleButtonClick("past")}
        >
          Past Orders
        </button>
        <button
          className={component === "logout" ? "active" : ""}
          onClick={() => handleButtonClick("logout")}
        >
          Logout
        </button>
      </div>
      <div className="right-column">
        {component === "students" && <Students />}
        {component === "add" && <AddStudent />}
        {component === "orders" && <Orders />}
        {component === "past" && <PastOrders />}
        {component === "logout" && <div>Logout Content</div>}
```

```
      </div>
    </div>
  );
}

export default Dashboard;
```

editstudent.js

```
import React, { useState } from "react";
import "./style.css";
import { toast } from "react-toastify";
import baseUrl from "../../constants";
import Loader from "../../common/Loader";

const EditStudent = ({ studentData, setStudentData, setShowUpdate }) => {
  console.log(studentData);
  const [loader, setLoader] = useState(false);
  const [imageChanged, setImageChanged] = useState(false);
  const handleInputChange = (e) => {
    const { name, value } = e.target;
    setStudentData({
      ...studentData,
      [name]: value,
    });
    console.log(value);
  };

  const handleImageChange = (e) => {
    const file = e.target.files[0];
    console.log(file, "studentimage");
    setStudentData({
      ...studentData,
      stuimg: file,
    });
    setImageChanged(true);
  };

  const handleUpdate = async (e) => {
    e.preventDefault();
    console.log(studentData);
    if (studentData.stuimg == "" || studentData.stuimg == null) {
      toast.error("Please upload student image");
      return;
    }
    const idRegex = /^[a-zA-Z0-9]+$/;
    if (!idRegex.test(studentData.stuId)) {
      console.log("entered");
```

```
      toast.error("Enter valid studentId");
      return;
    }
    const nameRegex = /\d/;
    if (nameRegex.test(studentData.firstname)) {
      toast.error("Enter valid first name");
      return;
    }
    if (nameRegex.test(studentData.lastname)) {
      toast.error("Enter valid last name");
      return;
    }
    if (nameRegex.test(studentData.fathername)) {
      toast.error("Enter valid father name");
      return;
    }
    const phoneNumberRegex = /^[6-9]\d{9}$/;
    if (!phoneNumberRegex.test(studentData.phno)) {
      toast.error("Enter valid phone number");
      return;
    }
    setLoader(true);
    try {
      const formData = new FormData();
      for (const key in studentData) {
        if (studentData.hasOwnProperty(key)) {
          formData.append(key, studentData[key]);
        }
      }
      console.log(formData);
      const url =
`${baseUrl}/admin/updatestudent/${studentData.stuid}/${imageChanged}`;
      console.log(url, imageChanged, "url");
      const response = await fetch(url, {
        method: "PUT",
        body: formData,
      });
      const data = await response.json();
      if (response.ok) {
        setLoader(false);
        setShowUpdate(false);
        toast.success("Student updated Successfully");
        console.log("Student Data Submitted:", data);
      } else {
        setLoader(false);
        toast.error("Error in updating student data");
      }
    } catch (error) {
```

```
    console.log("Error:", error);
    setLoader(false);
    toast.error(error);
  }
};

const handleDelete = async () => {
  setLoader(true);
  try {
    const url = `${baseUrl}/admin/deletestudent/${studentData.stuid}`;
    console.log(url, "url");
    const response = await fetch(url, {
      method: "DELETE",
      headers: {
        "content-type": "application/json",
      },
    });
    const data = await response.json();
    console.log(data);
    if (response.ok) {
      setLoader(false);
      setShowUpdate(false);
      toast.success("Student Deleted Successfully");
      console.log("Student Data Deleted:", data);
    } else {
      console.log("Error:", data.error);
      setLoader(false);
      toast.error(data.error);
    }
  } catch (error) {
    console.log("Error:", error.message);
    setLoader(false);
    toast.error("Error deleting student");
  }
};

const handleCancel = () => {
  setShowUpdate(false);
};
return (
  <div>
    <h2>Update student {studentData.stuid}</h2>
    <form onSubmit={handleUpdate} className="student-form">
      <div className="form-row">
        <div className="form-group">
          <label htmlFor="firstname">First Name:</label>
          <input
            type="text"
```

```
            id="firstname"
            name="firstname"
            value={studentData.firstname}
            onChange={handleInputChange}
            required
          />
        </div>
        <div className="form-group">
          <label htmlFor="lastname">Last Name:</label>
          <input
            type="text"
            id="lastname"
            name="lastname"
            value={studentData.lastname}
            onChange={handleInputChange}
            required
          />
        </div>
      </div>
      <div className="form-row">
        <div className="form-group ">
          <label htmlFor="course">Course :</label>
          <select
            id="course"
            name="course"
            value={studentData.course}
            onChange={handleInputChange}
            required
            className="blood-group"
          >
            <option value={setStudentData.course}>
              {studentData.course}
            </option>
            <option value="B.TECH">B.TECH</option>
          </select>
        </div>
        <div className="form-group ">
          <label htmlFor="branch">Branch :</label>
          <select
            id="branch"
            name="branch"
            value={studentData.branch}
            onChange={handleInputChange}
            required
            className="blood-group"
          >
            <option value={studentData.branch}>{studentData.branch}</option>
            <option value="CSE">CSE</option>
```

```
            <option value="MECH">MECH</option>
            <option value="EEE">EEE</option>
            <option value="ECE">ECE</option>
            <option value="CIVIL">CIVIL</option>
            <option value="CSE-DS">CSE-DS</option>
            <option value="CSE-AIML">CSE-AIML</option>
            <option value="CSE-CS">CSE-CS</option>
            <option value="CSE-IT">CSE-IT</option>
            <option value="CSE-IOT">CSE-IOT</option>
          </select>
        </div>
        <div className="form-group ">
          <label htmlFor="batch">Batch :</label>
          <select
            id="batch"
            name="batch"
            value={studentData.batch}
            onChange={handleInputChange}
            required
            className="blood-group"
          >
            <option value={studentData.batch}>{studentData.batch}</option>
            <option value="2020-2021">2020-2021</option>
            <option value="2021-2022">2021-2022</option>
            <option value="2022-2023">2022-2023</option>
            <option value="2023-2024">2023-2024</option>
          </select>
        </div>
        <div className="form-group ">
          <label htmlFor="bloodgroup">Blood Group:</label>
          <select
            id="bloodgroup"
            name="bloodgroup"
            value={studentData.bloodgroup}
            onChange={handleInputChange}
            required
            className="blood-group"
          >
            <option value="">Select Blood Group</option>
            <option value="A+">A+</option>
            <option value="A-">A-</option>
            <option value="B+">B+</option>
            <option value="B-">B-</option>
            <option value="AB+">AB+</option>
            <option value="AB-">AB-</option>
            <option value="O+">O+</option>
            <option value="O-">O-</option>
          </select>
```

```
        </div>
      </div>
      <div className="form-group address">
        <label htmlFor="address">Address:</label>
        <input
          type="text"
          id="address"
          name="address"
          value={studentData.address}
          onChange={handleInputChange}
          required
        />
      </div>
      <div className="form-row last-row">
        <div className="form-group">
          <label htmlFor="fathername">Father Name:</label>
          <input
            type="text"
            id="fathername"
            name="fathername"
            value={studentData.fathername}
            onChange={handleInputChange}
            required
          />
        </div>
        <div className="form-group">
          <label htmlFor="dob">Date of Birth:</label>
          <input
            type="date"
            id="dob"
            name="dob"
            value={studentData.dob}
            onChange={handleInputChange}
            required
          />
        </div>
        <div className="form-group">
          <label htmlFor="phno">Phone number:</label>
          <input
            type="text"
            id="phno"
            name="phno"
            value={studentData.phno}
            onChange={handleInputChange}
            required
          />
        </div>
      </div>
```

```
        <div className="form-group">
          <label htmlFor="stuimg">Student Image:</label>
          <div>
            <input
              type="file"
              id="stuimg"
              name="stuimg"
              onChange={handleImageChange}
              accept="image/jpeg"
              style={{ display: "none" }}
            />
            <label htmlFor="stuimg" className="image-upload-label">
              <img
                src={
                  studentData.stuimg
                    ? studentData.stuimg
                    : "https://www.pngfind.com/pngs/m/610-6104451_image-placeholder-
png-user-profile-placeholder-image-png.png"
                }
                alt="Student Image"
                className="uploaded-image"
              />
            </label>
          </div>
        </div>
        <button type="submit" className="update-button">
          Update
        </button>
      </form>

      <button type="button" className="delete-button" onClick={handleDelete}>
        Delete
      </button>
      <button type="button" className="delete-button" onClick={handleCancel}>
        Cancel
      </button>
      {loader && <Loader />}
    </div>
  );
};

export default EditStudent;
```

Login.js

```
import React, { useState } from "react";
import "./style.css";
import { useNavigate } from "react-router-dom";
```

```
import baseUrl from "../../constants";

function Login() {
  const [username, setUsername] = useState("");
  const [password, setPassword] = useState("");
  const navigate = useNavigate();

  const Login = async () => {
    try {
      const response = await fetch(`${baseUrl}/login`, {
        method: "POST",
        headers: {
          "Content-Type": "application/json",
        },
        body: JSON.stringify({ username, password }),
      });
      console.warn(username, password);
      console.log(username, password);
      console.log(response);
      console.warn(response);
      const data = await response.json();
      console.log(data);
      console.warn(data);
      if (response.ok) {
        console.log("Login successful");
        console.log(data);
        console.log(data.token);
        navigate("/admin/dashboard");
      } else {
        console.log("Login failed");
      }
    } catch (error) {
      console.error("Error during login:", error);
    }
  };

  return (
    <div className="login-container">
      <h2 style={{ textAlign: "center" }}>Login</h2>
      <div className="login-form">
        <label htmlFor="username">Username</label>
        <input
          type="text"
          id="username"
          name="username"
          value={username}
          onChange={(e) => setUsername(e.target.value)}
        />
```

```
        <label htmlFor="password">Password</label>
        <input
          type="password"
          id="password"
          name="password"
          value={password}
          onChange={(e) => setPassword(e.target.value)}
        />

        <button onClick={Login} className="login-button">
          Login
        </button>
      </div>
    </div>
  );
}

export default Login;
```

orders.js

```
import React, { useEffect, useState } from "react";
import baseUrl from "../../constants";
import "./style.css";
import { toast } from "react-toastify";
import Loader from "../../common/Loader";

function Orders() {
  const [orders, setOrders] = useState([]);
  const [loader, setLoader] = useState(true);

  useEffect(() => {
    getOrders();
  }, []);

  const getOrders = async () => {
    try {
      const url = `${baseUrl}/admin/getorders`;
      const response = await fetch(url, {
        method: "GET",
        headers: {
          "content-type": "application/json",
        },
      });
      const data = await response.json();
      console.log(data.data);
      setOrders(data.data);
      setLoader(false);
```

```
  } catch (error) {
    console.error("Error fetching orders:", error);
    setLoader(false);
    toast.error(error);
  }
};

const updateOrderStatus = async (orderId, newStatus) => {
  setLoader(true);
  try {
    const url = `${baseUrl}/admin/updateorder/${orderId}`;
    const response = await fetch(url, {
      method: "PUT",
      headers: {
        "content-type": "application/json",
      },
      body: JSON.stringify({ order_status: newStatus }),
    });
    setLoader(false);
    if (response.ok) {
      // If the update is successful, refresh the orders list
      getOrders();
      toast.success(`Order status updated successfully`);
    } else {
      toast.error(`Failed to update order ${orderId} status`);
    }
  } catch (error) {
    console.error("Error updating order status:", error);
    setLoader(false);
    toast.error(error);
  }
};

return (
  <div className="orders-container">
    <table className="orders-table">
      <thead>
        <tr>
          <th>Order ID</th>
          <th>Student ID</th>
          <th>Order Amount</th>
          <th>Payment Mode</th>
          <th>Items</th>
          <th>Reason</th>
          <th>Verification Img</th>
          <th>Student Img</th>
          <th>Status</th>
          <th>Placed At</th>
```

```
            <th>Actions</th>
          </tr>
        </thead>
        <tbody>
          {orders.map((order) => (
            <tr key={order.oid}>
              <td>#0000{order.oid}</td>
              <td>{order.stuid}</td>
              <td>{order.order_amount}</td>
              <td>{order.payment_mode}</td>
              <td>{order.order_items}</td>
              <td>{order.order_reason}</td>
              <td>
                <img
                  src={order.verification_img}
                  alt="Verification Image"
                  style={{ width: 100, height: 100 }}
                />
              </td>
              <td>
                <img
                  src={order.stuimg}
                  alt="Student Image"
                  style={{ width: 100, height: 100 }}
                />
              </td>
              <td>
                {order.order_status == 1
                  ? "Order Placed"
                  : order.order_status == 2
                  ? "Approved"
                  : order.order_status == 3
                  ? "Sent to Print"
                  : order.order_status == 4
                  ? "Completed"
                  : "Unknown"}
              </td>
              <td>{new Date(order.created_at).toLocaleDateString()}</td>

              <td>
                {order.order_status == 1 ? (
                  <div className="button-box">
                    <button
                      className="approve-btn"
                      onClick={() => updateOrderStatus(order.oid, "2")}
                    >
                      Approve
                    </button>
```

```
                  <button
                    className="reject-btn"
                    onClick={() => updateOrderStatus(order.oid, "7")}
                  >
                    Reject
                  </button>
                </div>
              ) : order.order_status == 2 ? (
                <button
                  className="approve-btn"
                  onClick={() => updateOrderStatus(order.oid, "3")}
                >
                  Send to Print
                </button>
              ) : order.order_status == 3 ? (
                <button
                  className="approve-btn"
                  onClick={() => updateOrderStatus(order.oid, "4")}
                >
                  Complete
                </button>
              ) : order.order_status == 4 ? (
                <button
                  className="approve-btn"
                  onClick={() => updateOrderStatus(order.oid, "5")}
                >
                  Deliver
                </button>
              ) : null}
            </td>
          </tr>
        ))}
      </tbody>
    </table>
    {loader && <Loader />}
  </div>
 );
}

export default Orders;
```

Pastorders.js

```
import React, { useEffect, useState } from "react";
import baseUrl from "../../constants";
import "./style.css";
import Loader from "../../common/Loader";
```

```
function PastOrders() {
  const [orders, setOrders] = useState([]);
  const [loader, setLoader] = useState(true);

  useEffect(() => {
    getOrders();
  }, []);

  const getOrders = async () => {
    try {
      const url = `${baseUrl}/admin/pastorders`;
      const response = await fetch(url, {
        method: "GET",
        headers: {
          "content-type": "application/json",
        },
      });
      const data = await response.json();
      console.log(data.data);
      setOrders(data.data);
      setLoader(false);
    } catch (error) {
      console.error("Error fetching orders:", error);
      setLoader(false);
      toast.error(error);
    }
  };

  return (
    <div className="orders-container">
      <table className="orders-table">
        <thead>
          <tr>
            <th>Order ID</th>
            <th>Student ID</th>
            <th>Order Amount</th>
            <th>Payment Mode</th>
            <th>Items</th>
            <th>Reason</th>
            <th>Verification Img</th>
            <th>Status</th>
            <th>Placed At</th>
            <th>Last Updated</th>
          </tr>
        </thead>
        <tbody>
          {orders.map((order) => (
            <tr key={order.oid}>
```

```
              <td>#0000{order.oid}</td>
              <td>{order.stuid}</td>
              <td>{order.order_amount}</td>
              <td>{order.payment_mode}</td>
              <td>{order.order_items}</td>
              <td>{order.order_reason}</td>
              <td>
                <img
                  src={order.verification_img}
                  alt="Verification Image"
                  style={{ width: 100, height: 100 }}
                />
              </td>
              <td>
                {order.order_status == 5
                  ? "Delivered"
                  : order.order_status == 6
                  ? "Student Cancelled"
                  : order.order_status == 7
                  ? "Order Rejected"
                  : "Unknown"}
              </td>
              <td>{new Date(order.created_at).toLocaleDateString()}</td>
              <td>{new Date(order.updated_at).toLocaleDateString()}</td>
            </tr>
          ))}
        </tbody>
      </table>
      {loader && <Loader />}
    </div>
  );
}

export default PastOrders;
```

students.js

```
import React, { useEffect, useState } from "react";
import baseUrl from "../../constants";
import "./style.css";
import { toast } from "react-toastify";
import Loader from "../../common/Loader";
import EditStudent from "../EditStudent/EditStudent";

function Students() {
  const [students, setStudents] = useState([]);
  const [editRowId, setEditRowId] = useState(null);
  const [loader, setLoader] = useState(true);
  const [showUpdate, setShowUpdate] = useState(false);
```

```
const [updatedValues, setUpdatedValues] = useState({
  firstname: "",
  lastname: "",
  course: "",
  branch: "",
  batch: "",
  phno: "",
  address: "",
  fathername: "",
});
const [updateData, setUpdateData] = useState();

useEffect(() => {
  if (!showUpdate) getStudents();
}, [showUpdate]);

const getStudents = async () => {
  setLoader(true);
  try {
    const url = `${baseUrl}/admin/students`;
    const response = await fetch(url, {
      method: "GET",
      headers: {
        "content-type": "application/json",
      },
    });
    const data = await response.json();
    setStudents(data.data);
    setLoader(false);
  } catch (error) {
    console.log(error, "not fetched student details");
    setLoader(false);
    toast.error(error);
  }
};

const handleEditClick = (student) => {
  setUpdateData(student);
  setShowUpdate(true);
  // setUpdate(true);
  // setEditRowId(student.id);
  // setUpdatedValues({
  //   firstname: student.firstname,
  //   lastname: student.lastname,
  //   course: student.course,
  //   branch: student.branch,
  //   batch: student.batch,
  //   phno: student.phno,
```

```
  //   address: student.address,
  //   fathername: student.fathername,
  // });
};

const updateDetails = async (stuid) => {
  console.log(stuid, "stuid");
  // try {
  //   const requestBody = {
  //     firstname: updatedValues.firstname,
  //     lastname: updatedValues.lastname,
  //     course: updatedValues.course,
  //     branch: updatedValues.branch,
  //     batch: updatedValues.batch,
  //     phno: updatedValues.phno,
  //     address: updatedValues.address,
  //     fathername: updatedValues.fathername,
  //   };
  //   const url = `${baseUrl}/admin/updatestudent/${editRowId}`;
  //   const response = await fetch(url, {
  //     method: "PUT",
  //     headers: {
  //       "content-type": "application/json",
  //     },
  //     body: JSON.stringify(requestBody),
  //   });
  //   if (response.status === 200) {
  //     setEditRowId(null);
  //     getStudents();
  //     console.log("Student details updated successfully!");
  //   } else {
  //     console.log(
  //       "Failed to update student details. Status code:",
  //       response.status
  //     );
  //   }
  // } catch (error) {
  //   console.error("Error updating student details:", error);
  // }
};
const handleInputChange = (e, fieldName) => {
  const { value } = e.target;
  setUpdatedValues({
    ...updatedValues,
    [fieldName]: value,
  });
};
return (
```

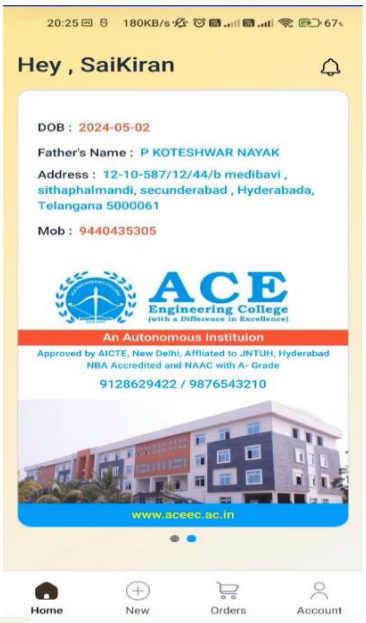
```
<>
  {!showUpdate ? (
    <div className="students-container">
      <table className="students-table">
        <thead>
          <tr>
            <th>ID</th>
            <th>Firstname</th>
            <th>Lastname</th>
            <th>Course</th>
            <th>Branch</th>
            <th>Batch</th>
            <th>Phone number</th>
            <th>Address</th>
            <th>Fathername</th>
            <th>Image</th>
            <th>Actions</th>
          </tr>
        </thead>
        <tbody>
          {students.map((student) => (
            <tr key={student.stuid}>
              <td>{student.stuid}</td>
              <td>
                {editRowId === student.stuid ? (
                  <input
                    type="text"
                    value={updatedValues.firstname}
                    onChange={(e) => handleInputChange(e, "firstname")}
                  />
                ) : (
                  student.firstname
                )}
              </td>
              <td>
                {editRowId === student.stuid ? (
                  <input
                    type="text"
                    value={updatedValues.lastname}
                    onChange={(e) => handleInputChange(e, "lastname")}
                  />
                ) : (
                  student.lastname
                )}
              </td>
              <td>
                {editRowId === student.stuid ? (
                  <input
```

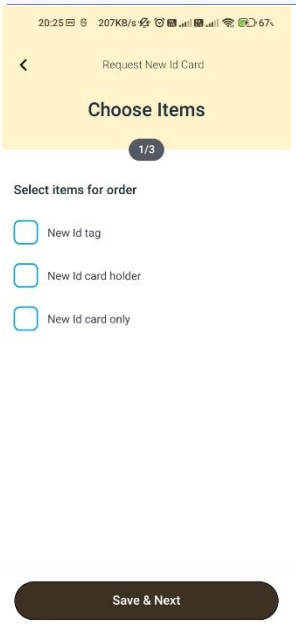
```
                type="text"
                value={updatedValues.course}
                onChange={(e) => handleInputChange(e, "course")}
              />
            ) : (
              student.course
            )}
          </td>
          <td>
            {editRowId === student.stuid ? (
              <input
                type="text"
                value={updatedValues.branch}
                onChange={(e) => handleInputChange(e, "branch")}
              />
            ) : (
              student.branch
            )}
          </td>
          <td>
            {editRowId === student.stuid ? (
              <input
                type="text"
                value={updatedValues.batch}
                onChange={(e) => handleInputChange(e, "batch")}
              />
            ) : (
              student.batch
            )}
          </td>
          <td>
            {editRowId === student.stuid ? (
              <input
                type="text"
                value={updatedValues.phno}
                onChange={(e) => handleInputChange(e, "phno")}
              />
            ) : (
              student.phno
            )}
          </td>
          <td>
            {editRowId === student.stuid ? (
              <input
                type="text"
                value={updatedValues.address}
                onChange={(e) => handleInputChange(e, "address")}
              />
```

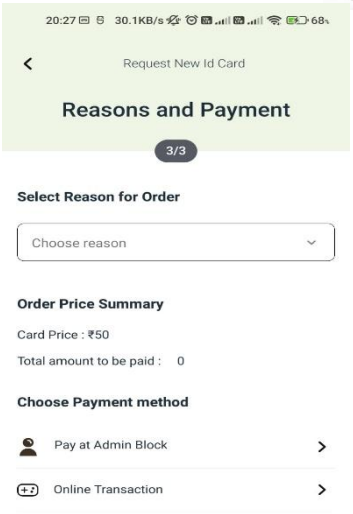
```
                  ) : (
                    student.address
                  )}
                </td>
                <td>
                  {editRowId === student.stuid ? (
                    <input
                      type="text"
                      value={updatedValues.fathername}
                      onChange={(e) => handleInputChange(e, "fathername")}
                    />
                  ) : (
                    student.fathername
                  )}
                </td>
                <td>
                  <img
                    src={student.stuimg}
                    alt="Student"
                    style={{ width: 50, height: 50 }}
                  />
                </td>
                <td>
                  <button
                    className="update-btn"
                    onClick={() => handleEditClick(student)}
                  >
                    Update
                  </button>
                </td>
              </tr>
            ))}
          </tbody>
        </table>
      </div>
    ) : (
      <EditStudent
        studentData={updateData}
        setStudentData={setUpdateData}
        setShowUpdate={setShowUpdate}
      />
    )}
    {loader && <Loader />}
  </>
 );
}

export default Students;
```

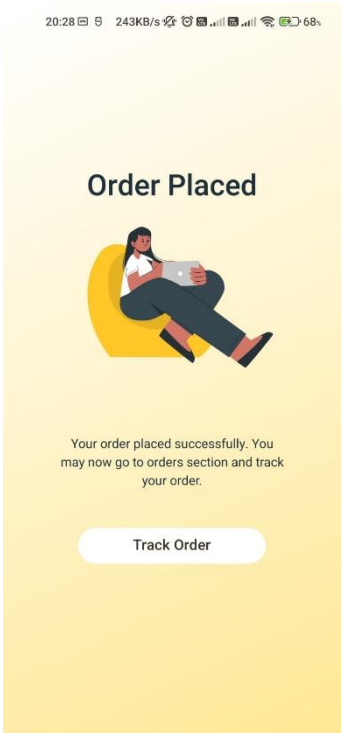
OUTPUT:

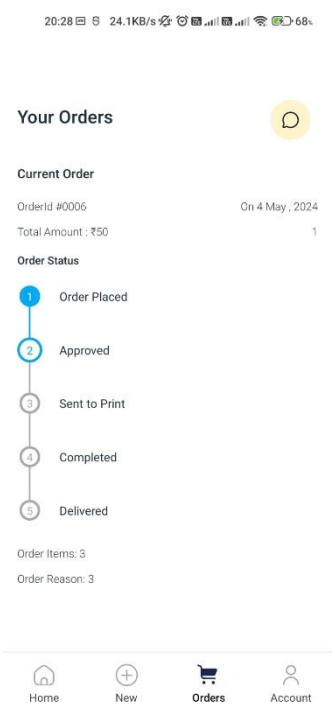

Hey , SaiKiran
ACE
Engineering College
An Autonomous Instituion
Approved by AICTE, New Delhi, Affliated to JNTUH, Hyderabad
NBA Accredited and NAAC with A- Grade
STUDENT
SaiKiran Nayak
Roll No : 21AG5A0502
Course/Branch : B.TECH/CSE
Batch : 2021-2024
Principal
Home
New
Orders
Account

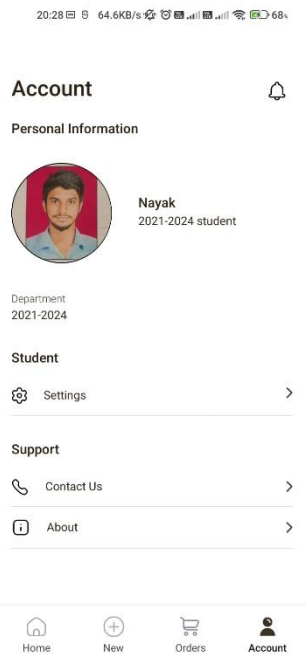

Hey , SaiKiran
DOB : 2024-05-02
Father's Name : P KOTESHWAR NAYAK
Address : 12-10-587/12/44/b medibavi , sithaphalmandi, secunderabad , Hyderabada, Telangana 5000061
Mob : 9440435305
ACE
Engineering College
An Autonomous Instituion
Approved by AICTE, New Delhi, Affliated to JNTUH, Hyderabad
NBA Accredited and NAAC with A- Grade
9128629422 / 9876543210
www.aceec.ac.in
Home
New
Orders
Account

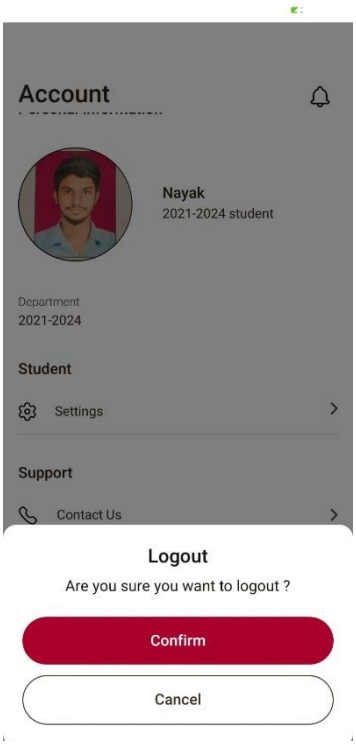

Request New Id Card
Choose Items
1/3
Select items for order
New Id tag
New Id card holder
New Id card only
Save & Next

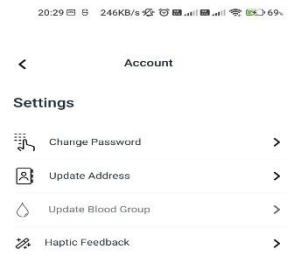

Request New Id Card
Reasons and Payment
3/3
Select Reason for Order
Choose reason
Order Price Summary
Card Price : ₹50
Total amount to be paid : 0
Choose Payment method
Pay at Admin Block
Online Transaction

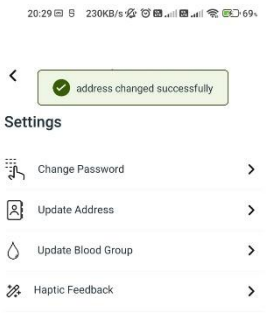

Order Placed
Your order placed successfully. You may now go to orders section and track your order.
Track Order

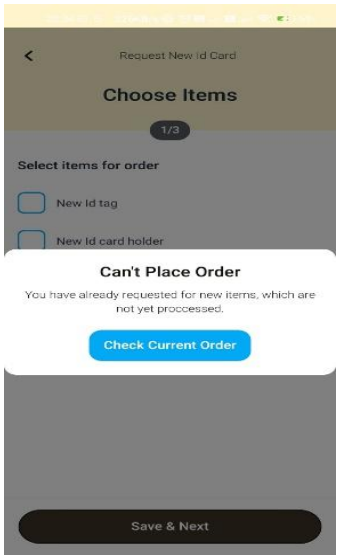

Your Orders
Current Order
OrderId #0006
On 4 May , 2024
Total Amount : ₹50
Order Status
Order Placed
Approved
Sent to Print
Completed
Delivered
Order Items: 3
Order Reason: 3
Home
New
Orders
Account


Account
Personal Information
Nayak
2021-2024 student
Department
2021-2024
Student
Settings
Support
Contact Us
About
Home
New
Orders
Account

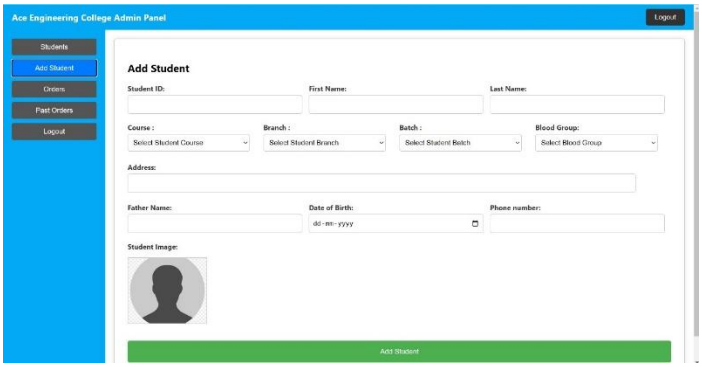

Account
Nayak
2021-2024 student
Department
2021-2024
Student
Settings
Support
Contact Us
Logout
Are you sure you want to logout ?
Confirm
Cancel

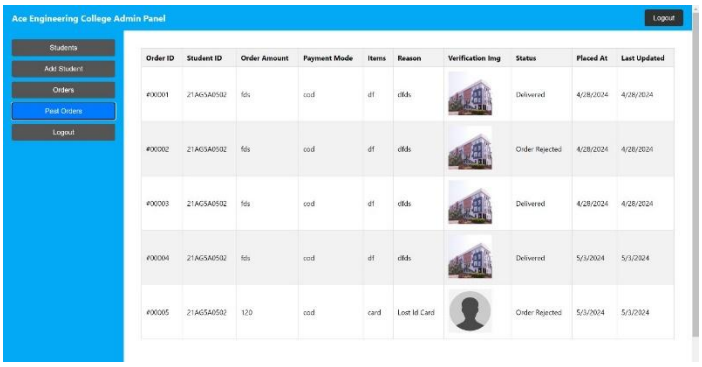

20:29 246KB/s 69%
Account
Settings
Change Password
Update Address
Update Blood Group
Haptic Feedback

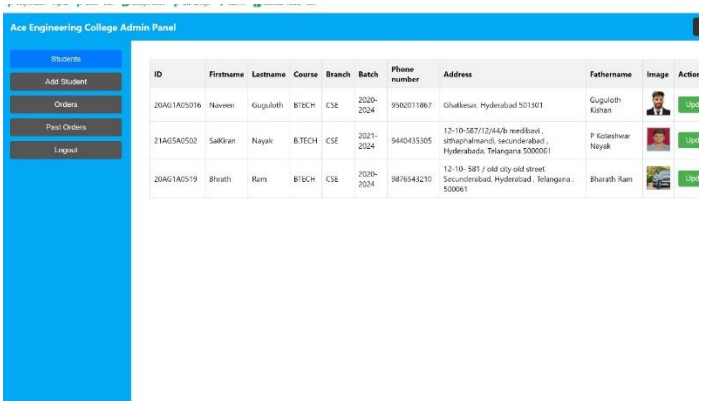

20:29 230KB/s 69%
address changed successfully
Settings
Change Password
Update Address
Update Blood Group
Haptic Feedback

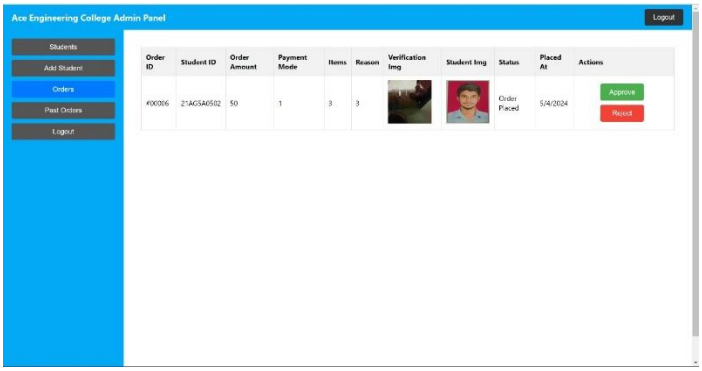

Request New Id Card
Choose Items
1/3
Select items for order
New Id tag
New Id card holder
Can't Place Order
You have already requested for new items, which are not yet proccessed.
Check Current Order
Save & Next


Ace Engineering College Admin Panel
Logout
Students
Add Student
Orders
Past Orders
Logout
ID
Firstname
Lastname
Course
Branch
Batch
Phone number
Address
Fathername
Image
Actions
20AG1A05316
Naveen
Guguloth
BTECH
CSE
2020-2024
9502011867
Ghatkesar, Hyderabad 501301
Guguloth Kishan
Update
21AG5A0502
SaiKiran
Nayak
B.TECH
CSE
2021-2024
9490435305
12-10-587/12/44/b inedibavi , sithaphalmandi, secunderabad , Hyderabadia, Telangana 5000061
P Koteshwar Nayak
Update
20AG1A0519
Bharath
Ram
BTECH
CSE
2020-2024
9876543210
12-10- 581 / old city old sheet Secunderabad, Hyderabad , Telangana , 500061
Bharath Ram
Update









11.TESTING

Functional Testing:

UI Testing:

Verify that all UI elements are displayed correctly on different screen sizes and resolutions.

Ensure consistency in UI design and layout across various screens and devices.

Request Submission Testing:

Test the submission of a new ID card request with valid and invalid input data.

Verify that all required fields are correctly validated before submission.
Payment Processing Testing:
Test the payment process with valid and invalid payment information.
Verify that payment transactions are processed securely and accurately.
Request Tracking Testing:
Test the real-time tracking of ID card requests at different stages (e.g., pending, processing, approved, delivered).
Verify that users receive timely updates on the status of their requests.
Compatibility Testing:
Test the application on different devices (e.g., smartphones, tablets) with various operating systems (iOS, Android).
Verify that the application's layout and functionality remain consistent across different screen sizes and resolutions.
Test the application on different web browsers (e.g., Chrome, Safari, Firefox) to ensure compatibility and functionality.
Integration Testing:
Test the integration with external systems, such as payment gateways and student databases.
Verify that data exchange between the application and external systems is seamless and accurate.
Test scenarios involving concurrent access to shared resources or data to ensure data integrity and consistency.
Security Testing:
Test for vulnerabilities such as SQL injection, cross-site scripting (XSS), and broken authentication.
Verify that user authentication and authorization mechanisms are robust and secure.
Conduct penetration testing to identify and address potential security loopholes or weaknesses.
Performance Testing:
Load Testing:
Test the application's performance under different levels of user traffic (low, moderate, high).
Verify that the application remains responsive and stable under peak load conditions.
Response Time Testing:
Measure the response time for key actions such as submitting a request, making a payment, and tracking a request.
Ensure that response times meet acceptable thresholds defined for the application.
Usability Testing:
Conduct usability testing sessions with representative users to gather feedback on the application's ease of use and intuitiveness.
Identify any usability issues or pain points encountered by users during the testing sessions.
Incorporate user feedback to make iterative improvements to the application's design and user experience.

11. CONCLUSION

In conclusion, the introduction of a dedicated mobile application for college students' ID card management represents a significant leap forward in enhancing the efficiency, convenience, and user experience within educational institutions. By streamlining the process of requesting and managing ID cards through a user-friendly digital platform, we are revolutionizing traditional administrative practices and empowering students with greater control over their identity and access to campus facilities.

Through rigorous testing and continuous improvement, we have ensured that our application meets the highest standards of quality, security, and usability. By embracing future opportunities for enhancement, such as integrating biometric authentication, expanding features, and exploring new partnerships, we can further elevate the value proposition of our application and address evolving needs in the educational landscape.

Ultimately, our commitment to innovation and user-centric design reflects our dedication to enhancing the college experience for students and administrators alike. As we continue to evolve and adapt to the ever-changing demands of higher education, we remain steadfast in our mission to provide cutting-edge solutions that redefine the boundaries of efficiency and convenience in campus administration.

With this mobile application, we are not only simplifying ID card management but also laying the foundation for a more connected, accessible, and inclusive campus community. Together, we embark on a journey towards a future where technology seamlessly integrates with tradition to create transformative experiences for all stakeholders in the educational ecosystem.

12. FUTURE SCOPE

The future scope for a mobile application dedicated to college students' ID card management is vast and promising. Here are some potential avenues for future development and expansion:

1. 1.*Enhanced Features:* Continuously improve the application by adding new features and functionalities based on user feedback and emerging trends. This could include features such as digital signatures for official documents, virtual campus tours, event registration, or integration with other campus services like parking permits or meal plans.
2. 2.*Biometric Authentication:* Implement biometric authentication methods such as fingerprint or facial recognition for enhanced security and convenience in accessing the application and verifying identity for ID card-related transactions.

3. *Integration with Campus Systems:* Explore opportunities to integrate the application with other campus systems and services, such as student information systems (SIS), learning management systems (LMS), or campus security systems. This integration can streamline administrative processes and provide students with a unified platform for accessing various campus services.

13. REFERENCES :

[1] https://reactnative.dev/docs/getting-started
[2] https://reactnavigation.org/docs/getting-started/ .
[3] https://reactnavigation.org/docs/native-stack-navigator.
[4] https://reactnavigation.org/docs/drawer-navigator.
[5] https://reactnavigation.org/docs/bottom-tab-navigator
[6]https://www.npmjs.com/package/@react-nativ asyncstorage/async-storage
[7]https://redux-toolkit.js.org/usage/usage-guide.
[8] https://www.npmjs.com/package/react-native-scan-barcode
[9] https://github.com/react-native-camera/react-native-camera
[10] https://www.npmjs.com/package/react-native-chart-kit
[11] https://www.npmjs.com/package/react-native-gifted-charts
[12] https://github.com/jestjs/jest