ДОДАТОК А. FrontCode

1. App.js

```
import { Button, Space } from 'antd';
import './stylesheets/theme.css';
import './stylesheets/alignments.css';
import './stylesheets/textelements.css';
import './stylesheets/custom-components.css';
import './stylesheets/form-elements.css';
import './stylesheets/layout.css';
import { Routes, Route, BrowserRouter } from 'react-router-dom';
import Login from './pages/common/Login';
import Register from './pages/common/Register';
import ProtectedRoute from './components/ProtectedRoute';
import Home from "./pages/common/Home";
import Exams from './pages/admin/Exams';
import AddEditExam from './pages/admin/Exams/AddEditExam';
import Loader from './components/Loader';
import { useSelector } from 'react-redux';
import WriteExam from './pages/user/WriteExam';
import UserReports from './pages/user/UserReports';
import AdminReports from './pages/admin/AdminReports';

function App() {
  const { loading } = useSelector(state => state.loader)
  return (
    <>
      {loading && <Loader />}
      <BrowserRouter>
        <Routes>
          {/* Common Routes */}
          <Route path="/login" element={<Login />} />
          <Route path="/register" element={<Register />} />

          {/* User Routes */}
          <Route
            path="/"
            element={
              <ProtectedRoute>
                <Home />
              </ProtectedRoute>
            }
          />
          <Route
            path="/user/write-exam/:id"
```

```
  element={
    <ProtectedRoute>
      <WriteExam />
    </ProtectedRoute>
  }
/>
<Route
  path="/user/reports"
  element={
    <ProtectedRoute>
      <UserReports />
    </ProtectedRoute>
  }
/>
{/* Admin Routes */}
<Route
  path="/admin/tests"
  element={
    <ProtectedRoute>
      <Exams />
    </ProtectedRoute>
  }
/>
<Route
  path="/admin/exams/add"
  element={
    <ProtectedRoute>
      <AddEditExam />
    </ProtectedRoute>
  }
/>

<Route
  path="/admin/exams/edit/:id"
  element={
    <ProtectedRoute>
      <AddEditExam />
    </ProtectedRoute>
  }
/>
<Route
  path="/admin/reports"
  element={
    <ProtectedRoute>
      <AdminReports />
    </ProtectedRoute>
  }
```

```
        />
      </Routes>
    </BrowserRouter>
  </>
 );
}

export default App;
```

2. exam.js

```
import axios from "axios";

const { default: axiosInstance } = require(".");

// add test

export const addExam = async (payload) => {
  try {
    const response = await axiosInstance.post("/api/exams/add", payload);
    return response.data;
  } catch (error) {
    return error.response.data;
  }
};

// get all tests
export const getAllExams = async () => {
  try {
    const response = await axiosInstance.post("/api/exams/get-all-exams");
    return response.data;
  } catch (error) {
    return error.response.data;
  }
};

// get test by id

export const getExamById = async (payload) => {
  try {
    const response = await axiosInstance.post("/api/exams/get-exam-by-id", payload);
    return response.data;
  } catch (error) {
    return error.response.data;
  }
};
```

```
//edit test by id

export const editExamById = async (payload) => {
    try {
        const response = await axiosInstance.post("/api/exams/edit-exam-by-id", payload)
        return response.data;
    } catch (error) {
        return error.response.data;
    }
}

//delete test by id

export const deleteExamById = async (payload) => {
    try {
        const response = await axiosInstance.post("/api/exams/delete-exam-by-id", payload);
        return response.data;
    } catch (error) {
        return error.response.data;
    }
}

// add question to test
export const addQuestionToExam = async (payload) => {
    try {
        const response = await axiosInstance.post("/api/exams/add-question-to-exam", payload);
        return response.data;
    } catch (error) {
        return error.response.data;
    }
}

export const editQuestionById = async (payload) => {
    try {
        const response = await axiosInstance.post("/api/exams/eddit-question-in-exam", payload);
        return response.data;
    } catch (error) {
        return error.response.data;
    }
}

export const deleteQuestionById = async(payload) => {
    try {
        const response = await axiosInstance.post("/api/exams/delete-question-in-exam", payload);
        return response.data;
    } catch (error) {
        return error.response.data; }};
```

3. apicalls → index.js

```
import axios from 'axios';

const axiosInstance = axios.create({
   headers:{
      Authorization : `Bearer ${localStorage.getItem('token')}`
   }
})

export default axiosInstance;
```

4. reports.js

```
const { default: axiosInstance } = require(".");

//add report
export const addReport = async(payload) => {
   try {
      const response = await axiosInstance.post("/api/reports/add-report", payload);
      return response.data;
   } catch (error) {
      return error.response.data;
   }
}

// get all reports
export const getAllReports = async(filters) => {
   try {
      const response = await axiosInstance.post("/api/reports/get-all-reports", filters);
      return response.data;
   } catch (error) {
      return error.response.data;
   }
}

//get all reports by user
export const getAllReportsByUser = async() => {
   try {
      const response = await axiosInstance.post("/api/reports/get-all-reports-by-user");
      return response.data;
   } catch (error) {
      return error.response.data;
   }
}
```

4. user.js

```
const { default: axiosInstance } = require(".");

export const registerUser = async (payload) => {
   try {
      const response = await axiosInstance.post('/api/users/register', payload);
      return response.data;
   } catch (error) {
      return error.response.data;
   }
}

export const loginUser = async (payload) => {
   try {
      const response = await axiosInstance.post('/api/users/login', payload);
      return response.data;
   } catch (error) {
      return error.response.data;
   }
}

export const getUserInfo = async () => {
   try {
      const response = await axiosInstance.post('/api/users/get-user-info');
      return response.data;
   } catch (error) {
      return error.response.data;
   }
}
```

5. Loader.js

```
import React from 'react'

function Loader() {
  return (
    <div className='loader-parent'>
      <div className='loader'></div>
    </div>
  )
}
```

6. PageTitle.js

```
import React from 'react'

function PageTitle({title}) {
  return (
    <div className='mt-2'>
```

```
      <h1>{title}</h1>
    </div>
  )
}

export default PageTitle
```

7. ProtectedRoute.js

```
import { message } from "antd";
import React, { useEffect, useState } from "react";
import { getUserInfo } from "../apicalls/users";
import { useDispatch, useSelector } from "react-redux";
import { SetUser } from "../redux/usersSlice.js";
import { useNavigate } from "react-router-dom";
import { HideLoading, ShowLoading } from "../redux/loaderSlice";


function ProtectedRoute({ children }) {
  const { user } = useSelector((state) => state.users);
  const [menu, setMenu] = useState([]);
  const [collapsed, setCollapsed] = useState(false);
  const dispatch = useDispatch();
  const navigate = useNavigate();

  const userMenu = [
    {
      title: "Home",
      paths: ["/", "/user/write-exam"],
      icon: <i className="ri-home-line"></i>,
      onClick: () => navigate("/"),
    },
    {
      title: "Reports",
      paths: ["/user/reports"],
      icon: <i className="ri-bar-chart-line"></i>,
      onClick: () => navigate("/user/reports"),
    },

    {
      title: "Logout",
      paths: ["/logout"],
      icon: <i className="ri-logout-box-line"></i>,
      onClick: () => {
        localStorage.removeItem("token");
        navigate("/login");
      },
    },
```

```
];
const adminMenu = [
  {
    title: "Home",
    paths: ["/", "/user/write-exam"],
    icon: <i className="ri-home-line"></i>,
    onClick: () => navigate("/"),
  },
  {
    title: "Tests",
    paths: ["/admin/tests", "/admin/tests/add"],
    icon: <i className="ri-file-list-line"></i>,
    onClick: () => navigate("/admin/tests"),
  },
  {
    title: "Reports",
    paths: ["/admin/reports"],
    icon: <i className="ri-bar-chart-line"></i>,
    onClick: () => navigate("/admin/reports"),
  },

  {
    title: "Logout",
    paths: ["/logout"],
    icon: <i className="ri-logout-box-line" ></i>,
    onClick: () => {
      localStorage.removeItem("token");
      navigate("/login");
    },
  },
]

const getUserData = async () => {
  try {
    dispatch(ShowLoading());
    const response = await getUserInfo()
    dispatch(HideLoading());
    if (response.success) {
      dispatch(SetUser(response.data))
      if (response.data.isAdmin) {
        setMenu(adminMenu);
      } else {
        setMenu(userMenu);
      }
    } else {
      message.error(response.message)
    }
```

```
  } catch (error) {
    navigate("/login");
    dispatch(HideLoading());
    message.error(error.message)
  }
}


useEffect(() => {
  if (localStorage.getItem("token")) {
    getUserData();
  } else {
    navigate("/login");
  }
}, []);

const activeRoute = window.location.pathname;

const getIsActiveOrNot = (paths) => {
  if (paths.includes(activeRoute)) {
    return true;
  } else {
    if (activeRoute.includes("/admin/tests/edit") && paths.includes("/admin/tests")) {
      return true;
    }
    if (activeRoute.includes("/user/write-exam") && paths.includes("/user/write-exam")) {
      return true;
    }
  }
  return false;
}

return (
  <div className='layout'>
    <div className='flex gap-2 w-full h-full h-100'>
      <div className='sidebar'>
        <div className='menu'>
          {menu.map((item, index) => {
            return (
              <div
                className={`menu-item ${getIsActiveOrNot(item.paths) && "active-menu-item"
                  }`}
                key={index}
                onClick={item.onClick}
              >
                {item.icon}
                {!collapsed && <span>{item.title}</span>}
```

```
            </div>
          );
        })}
      </div>
    </div>
    <div className='body'>
      <div className="header flex justify-between">
        {!collapsed && (
          <i
            className="ri-close-line"
            onClick={() => setCollapsed(true)}
          ></i>
        )}
        {collapsed && (
          <i
            className="ri-menu-line"
            onClick={() => setCollapsed(false)}
          ></i>
        )}
        <h1 className='text-2xl'>QUICKI</h1>
        <div>
          <div className="flex gap-1 items-center">
            <h1 className='text-md '>{user?.name}</h1>
          </div>
          <span>
            Role : {user?.isAdmin ? "Admin" : "User"}
          </span>
        </div>
      </div>
      <div className='content'>{children}</div>
    </div>
  </div>
 );
}

export default ProtectedRoute
```

8. AdminReports → index.js

```
import React from 'react'
import PageTitle from '../../../components/PageTitle'
import { Table, message } from 'antd'
import { useDispatch } from 'react-redux'
import { HideLoading, ShowLoading } from '../../../redux/loaderSlice'
import { getAllReports } from '../../../apicalls/reports'
import { useEffect } from 'react'
import moment from "moment"
```

```
function AdminReports() {
  const [reportsData, setReportsData] = React.useState([])
  const dispatch = useDispatch();
  const [filters, setFilters] = React.useState({
    examName: "",
    userName: "",
  });

  const columns = [
    {
      title: "Test Name",
      dataIndex: "examName",
      render: (text, record) => <>{record.exam.name}</>,
    },
    {
      title: "User Name",
      dataIndex: "userName",
      render: (text, record) => <>{record.user.name}</>,
    },
    {
      title: "Date",
      dataIndex: "date",
      render: (text, record) => (
        <>{moment(record.createdAt).format("DD-MM-YYYY hh:mm:ss")}</>
      ),
    },
    {
      title: "Total Marks",
      dataIndex: "totalQuestions",
      render: (text, record) => <>{record.exam.totalMarks}</>,
    },
    {
      title: "Passing Marks",
      dataIndex: "correctAnswers",
      render: (text, record) => <>{record.exam.passingMarks}</>,
    },
    {
      title: "Obtained Marks",
      dataIndex: "correctAnswers",
      render: (text, record) => <>{record.result.correctAnswers.length}</>,
    },
    {
      title: "Verdict",
      dataIndex: "verdict",
      render: (text, record) => <>{record.result.verdict}</>,
    },
```

```
];

const getData = async (tempFilters) => {
   try {
      dispatch(ShowLoading());
      const response = await getAllReports(tempFilters);
      if (response.success) {
         setReportsData(response.data);
      } else {
         message.error(response.message);
      }
      dispatch(HideLoading());
   } catch (error) {
      dispatch(HideLoading());
      message.error(error.message);
   }
};

useEffect(() => {
   getData(filters);
}, []);

return (
   <div>
      <PageTitle title="Reports" />
      <div className="divider"></div>
      <div className='flex gap-2'>
         <input type='text' placeholder='Test Name'
            value={filters.examName}
            onChange={(e) => setFilters({ ...filters, examName: e.target.value })}
         />
         <input type='text' placeholder='User'
            value={filters.userName}
            onChange={(e) => setFilters({ ...filters, userName: e.target.value })}
         />
         <button className='primary-outlined-btn'
            onClick={() => {
               setFilters({
                  examName : "",
                  userName : "",
               })
               getData({
                  examName : "",
                  userName : "",
               });
            }}
         >Clear
```

```
            </button>
            <button className='primary-contained-btn'
              onClick={() => getData(filters)}
            >Search
            </button>
          </div>
          <Table columns={columns} dataSource={reportsData} className='mt-2' />
        </div>
    );
}

export default AdminReports;
```

9. AddEditExam.js

```
import React, { useEffect } from 'react'
import PageTitle from '../../../components/PageTitle'
import { Col, Form, Row, Select, message, Table } from 'antd'
import { addExam, deleteQuestionById, editExamById, getExamById, getExamData } from '../../../apicalls/exams';
import { useNavigate, useParams } from "react-router-dom"
import { useDispatch } from 'react-redux';
import { HideLoading, ShowLoading } from '../../../redux/loaderSlice';
import { Tabs } from 'antd'
import TabPane from 'antd/es/tabs/TabPane';
import AddEditQuestion from './AddEditQuestion';


function AddEditExam() {
    const dispatch = useDispatch();
    const navigate = useNavigate();
    const [examData, setExamData] = React.useState(null);
    const [showAddEditQuestionModal, setShowAddEditQuestionModal] = React.useState(false)
    const [selectedQuestion, setSelectedQuestion] = React.useState(null);
    const params = useParams();
    const onFinish = async (values) => {
        try {
            dispatch(ShowLoading());
            let response;
            if (params.id) {
                response = await editExamById({
                    ...values,
                    examId: params.id
                });
            } else {
                response = await addExam(values);
            }
            if (response.success) {
```

```
            message.success(response.message);
            navigate("/admin/tests");
        } else {
            message.error(response.message);
        }
        dispatch(HideLoading());
    } catch (error) {
        dispatch(HideLoading());
        message.error(error.message);
    }
};

const getExamData = async () => {
    try {
        dispatch(ShowLoading());
        const response = await getExamById({
            examId: params.id,
        });
        dispatch(HideLoading());
        if (response.success) {
            setExamData(response.data);
        } else {
            message.error(response.message);
        }
    } catch (error) {
        dispatch(HideLoading());
        message.error(error.message);
    }
};
useEffect(() => {
    if (params.id) {
        getExamData();
    }
}, []);

const deleteQuestion = async (questionId) => {
    try {
        dispatch(ShowLoading());
        const response = await deleteQuestionById({
            questionId,
            examId: params.id
        });
        dispatch(HideLoading());
        if (response.success) {
            message.success(response.message);
            getExamData();
        } else {
```

```
            message.error(response.message);
        }
    } catch (error) {
        dispatch(HideLoading());
        message.error(error.message);
    }
}

const questionColumns = [
    {
        title: "Question",
        dataIndex: "name",
    },
    {
        title: "Options",
        dataIndex: "options",
        render: (text, record) => {
            return Object.keys(record.options).map((key) => {
                return (
                    <div>
                        {key} : {record.options[key]}
                    </div>
                );
            });
        },
    },
    {
        title: "Correct Option",
        dataIndex: "correctOption",
        render: (text, record) => {
            return ` ${record.correctOption}: ${record.options[record.correctOption]}`;
        },
    },
    {
        title: "Action",
        dataIndex: "action",
        render: (text, record) => (
            <div className="flex gap-2">
                <i
                    className="ri-pencil-line"
                    onClick={() => {
                        setSelectedQuestion(record);
                        setShowAddEditQuestionModal(true);
                    }}
                ></i>
                <i
                    className="ri-delete-bin-line"
```

```
                onClick={() => {
                    deleteQuestion(record._id);
                }}
            ></i>
        </div>
    )

  }
]
return (
  <div>
    <PageTitle title=
      {params.id ? "Edit Test" : "Add Test"}
    />
    <div className='divider'></div>

    {(examData || !params.id) && (
      <Form layout='vertical ' onFinish={onFinish} initialValues={examData}>
        <Tabs defaultActiveKey='1'>
          <TabPane tab="Test Components " key="1" >
            <Row gutter={[10, 10]}>
              <Col span={8}>
                <Form.Item label="Test Name " name="name" >
                  <input type='text ' />
                </Form.Item>
              </Col>
              <Col span={8}>
                <Form.Item label="Test Duration" name="duration">
                  <input type='number' />
                </Form.Item>
              </Col>
              <Col span={8}>
                <Form.Item label="Category" name="category">
                  <select name="" id="">
                    <option value="">Select Category</option>
                    <option value="Math">Math tests</option>
                    <option value="Verbal">Verbal tests</option>
                    <option value="Puzzles">Puzzles</option>
                    <option value="Logic">Logic tests</option>
                  </select>
                </Form.Item>
              </Col>
              <Col span={8}>
                <Form.Item label="Total Marks" name="totalMarks">
                  <input type='number' />
                </Form.Item>
              </Col>
```

```
                    <Col span={8}>
                      <Form.Item label="Passing Marks" name="passingMarks">
                        <input type='number' />
                      </Form.Item>
                    </Col>
                  </Row>
                  <div className='flex justify-end gap-2' style={{ marginTop: '280px', marginRight:
'500px' }}>
                    <button className='primary-outlined-btn-1 text-md' type='button'
                      onClick={() => navigate("/admin/tests")}
                    >
                      Cancel</button>
                      <button      className='primary-contained-btn-1      text-md'      type='submit'
>Save</button>
                  </div>
                </TabPane>
                {params.id && (
                  <TabPane tab="Questions" key="2">
                    <div className='flex justify-end'>

                      <button className='primary-outlined-btn'
                        type='button'
                        onClick={() => setShowAddEditQuestionModal(true)}
                      >Add Question</button>
                    </div>
                    <Table
                      columns={questionColumns}
                      dataSource={examData?.questions || []}
                    />
                  </TabPane>
                )}
              </Tabs>
            </Form>
          )}

          {showAddEditQuestionModal && (
            <AddEditQuestion
              setShowAddEditQuestionModal={setShowAddEditQuestionModal}
              showAddEditQuestionModal={showAddEditQuestionModal}
              examId={params.id}
              refreshData={getExamData}
              selectedQuestion={selectedQuestion}
              setSelectedQuestion={setSelectedQuestion}
            />
          )}
        </div>
    );
```

```
}

export default AddEditExam
export default Loader
```

10. AddEditQuestion.js

```
import { Form, Modal, message } from 'antd'

import React from 'react'

import { addQuestionToExam, editQuestionById } from '../../../apicalls/exams';

import { useDispatch } from 'react-redux';

import { HideLoading, ShowLoading } from '../../../redux/loaderSlice';


function AddEditQuestion({
    showAddEditQuestionModal,
    setShowAddEditQuestionModal,
    refreshData,
    examId,
    selectedQuestion,
    setSelectedQuestion,
}) {
    const dispatch = useDispatch();
    const onFinish = async (values) => {
        try {
            dispatch(ShowLoading());
            const requiredPayload = {
                name: values.name,
                correctOption: values.correctOption,
                options: {
                    A: values.A,
                    B: values.B,
                    C: values.C,
```

```
            D: values.D,
          },
          exam: examId,
        };

        let response
        if (selectedQuestion) {
          response = await editQuestionById({
            ...requiredPayload,
            questionId: selectedQuestion._id
          })
        } else {
          response = await addQuestionToExam(requiredPayload);
        }
        if (response.success) {
          message.success(response.message);
          refreshData();
          setShowAddEditQuestionModal(false);
        } else {
          message.error(response.message);
        }

        setSelectedQuestion(null)
        dispatch(HideLoading());
      } catch (error) {
        dispatch(HideLoading());
        message.error(error.message);
      }
    };
    return (
```

```
<Modal title={selectedQuestion ? "Edit Question" : "Add Question"}
    visible={showAddEditQuestionModal}
    footer={false}
    onCancel={() => {
        setShowAddEditQuestionModal(false)
        setSelectedQuestion(null)
    }}>
    <Form onFinish={onFinish} layout='vertical'
        initialValues={{
            name: selectedQuestion?.name,
            A: selectedQuestion?.options?.A,
            B: selectedQuestion?.options?.B,
            C: selectedQuestion?.options?.C,
            D: selectedQuestion?.options?.D,
            correctOption: selectedQuestion?.correctOption,
        }}
    >
        <Form.Item name="name" label="Question">
            <input type='text' />
        </Form.Item>
        <Form.Item name="correctOption" label="Correct Option">
            <input type='text' />
        </Form.Item>
        <div className='flex gap-8' >
            <Form.Item name="A" label="Option A">
                <input type='text' />
            </Form.Item>
            <Form.Item name="B" label="Option B" style={{marginLeft:'77px'}}>
                <input type='text' />
            </Form.Item>
```

```
            </div>
            <div className='flex gap-8 ' >
              <Form.Item name="C" label="Option C">
                <input type='text' />
              </Form.Item>
              <Form.Item name="D" label="Option D" style={{marginLeft:'77px'}}>
                <input type='text' />
              </Form.Item>
            </div>

            <div className='flex justify-end mt-2 gap-3'>
              <button
                className='primary-outlined-btn'
                type='button'
                onClick={() => setShowAddEditQuestionModal(false)}
              >
                Cancel
              </button>
              <button className='primary-contained-btn'>
                Save
              </button>
            </div>
          </Form>
        </Modal>
    );
}

export default AddEditQuestion
```

11. Exams → index.js

```
import React from 'react'
import { message, Table } from "antd";
import PageTitle from '../../../components/PageTitle';
import { useNavigate } from 'react-router-dom';
import { useEffect } from "react";
import { useDispatch } from "react-redux";
import { deleteExamById, getAllExams } from "../../../apicalls/exams";
import { HideLoading, ShowLoading } from "../../../redux/loaderSlice";

function Exams() {
  const navigate = useNavigate();
  const [exams, setExams] = React.useState([]);
  const dispatch = useDispatch();

  const getExamsData = async () => {
    try {
      dispatch(ShowLoading());
      const response = await getAllExams();
      dispatch(HideLoading());
      if (response.success) {
        setExams(response.data);
      } else {
        message.error(response.message);
      }
    } catch (error) {
      dispatch(HideLoading());
      message.error(error.message);
    }
  };

  const deleteExam = async (examId) => {
    try {
      dispatch(ShowLoading());
      const response = await deleteExamById({
        examId,
      });
      dispatch(HideLoading());
      if (response.success) {
        message.success(response.message);
        getExamsData();
      } else {
        message.error(response.message);
      }
    } catch (error) {
      dispatch(HideLoading());
```

```
    message.error(error.message);
  }
};
const columns = [
  {
    title: "Training Test",
    dataIndex: "name",
  },
  {
    title: "Duration",
    dataIndex: "duration",
  },
  {
    title: "Category",
    dataIndex: "category",
  },
  {
    title: "Total Marks",
    dataIndex: "totalMarks",
  },
  {
    title: "Passing Marks",
    dataIndex: "passingMarks",
  },
  {
    title: "Action",
    dataIndex: "action",
    render: (text, record) => (
      <div className="flex gap-2">
        <i
          className="ri-pencil-line"
          onClick={() => navigate(`/admin/exams/edit/${record._id}`)}
        ></i>
        <i
          className="ri-delete-bin-line"
          onClick={() => deleteExam(record._id)}
        ></i>
      </div>
    ),
  },

];

useEffect(() => {
  getExamsData();
}, []);
return (
```

```
    <div>
      <div className="flex justify-between mt-2 items-end">
        <PageTitle title="Training Tests" />

        <button
          className="primary-outlined-btn flex items-center"
          onClick={() => navigate("/admin/exams/add")}
        >
          <i className="ri-add-line"></i>
          Add Test
        </button>
      </div>
      <div className="divider"></div>

      <Table columns={columns} dataSource={exams} />
    </div>
  );
}

export default Exams;
```

12. Home → index.js

```
import { Col, message, Row } from "antd";
import React, { useEffect } from "react";
import { useDispatch, useSelector } from "react-redux";
import { getAllExams } from "../../../apicalls/exams";
import { HideLoading, ShowLoading } from "../../../redux/loaderSlice";
import PageTitle from "../../../components/PageTitle";
import { useNavigate } from "react-router-dom";
function Home() {
  const [exams, setExams] = React.useState([]);
  const navigate = useNavigate();
  const dispatch = useDispatch();
  const { user } = useSelector((state) => state.users);
  const getExams = async () => {
    try {
      dispatch(ShowLoading());
      const response = await getAllExams();
      if (response.success) {
        setExams(response.data);
      } else {
        message.error(response.message);
      }
      dispatch(HideLoading());
    } catch (error) {
      dispatch(HideLoading());
      message.error(error.message);
```

```
    }
  };

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

  return (
    user && (
      <div>
        <PageTitle title={`Hi ${user.name}, Welcome to Quicki!`} />
        <div className="divider"></div>
        <Row gutter={[16, 16]}>
          {exams.map((exam) => (
            <Col span={6}>
              <div className="card-lg flex flex-col gap-1 p-2">
                <h1 className="text-2xl">{exam?.name}</h1>

                <h1 className="text-md">Category : {exam.category}</h1>

                <h1 className="text-md">Total Marks : {exam.totalMarks}</h1>
                <h1 className="text-md">Passing Marks : {exam.passingMarks}</h1>
                <h1 className="text-md">Duration : {exam.duration}</h1>
                <button
                  className="primary-outlined-btn text-md"
                  onClick={() => navigate(`/user/write-exam/${exam._id}`)}
                >
                  Start Test
                </button>
              </div>
            </Col>
          ))}
        </Row>
      </div>
    )
  );
}

export default Home;
```

13. Login → index.js

```
import React from 'react'
import { Form, message } from 'antd'
import { Link } from 'react-router-dom'
import { loginUser } from '../../../apicalls/users';
import { useDispatch } from 'react-redux';
import { HideLoading, ShowLoading } from '../../../redux/loaderSlice';
```

```
function Login() {
  const dispatch = useDispatch();
  const onFinish = async (values) => {
    try {
      dispatch(ShowLoading());
      const response = await loginUser(values);
      dispatch(HideLoading());
      if (response.success) {
        message.success(response.message);
        localStorage.setItem("token", response.data);
        window.location.href = "/";
      } else {
        message.error(response.message);
      }
    } catch (error) {
      dispatch(HideLoading());
      message.error(error.message);
    }
  }
  return (
    <div className="flex justify-center items-center h-screen w-screen bg-primary">
      <div className="card w-400 p-3 bg-white">
        <div className="flex flex-col">
          <div className="flex">
            <h1 className="text-2xl">QUICKI - LOGIN <i class="ri-login-circle-line"></i></h1>

          </div>
          <div className="divider"></div>
          <Form layout="vertical" className="mt-2" onFinish={onFinish}>
            <Form.Item name="email" label="Email">
              <input type="text" />
            </Form.Item>
            <Form.Item name="password" label="Password">
              <input type="password" />
            </Form.Item>
            <div className='flex flex-col gap-2'>
              <button type='submit' className='primary-contained-btn mt-2 w-100' >Log in</button>
              <Link to="/register" className='underline'>Don't have an account?Register here.</Link>
            </div>
          </Form>
        </div>
      </div>

    </div>
  )
}
```

export default Login

14. Register → index.js

```
import React from 'react'
import { Form, message } from 'antd'
import { Link, useNavigate } from 'react-router-dom'
import { registerUser } from '../../../apicalls/users'
import { useDispatch } from 'react-redux'
import { HideLoading, ShowLoading } from '../../../redux/loaderSlice'

function Register() {
  const dispatch = useDispatch();
  const navigate = useNavigate();
  const onFinish = async (values) => {
    try {
      dispatch(ShowLoading());
      const response = await registerUser(values);
      dispatch(HideLoading());

      if (response.success) {
        message.success(response.message);
        navigate("/login");
      } else {
        message.error(response.message);
      }
    } catch (error) {
      dispatch(HideLoading());
      message.error(error.message);
    }};
  return (
    <div className='flex justify-center items-center h-screen w-screen bg-primary'>
      <div className='card w-400 p-3 bg-white'>
        <div className="flex flex-col">
          <div className='flex'>
            <h1 className='text-2xl'>QUICKI - REGISTER
              <i class="ri-user-add-line"></i>
            </h1>
          </div>
          <div className='divider'></div>
          <Form layout='vertical' className='mt-2' onFinish={onFinish}>
            <Form.Item name='name' label='Name'>
              <input type='text' />
            </Form.Item>
            <Form.Item name='email' label='Email'>
              <input type='text' />
            </Form.Item>
```

```
          <Form.Item name='password' label='Password'>
            <input type='password' />
          </Form.Item>
          <div className='flex flex-col gap-2'>
            <button type='submit' className='primary-contained-btn mt-2 w-100'>Register</button>
            <Link to="/login" className='underline'>Already have an account?Login here.</Link>
          </div>
        </Form>
      </div>
    </div>

  </div>)}

export default Register;
```

15. UserReports → index.js

```
import React from "react";

import PageTitle from "../../../components/PageTitle";

import { message, Modal, Table } from "antd";

import { useDispatch } from "react-redux";

import { HideLoading, ShowLoading } from "../../../redux/loaderSlice";

import { getAllReportsByUser } from "../../../apicalls/reports";

import { useEffect } from "react";

import moment from "moment";


function UserReports() {

  const [reportsData, setReportsData] = React.useState([]);

  const dispatch = useDispatch();

  const columns = [

    {

      title: "Tests Name",

      dataIndex: "examName",

      render: (text, record) => <>{record.exam.name}</>,

    },

    {
```

```
    title: "Date",
    dataIndex: "date",
    render: (text, record) => (
      <>{moment(record.createdAt).format("DD-MM-YYYY hh:mm:ss")}</>
    ),
  },
  {
    title: "Total Marks",
    dataIndex: "totalQuestions",
    render: (text, record) => <>{record.exam.totalMarks}</>,
  },
  {
    title: "Passing Marks",
    dataIndex: "correctAnswers",
    render: (text, record) => <>{record.exam.passingMarks}</>,
  },
  {
    title: "Obtained Marks",
    dataIndex: "correctAnswers",
    render: (text, record) => <>{record.result.correctAnswers.length}</>,
  },
  {
    title: "Verdict",
    dataIndex: "verdict",
    render: (text, record) => <>{record.result.verdict}</>,
  },
];

const getData = async () => {
  try {
```

```
      dispatch(ShowLoading());
      const response = await getAllReportsByUser();
      if (response.success) {
        setReportsData(response.data);
      } else {
        message.error(response.message);
      }
      dispatch(HideLoading());
    } catch (error) {
      dispatch(HideLoading());
      message.error(error.message);
    }
  };

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

  return (
    <div>
      <PageTitle title="Reports" />
      <div className="divider"></div>
      <Table columns={columns} dataSource={reportsData} />
    </div>
  );
}

export default UserReports;
```

16. index.jsx

```
import React, { useEffect, useState } from 'react'
import { useDispatch, useSelector } from 'react-redux';
import { useNavigate, useParams } from 'react-router-dom';
import { HideLoading, ShowLoading } from '../../../redux/loaderSlice';
import { getExamById } from '../../../apicalls/exams';
import { addReport } from "../../../apicalls/reports";
import { message } from 'antd';
import Instructions from './Instructions';

function WriteExam() {
    const [examData, setExamData] = React.useState(null);
    const [questions = [], setQuestions] = React.useState([]);
    const [selectedQuestionIndex, setSelectedQuestionIndex] = React.useState(0);
    const [selectedOptions, setSelectedOptions] = React.useState({});
    const [result = {}, setResult] = React.useState({});
    const params = useParams();
    const dispatch = useDispatch();
    const navigate = useNavigate();
    const [view, setView] = useState("instructions");
    const [secondsLeft = 0, setSecondsLeft] = useState(0);
    const [timeUp, setTimeUp] = useState(false);
    const [intervalId, setIntervalId] = useState(null);
    const { user } = useSelector((state) => state.users);

    const getExamData = async () => {
        try {
            dispatch(ShowLoading());
            const response = await getExamById({
                examId: params.id,
```

```
      });
      dispatch(HideLoading());
      if (response.success) {
        setQuestions(response.data.questions);
        setExamData(response.data);
        setSecondsLeft(response.data.duration);
      } else {
        message.error(response.message);
      }
    } catch (error) {
      dispatch(HideLoading());
      message.error(error.message);
    }
};

const calculateResult = async () => {
    try {
      let correctAnswers = [];
      let wrongAnswers = [];

      questions.forEach((question, index) => {
        if (question.correctOption === selectedOptions[index]) {
          correctAnswers.push(question);
        } else {
          wrongAnswers.push(question);
        }
      });

      let verdict = "Pass";
      if (correctAnswers.length < examData.passingMarks) {
```

```
            verdict = "Fail";
        }

        const tempResult = {
            correctAnswers,
            wrongAnswers,
            verdict,

        }
        setResult(tempResult);
        dispatch(ShowLoading());
        const response = await addReport({
            exam: params.id,
            result: tempResult,
            user: user._id
        });
        dispatch(HideLoading());
        if (response.success) {
            setView("result");
        } else {
            message.error(response.message);
        }
    } catch (error) {
        dispatch(HideLoading());
        message.error(error.message);
    }
};

const startTimer = () => {
    let totalSeconds = examData.duration;
```

```
    const intervalId = setInterval(() => {
        if (totalSeconds > 0) {
            totalSeconds = totalSeconds - 1;
            setSecondsLeft(totalSeconds);
        } else {
            setTimeUp(true);
        }
    }, 1000);
    setIntervalId(intervalId);
};

useEffect(() => {
    if (timeUp && view === "questions") {
        clearInterval(intervalId);
        calculateResult();
    }
}, [timeUp]);

useEffect(() => {
    if (params.id) {
        getExamData();
    }
}, []);
return (
    examData && (
        <div className='mt-2'>
            <div className='divider'></div>
            <h1 className='text-center'>{examData.name}</h1>
            <div className='divider'></div>
```

```
{view === "instructions" && (
  <Instructions
    examData={examData}
    setView={setView}
    startTimer={startTimer}
  />
)}
{view === "questions" && (
  <div className="flex flex-col gap-2">
    <div className='flex justify-between'>
      <h1 className="text-2xl">
        {selectedQuestionIndex + 1} :{" "}
        {questions[selectedQuestionIndex].name}
      </h1>
      <div className='timer'>
        <span className='text-2xl'>{secondsLeft}</span>
      </div>


    </div>

    <div className="flex flex-col gap-2">
      {Object.keys(questions[selectedQuestionIndex].options).map(
        (option, index) => {
          return (
            <div
              className={`flex gap-2 flex-col ${selectedOptions[selectedQuestionIndex]
=== option
                ? "selected-option"
                : "option"
```

```
                }`}
                key={index}
                onClick={() => {
                  setSelectedOptions({
                    ...selectedOptions,
                    [selectedQuestionIndex]: option,
                  });
                }}
              >
                <h1 className="text-xl">
                  {option} :{" "}
                  {questions[selectedQuestionIndex].options[option]}
                </h1>
              </div>
            );
          }
        )}
      </div>

      <div className="flex justify-between">
        {selectedQuestionIndex > 0 && (
          <button
            className="primary-outlined-btn"
            onClick={() => {
              setSelectedQuestionIndex(selectedQuestionIndex - 1);
            }}
          >
            Previous
          </button>
        )}
```

```
            {selectedQuestionIndex < questions.length - 1 && (
              <button
                 className="primary-contained-btn"
                 onClick={() => {
                    setSelectedQuestionIndex(selectedQuestionIndex + 1);
                 }}
              >
                 Next
              </button>
            )}

            {selectedQuestionIndex === questions.length - 1 && (
              <button
                 className="primary-contained-btn"
                 onClick={() => {
                    clearInterval(intervalId);
                    setTimeUp(true);
                 }}
              >
                 Submit
              </button>
            )}
         </div>
      </div>
   )}

   {view === "result" && (
      <div className='flex items-center mt-2 justify-center result' >
         <div className='flex flex-col gap-2 '>
```

```
<h1 className='text-2xl'>RESULT</h1>
<div className='divider'></div>
<div className='marks'>
   <h1 className='text-md'>Total Marks : {examData.totalMarks}</h1>
   <h1 className='text-md'>Obtained Marks :
      {result.correctAnswers.length}
   </h1>
   <h1 className='text-md'>
      Wrong Answers : {result.wrongAnswers.length}
   </h1>
   <h1 className='text-md'>Passing Marks : {examData.passingMarks}</h1>
   <h1 className='text-md'>VERDICT :
      {result.verdict}
   </h1>
   <div className='flex gap-2 mt-2'>
      <button className='primary-outlined-btn-2'
         onClick={() => {
            setView("instructions");
            setSelectedQuestionIndex(0);
            setSelectedOptions({});
            setSecondsLeft(examData.duration);
         }}
      >
         Retake Test
      </button>
      <button className='primary-contained-btn-2'
         onClick={() => {
            setView("review");
         }}
      >
```

```
                    Review Answers
                  </button>
                </div>
              </div>
            </div>
            <div className='lottie-animation'>
              {result.verdict === "Pass" && (<lottie-player
                src="https://assets2.lottiefiles.com/packages/lf20_uu0x8lqv.json"
                background="transparent"
                speed="1"
                loop
                autoplay>

              </lottie-player>
              )}

              {result.verdict === "Fail" && (<lottie-player
                src="https://assets2.lottiefiles.com/packages/lf20_e5pm2x8q.json"
                background="transparent"
                speed="1"
                loop
                autoplay>
              </lottie-player>
              )}
            </div>
          </div>
        )}

        {view === "review" && (
          <div className='flex flex-col gap-2'>
```

```
{questions.map((question, index) => {
  const isCorrect = question.correctOption === selectedOptions[index];
  return (
    <div className={`flex flex-col gap-1 p-2 ${isCorrect ? "bg-success" : "bg-error"}`}>
      <h1 className="text-xl">
        {index + 1} : {question.name}
      </h1>
      <h1 className="text-md">
        Submitted Answer : {selectedOptions[index]} -{" "}
        {question.options[selectedOptions[index]]}
      </h1>
      <h1 className="text-md">
        Correct Answer : {question.correctOption} -{" "}
        {question.options[question.correctOption]}
      </h1>
    </div>
  );
})}

<div className="flex justify-center gap-2">
  <button
    className="primary-outlined-btn"
    onClick={() => {
      navigate("/");
    }}
  >
    Close
  </button>
  <button
```

```
                        className="primary-contained-btn"
                        onClick={() => {
                            setView("instructions");
                            setSelectedQuestionIndex(0);
                            setSelectedOptions({});
                            setSecondsLeft(examData.duration);
                        }}
                    >
                        Retake Test
                    </button>
                </div>
            </div>
        )}

    </div>
  )
 );
}

export default WriteExam;
```

16. loaderSlice.js

```
import { createSlice } from "@reduxjs/toolkit";

export const loaderSlice = createSlice({
  name: 'loader',
  initialState: {
    loading: false,
  },
  reducers: {
```

```
        ShowLoading(state) {
            state.loading = true;
        },
        HideLoading(state) {
            state.loading = false;
        }
    }
})

export const { ShowLoading, HideLoading } = loaderSlice.actions
export default loaderSlice.reducer
```

17. store.js

```
import loaderSlice from "./loaderSlice";
import { configureStore } from "@reduxjs/toolkit";
import usersSlice from "./usersSlice";

const store = configureStore({
    reducer: {
        users: usersSlice,
        loader: loaderSlice
    },
});

export default store;
```

18. usersSlice.js

```
import { createSlice } from '@reduxjs/toolkit';
const usersSlice = createSlice({
    name: 'users',
```

```
    initialState: {
        user: null
    },
    reducers: {
        SetUser: (state, action) => {
            state.user = action.payload
        }
    }
})

export const { SetUser } = usersSlice.actions
export default usersSlice.reducer
```

19. alignments.css

```
.h-screen{
    height: 100vh;
}
.h-100{
    height: 100%;
}
.h-75{
    height: 75%;
}
.h-50{
    height: 50%;
}
.h-25{
    height: 25%;
```

```
}

.w-screen{
    width: 100vw;
}
.w-100{
    width: 100%;
}
.w-75{
    width: 75%;
}
.w-50{
    width: 50%;
}
.w-25{
    width: 25%;
}

.w-400{
    width: 400px;
}

.flex{
    display: flex;
}
.flex-col{
    flex-direction: column;
```

```
}

.justify-center{
    justify-content: center;
}
.justify-between{
    justify-content: space-between;
}
.justify-start{
    justify-content: flex-start;
}
.justify-end{
    justify-content: flex-end;
}

.items-center{
    align-items: center;
}
.items-start{
    align-items: flex-start;
}
.items-end{
    align-items: flex-end;
}

.gap-1{
    gap: 10px;
```

```
}
.gap-2{
    gap: 20px;
}
.gap-3{
    gap: 30px;
}
.gap-4{
    gap: 40px;
}
.gap-5{
    gap: 50px;
}

.p-5{
    padding: 50px;
}
.p-4{
    padding: 40px;
}
.p-3{
    padding: 30px;
}
.p-2{
    padding: 20px;
}
.p-1{
```

```
    padding: 10px;
}

.mt-1{
    margin-top: 10px;
}
.mt-2{
    margin-top: 20px;
}
```

20. custom-components.css

```
.card {
    box-shadow: 0 0 1px gray;
    border-radius: 5px;
}

.card-lg {
    box-shadow: 0 0 2px rgb(27, 27, 27);
    border-radius: 5px;
    /* width: 400px; */
}

.divider {
    border-bottom: 1px solid gray;
    margin: 10px 0;
}

.loader-parent {
    position: absolute;
```

```
    inset: 0;
    background-color: rgba(0, 0, 0, 0.6);
    z-index: 10000;
    display: flex;
    justify-content: center;
    align-items: center;
}

.loader {
    height: 80px;
    width: 80px;
    border: 5px solid white;
    border-left-color: transparent;
    border-radius: 50%;
    animation: loader 1s linear infinite;
}

.option{
    box-shadow: 0px 0px 1.5px;
    padding: 10px;
    border-radius: 2px;
}

.selected-option{
    box-shadow: 0px 0px 1 px;
    padding: 10px;
    border-radius: 2px;
    border: 2px solid var(--primary);
}
.result{
```

```
    box-shadow: black 0px 0px 2px;
    padding: 15px;
    /* color: white !important; */
    border-radius: 5px;
    height: 470px;
}

.lottie-animation{
    height: 200px;
    padding: 30px;
}
.timer{
    background-color: var(--color);
    color: white !important;
    padding: 10px;
    display: flex;
    align-items: center;
    justify-content: center;
    height: 50px;
    width: 50px;
    border-radius: 50%;
}
@keyframes loader {
    0% {
        transform: rotate(0deg);
    }
    100% {
        transform: rotate(360deg);
    }
}
```

21. form-elements.css

```
input {
    height: 25px !important;
    padding: 5px 10px !important;
    border: 2px solid #676767 !important;
    width: 94%;
    border-radius: 7px !important;

}

select {
    height: 38px !important;
    padding: 5px 10px !important;
    border: 2px solid #676767 !important;
    width: 94%;
    border-radius: 2px !important;
}

.ant-form-item {
    margin-bottom: 5px !important;
}

.button {
    border-radius: 2px;
    padding: 0px 25px;
    height: 40px !important;
}

.primary-contained-btn {
    background-color: var(--color);
```

```
    color: white;
    padding: 10px 25px;
    border-color: var(--color);
    border-radius: 7px;
}
.primary-contained-btn-1{
    background-color: var(--primary);
    color: white;
    padding: 10px 80px;
    border-color: var(--primary);
    border-radius: 7px;
}

.primary-outlined-btn {
    background-color: white;
    color: var(--color);
    border: 2px solid var(--color);
    padding: 10px 25px;
    border-radius: 7px;
}
.primary-outlined-btn-1 {
    background-color: white;
    color: var(--primary);
    border: 2px solid var(--primary);
    padding: 10px 80px;
    border-radius: 7px;
}

.primary-contained-btn-2{
    background-color: var(--color);
```

```
    color: white;
    padding: 10px 80px;
    border-color: var(--color);
    border-radius: 7px;
}

.primary-outlined-btn-2{
    background-color: white;
    color: var(--color);
    border: 2px solid var(--color);
    padding: 10px 80px;
    border-radius: 7px;
}
```

22. layout.css

```
.layout {
    padding: 0px;
    height: 100;
    width: 100;
  }

  .sidebar {
    background-color: var(--primary);
    padding: 10px;
    padding-top: 250px;
    border-radius: 5px;
    height: 100;
    display: flex;
    align-items: flex-start;
    justify-content: center;
```

```
}

.menu-item {
  display: flex;
  align-items: center;
  padding: 10px;
  margin:5px;
  cursor: pointer;
  transition: all 0.2s ease-in-out;
  gap: 15px;
  color: white !important;
  /* margin-top: 100px; */
}

.active-menu-item{
    border: 2px solid white;
    border-radius: 5px;
}

.body {
  width: 100% !important;
}
.header {
  background-color: var(--primary);
  color: white !important;
  width: 97% !important;
  padding: 15px;
  border-radius: 5px;
  align-items: center !important;
}
```

23. textelements.css

```
.text-2xl{
    font-size: 1.5rem;
    line-height: 2rem;
}
.text-xl{
    font-size: 1.25rem;
    line-height: 1.75rem;
}
.text-lg{
    font-size: 1.25rem;
    line-height: 1.5rem;
}
.text-md{
    font-size: 1rem;
    line-height: 1.25rem;
}
.text-sm{
    font-size: 0.875rem;
    line-height: 1rem;
}
.text-xs{
    font-size: 0.75rem;
    line-height: 0.875rem;
}

h1{
    margin-bottom: 0px !important;
    padding-bottom:0 !important ;
}
```

```
.text-center{
    text-align: center;
}
a{
    color: var(--primary) !important;
}
a:hover{
    color: var(--primary) !important;
    text-decoration: underline !important;
}

i{
    font-size: 18px !important;
    cursor: pointer !important;
}

.underline{
    text-decoration: underline !important;
}
```

24. theme.css

```
:root {
    --primary: #19A7CE;
    --color: #B799FF;
    --secondary: #ff5722;
    --success: #B3FFAE;
    --info: #00bcd4;
    --warning: #ff9800;
    --dander: #FF6464;
    --light: #f1f1f1;
```

```
    --dark: #212121;
}

.bg-primary{
    background-color: var(--primary) !important;
}

.text.white{
    color: white !important;
}

.bg-success{
    background-color: var(--success);
}

.bg-white{
    background-color: white !important;
}

.bg-error{
    background-color: var(--dander);
}
```

25. index.css

```
@import
url('http://fonts.googleapis.com/css2?family=Montserrat:wght@500&display=swap');
* {
    font-family: 'Montserrat', sans-serif !important;
}
```

26. index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';
import 'antd/dist/antd.min.js';
import store from './redux/store';
import { Provider } from 'react-redux';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <Provider store={store}>
    <App />
  </Provider>
);
reportWebVitals();
```

ДОДАТОК Б. BackCode

1. server.js

```
const express = require("express");
const app = express();
require("dotenv").config();
app.use(express.json());
const dbConfig = require("./config/dbConfig");

const usersRoute = require("./routes/usersRoute");
const examsRoute = require("./routes/examsRoute");
const reportsRoute = require("./routes/reportsRoute");

app.use("/api/users", usersRoute);
app.use("/api/exams", examsRoute);
app.use("/api/reports", reportsRoute);

const port = process.env.PORT || 5000;
app.listen(port, () => {
  console.log(`Server listening on port ${port}`);
});
```

2. dbConfig.js

```
const mongoose = require('mongoose');

mongoose.connect(process.env.MONGO_URL)

const connection = mongoose.connection;

connection.on('connected', () => {

    console.log('Mongo DB Connection Successful');

})

connection.on('error', (err) => {

    console.log('Mongo DB Connection Faild');

})

module.exports = connection;
```

3. examModel.js

```
const mongoose = require('mongoose');

const examSchema = new mongoose.Schema({

    name: {
```

```
        type: String,
        required: true,
    },
    duration: {
        type: Number,
        required: true,
    },
    category: {
        type: String,
        required: true,
    },
    totalMarks: {
        type: Number,
        required: true,
    },
    passingMarks: {
        type: Number,
        required: true,
    },
    questions: {
        type: [mongoose.Schema.Types.ObjectId],
        ref: "questions",
        required: true,
    },
}, {
    timestamps: true,
});

const Exam = mongoose.model("exams", examSchema);
```

```
module.exports = Exam;
```

4. questionModel.js

```
const mongoose = require('mongoose');

const questionSchema = new mongoose.Schema({
    name: {
        type: String,
        require: true
    },
    correctOption: {
        type: String,
        require: true
    },
    options: {
        type: Object,
        require: true
    },
    exam: {
        type: mongoose.Schema.Types.ObjectId,
        ref: "exams",
    },

}, {
    timestamps: true,
});

const Question = mongoose.model("questions", questionSchema);
module.exports = Question;
```

5. reportModel.js

```
const mongoose = require('mongoose');


const reportSchema = new mongoose.Schema({
   user: {
      type: mongoose.Schema.Types.ObjectId,
      ref: "users"
   },
   exam: {
      type: mongoose.Schema.Types.ObjectId,
      ref: "exams",
   },
   result: {
      type: Object,
      required: true,
   },
},{
   timestamps: true,
});


const Report = mongoose.model("reports", reportSchema);


module.exports = Report;
```

6. userModel.js

```
const mongoose = require("mongoose");
const userSchema = new mongoose.Schema(
  {
    name: {
      type: String,
      required: true,
```

```
    },
    email: {
      type: String,
      required: true,
      unique: true,
    },
    password: {
      type: String,
      required: true,
    },
    isAdmin: {
      type: Boolean,
      default: false,
    },
  },
  {
    timestamps: true,
  }
);


const userModel = mongoose.model("users", userSchema);


module.exports = userModel;
```

7. examsRoute.js

```
const router = require("express").Router();
const Exam = require("../models/examModel");
const authMiddleware = require("../middlewares/authMiddleware");
const Question = require("../models/questionModel");

// add test
```

```
router.post("/add", authMiddleware, async (req, res) => {
    try {
        // check if exam already exists
        const examExists = await Exam.findOne({ name: req.body.name });
        if (examExists) {
            return res
                .status(200)
                .send({ message: "Exam already exists", success: false });
        }
        req.body.questions = [];
        const newExam = new Exam(req.body);
        await newExam.save();
        res.send({
            message: "Exam added successfully",
            success: true,
        });
    } catch (error) {
        res.status(500).send({
            message: error.message,
            data: error,
            success: false,
        });
    }
});

// get all tests
router.post("/get-all-exams", authMiddleware, async (req, res) => {
    try {
        const exams = await Exam.find({});
        res.send({
            message: "Exams fetched successfully",
            data: exams,
            success: true,
        });
    } catch (error) {
        res.status(500).send({
            message: error.message,
            data: error,
            success: false,
        });
    }
});

// get test by id
router.post("/get-exam-by-id", authMiddleware, async (req, res) => {
    try {
        const exam = await Exam.findById(req.body.examId).populate("questions");
```

```
        res.send({
            message: "Exam fetched successfully",
            data: exam,
            success: true,
        });
    } catch (error) {
        res.status(500).send({
            message: error.message,
            data: error,
            success: false,
        });

    }
})

// edit test by id
router.post("/edit-exam-by-id", authMiddleware, async (req, res) => {
    try {
        await Exam.findByIdAndUpdate(req.body.examId, req.body);
        res.send({
            message: "Exam edited successfully",
            success: true,
        });
    } catch (error) {
        res.status(500).send({
            message: error.message,
            data: error,
            success: false,
        });

    }
});

// delete test by id
router.post("/delete-exam-by-id", authMiddleware, async (req, res) => {
    try {
        await Exam.findByIdAndDelete(req.body.examId);
        res.send({
            message: "Exam deleted successfully",
            success: true,
        });
    } catch (error) {
        res.status(500).send({
            message: error.message,
            data: error,
            success: false,
        });
```

```
    }
});

// add question to test
router.post("/add-question-to-exam", authMiddleware, async (req, res) => {
    try {
        // add question to Questions collection
        const newQuestion = new Question(req.body);
        const question = await newQuestion.save();

        // add question to test
        const exam = await Exam.findById(req.body.exam);
        exam.questions.push(question._id);
        await exam.save();
        res.send({
            message: "Question added successfully",
            success: true,
        });
    } catch (error) {
        res.status(500).send({
            message: error.message,
            data: error,
            success: false,
        });
    }

});

// edit questions in test
router.post("/eddit-question-in-exam", authMiddleware, async (req, res) => {
    try {
        // edit question in Questions collection
        await Question.findByIdAndUpdate(req.body.questionId, req.body);
        res.send({
            message: "Question edited successfully",
            success: true,
        });
    } catch (error) {
        res.status(500).send({
            message: error.message,
            data: error,
            success: false,
        });
    }

});
```

```
// delete question in test
router.post("/delete-question-in-exam", authMiddleware, async (req, res) => {
   try {
      // delete question in Questions collection
      await Question.findByIdAndDelete(req.body.questionId);

      // delete question in test
      const exam = await Exam.findById(req.body.examId);
      exam.questions = exam.questions.filter(
         (question) => question._id != req.body.questionId
      );
      await exam.save();
      res.send({
         message: "Question deleted successfully",
         success: true,
      });
   } catch (error) {

   }});

module.exports = router;
```

8. reportsRoute.js

```
const authMiddleware = require("../middlewares/authMiddleware");
const Exam = require("../models/examModel");
const User = require("../models/userModel");
const Report = require("../models/reportModel");
const router = require("express").Router();

//add report

router.post("/add-report", authMiddleware, async (req, res) => {
   try {
      const newReport = new Report(req.body);
      await newReport.save();
      res.send({
         message: "Attempt added successfully",
         success: true,
      });
   } catch (error) {
      res.status(500).send({
         message: error.message,
         data: error,
         success: false,
      });
   }
});
```

```
//get all reports

router.post("/get-all-reports", authMiddleware, async (req, res) => {
    try {
        const { examName, userName } = req.body;
        const exams = await Exam.find({
            name: {
                $regex: examName,
            }
        });

        const matchedExamIds = exams.map((exam) => exam._id);

        const users = await User.find({
            name: {
                $regex: userName,
            },
        });

        const matchedUserIds = users.map((user) => user._id);

        const reports = await Report.find({
            exam: {
                $in: matchedExamIds,
            },
            user: {
                $in: matchedUserIds,
            },
        }).populate("exam").populate("user").sort({ createdAt: -1 });
        res.send({
            message: "Attempt fetched successfully",
            data: reports,
            success: true,
        });
    } catch (error) {
        res.status(500).send({
            message: error.message,
            data: error,
            success: false,
        });
    }
});

//get all reports by user
router.post("/get-all-reports-by-user", authMiddleware, async (req, res) => {
    try {
```

```
        const reports = await Report.find({ user: req.body.userId
}).populate("exam").populate("user").sort({ createdAt: -1 });
        res.send({
            message: "Attempt fetched successfully",
            data: reports,
            success: true,
        });
    } catch (error) {
        res.status(500).send({
            message: error.message,
            data: error,
            success: false,
        });
    }
});

module.exports = router;
```

9. usersRoute.js

```
const router = require('express').Router();
const User = require('../models/userModel');
const bcrypt = require('bcryptjs');
const jwt = require("jsonwebtoken");
const authMiddleware = require('../middlewares/authMiddleware');

// user registration

router.post("/register", async (req, res) => {
  try {
    // check if user already exists
    const userExists = await User.findOne({ email: req.body.email });
    if (userExists) {
      return res
        .status(200)
        .send({ message: "User already exists", success: false });
    }

    // hash password
    const salt = await bcrypt.genSalt(10);
    const hashedPassword = await bcrypt.hash(req.body.password, salt);
    req.body.password = hashedPassword;

    // create new user
    const newUser = new User(req.body);
    await newUser.save();
    res.send({
```

```
      message: "User created successfully",
      success: true,
    });
  } catch (error) {
    res.status(500).send({
      message: error.message,
      data: error,
      success: false,
    });
  }
});

// user login

router.post("/login", async (req, res) => {
  try {
    // check if user exists
    const user = await User.findOne({ email: req.body.email });
    if (!user) {
      return res
        .status(200)
        .send({ message: "User does not exist", success: false });
    }

    // check password
    const validPassword = await bcrypt.compare(
      req.body.password,
      user.password
    );
    if (!validPassword) {
      return res
        .status(200)
        .send({ message: "Invalid password", success: false });
    }

    const token = jwt.sign({ userId: user._id }, process.env.JWT_SECRET, {
      expiresIn: "1d",
    });

    res.send({
      message: "User logged in successfully",
      success: true,
      data: token,
    });
  } catch (error) {
    res.status(500).send({
      message: error.message,
```

```
      data: error,
      success: false,
    });
  }
});

// get user info
router.post("/get-user-info", authMiddleware, async (req, res) => {
  try {
    const user = await User.findById(req.body.userId);
    res.send({
      message: "User info fetched successfully",
      success: true,
      data: user,
    });
  } catch (error) {
    res.status(500).send({
      message: error.message,
      data: error,
      success: false,
    });
  }
});

module.exports = router;
```