\*NextAuth Configuration\*

`pages/api/[...nextauth].js`

```

jsx

import NextAuth from "next-auth";

import Providers from "next-auth/providers";

import { db } from "../../../lib/db";

export default NextAuth({

providers: [

Providers.CredentialsProvider({

name: "Credentials",

credentials: {

username: { label: "Username", type: "text" },

password: { label: "Password", type: "password" },

},

async authorize(credentials) {

const user = await db.getUser(credentials.username, credentials.password);

return user;

},

}),

],

database: process.env.DATABASE\_URL,

session: {

jwt: true,

},

jwt: {

secret: process.env.SECRET,

},

callbacks: {

async jwt(token, user, account, isNewUser) {

if (user.role) {

token.role = user.role;

}

return token;

},

async session(session, token) {

session.role = token.role;

return session;

},

},

});

```

\*Database Schema\*

`lib/db.js`

```

import sqlite3 from "sqlite3";

import { Database } from "sqlite3";

const db = new Database("financial.db");

db.serialize(() => {

db.run(`

CREATE TABLE IF NOT EXISTS users (

id INTEGER PRIMARY KEY AUTOINCREMENT,

username TEXT NOT NULL,

password TEXT NOT NULL,

role TEXT NOT NULL

);

`);

db.run(`

CREATE TABLE IF NOT EXISTS transactions (

id INTEGER PRIMARY KEY AUTOINCREMENT,

user\_id INTEGER NOT NULL,

amount REAL NOT NULL,

description TEXT NOT NULL,

status TEXT NOT NULL,

FOREIGN KEY (user\_id) REFERENCES users (id)

);

`);

});

export { db };

```

\*Transaction Submission Page\*

`pages/submit-transaction.js`

```

jsx

import { useSession } from "next-auth/client";

import { useMutation } from "react-query";

import { db } from "../../lib/db";

const SubmitTransaction = () => {

const { data: session } = useSession({ required: true });

const mutation = useMutation(

async (transaction) => {

await db.run(`

INSERT INTO transactions (user\_id, amount, description, status)

VALUES (?, ?, ?, ?);

`, [(link unavailable), transaction.amount, transaction.description, "pending"]);

},

{

onSuccess: () => {

alert("Transaction submitted successfully!");

},

onError: (error) => {

alert("Error submitting transaction: " + error.message);

},

}

);

const handleSubmit = (event) => {

event.preventDefault();

const transaction = {

amount: event.target.amount.value,

description: event.target.description.value,

};

mutation.mutate(transaction);

};

return (

<div>

<h1>Submit Transaction</h1>

<form onSubmit={handleSubmit}>

<label>

Amount:

<input type="number" name="amount" />

</label>

<br />

<label>

Description:

<input type="text" name="description" />

</label>

<br />

<button type="submit">Submit</button>

</form>

</div>

);

};

export default SubmitTransaction;

```

\*Transaction Approval Page\*

`pages/approve-transaction.js`

```

jsx

import { useSession } from "next-auth/client";

import { useQuery, useMutation } from "react-query";

import { db } from "../../lib/db";

import { Table } from "../../components/Table";

const ApproveTransaction = () => {

const { data: session } = useSession({ required: true });

const { data: transactions } = useQuery(

"transactions",

async () => {

const transactions = await db.all(`

SELECT \* FROM transactions

WHERE status = ?;

`, ["pending"]);

return transactions;

},

{

refetchInterval: 1000,

}

);

const mutation = useMutation(

async (transaction) => {

await db.run(`

UPDATE transactions

SET status = ?

WHERE id = ?;

`, ["approved", (link unavailable)]);

},

{

onSuccess: () => {

alert("Transaction approved successfully!");

},

onError: (error) => {

alert("Error approving transaction: " + error.message);

},

}

);

const handleApprove = (transaction) => {

mutation.mutate(transaction);

};

return (

<div>

<h1>Approve Transactions</h1>

<Table

```

\_Transaction Approval Page\_

```

jsx

import { useSession } from "next-auth/client";

import { useQuery, useMutation } from "react-query";

import { db } from "../../lib/db";

import { Table } from "../../components/Table";

const ApproveTransaction = () => {

const { data: session } = useSession({ required: true });

const { data: transactions } = useQuery(

"transactions",

async () => {

const transactions = await db.all(`

SELECT \* FROM transactions

WHERE status = ?;

`, ["pending"]);

return transactions;

},

{

refetchInterval: 1000,

}

);

const mutation = useMutation(

async (transaction) => {

await db.run(`

UPDATE transactions

SET status = ?

WHERE id = ?;

`, ["approved", (link unavailable)]);

},

{

onSuccess: () => {

alert("Transaction approved successfully!");

},

onError: (error) => {

alert("Error approving transaction: " + error.message);

},

}

);

const handleApprove = (transaction) => {

mutation.mutate(transaction);

};

return (

<div>

<h1>Approve Transactions</h1>

<Table

data={transactions}

columns={[

{

Header: "ID",

accessor: "id",

},

{

Header: "Amount",

accessor: "amount",

},

{

Header: "Description",

accessor: "description",

},

{

Header: "Status",

accessor: "status",

},

{

Header: "Actions",

Cell: ({ row }) => (

<button onClick={() => handleApprove(row.original)}>Approve</button>

),

},

]}

/>

</div>

);

};

export default ApproveTransaction;

```

\_Table Component\_

```

jsx

import React from "react";

import { useTable } from "@tanstack/react-table";

const Table = ({ data, columns }) => {

const { getTableProps, getTableBodyProps, headerGroups, rows, prepareRow } = useTable({

columns,

data,

});

return (

<table {...getTableProps()} className="table-auto w-full">

<thead>

{headerGroups.map((headerGroup) => (

<tr {...headerGroup.getHeaderGroupProps()} className="bg-gray-100">

{headerGroup.headers.map((column) => (

<th {...column.getHeaderProps()} className="px-4 py-2">

{column.render("Header")}

</th>

))}

</tr>

))}

</thead>

<tbody {...getTableBodyProps()}>

{rows.map((row) => {

prepareRow(row);

return (

<tr {...row.getRowProps()} className="even:bg-gray-50">

{row.cells.map((cell) => (

<td {...cell.getCellProps()} className="px-4 py-2">

{cell.render("Cell")}

</td>

))}

</tr>

);

})}

</tbody>

</table>

);

};

export default Table;

```

\_Role-Based Access Control\_

```

jsx

import { hasPermission } from "../utils/roles";

const SubmitTransaction = () => {

const { data: session } = useSession({ required: true });

if (!hasPermission(session.role, "submit-transaction")) {

return <p>Access denied.</p>;

}

// ...

};

const ApproveTransaction = () => {

const { data: session } = useSession({ required: true });

if (!hasPermission(session.role, "approve-transaction")) {

return <p>Access denied.</p>;

}

// ...

};

```

\*Permissions\*

| Role | Permissions |

|---------|---------------------|

| admin | submit-transaction, approve-transaction |

| manager | approve-transaction |

| user | submit-transaction |

\*Pages\*

| Page | Required Role |

|-----------|----------------|

| /submit-transaction | admin, user |

| /approve-transaction | admin, manager |

\_utils/roles.js\_

```

const roles = {

admin: ["submit-transaction", "approve-transaction"],

manager: ["approve-transaction"],

user: ["submit-transaction"],

};

export const hasPermission = (role, permission) => {

return roles[role].includes(permission);

};

```

\_components/TransactionTable.js\_

```

jsx

import React from "react";

import { useTable } from "@tanstack/react-table";

import { Table } from "../../components/Table";

const TransactionTable = ({ transactions }) => {

const columns = [

{

Header: "ID",

accessor: "id",

},

{

Header: "Amount",

accessor: "amount",

},

{

Header: "Description",

accessor: "description",

},

{

Header: "Status",

accessor: "status",

},

];

return (

<Table

data={transactions}

columns={columns}

/>

);

};

export default TransactionTable;

```

\_pages/transactions.js\_

```

jsx

import React from "react";

import { useQuery } from "react-query";

import { db } from "../../lib/db";

import TransactionTable from "../../components/TransactionTable";

const Transactions = () => {

const { data: transactions } = useQuery(

"transactions",

async () => {

const transactions = await db.all("SELECT \* FROM transactions");

return transactions;

},

{

refetchInterval: 1000,

}

);

return (

<div>

<h1>Transactions</h1>

<TransactionTable transactions={transactions} />

</div>

);

};

export default Transactions;

```

This code sets up the roles and permissions, creates a transaction table component, and displays all transactions on the transactions page.

\_pages/approve-transaction.js\_

```

jsx

jsx

import React, { useState } from "react";

import { useQuery, useMutation } from "react-query";

import { db } from "../../lib/db";

import TransactionTable from "../../components/TransactionTable";

const ApproveTransaction = () => {

const { data: transactions } = useQuery(

"transactions",

async () => {

const transactions = await db.all("SELECT \* FROM transactions WHERE status = ?", ["pending"]);

return transactions;

},

{

refetchInterval: 1000,

}

);

const [selectedTransaction, setSelectedTransaction] = useState(null);

const mutation = useMutation(

async (transaction) => {

await db.run("UPDATE transactions SET status = ? WHERE id = ?", ["approved", (link unavailable)]);

},

{

onSuccess: () => {

alert("Transaction approved successfully!");

},

onError: (error) => {

alert("Error approving transaction: " + error.message);

},

}

);

const handleApprove = (transaction) => {

setSelectedTransaction(transaction);

mutation.mutate(transaction);

};

return (

<div>

<h1>Approve Transactions</h1>

<TransactionTable transactions={transactions} onApprove={handleApprove} />

{selectedTransaction && (

<div>

<p>Approving transaction {(link unavailable)}...</p>

</div>

)}

</div>

);

};

export default ApproveTransaction;

```

\_components/TransactionTable.js (updated)\_

```

jsx

jsx

import React from "react";

import { useTable } from "@tanstack/react-table";

const TransactionTable = ({ transactions, onApprove }) => {

const columns = [

{

Header: "ID",

accessor: "id",

},

{

Header: "Amount",

accessor: "amount",

},

{

Header: "Description",

accessor: "description",

},

{

Header: "Status",

accessor: "status",

},

{

Header: "Actions",

Cell: ({ row }) => (

<button onClick={() => onApprove(row.original)}>Approve</button>

),

},

];

return (

<Table

data={transactions}

columns={columns}

/>

);

};

export default TransactionTable;

```

This code sets up the approve transaction page, which displays a table of pending transactions and allows administrators to approve them.

\_pages/reports.js\_

```

jsx

import React from "react";

import { useQuery } from "react-query";

import { db } from "../../lib/db";

import ReportTable from "../../components/ReportTable";

const Reports = () => {

const { data: reports } = useQuery(

"reports",

async () => {

const reports = await db.all("SELECT \* FROM reports");

return reports;

},

{

refetchInterval: 1000,

}

);

return (

<div>

<h1>Reports</h1>

<ReportTable reports={reports} />

</div>

);

};

export default Reports;

```

\_components/ReportTable.js\_

```

jsx

import React from "react";

import { useTable } from "@tanstack/react-table";

const ReportTable = ({ reports }) => {

const columns = [

{

Header: "ID",

accessor: "id",

},

{

Header: "Transaction ID",

accessor: "transaction\_id",

},

{

Header: "User ID",

accessor: "user\_id",

},

{

Header: "Report Type",

accessor: "report\_type",

},

{

Header: "Description",

accessor: "description",

},

];

return (

<Table

data={reports}

columns={columns}

/>

);

};

export default ReportTable;

```

\_lib/db.js (updated)\_

```

import sqlite3 from "sqlite3";

import { Database } from "sqlite3";

const db = new Database("financial.db");

db.serialize(() => {

db.run(`

CREATE TABLE IF NOT EXISTS reports (

id INTEGER PRIMARY KEY AUTOINCREMENT,

transaction\_id INTEGER NOT NULL,

user\_id INTEGER NOT NULL,

report\_type TEXT NOT NULL,

description TEXT NOT NULL,

FOREIGN KEY (transaction\_id) REFERENCES transactions (id),

FOREIGN KEY (user\_id) REFERENCES users (id)

);

`);

});

export { db };

```

This code sets up the reports page, which displays a table of reports.

\_pages/report-transaction.js\_

```

jsx

import React, { useState } from "react";

import { useMutation } from "react-query";

import { db } from "../../lib/db";

const ReportTransaction = () => {

const [transactionId, setTransactionId] = useState("");

const [reportType, setReportType] = useState("");

const [description, setDescription] = useState("");

const mutation = useMutation(

async (report) => {

await db.run(`

INSERT INTO reports (transaction\_id, user\_id, report\_type, description)

VALUES (?, ?, ?, ?);

`, [report.transactionId, report.userId, report.reportType, report.description]);

},

{

onSuccess: () => {

alert("Report submitted successfully!");

},

onError: (error) => {

alert("Error submitting report: " + error.message);

},

}

);

const handleSubmit = (event) => {

event.preventDefault();

const report = {

transactionId,

userId: 1, // Replace with actual user ID

reportType,

description,

};

mutation.mutate(report);

};

return (

<div>

<h1>Report Transaction</h1>

<form onSubmit={handleSubmit}>

<label>

Transaction ID:

<input type="number" value={transactionId} onChange={(e) => setTransactionId(e.target.value)} />

</label>

<br />

<label>

Report Type:

<select value={reportType} onChange={(e) => setReportType(e.target.value)}>

<option value="">Select report type</option>

<option value="fraud">Fraud</option>

<option value="dispute">Dispute</option>

</select>

</label>

<br />

<label>

Description:

<textarea value={description} onChange={(e) => setDescription(e.target.value)} />

</label>

<br />

<button type="submit">Submit Report</button>

</form>

</div>

);

};

export default ReportTransaction;

```

\_components/ReportForm.js\_

```

jsx

import React from "react";

const ReportForm = () => {

return (

<form>

<label>

Transaction ID:

<input type="number" />

</label>

<br />

<label>

Report Type:

<select>

<option value="">Select report type</option>

<option value="fraud">Fraud</option>

<option value="dispute">Dispute</option>

</select>

</label>

<br />

<label>

Description:

<textarea />

</label>

<br />

<button type="submit">Submit Report</button>

</form>

);

};

export default ReportForm;

```

This code sets up the report transaction page, which allows users to submit reports.

\*\_pages/user-transactions.js\_\*

```

jsx

import React from "react";

import { useQuery } from "react-query";

import { db } from "../../lib/db";

import TransactionTable from "../../components/TransactionTable";

const UserTransactions = () => {

const { data: transactions } = useQuery(

"transactions",

async () => {

const transactions = await db.all("SELECT \* FROM transactions WHERE user\_id = ?", [1]); // Replace with actual user ID

return transactions;

},

{

refetchInterval: 1000,

}

);

return (

<div>

<h1>My Transactions</h1>

<TransactionTable transactions={transactions} />

</div>

);

};

export default UserTransactions;

```

\*\_components/TransactionDetails.js\_\*

```

jsx

import React from "react";

const TransactionDetails = ({ transaction }) => {

return (

<div>

<h2>Transaction Details</h2>

<p>ID: {(link unavailable)}</p>

<p>Amount: {transaction.amount}</p>

<p>Description: {transaction.description}</p>

<p>Status: {transaction.status}</p>

</div>

);

};

export default TransactionDetails;

```

\*\_pages/transaction-details.js\_\*

```

jsx

import React from "react";

import { useQuery } from "react-query";

import { db } from "../../lib/db";

import TransactionDetails from "../../components/TransactionDetails";

const TransactionDetailsPage = () => {

const { data: transaction } = useQuery(

"transaction",

async () => {

const transaction = await db.get("SELECT \* FROM transactions WHERE id = ?", [1]); // Replace with actual transaction ID

return transaction;

},

{

refetchInterval: 1000,

}

);

return (

<div>

<TransactionDetails transaction={transaction} />

</div>

);

};

export default TransactionDetailsPage;

```

\*\_lib/auth.js\_\*

```

import { NextAuth } from "next-auth";

const authOptions = {

// Configure authentication providers

providers: [

NextAuth.Providers.Credentials({

name: "Credentials",

credentials: {

username: { label: "Username", type: "text" },

password: { label: "Password", type: "password" },

},

async authorize(credentials) {

// Add authentication logic here

},

}),

],

// Configure database

database: process.env.DATABASE\_URL,

// Configure session

session: {

jwt: true,

},

// Configure JWT

jwt: {

secret: process.env.SECRET,

},

};

export default authOptions;

```

These code snippets enhance the system by:

1. Displaying user transactions

2. Showing transaction details

3. Implementing authentication

\*Next.js Pages\*

`pages/index.js`

```

jsx

import Link from "next/link";

const Home = () => {

return (

<div>

<h1>Welcome to Financial Transaction System</h1>

<ul>

<li>

<Link href="/submit-transaction">

<a>Submit Transaction</a>

</Link>

</li>

<li>

<Link href="/approve-transaction">

<a>Approve Transaction</a>

</Link>

</li>

<li>

<Link href="/reports">

<a>Reports</a>

</Link>

</li>

</ul>

</div>

);

};

export default Home;

```

`pages/submit-transaction.js`

```

jsx

import React, { useState } from "react";

import { useMutation } from "react-query";

import { db } from "../../lib/db";

const SubmitTransaction = () => {

const [amount, setAmount] = useState(0);

const [description, setDescription] = useState("");

const mutation = useMutation(

async (transaction) => {

await db.run(`

INSERT INTO transactions (amount, description, status)

VALUES (?, ?, ?);

`, [transaction.amount, transaction.description, "pending"]);

},

{

onSuccess: () => {

alert("Transaction submitted successfully!");

},

onError: (error) => {

alert("Error submitting transaction: " + error.message);

},

}

);

const handleSubmit = (event) => {

event.preventDefault();

const transaction = {

amount,

description,

};

mutation.mutate(transaction);

};

return (

<div>

<h1>Submit Transaction</h1>

<form onSubmit={handleSubmit}>

<label>

Amount:

<input type="number" value={amount} onChange={(e) => setAmount(e.target.value)} />

</label>

<br />

<label>

Description:

<textarea value={description} onChange={(e) => setDescription(e.target.value)} />

</label>

<br />

<button type="submit">Submit Transaction</button>

</form>

</div>

);

};

export default SubmitTransaction;

```

`pages/approve-transaction.js`

```

jsx

import React from "react";

import { useQuery, useMutation } from "react-query";

import { db } from "../../lib/db";

import TransactionTable from "../../components/TransactionTable";

const ApproveTransaction = () => {

const { data: transactions } = useQuery(

"transactions",

async () => {

const transactions = await db.all("SELECT \* FROM transactions WHERE status = ?", ["pending"]);

return transactions;

},

{

refetchInterval: 1000,

}

);

const mutation = useMutation(

async (transaction) => {

await db.run(`

UPDATE transactions

SET status = ?

WHERE id = ?;

`, ["approved", (link unavailable)]);

},

{

onSuccess: () => {

alert("Transaction approved successfully!");

},

onError: (error) => {

alert("Error approving transaction: " + error.message);

},

}

);

const handleApprove = (transaction) => {

mutation.mutate(transaction);

};

return (

<div>

<h1>Approve Transactions</h1>

<TransactionTable transactions={transactions} onApprove={handleApprove} />

</div>

);

};

export default ApproveTransaction;

```

\*Components\*

`components/TransactionTable.js`

```

jsx

import React from "react";

import { useTable } from "@tanstack/react-table";

const TransactionTable = ({ transactions, onApprove }) => {

const columns = [

{

Header: "ID",

accessor: "id",

},

{

Header: "Amount",

accessor: "amount",

},

{

Header: "Description",

accessor: "description",

},

{

Header: "Status",

accessor: "status",

},

{

Header: "Actions",

Cell: ({ row }) => (

<button onClick={() => onApprove(row.original)}>Approve</button>

),

},

];

return (

<Table

data={transactions}

columns={columns}

/>

);

};

export default TransactionTable;

```

\*Database\*

`lib/db.js`

```

import sqlite3 from "sqlite3";

import { Database } from "sqlite3";

const db = new Database("financial.db");

db.serialize(() => {

db.run(`

CREATE TABLE IF NOT EXISTS transactions (

id INTEGER PRIMARY KEY AUTOINCREMENT,

amount REAL NOT NULL

```

`lib/db.js`

```

import sqlite3 from "sqlite3";

import { Database } from "sqlite3";

const db = new Database("financial.db");

db.serialize(() => {

db.run(`

CREATE TABLE IF NOT EXISTS transactions (

id INTEGER PRIMARY KEY AUTOINCREMENT,

amount REAL NOT NULL,

description TEXT NOT NULL,

status TEXT NOT NULL,

created\_at DATETIME DEFAULT CURRENT\_TIMESTAMP

);

`);

db.run(`

CREATE TABLE IF NOT EXISTS reports (

id INTEGER PRIMARY KEY AUTOINCREMENT,

transaction\_id INTEGER NOT NULL,

user\_id INTEGER NOT NULL,

report\_type TEXT NOT NULL,

description TEXT NOT NULL,

created\_at DATETIME DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (transaction\_id) REFERENCES transactions (id),

FOREIGN KEY (user\_id) REFERENCES users (id)

);

`);

});

export { db };

```

`lib/auth.js`

```

import NextAuth from "next-auth";

const authOptions = {

providers: [

NextAuth.Providers.Credentials({

name: "Credentials",

credentials: {

username: { label: "Username", type: "text" },

password: { label: "Password", type: "password" },

},

async authorize(credentials) {

// Add authentication logic here

},

}),

],

database: process.env.DATABASE\_URL,

session: {

jwt: true,

},

jwt: {

secret: process.env.SECRET,

},

};

export default authOptions;

```

`pages/\_app.js`

```

jsx

import { Provider } from "react-redux";

import store from "../store";

import { SessionProvider } from "next-auth/react";

function MyApp({ Component, pageProps }) {

return (

<SessionProvider>

<Provider store={store}>

<Component {...pageProps} />

</Provider>

</SessionProvider>

);

}

export default MyApp;

```

`store.js`

```

import { createStore, combineReducers } from "redux";

import transactionReducer from "./transactionReducer";

const rootReducer = combineReducers({

transactions: transactionReducer,

});

const store = createStore(rootReducer);

export default store;

```

`transactionReducer.js`

```

const initialState = [];

const transactionReducer = (state = initialState, action) => {

switch (action.type) {

case "ADD\_TRANSACTION":

return [...state, action.transaction];

case "UPDATE\_TRANSACTION":

return state.map((transaction) =>

(link unavailable) === (link unavailable) ? action.transaction : transaction

);

default:

return state;

}

};

export default transactionReducer;

```