

## ABSTRACT

This project presents a comprehensive system designed to manage and optimize the production workflow of composite metal foams (CMF), focusing on secure data handling, resource analysis, design support, manufacturing, and quality control. The system integrates five critical modules—Admin, Resource Analyst, Design Support, Production & Assembly, and Quality Control—to ensure that sensitive data related to material properties and production processes is securely uploaded, processed, and shared across departments. Each module plays a crucial role in maintaining the integrity, accuracy, and confidentiality of data, utilizing encryption and controlled access to facilitate seamless collaboration across the production pipeline.

The Admin Module acts as the system's foundation, facilitating the secure upload of CSV datasets containing essential material details such as density, tensile strength, porosity, and thermal conductivity. This data is then made available to other modules after encryption, with the Admin also overseeing employee onboarding and inter-module report management. Once decrypted, the Resource Analyst Module meticulously evaluates material requirements, performing calculations on density, mass, volume, and other key parameters. This ensures optimal material utilization and prepares datasets for downstream design and production stages.

The Design Support Module focuses on engineering calculations essential for CMF structural integrity. By decrypting data from the Resource Analyst Module, this module computes material strength metrics such as tensile, yield, and flexural strength. The module also collaborates with the Production & Assembly team by providing critical design specifications and welding parameters. These include calculations for heat input, cooling time, and welding strength, all of which are vital to achieving structurally sound and high-performance CMF components.

Finally, the Quality Control Module validates the entire production process by rigorously testing material samples for Young's Modulus, corrosion resistance, tensile strength, and weight efficiency. This step ensures the CMF products meet industry standards before deployment in demanding applications such as aerospace and automotive sectors. Throughout the system, secure communication and encrypted data exchanges safeguard sensitive production information, creating a robust, efficient, and reliable platform for CMF production and quality assurance.

## **CHAPTER – 1**

### **INTRODUCTION**

The manufacturing of composite metal foams (CMFs) plays a significant role in industries that require materials with superior mechanical properties, such as aerospace, automotive, and defense. CMFs are valued for their high strength-to-weight ratio, excellent energy absorption, and thermal resistance, making them ideal for demanding engineering applications. However, the production and quality control of CMFs require an intricate and secure workflow, as the materials and processes involved are highly sensitive and demand precise management. This project addresses this complexity by developing a modular system that handles everything from secure data entry to production oversight and quality testing, ensuring that every stage of the CMF production lifecycle is systematically managed and optimized.

The project is structured into five interconnected modules: Admin, Resource Analyst, Design Support, Production & Assembly, and Quality Control. Each module performs a specialized function, contributing to the seamless flow of encrypted data and secure communication between teams. The Admin Module is responsible for securely uploading product information and managing employee access. The Resource Analyst Module focuses on material resource analysis and preparation, ensuring precise calculations for density, mass, volume, and other material metrics. The Design Support Module then takes these insights to perform advanced engineering calculations for tensile strength, yield strength, and flexural strength, as well as providing welding parameters essential for production. This modular approach ensures that only authorized teams can access and process sensitive production data.

Additionally, the Production & Assembly Module oversees the manufacturing and welding of CMF components, ensuring that welding parameters and production specifications are accurately calculated and securely recorded. The final stage, managed by the Quality Control Module, involves rigorous testing of the produced CMF to ensure compliance with industry standards. Throughout the entire system, encryption protocols and secure data handling practices are enforced to protect intellectual property and critical production data. This project not only optimizes production processes but also enhances the security, accuracy, and reliability of CMF manufacturing, providing a robust solution tailored to the challenges faced by modern engineering and material science industries.

## CHAPTER – 2

### SYSTEM REQUIREMENTS

#### HARDWARE REQUIREMENTS:

- **Processor** : Intel Core i5 or higher / AMD Ryzen 5 or higher
- **RAM** : Minimum 8 GB (16 GB recommended for smooth multitasking and blockchain simulations)
- **Storage** : At least 250 GB SSD (for faster read/write operations and better performance)
- **Display** : 13-inch or larger monitor with minimum 1080p resolution
- **Network**: Stable internet connection for blockchain interactions and web-based operations
- **Graphics**: Integrated graphics sufficient (dedicated GPU recommended for heavy frontend development, but not mandatory)

#### SOFTWARE REQUIREMENTS:

- **Operating System**: Windows 10/11, macOS, or Linux (Ubuntu preferred for blockchain development)
- **Backend**:
  - Node.js (v14.x or higher)
  - Express.js (v4.x or higher)
  - MongoDB (v5.x or higher)
- **Blockchain**:
  - Ganache (for local blockchain network simulation)
  - Truffle Suite (for smart contract development, testing, and deployment)
- **Frontend**:
  - React (v18.x or higher)
- **Development Tools**:
  - Visual Studio Code or any preferred IDE
  - Postman (for API testing)
  - Git (for version control)
  - Browser (Google Chrome, Mozilla Firefox, or equivalent)

## **CHAPTER – 3**

### **METHODOLOGY**

#### **1. System Design**

The project is developed using a modular system design approach where each module operates independently while being securely connected to others. The entire system is divided into five modules: Admin, Resource Analyst, Design Support, Production & Assembly, and Quality Control. Each module is responsible for specific functionalities and interacts with other modules through encrypted data transfers and blockchain-based access management. This ensures data integrity, traceability, and secure communication throughout the workflow.

#### **2. Backend and Blockchain Development**

The backend is built using Node.js and Express.js, providing RESTful APIs for handling data operations, user authentication, and inter-module communication. MongoDB is used as the database to securely store encrypted material datasets and operational records. Blockchain smart contracts, developed using Truffle and deployed on Ganache, manage encryption keys, track data access events, and ensure decentralized data control. This setup guarantees that sensitive data is only accessible to authorized users based on smart contract permissions.

#### **3. Frontend Development**

The frontend is developed using React.js, providing user-friendly dashboards for Admins, Resource Analysts, Designers, Production teams, and Quality Control personnel. Each user role has a dedicated interface with secured login features. React interacts with backend APIs to fetch and display data dynamically. MetaMask integration is used to facilitate secure blockchain interactions directly from the frontend, allowing users to request or approve decryption keys through smart contract transactions.

#### **4. Data Flow & Workflow Execution**

The system starts with the Admin uploading CSV datasets containing product details, which are encrypted and made accessible to other modules via blockchain-based key requests. The Resource Analyst decrypts and analyzes the data to compute resource metrics. The Design Support team then calculates material strength and welding parameters. Subsequently, the Production & Assembly team handles manufacturing and uploads welding

## CHAPTER – 4

### REQUIRED PACKAGES

#### 1. Backend Packages (Node.js + Express + MongoDB)

- **express**: Handles routing and server creation for RESTful APIs.
- **mongoose**: Connects and interacts with MongoDB for database operations.
- **jsonwebtoken**: Manages authentication using JSON Web Tokens (JWT).
- **bcrypt** / **bcryptjs**: Used for securely hashing passwords.
- **dotenv**: Loads environment variables from a `.env` file.
- **multer**: Handles file uploads (e.g., CSV).
- **csv-parser**: Parses CSV files for material data ingestion.
- **cookie-parser**: Parses cookies for session management.
- **nodemailer**: Sends automated emails (e.g., user credentials).
- **cors**: Enables Cross-Origin Resource Sharing for secure API consumption.
- **web3**: Facilitates blockchain interactions from the backend.
- **nodemon (devDependency)**: Automatically restarts the server during development.

#### 2. Blockchain Packages (Ganache + Truffle)

- **ganache**: A local in-memory blockchain for testing and development.
- **truffle**: A development framework for compiling, deploying, and testing smart contracts.

#### 3. Frontend Packages (React + Vite)

- **react** / **react-dom**: Core libraries for building the frontend user interface.
- **react-router-dom**: Enables routing and navigation within the React app.
- **axios**: Used to make HTTP requests to backend APIs.
- **framer-motion**: Provides animations and transitions for a smoother user experience.
- **@shadcn/ui**, **tailwindcss**, **tailwindcss-animate**: Used for modern UI components and styling.
- **recharts** / **react-chartjs-2** / **chart.js**: For data visualization and dashboard charts.
- **pdfmake**: Generates PDF reports from the frontend.
- **react-icons** / **lucide-react**: Provides a wide collection of icons.
- **date-fns**: Simplifies date formatting and manipulation.
- **react-paginate**: Implements pagination components.
- **eslint** + **plugins**: Ensures code quality and enforces coding standards.
- **vite**: A fast build tool and development server.
- **nodemon** (used in frontend scripts): Watches for Vite configuration changes during development.

## CHAPTER – 5

### SYSTEM DESIGN

#### ARCHITECTURE :

##### 1)Frontend Layer

The **Frontend** is designed using **React.js**, a modern JavaScript library for building interactive user interfaces. It provides individual dashboards for different user roles such as Admin, Resource Analyst, Design Support, Production & Assembly, and Quality Control teams. React handles routing using **React Router**, making navigation between different modules smooth and dynamic. The frontend interacts with backend APIs via **Axios** over secure HTTPS protocols. In addition, the frontend is integrated with **MetaMask**, allowing users to interact directly with blockchain smart contracts when requesting decryption keys or approving blockchain transactions. The frontend also includes visualization tools (e.g., **Recharts**, **Chart.js**) for presenting material metrics and quality control results in a user-friendly way.

##### 2)Backend Layer

The **Backend** is developed using **Node.js** with the **Express.js** framework, enabling the development of lightweight and high-performance RESTful APIs. This layer serves as the core of the application, performing essential operations like user authentication, business logic execution, file processing (CSV file parsing with **csv-parser** and **multer**), and encryption/decryption of material data using cryptographic libraries. The backend also manages communication between the frontend and the blockchain layer. It ensures that each API endpoint is secure, utilizing **JWT (JSON Web Tokens)** for authentication and **CORS** for secure cross-origin resource sharing. Processed data is securely stored in **MongoDB**, a NoSQL database that supports fast querying, indexing, and scalability for handling complex material datasets.

##### 3)Blockchain Integration Layer

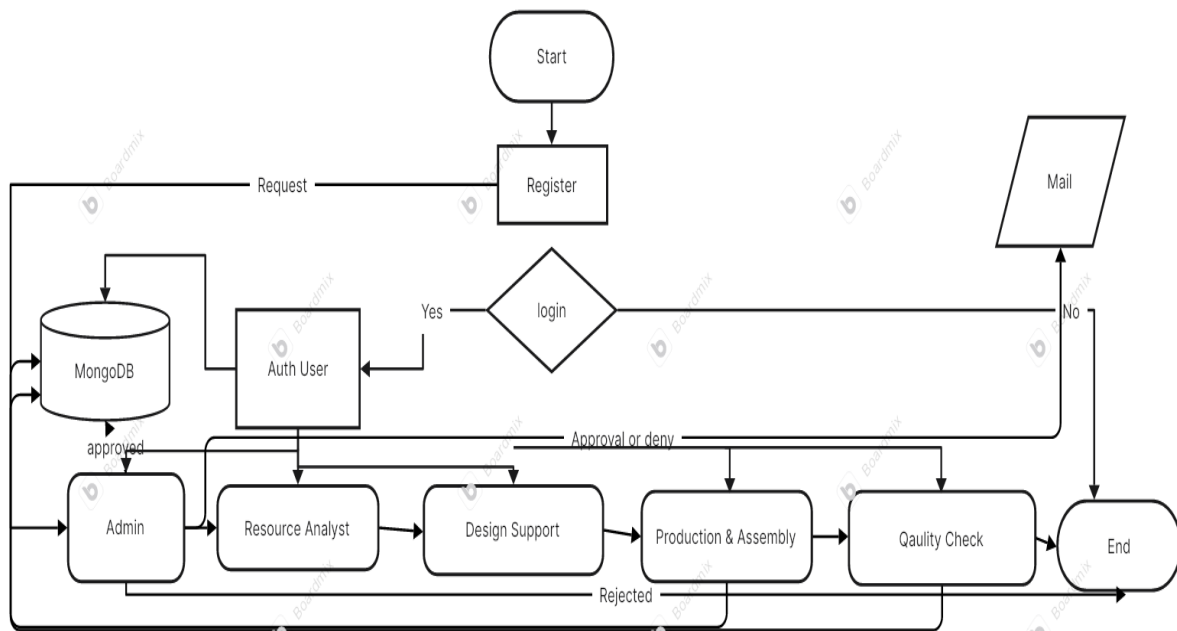
The **Blockchain Layer** is implemented using **Truffle Suite** and deployed locally on **Ganache**, a personal Ethereum blockchain simulator. This layer is responsible for managing decryption key generation, key access requests, and maintaining an immutable audit trail of all transactions between modules. Custom smart contracts handle the issuance of encryption keys and log every key access or approval event, ensuring that critical actions are recorded transparently. By using **Web3.js**, the backend securely interacts with these smart contracts, automating key authorization workflows and strengthening trust in the system's operations.

## DATA FLOW:

The data flow in the system is designed to maintain security while ensuring smooth operations:

- **Step 1:** Admin uploads CSV files → Data is encrypted and stored in MongoDB.
- **Step 2:** Resource Analyst requests a decryption key → Blockchain validates and provides access.
- **Step 3:** Resource Analyst processes data → Uploads re-encrypted data.
- **Step 4:** Design Support decrypts → Calculates strength & welding metrics → Uploads encrypted design data.
- **Step 5:** Production & Assembly decrypts → Performs welding → Uploads production metrics.
- **Step 6:** Quality Control decrypts → Performs testing → Uploads final test results.
- **Step 7:** Admin downloads consolidated reports.

## FLOW CHART:



## SECURITY:

- **Data Encryption:** All datasets are encrypted before storage and sharing between modules.
- **Blockchain Key Management:** Smart contracts manage decryption key requests and ensure that only authorized modules can access sensitive data.
- **Role-Based Access Control:** Different user roles (Admin, Analyst, Designer, Production, QC) have distinct access permissions.
- **Secure API Communication:** Backend and frontend communicate over HTTPS with token-based authentication using JWT.
- **Audit Trail via Blockchain:** All key requests and approvals are logged on-chain to ensure traceability and transparency.

## FUTURE ENHANCEMENT:

- **Smart Contract Automation:** Automate more decision-making processes using advanced smart contracts.
- **Multi-chain Support:** Expand beyond Ganache to deploy smart contracts on testnets like Polygon or Ethereum.
- **AI-Powered Analysis:** Integrate AI models to automate material quality assessments and predictive maintenance.
- **Microservices Conversion:** Convert monolithic backend services into microservices for better scalability.
- **Advanced User Analytics:** Add analytics dashboards to monitor workflow efficiency, resource usage, and performance trends.



## CHAPTER – 6

### SOURCE CODE

#### FRONTEND

##### MAIN.JSX

```
import { createRoot } from 'react-dom/client'

import { BrowserRouter } from 'react-router-dom'

import './index.css'

import App from './App.jsx'

import "bootstrap/dist/css/bootstrap.min.css";
import "bootstrap/dist/js/bootstrap.bundle.min.js";

import { LoaderProvider } from "../context/LoaderContext";

import GlobalLoader from "../components/GlobalLoader";

import './index.css';

createRoot(document.getElementById("root")).render(

  <LoaderProvider>

  <BrowserRouter>

  <App />

  </BrowserRouter>

  <GlobalLoader />

  </LoaderProvider>

)
```

##### APP.JSX

```
import './App.css';

import AppRoutes from "../Router/AppRoutes";

import { useLocation } from "react-router-dom";

import { useEffect } from "react";

import { AuthProvider } from '../context/AuthContext';

import { Toaster } from "sonner";

function App() {

  const location = useLocation();

  const backgroundColors = {

    "/": "#232A58",

    "/register": "#232A58",

    "/change-password": "#232A58",

    "/adminHome": "#6286AA",

    "/adminProduct": "#6286AA",

    "/adminUserManagement": "#6286AA",

    "/adminReports": "#6286AA",

    "/adminRejectedProducts": "#6286AA",

  }
```

```

"/adminDataAccess": "#6286AA",
'/raHome': '#CD7903',
'/raProduct': '#CD7903',
'/raAnalysis': '#CD7903',
'/raAnalysisReport': '#CD7903',
'/raDataAccess': '#CD7903',
'/dsHome': '#09B478',
'/dsProduct': '#09B478',
'/dsAnalysis': '#09B478',
'/dsAnalysisReport': '#09B478',
'/dsDataAccess': '#09B478',
'/prHome': '#6B6C6C',
"/prProduction": '#6B6C6C',
"/prProgress": '#6B6C6C',
"/prQualityCheck": '#6B6C6C',
"/prReport": '#6B6C6C',
"/qcHome": "#02542D",
"/qcQualityCheck": "#02542D",
"/qcProgress": "#02542D",
"/qcReport": "#02542D"
};

useEffect(() => {
  document.body.style.backgroundColor = backgroundColors[location.pathname] || "#232A58";
  return () => {
    document.body.style.backgroundColor = "";
  };
}, [location.pathname]);
return (
  <div >
    <Toaster position="top-right" />
    <AuthProvider>
      <AppRoutes />
    </AuthProvider>
  </div>
);
}
export default App;

```

**APPROUTER.JSX**

```

import { Routes, Route } from "react-router-dom";
import LandingPage from '../pages/LandingPage/LandingPage';
import AdminHomePage from '../pages/AdminHomePage/AdminHomePage';
import ProtectedRoute from '../utils/ProtectedRoute';
import RegistrationPage from '../pages/RegistrationPage/RegistrationPage';
import AdminProductPage from '../pages/AdminProductPage/AdminProductPage';

```

```

import AdminUserManagement from "../pages/AdminUserManagement/AdminUserManagement";
import AdminReportsPage from "../pages/AdminReportsPage/AdminReportsPage";
import AdminRejectedPage from "../pages/AdminRejectedPage/AdminRejectedPage";
import AdminDataAccessPage from "../pages/AdminDataAccessPage/AdminDataAccessPage";
import ChangePasswordPage from "../pages/ChangePasswordPage/ChangePasswordPage";
import RAHome from "../pages/RAHome/RAHome";
import RAProductPage from "../pages/RAProductPage/RAProductPage";
import RAAnalysisPage from "../pages/RAAnalysisPage/RAAnalysisPage";
import RAAnalysisReportPage from "../pages/RAAnalysisReportPage/RAAnalysisReportPage";
import RADataAccessPage from "../pages/RADataAccessPage/RADataAccessPage";
import DSHomePage from "../pages/DSHomePage/DSHomePage";
import DSProduct from "../pages/DSProduct/DSProduct";
import DSAnalysis from "../pages/DSAnalysis/DSAnalysis";
import DSDataAccess from "../pages/DSDataAccess/DSDataAccess";
import DSAnalysisReport from "../pages/DSAnalysisReport/DSAnalysisReport";
import PRHomePage from "../pages/PRHomePage/PRHomePage";
import PRProductionPage from "../pages/PRProductionPage/PRProductionPage";
import PRProgressPage from "../pages/PRProgressPage/PRProgressPage";
import PRQualityCheckPage from "../pages/PRQualityCheckPage/PRQualityCheckPage";
import PRRReportPage from "../pages/PRRReportPage/PRRReportPage";
import QCHomePage from "../pages/QCHomePage/QCHomePage";
import QCQualityCheckPage from "../pages/QCQualityCheckPage/QCQualityCheckPage";
import QCProgressPage from "../pages/QCProgressPage/QCProgressPage";
import QCReportPage from "../pages/QCReportPage/QCReportPage";

const AppRoutes = () => {
  return (
    <Routes>
      <Route path="/" element={ <LandingPage/> } />
      <Route path="/register" element={ <RegistrationPage/> } />
      <Route path = "/change-password" element={ <ChangePasswordPage/> } />
      { /* Admin Routes */ }
      <Route path="/adminHome" element={ <ProtectedRoute><AdminHomePage/></ProtectedRoute> } />
      <Route path="/adminProduct" element={ <ProtectedRoute><AdminProductPage/></ProtectedRoute> } />
      <Route path="/adminUserManagement" element={ <ProtectedRoute><AdminUserManagement/></ProtectedRoute> } />
      <Route path="/adminReports" element={ <ProtectedRoute><AdminReportsPage/></ProtectedRoute> } />
      <Route path="/adminRejectedProducts" element={ <ProtectedRoute><AdminRejectedPage/></ProtectedRoute> } />
      <Route path="/adminDataAccess" element={ <ProtectedRoute><AdminDataAccessPage/></ProtectedRoute> } />
      { /* Resource Analyst Routes */ }
      <Route path="/raHome" element={ <ProtectedRoute><RAHome/></ProtectedRoute> } />
      <Route path="/raProduct" element={ <ProtectedRoute><RAProductPage/></ProtectedRoute> } />
      <Route path="/raAnalysis" element={ <ProtectedRoute><RAAnalysisPage/></ProtectedRoute> } />
      <Route path="/raAnalysisReport" element={ <ProtectedRoute><RAAnalysisReportPage/></ProtectedRoute> } />
      <Route path="/raDataAccess" element={ <ProtectedRoute><RADataAccessPage/></ProtectedRoute> } />
      { /* Design Support */ }
    </Routes>
  )
}

```

```

    <Route path="/dsHome" element={<ProtectedRoute><DSHomePage/></ProtectedRoute>}></Route>
    <Route path="/dsProduct" element={<ProtectedRoute><DSProduct/></ProtectedRoute>}></Route>
    <Route path="/dsAnalysis" element={<ProtectedRoute><DSAnalysis/></ProtectedRoute>}></Route>
    <Route path="/dsAnalysisReport" element={<ProtectedRoute><DSAnalysisReport/></ProtectedRoute>}></Route>
    <Route path="/dsDataAccess" element={<ProtectedRoute><DSDataAccess/></ProtectedRoute>}></Route>
    <Route path="/prHome" element={<ProtectedRoute><PRHomePage/></ProtectedRoute>}></Route>
    <Route path="/prProduction" element={<ProtectedRoute><PRProductionPage/></ProtectedRoute>}></Route>
    <Route path="/prProgress" element={<ProtectedRoute><PRProgressPage/></ProtectedRoute>}></Route>
    <Route path="/prQualityCheck" element={<ProtectedRoute><PRQualityCheckPage/></ProtectedRoute>}></Route>
    <Route path="/prReport" element={<ProtectedRoute><PRReportPage/></ProtectedRoute>}></Route>
    <Route path="/qcHome" element={<ProtectedRoute><QCHomePage/></ProtectedRoute>}></Route>
    <Route path="/qcQualityCheck" element={<ProtectedRoute><QCQualityCheckPage/></ProtectedRoute>}></Route>
    <Route path="/qcProgress" element={<ProtectedRoute><QCProgressPage/></ProtectedRoute>}></Route>
    <Route path="/qcReport" element={<ProtectedRoute><QCReportPage/></ProtectedRoute>}></Route>
  </Routes>
)
}

```

export default AppRoutes

#### **AXIOSINSTANCE.JS**

import axios from "axios";

import { navigateToLogin } from "../utils/navigation";

```

const axiosInstance = axios.create({
  baseURL: import.meta.env.VITE_API_BASE_URL,
  headers: {
    "Content-Type": "application/json",
  },
  withCredentials: true,
});

```

// Create a separate Axios instance for refreshing tokens

```

const refreshInstance = axios.create({
  baseURL: import.meta.env.VITE_API_BASE_URL,
  headers: {
    "Content-Type": "application/json",
  },
  withCredentials: true,
});

```

// Set the access token in headers before requests

```

axiosInstance.interceptors.request.use(
  (config) => {
    const token = localStorage.getItem("authToken");
    if (token) {
      config.headers.Authorization = `Bearer ${token}`;
    }
  }
);

```

```

    }

    return config;

  },

  (error) => Promise.reject(error)
);

// Refresh token logic
let isRefreshing = false;
let refreshSubscribers = [];

const onRefreshed = (token) => {
  refreshSubscribers.forEach((callback) => callback(token));
  refreshSubscribers = [];
};

axiosInstance.interceptors.response.use(
  (response) => response,
  async (error) => {
    const originalRequest = error.config;

    // Check if the error is due to unauthorized access (401)
    if (error.response?.status === 401 && !originalRequest._retry) {
      originalRequest._retry = true; // Mark request as retried

      if (!isRefreshing) {
        isRefreshing = true;

        try {
          console.log("Attempting token refresh...");
          const { data } = await refreshInstance.post("/auth/refreshToken"); // Use separate instance
          const newToken = data.access_token;

          localStorage.setItem("authToken", newToken);
          onRefreshed(newToken);
          isRefreshing = false;
        } catch (refreshError) {
          console.error("Token refresh failed, redirecting to login.");
          localStorage.removeItem("authToken");
          navigateToLogin(); // Redirect user to login
          isRefreshing = false;
          return Promise.reject(refreshError);
        }
      }
    }

    // Wait for token refresh to complete, then retry original request
    return new Promise((resolve) => {

```

```

        refreshSubscribers.push((token) => {
            originalRequest.headers.Authorization = `Bearer ${token}`;
            resolve(axiosInstance(originalRequest));
        });
    });
}

return Promise.reject(error);
}
);

```

```
export default axiosInstance;
```

## BACKEND

### SERVER.JS

```

import express from "express";
import cors from "cors";
import dotenv from "dotenv";
import path from 'path';
import { fileURLToPath } from 'url';
import adminRouter from './routes/adminRoutes.js';
import employeeRouter from './routes/employeeRoute.js';
import connectDB from './config/db.js';
import authRouter from './routes/authRoute.js';
import resourceAnalystRouter from './routes/resourceAnalystRoute.js';
import cookieParser from 'cookie-parser';
import designSupportRouter from './routes/designSupportRoutes.js';
import productionRouter from './routes/productionRoute.js';
import qualityRouter from './routes/qualityRoute.js';
import dashboardRouter from './routes/dashboardRouter.js'

dotenv.config();

const app = express();

// Middlewares
app.use(cors({
    origin: "http://localhost:5173",
    credentials: true,
    methods: ["GET", "POST", "PUT", "DELETE"],
    allowedHeaders: ["Content-Type", "Authorization"]
}));
app.use(express.json());
app.use(cookieParser());
app.use('/uploads', express.static(path.join(path.dirname(fileURLToPath(import.meta.url)), 'uploads')));

```

```

// Connect to MongoDB
connectDB();

app.use('/api/admin', adminRouter);
app.use('/api/employee', employeeRouter);
app.use('/api/auth', authRouter);
app.use('/api/resource-analyst', resourceAnalystRouter);
app.use('/api/design-support', designSupportRouter);
app.use('/api/production', productionRouter);
app.use('/api/quality', qualityRouter);
app.use('/api/dashboard', dashboardRouter);

app.listen(3000, () => {
  console.log("Server is running on port 3000");
});

BLOCKCHAINSERVICE.JS

import web3 from './config/Web3Config.js';
import fs from "fs";
import path from "path";
import dotenv from "dotenv";

import contractDetails from './config/contractDetails.json' assert { type: 'json' };
import { fileURLToPath } from 'url';

dotenv.config();

// Create __dirname equivalent for ES modules
const __filename = fileURLToPath(import.meta.url);
const __dirname = path.dirname(__filename);

export const getContractInstance = (contractName) => {
  try {
    const contractInfo = contractDetails[contractName];

    if (!contractInfo || !contractInfo.address || !contractInfo.abiPath) {
      throw new Error(`Invalid contract details for ${contractName}`);
    }

    const contractData = JSON.parse(
      fs.readFileSync(path.resolve(__dirname, contractInfo.abiPath)).toString()
    );

    return new web3.eth.Contract(contractData.abi, contractInfo.address);
  } catch (error) {
    console.error('Error getting contract instance:', error);
    throw error;
  }
}

```

### RESOURCEANALYSTSERVICE.JSX

```
import { getContractInstance } from './blockchainService.js';
import web3 from './config/Web3Config.js';

export const saveAnalysisData = async (encryptedHash, initialVector, timestamp) => {
  const contract = getContractInstance('AnalysisStorage');

  try {
    const receipt = await contract.methods.storeAnalysis(encryptedHash, initialVector, timestamp)
      .send({ from: web3.eth.defaultAccount });

    if (!receipt) {
      throw new Error('Transaction failed - no receipt received');
    }

    return {
      response: true,
      receipt: {
        transactionHash: receipt.transactionHash,
        blockNumber: receipt.blockNumber,
        gasUsed: receipt.gasUsed,
        status: receipt.status,
        transactionIndex: receipt.transactionIndex || 0,
        cumulativeGasUsed: receipt.cumulativeGasUsed || 0,
        effectiveGasPrice: receipt.effectiveGasPrice || 0,
        type: receipt.type || 0
      }
    };
  } catch (error) {
    return { response: false, error };
  }
}

export const getTotal = async () => {
  const contract = getContractInstance('AnalysisStorage');

  try {
    const total = await contract.methods.getTotal().call();
    const count = Number(total);

    return { response: true, total: count };
  } catch (error) {
    return { response: false, error };
  }
}

export const fetchAllData = async (startIndex, endIndex) => {
  const contract = getContractInstance('AnalysisStorage');

  try {
```



```

const result = await contract.methods.getAllAnalysis(startIndex, endIndex).call();

// Destructure the returned tuple
const txnHashes = result[0];
const encryptedHashes = result[1];
const initialVectors = result[2];
const timestamps = result[3];

// Convert the tuple of arrays into an array of objects.
let formattedData = [];
for (let i = 0; i < txnHashes.length; i++) {
  formattedData.push({
    transactionHash: txnHashes[i],
    encryptedHash: encryptedHashes[i],
    initialVector: initialVectors[i],
    timestamp: timestamps[i]
  });
}
return { response: true, data: formattedData };
} catch (error) {
  return { response: false, error };
}
}

export const getData = async (transactionHash) => {
  const contract = getContractInstance('AnalysisStorage');
  try {
    const result = await contract.methods.getAnalysis(transactionHash).call();
    const encryptedHash = result[0];
    const initialVector = result[1];
    const timestamp = result[2];
    const data = {
      encryptedHash, initialVector, timestamp
    }
    return { response: true, data }
  } catch (error) {
    return { response: false, error }
  }
}

export const updateData = async (transactionHash, encryptedHash, initialVector, timestamp) => {
  const contract = getContractInstance('AnalysisStorage');
  try {
    const receipt = await contract.methods.updateAnalysis(transactionHash, encryptedHash, initialVector, timestamp)
      .send({ from: web3.eth.defaultAccount });
    if (!receipt) {

```

```

        throw new Error("Transaction failed - no receipt received");
    }

    return {
        response: true,
        receipt: {
            transactionHash: receipt.transactionHash,
            blockNumber: receipt.blockNumber,
            gasUsed: receipt.gasUsed,
            status: receipt.status,
            transactionIndex: receipt.transactionIndex || 0,
            cumulativeGasUsed: receipt.cumulativeGasUsed || 0,
            effectiveGasPrice: receipt.effectiveGasPrice || 0,
            type: receipt.type || 0
        }
    };
} catch (error) {
    return { response: false, error };
}
}

export const deleteData = async (transactionHash) => {
    const contract = getContractInstance('AnalysisStorage');

    try {
        const receipt = await contract.methods.deleteAnalysis(transactionHash)
            .send({ from: web3.eth.defaultAccount });

        if (!receipt) {
            throw new Error("Transaction failed - no receipt received");
        }

        return {
            response: true,
            receipt: {
                transactionHash: receipt.transactionHash,
                blockNumber: receipt.blockNumber,
                gasUsed: receipt.gasUsed,
                status: receipt.status,
                transactionIndex: receipt.transactionIndex || 0,
                cumulativeGasUsed: receipt.cumulativeGasUsed || 0,
                effectiveGasPrice: receipt.effectiveGasPrice || 0,
                type: receipt.type || 0
            }
        };
    } catch (error) {
        if (error.message && error.message.includes("Analysis not found")) {

```

```

// Optionally, return a dummy receipt or an appropriate success message.
return {
  response: true,
  receipt: {
    transactionHash: transactionHash,
    message: "Analysis not found. It may have already been deleted."
  }
};
}
return { response: false, error };
}
}

```

#### ADMINROUTES.JS

```

import express from "express";
import multer from "multer";

import { uploadCSV, getAllProductData, addProduct, deleteAllProductData, getProduct, deleteProduct, updateProduct, approveEmployee, denyEmployee, getAllEmployeeData, deleteEmployee, grantAccess, getAccessRequests, denyAccess, getLoggedInUsers, getTransactionHistory, getRejectedProducts, qualityReport, productionReport, featuredMaterial, removeFeaturedMaterial, getFeaturedMaterial } from "../controllers/adminController.js";

import { getAllData as raController } from "../controllers/resourceAnalystController.js";
import { getAllData as dsController } from "../controllers/designSupportController.js";
import { authMiddleware } from "../middlewares/authMiddleware.js";
import { adminMiddleware } from "../middlewares/adminMiddleware.js";

const router = express.Router();
const upload = multer({ dest: "uploads/" });

// CSV upload route
router.post("/upload-csv", upload.single("file"), uploadCSV);

// Get products with pagination
// Example: /get-products-data?page=1&limit=10
router.get("/get-products-data", authMiddleware, adminMiddleware, getAllProductData);

// Add single product
router.post("/add-product", express.json(), authMiddleware, adminMiddleware, addProduct);

// Get single product
router.get("/get-product/:productId", authMiddleware, adminMiddleware, getProduct);

// Delete single product
router.delete("/delete-product/:productId", authMiddleware, adminMiddleware, deleteProduct);

// Delete all products
router.delete("/delete-all-products", authMiddleware, adminMiddleware, deleteAllProductData);

```

```

// Update product
router.put("/update-product/:productId", express.json(), updateProduct);

//approve employee
router.put("/approve-employee/:employeeId",authMiddleware,adminMiddelware, approveEmployee);

//deny employee
router.put("/deny-employee/:employeeId",authMiddleware,adminMiddelware, denyEmployee);

//get all employee data Example: /get-all-employee-data?status=pending
router.get("/get-all-employee-data",authMiddleware,adminMiddelware, getAllEmployeeData);

//delete employee
router.delete("/delete-employee/:employeeId",authMiddleware,adminMiddelware, deleteEmployee);

// Access control routes
router.get("/get-access-requests",authMiddleware,adminMiddelware, getAccessRequests);
router.put("/grant-access/:employeeId",authMiddleware,adminMiddelware, grantAccess);
router.put("/deny-access/:employeeId",authMiddleware,adminMiddelware, denyAccess);

// Get logged in users
router.get("/logged-in-users",authMiddleware,adminMiddelware, getLoggedInUsers);

// Get the transaction history
router.get("/get-transaction-history",authMiddleware,adminMiddelware, getTransactionHistory);
export default router;

//Get the rejected Products
router.get('/get-rejected-products',authMiddleware,adminMiddelware,getRejectedProducts)

// get the Resource Analyst Reports
router.get('/get-ra-report',authMiddleware,adminMiddelware,raController)

//ger the Design Support Reports
router.get('/get-ds-report',authMiddleware,adminMiddelware,dsController)

//get the Production and Assembly report
router.get('/get-pa-report',authMiddleware,adminMiddelware,productionReport)

//get the quality report
router.get('/get-qa-report',authMiddleware,adminMiddelware,qualityReport)

//set featured material
router.post('/set-featured-material',authMiddleware,adminMiddelware, featuredMaterial)

```

```
//delete feature material

router.delete('/remove-featured-material/:productId',authMiddleware,adminMiddleware,removeFeaturedMaterial)
```

```
//check featured material

router.get('/check-featured-material/:productId',authMiddleware,adminMiddleware,getFeaturedMaterial)
```

#### ADMINMIDDLEWARE.JS

```
export const adminMiddleware = async (req, res, next) => {

  if (req.role !== 'admin') {

    return res.status(403).json({ error: 'Forbidden', message: 'You do not have permission to admin' });

  }

  return next();

}
```

#### ANALYSTMIDDLEWARE.JS

```
export const analystMiddleware = async (req, res, next) => {

  if (req.role !== 'resource_analyst') {

    return res.status(403).json({ error: 'Forbidden', message: 'You do not have permission to resource analyst' });

  }

  return next();

}
```

#### CONTRACTDETAILS.JSON

```
{

  "ProductStorage": {

    "address": "0x8662A1649bb37b7755Fd448942C8f0BfCd049F4",

    "abiPath": "../blockchain/build/contracts/ProductStorage.json"

  },

  "AnalysisStorage":{

    "address":"0xBa3D5e92337663cF734E25eA39591a83a827d1BC",

    "abiPath":"../blockchain/build/contracts/AnalysisStorage.json"

  },

  "DesignerSupport":{

    "address":"0xf6C131298A2eb46037F5530A1267f73CDeC67815",

    "abiPath":"../blockchain/build/contracts/DesignerSupport.json"

  }

}
```

#### WEB3CONFIG.JS

```
import Web3 from "web3";
```

```
import dotenv from "dotenv";
```

```
dotenv.config();
```

```
if (!process.env.WEB3_PROVIDER_URL) {
```

```
  console.error("WEB3_PROVIDER_URL is not set in the environment variables");
```

```
}
```

```

const web3 = new Web3(new Web3.providers.HttpProvider(process.env.WEB3_PROVIDER_URL));

if(!process.env.BLOCKCHAIN_PRIVATE_KEY){
  console.error('BLOCKCHAIN_PRIVATE_KEY is not set in the environment variables');
}

const account = web3.eth.accounts.privateKeyToAccount(process.env.BLOCKCHAIN_PRIVATE_KEY);
web3.eth.accounts.wallet.add(account);
web3.eth.defaultAccount = account.address;

//Check balance
// web3.eth.getBalance(process.env.BLOCKCHAIN_PUBLIC_KEY)
// .then(balance => {
//   console.log('Balance: ${web3.utils.fromWei(balance, "ether")} ETH');
// })
// .catch(err => console.error("Error fetching balance:", err));

export default web3;

```

## BLOCKCHAIN

### ANALYSISSTORAGE.SOL

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.20;

```

contract AnalysisStorage {
    struct AnalysisData {
        string encryptedHash;
        string initialVector;
        uint256 timestamp;
        bool exists;
    }

    mapping(bytes32 => AnalysisData) private storedAnalysis;
    bytes32[] private transactionHashes;

    event AnalysisStored(bytes32 indexed transactionHash, string encryptedHash, string initialVector, uint256 timestamp);
    event AnalysisUpdated(bytes32 indexed transactionHash, string newEncryptedHash, string newInitialVector, uint256 newTimestamp);
    event AnalysisDeleted(bytes32 indexed transactionHash);

    function storeAnalysis(
        string memory _encryptedHash,
        string memory _initialVector,
        uint256 _timestamp
    ) public {
        bytes32 transactionHash = keccak256(abi.encodePacked(msg.sender, _timestamp));

        require(!storedAnalysis[transactionHash].exists, "Analysis already exists for this transaction hash");
    }
}

```

```

        storedAnalysis[transactionHash] = AnalysisData(_encryptedHash, _initialVector, _timestamp, true);
        transactionHashes.push(transactionHash);

        emit AnalysisStored(transactionHash, _encryptedHash, _initialVector, _timestamp);
    }

function getAnalysis(bytes32 _transactionHash)
    public
    view
    returns (string memory, string memory, uint256)
{
    require(storedAnalysis[_transactionHash].exists, "Analysis not found");

    AnalysisData memory analysis = storedAnalysis[_transactionHash];
    return (analysis.encryptedHash, analysis.initialVector, analysis.timestamp);
}

function getAllAnalysis(uint256 startIndex, uint256 endIndex)
    public
    view
    returns (bytes32[] memory, string[] memory, string[] memory, uint256[] memory)
{
    require(startIndex < endIndex, "Invalid range");
    require(startIndex < transactionHashes.length, "Start index out of bounds");
    require(endIndex <= transactionHashes.length, "End index exceeds total product count");

    uint256 size = endIndex - startIndex;

    bytes32[] memory paginatedTxnHashes = new bytes32[](size);
    string[] memory encryptedHashes = new string[](size);
    string[] memory initialVectors = new string[](size);
    uint256[] memory timestamps = new uint256[](size);

    for (uint256 i = 0; i < size; i++) {
        bytes32 txnHash = transactionHashes[startIndex + i];
        AnalysisData memory data = storedAnalysis[txnHash];

        paginatedTxnHashes[i] = txnHash;
        encryptedHashes[i] = data.encryptedHash;
        initialVectors[i] = data.initialVector;
        timestamps[i] = data.timestamp;
    }

    return (paginatedTxnHashes, encryptedHashes, initialVectors, timestamps);
}

```

```

function updateAnalysis(
    bytes32 _transactionHash,
    string memory _newEncryptedHash,
    string memory _newInitialVector,
    uint256 _newTimestamp
) public {
    require(storedAnalysis[_transactionHash].exists, "Analysis not found");

    storedAnalysis[_transactionHash].encryptedHash = _newEncryptedHash;
    storedAnalysis[_transactionHash].initialVector = _newInitialVector;
    storedAnalysis[_transactionHash].timestamp = _newTimestamp;

    emit AnalysisUpdated(_transactionHash, _newEncryptedHash, _newInitialVector, _newTimestamp);
}

function deleteAnalysis(bytes32 _transactionHash) public {
    require(storedAnalysis[_transactionHash].exists, "Analysis not found");

    delete storedAnalysis[_transactionHash];

    for (uint256 i = 0; i < transactionHashes.length; i++) {
        if (transactionHashes[i] == _transactionHash) {
            transactionHashes[i] = transactionHashes[transactionHashes.length - 1];
            transactionHashes.pop();
            break;
        }
    }

    emit AnalysisDeleted(_transactionHash);
}

function getTotal() public view returns (uint256) {
    return transactionHashes.length;
}
}

```

#### **DESIGNSUPPORTSTORAGE.SOL**

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.20;

```

contract DesignerSupport {
    struct DataEntry {
        string encryptedHash;
        string initialVector;
    }
}

```



```

    uint256 timeStamp;
}

mapping(bytes32 => DataEntry) private dataStore;
bytes32[] private transactionHashes;
address public owner;

event DataStored(bytes32 indexed transactionHash, string encryptedHash, string initialVector, uint256 timeStamp);
event DataUpdated(bytes32 indexed transactionHash, string newEncryptedHash, string newInitialVector, uint256 newTimeStamp);
event DataDeleted(bytes32 indexed transactionHash);

// Store data
function storeData(string memory encryptedHash, string memory initialVector, uint256 timeStamp) public {
    bytes32 transactionHash = keccak256(abi.encodePacked(encryptedHash, initialVector, timeStamp, block.timestamp));
    require(dataStore[transactionHash].timeStamp == 0, "Data already exists");

    dataStore[transactionHash] = DataEntry(encryptedHash, initialVector, timeStamp);
    transactionHashes.push(transactionHash);

    emit DataStored(transactionHash, encryptedHash, initialVector, timeStamp);
}

// Retrieve multiple data entries
function getAllData(uint256 startIndex, uint256 endIndex) public view returns (bytes32[] memory, string[] memory, string[] memory, uint256[] memory) {
    require(endIndex < transactionHashes.length, "Invalid index range");
    require(startIndex <= endIndex, "Start index must be less than or equal to end index");

    uint256 size = endIndex - startIndex + 1;
    bytes32[] memory paginatedTxnHashes = new bytes32[](size);
    string[] memory encryptedHashes = new string[](size);
    string[] memory initialVectors = new string[](size);
    uint256[] memory timeStamps = new uint256[](size);

    for (uint256 i = startIndex; i <= endIndex; i++) {
        bytes32 txHash = transactionHashes[i];
        DataEntry storage entry = dataStore[txHash];
        paginatedTxnHashes[i] = txHash;
        encryptedHashes[i - startIndex] = entry.encryptedHash;
        initialVectors[i - startIndex] = entry.initialVector;
        timeStamps[i - startIndex] = entry.timeStamp;
    }

    return (paginatedTxnHashes, encryptedHashes, initialVectors, timeStamps);
}

```

```

// Retrieve a single data entry

function getSingleData(bytes32 transactionHash) public view returns (string memory, string memory, uint256) {
    require(dataStore[transactionHash].timeStamp != 0, "Data not found");

    DataEntry storage entry = dataStore[transactionHash];

    return (entry.encryptedHash, entry.initialVector, entry.timeStamp);
}

// Update an existing entry

function updateData(bytes32 transactionHash, string memory newEncryptedHash, string memory newInitialVector, uint256 newTimeStamp) public {
    require(dataStore[transactionHash].timeStamp != 0, "Data not found");

    dataStore[transactionHash] = DataEntry(newEncryptedHash, newInitialVector, newTimeStamp);

    emit DataUpdated(transactionHash, newEncryptedHash, newInitialVector, newTimeStamp);
}

// Delete a data entry

function deleteData(bytes32 transactionHash) public {
    require(dataStore[transactionHash].timeStamp != 0, "Data not found");

    delete dataStore[transactionHash];

    // Remove transactionHash from the array
    for (uint256 i = 0; i < transactionHashes.length; i++) {
        if (transactionHashes[i] == transactionHash) {
            transactionHashes[i] = transactionHashes[transactionHashes.length - 1];
            transactionHashes.pop();
            break;
        }
    }

    emit DataDeleted(transactionHash);
}

// Get total stored data count

function getTotalStoredData() public view returns (uint256) {
    return transactionHashes.length;
}
}

PRODUCTSTORE.SOL

pragma solidity ^0.8.0;

contract ProductStorage {
    // Struct to store encrypted product data

    struct Product {

```

```

uint256 productId; // Unique product ID

string encryptedHash; // The encrypted hash of the product data

string initialVector; // The initial vector (IV) used in encryption

uint256 timestamp; // Timestamp of when the data was stored
}

// Counter for generating unique product IDs
uint256 private nextProductId = 1;

// Mapping to store product data by product ID
mapping(uint256 => Product) public productsById;

// Mapping to store product data by encrypted hash (for backward compatibility)
mapping(string => uint256) private hashToId;

// Array to store all product IDs for iteration
uint256[] public productIds;

// Counter to keep track of the number of products
uint256 public productCount;

// Function to get the number of products
function getProductCount() public view returns (uint256) {
    return productCount;
}

// Function to store encrypted product data
function storeProduct(
    string memory encryptedHash,
    string memory initialVector,
    uint256 timestamp
) public returns (uint256) {
    require(hashToId[encryptedHash] == 0, "Product already exists");

    uint256 productId = nextProductId++;
    productsById[productId] = Product(productId, encryptedHash, initialVector, timestamp);
    hashToId[encryptedHash] = productId;
    productIds.push(productId);
    productCount++;

    return productId;
}

// Store multiple products at once (Batch Insert)
function storeMultipleProducts(

```

```

    string[] memory encryptedHashes,
    string[] memory initialVectors,
    uint256[] memory timestamps
) public returns (uint256[] memory) {
    require(
        encryptedHashes.length == initialVectors.length &&
        initialVectors.length == timestamps.length,
        "Array lengths must match"
    );

    uint256[] memory newProductIds = new uint256[](encryptedHashes.length);

    for (uint256 i = 0; i < encryptedHashes.length; i++) {
        if (hashToId[encryptedHashes[i]] == 0) {
            uint256 productId = nextProductId++;
            productsById[productId] = Product(
                productId,
                encryptedHashes[i],
                initialVectors[i],
                timestamps[i]
            );
            hashToId[encryptedHashes[i]] = productId;
            productIds.push(productId);
            productCount++;
            newProductIds[i] = productId;
        }
    }

    return newProductIds;
}

// Function to get all product records in a range
function getAllProducts(uint256 startIndex, uint256 endIndex)
    public
    view
    returns (
        uint256[] memory ids,
        string[] memory encryptedHashes,
        string[] memory initialVectors,
        uint256[] memory timestamps
    )
{
    require(startIndex < endIndex, "Invalid range");
    require(startIndex < productIds.length, "Start index out of bounds");
    require(endIndex <= productIds.length, "End index exceeds total product count");
}

```

```

uint256 size = endIndex - startIndex;

ids = new uint256[](size);

encryptedHashes = new string[](size);

initialVectors = new string[](size);

timestamps = new uint256[](size);

for (uint256 i = 0; i < size; i++) {

    uint256 productId = productIds[startIndex + i];

    Product memory product = productsById[productId];

    ids[i] = product.productId;

    encryptedHashes[i] = product.encryptedHash;

    initialVectors[i] = product.initialVector;

    timestamps[i] = product.timestamp;

}

return (ids, encryptedHashes, initialVectors, timestamps);

}

// Function to get a product by ID
function getProductById(uint256 productId)

public

view

returns (Product memory)

{

    require(productsById[productId].productId != 0, "Product not found");

    return productsById[productId];

}

// Function to delete a product by ID
function deleteProduct(uint256 productId) public {

    require(productsById[productId].productId != 0, "Product not found");

    // Remove from hashToId mapping

    string memory hash = productsById[productId].encryptedHash;

    delete hashToId[hash];

    // Remove from productsById mapping

    delete productsById[productId];

    // Remove from productIds array

    for (uint256 i = 0; i < productIds.length; i++) {

        if (productIds[i] == productId) {

            productIds[i] = productIds[productIds.length - 1];

            productIds.pop();

```

```

        break;
    }
}

if (productCount > 0) {
    productCount--;
}
}

//Function to delete all products
function deleteAllProducts() public {
    // Delete all products from mappings
    for(uint256 i = 0; i < productIds.length; i++) {
        uint256 productId = productIds[i];
        string memory hash = productsById[productId].encryptedHash;
        delete hashToId[hash];
        delete productsById[productId];
    }

    // Clear the productIds array
    while(productIds.length > 0) {
        productIds.pop();
    }

    // Reset counters
    productCount = 0;
    nextProductId = 1;
}

// Function to update specific product fields
function updateProduct(
    uint256 productId,
    string memory encryptedHash,
    string memory initialVector,
    uint256 timestamp
) public returns (bool) {
    require(productsById[productId].productId != 0, "Product not found");

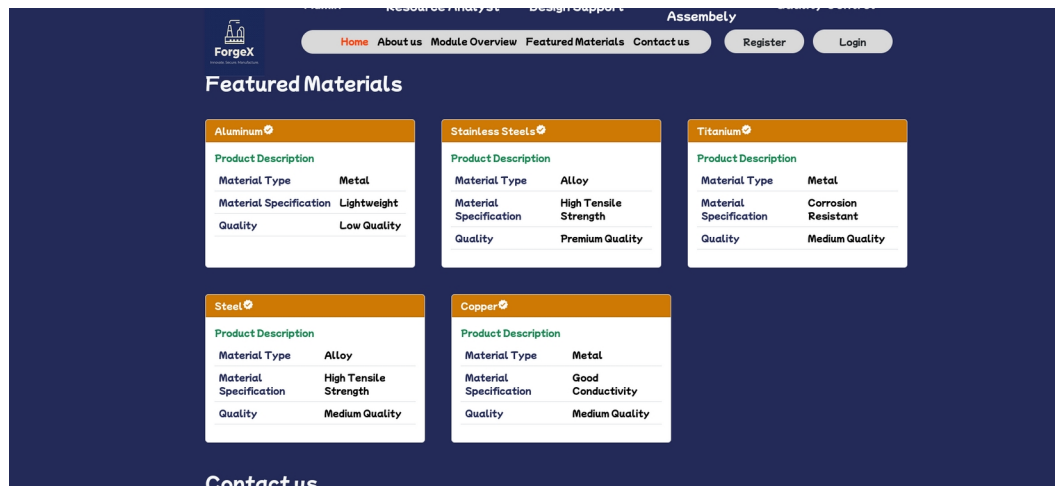
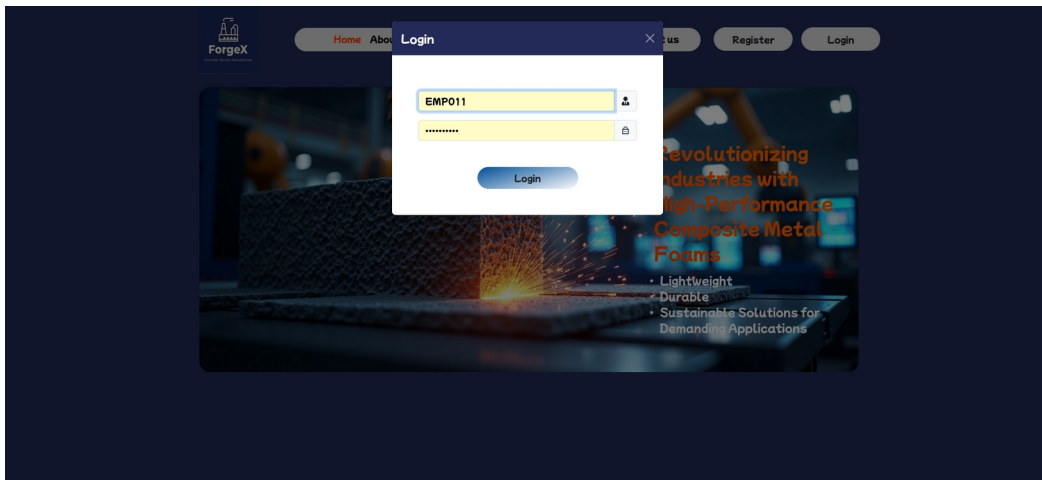
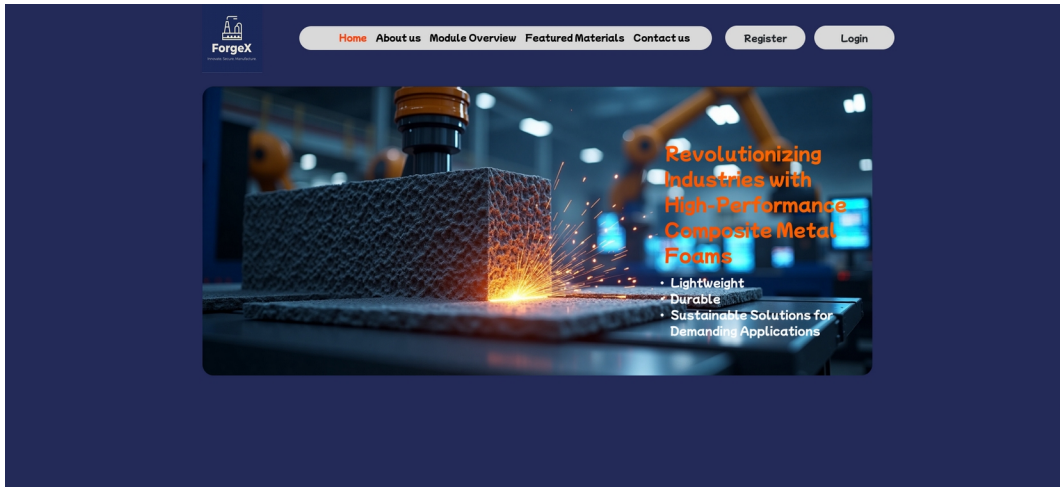
    // Remove old hash mapping
    string memory oldHash = productsById[productId].encryptedHash;
    delete hashToId[oldHash];


    // Update product
    productsById[productId] = Product(
        productId,

```

## CHAPTER – 7

### RESULT





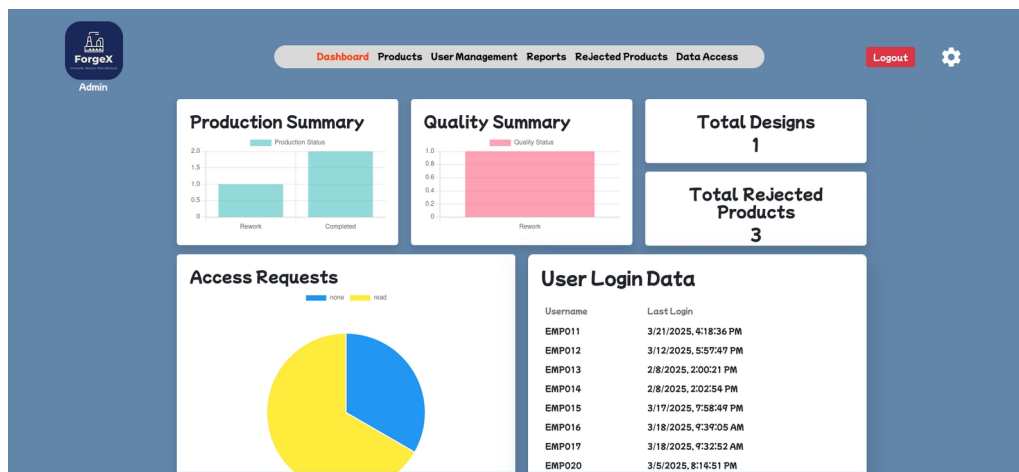
**ForgeX**  
Innovate. Secure. Manufacture.


### Create an Account

Invalid email format

[Register](#)


Already have an account? [Login](#)




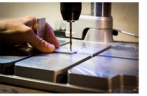
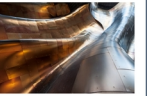







Admin

[Dashboard](#)
[Products](#)
[User Management](#)
[Reports](#)
[Rejected Products](#)
[Data Access](#)

[Logout](#)


[Upload CSV](#)
[Add Product](#)
[Delete All](#)


3/1/2025, 9:26:34 PM	3/1/2025, 9:26:34 PM	3/1/2025, 9:26:34 PM	3/1/2025, 9:26:34 PM
			
Product Name: <b>Steel</b>	Product Name: <b>Aluminum</b>	Product Name: <b>Titanium</b>	Product Name: <b>Copper</b>
Product ID: 1	Product ID: 2	Product ID: 3	Product ID: 4
<a href="#">More Info...</a>	<a href="#">More Info...</a>	<a href="#">More Info...</a>	<a href="#">More Info...</a>
			

← Prev


1
2
3
4


Next →
10 per page ▾




Admin


[Dashboard](#)
[Products](#)
[User Management](#)
[Reports](#)
[Rejected Products](#)
[Data Access](#)

[Logout](#)






Employee ID	Name	Email	Role	Status	Actions
EMP002	Sarah Johnson	sarah.johnson@company.com	design_support	denied	No Actions
EMP004	Emily Davis	emily.davis@company.com	design_support	approved	No Actions
EMP005	David Wilson	david.wilson@company.com	admin	denied	No Actions
EMP006	Lisa Anderson	lisa.anderson@company.com	resource_analyst	denied	No Actions
EMP007	Robert Taylor	robert.taylor@company.com	design_support	denied	No Actions
EMP008	Jennifer Martin	jennifer.martin@company.com	resource_analyst	denied	No Actions
EMP009	William Thompson	william.thompson@company.com	design_support	approved	No Actions
EMP010	Jessica White	Jessica.white@company.com	admin	approved	No Actions
EMP001	John Smith	John.smith@company.com	resource_analyst	approved	No Actions
EMP023	fdgsdfg	sgdfgds@gmail.com	design_support	approved	No Actions

[Prev](#)
[Next](#)


Admin


[Dashboard](#)
[Products](#)
[User Management](#)
[Reports](#)
[Rejected Products](#)
[Data Access](#)

[Logout](#)





Employee ID	Name	Email	Role	Status	Data Access
EMP022	MothishTestRA	mothishhellokumar@gmail.com	resource_analyst	pending	<a href="#">Approve</a> <a href="#">Deny</a>

[Prev](#)
[Next](#)


Resource Analyst

[Home](#)
[Products](#)
[Analysis](#)
[Analysis Report](#)
[Data Access](#)

[Logout](#)


### Quality Criteria Threshold

Density (g/cm<sup>3</sup>)  
7.43

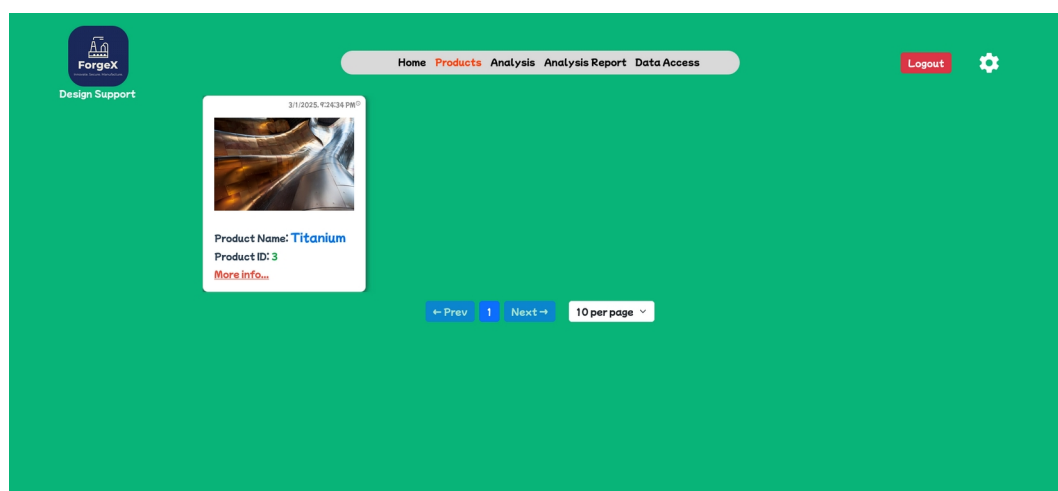
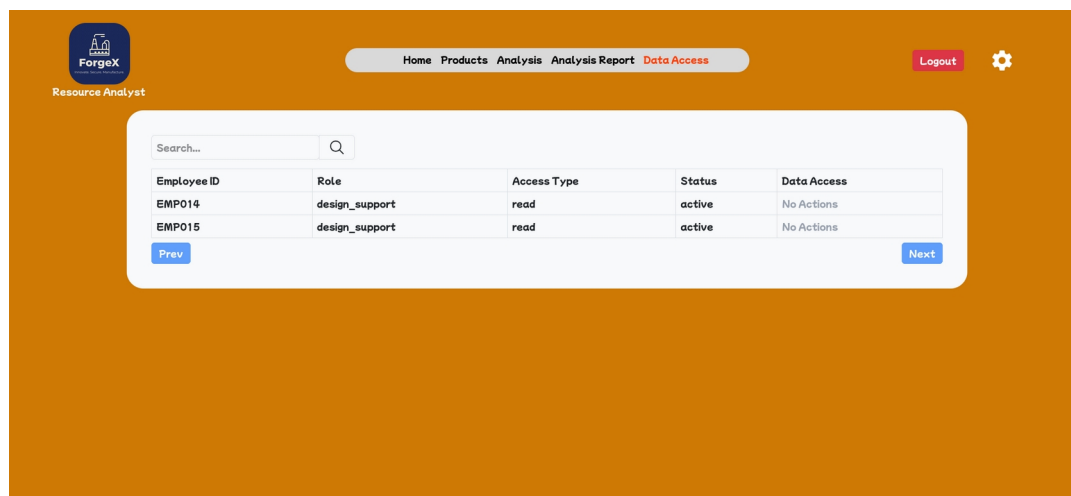
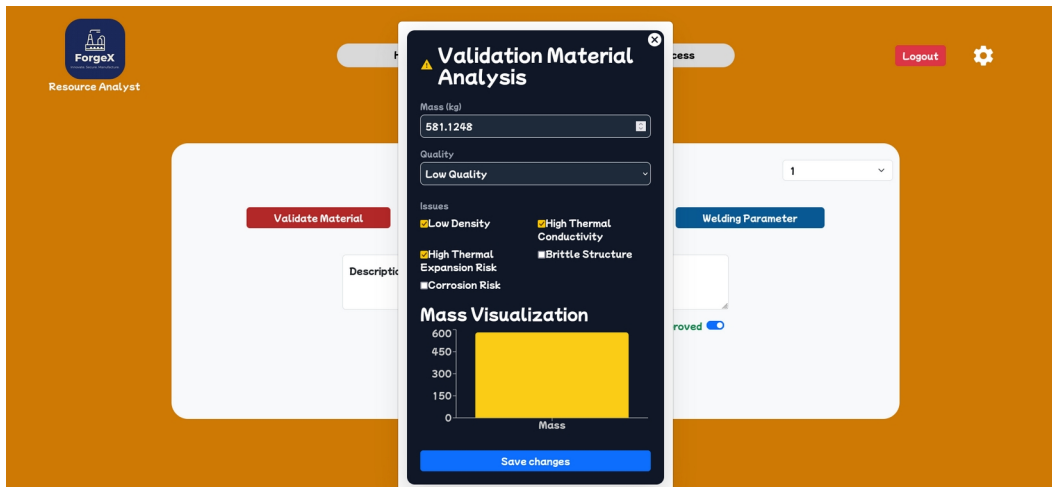
Flexural Strength (MPa)  
915

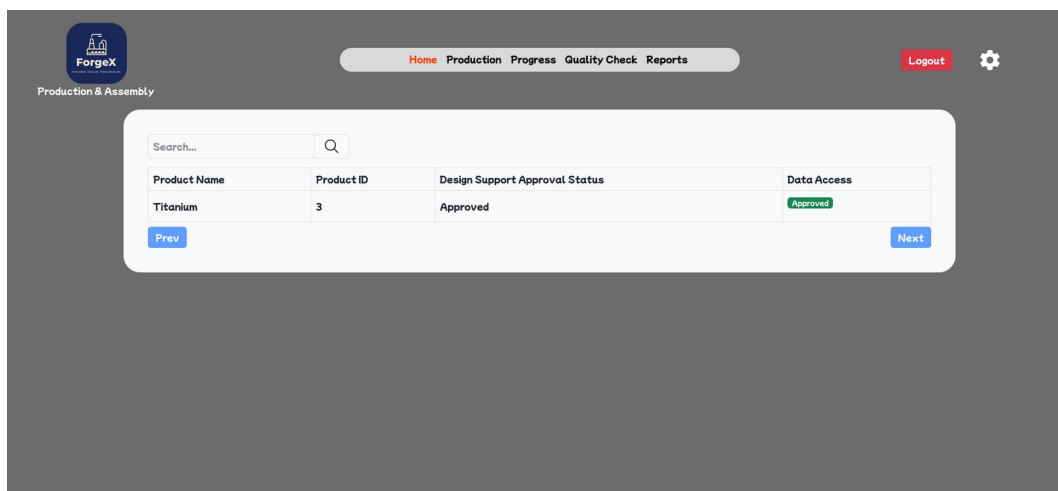
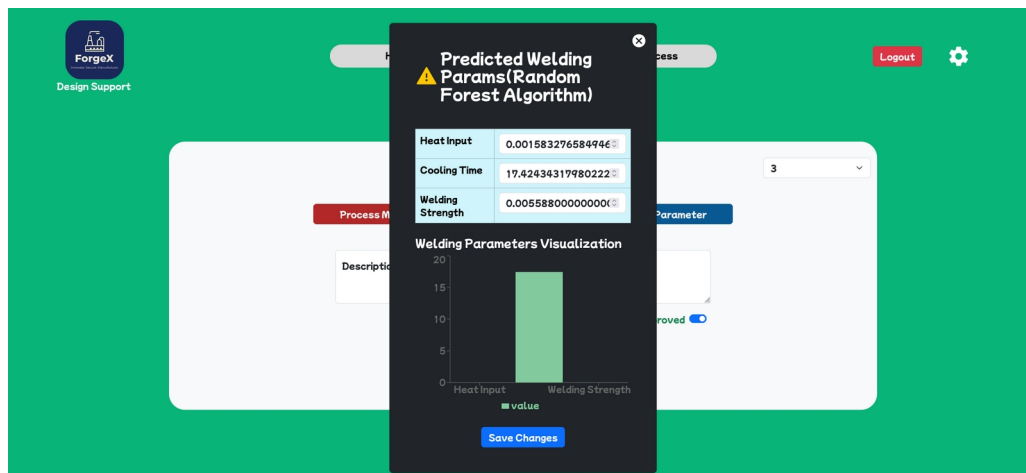
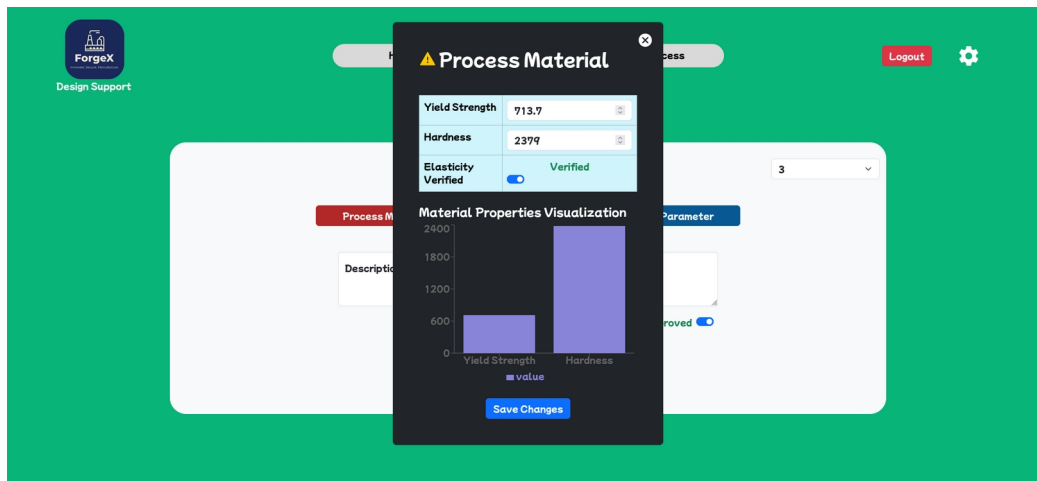
Tensile Strength (MPa)  
239

Porosity  
0.0128


Thermal Conductivity (W/mK)  
126.4

[Edit](#)









Quality Check


Home

Quality Check


Progress

Reports

Logout




Search...



Product Name	Product ID	Design Support Approval Status	Data Access
Titanium	3	Approved	Approved

Prev

Next



Quality Check

Home

Quality Check

Progress

Reports

No Production is found for quality checks

Choose Production ID

Young Modulus

Corrosion Resistance

Weight Efficiency

Tensile Strength


Weld Integrity

Corrosion Impact

Weight Retention

Approve

Rework



Quality Check


Home

Quality Check


Progress

Reports

Logout



Search...



Production ID	Production Name	Completed At	Action
PR002	qualitycheckTestProduction	1742227766385	Download Report
PR004	testproduction	1742269000226	Download Report

Prev

Next

## **CHAPTER – 8**

### **CONCLUSION**

This project successfully delivers a secure, modular, and efficient system for managing the production and quality control of composite metal foams (CMF) by integrating modern web technologies with blockchain-based security. By leveraging a layered architecture that includes React.js for the frontend, Node.js with Express.js for the backend, MongoDB for secure data storage, and Ganache with Truffle for blockchain-enabled key management, the system ensures data confidentiality, integrity, and traceability at every stage of the workflow.

The modular design allowed each department (Admin, Resource Analyst, Design Support, Production & Assembly, and Quality Control) to work independently yet collaboratively through secure data exchanges and decentralized key access. The inclusion of smart contracts for encryption key management has enhanced the system's security, eliminating the risk of unauthorized access and providing a transparent, tamper-proof audit trail.

Through features like encrypted data transfers, role-based access control, and automated reporting, the system addresses common challenges in the CMF production industry, including data breaches, manual errors, and inefficient workflows. The project not only improves the accuracy and security of data handling but also lays the foundation for scalable enhancements, such as AI integration and multi-chain blockchain support.

Overall, this system represents a modern, technology-driven approach to industrial production management, offering improved operational efficiency, secure collaboration between teams, and strong data protection mechanisms tailored to the needs of high-performance material production environments.

## REFERENCE

- Express.js Official Guide - <https://expressjs.com/>
- MongoDB Documentation - <https://www.mongodb.com/docs/>
- React.js Official Docs - <https://react.dev/>
- Ganache Documentation - <https://trufflesuite.com/ganache/>
- Truffle Suite Documentation - <https://trufflesuite.com/docs/truffle/>
- Web3.js Documentation - <https://web3js.readthedocs.io/>
- JWT Authentication - <https://jwt.io/introduction>
- CSV Parser for Node.js - <https://www.npmjs.com/package/csv-parser>
- Nodemailer - <https://nodemailer.com/about/>
- bcrypt.js Library - <https://www.npmjs.com/package/bcryptjs>
- MetaMask Documentation - <https://docs.metamask.io/>
- Node.js Official Documentation - <https://nodejs.org/en/docs/>