**📘 SiteOps Manager – Run Book**

**🔰 Project Overview**

**SiteOps Manager** is a web-based internal tool to manage websites, client tickets, risk tracking, and secure file uploads, mapped by employee ID (empID). Each logged-in user interacts with the system based on their role (admin, developer, or client), and all data interactions are scoped to the current user’s empID.

**📁 Project Structure**

SiteOps/   
│   
├── backend/   
│   ├── controllers/   
│   ├── models/   
│   ├── routes/   
│   ├── uploads/   
│   ├── server.js   
│   └── .env   
│   
├── frontend/   
│   ├── public/   
│   ├── src/   
│   │   ├── pages/        // Dashboard, Login, Register, Sites, Tickets, Files, Risks   
│   │   ├── context/      // AuthContext.js   
│   │   └── App.jsx   
│   
├── README.md   
└── package.json 

**⚙️ Tech Stack**

* **Frontend**: React.js, Tailwind CSS, FilePond
* **Backend**: Node.js, Express.js
* **Database**: MongoDB (Mongoose)
* **Auth**: JWT + Cookies + empID mapping
* **File Uploads**: multer, FilePond UI
* **Roles**: admin, developer, client

**🛠️ Initial Setup**

**1. Clone the Repository**

git clone <https://github.com/your-org/siteops-manager.git>   
cd siteops 

**2. Backend Setup (/backend)**

cd backend   
npm install 

Create a .env file:

.env

PORT=5000   
MONGO\_URI=mongodb://localhost:27017/siteops   
JWT\_SECRET=your\_secret\_key 

Start the backend:

npm start 

**3. Frontend Setup (/frontend)**

cd frontend   
npm install   
npm run dev  

The app will run at: [http://localhost:5173](http://localhost:5173/)

**🔐 Authentication Flow**

* POST /api/auth/login: Validates email/password and sets:
* token in localStorage
* empID in localStorage
* Cookie empID (optional)
* POST /api/auth/register: Only admin can register. Captures createdBy using current empID.

localStorage.setItem("empID", user.employeeId); 

**📥 File Upload Flow**

* Files are uploaded through FilePond to /upload.
* Backend uses multer to store and record files in MongoDB.
* Every upload includes empID from localStorage.

formData.append("empID", localStorage.getItem("empID")); 

**📌 empID Mapping Logic**

* Every record (site, ticket, file, risk) is tagged with empID.
* Only the current user's records are fetched on the UI using:

const empID = localStorage.getItem("empID"); 

* Server filters like:

const data = await Model.find({ empID: req.headers.empid }); 

**📋 Modules and Features**

|  |  |
| --- | --- |
| **Page** | **Features** |
| **Login / Register** | JWT login, password hashing, empID setting |
| **Dashboard** | Welcome + quick links |
| **Sites** | CRUD for site records. Scoped by empID |
| **Tickets** | Submit & delete issues. Sites dropdown scoped by empID |
| **File Upload** | FilePond uploader scoped by empID |
| **Risks** | Risk logging & tracking with priority/status |
| **Admin Features** | Register users and assign roles |
| **Auth Context** | Maintains login state |

**✅ User Roles**

|  |  |
| --- | --- |
| **Role** | **Permissions** |
| **Admin** | Full access to register users, manage all modules |
| **Developer** | Submit tickets, manage assigned sites/files |
| **Client** | View-only access (optional future support) |

**🧪 Testing Checklist**

* Login → empID set in localStorage
* Sites CRUD works for current empID
* File uploads stored in uploads/ folder
* Only logged-in users can access pages
* Ticket creation and filtering
* Risk records created and viewed
* Register user stores createdBy (empID)

**🚀 Deployment Guide**

**Backend (Node.js)**

cd backend   
npm install   
npm run build  # optional   
pm2 start server.js 

**Frontend (React)**

cd frontend   
npm run build   
serve -s dist  # If using Vite 

Use NGINX, Apache, or a cloud platform (e.g., Azure, Heroku, Netlify) to host frontend and backend separately or together via reverse proxy.

**🔐 Security Notes**

* Use HTTPS in production
* Set secure: true on cookies in auth routes
* Implement rate limiting on auth routes
* Use helmet middleware in Express

**📎 Useful APIs**

|  |  |  |
| --- | --- | --- |
| **API Route** | **Method** | **Description** |
| /api/auth/login | POST | Login with email & password |
| /api/auth/register | POST | Admin registers a user |
| /api/sites | GET/POST/PUT/DELETE | CRUD Sites per empID |
| /api/tickets | GET/POST/DELETE | Submit and manage tickets |
| /api/files | GET | List uploaded files by empID |
| /upload | POST | File upload handler |
| /api/risks | GET/POST/DELETE | Manage project risks |

**📌 Future Improvements**

* Add password reset feature
* Audit logs for admin actions
* Email notifications on ticket creation
* Role-based component rendering
* Pagination for large data tables