**✨ Savannah Photo App Documentation**

**Version**: 1.0.0  
**Author**: Ian Mathews/ ianmathew186@gmail.com/0793903636  
**Last Updated**: *07/02/2025***Last Site**: https://savannah-photo-app.vercel.app/

**📌 Overview**

The **Savannah Photo App** is a **full-stack web application** that allows users to:

* ✅ Register and log in to manage their albums and photos
* ✅ Upload photos, like, comment, and delete them
* ✅ View other users' albums and photos
* ✅ Edit photo titles and manage their content

**Access Restrictions:**

* ❌ Only authenticated users (active sessions) can **create their own albums** and **edit or delete their own photos**.
* ❌ Users **cannot edit or delete other users' albums or photos**.

The backend is built with **Node.js, Express, and MongoDB**, while the frontend is powered by **Next.js and Tailwind CSS**.

**📌 Tech Stack**

| **Component** | **Technology** |
| --- | --- |
| **Frontend** | Next.js, React, Tailwind CSS |
| **Backend** | Node.js, Express.js, Mongoose |
| **Database** | MongoDB Atlas |
| **State Management** | Redux Toolkit |
| **Image Uploads** | Cloudinary |
| **Deployment** | Backend on **Render**, Frontend on **Vercel** |
| **Authentication** | JWT (JSON Web Token) |
| **Docker Support** | Yes |

**📌 Features**

**🛡️ User Authentication**

* Users can **sign up, log in, and manage sessions** securely
* Authentication is handled via **JWT tokens**
* User sessions are **protected**

**🌟 Album & Photo Management**

* Users can **create, edit, and delete albums**
* Users can **upload images** to Cloudinary
* Users can **like and comment** on photos
* Users can **edit photo titles**
* **Users cannot edit or delete other users' albums or photos**

**🛠️ Navigation & UI**

* **Modern UI design** with Tailwind CSS and ShadCN UI components
* **Protected routes** ensure unauthorized users can't access private data

**📌 Installation & Setup**

**💡 1. Clone the Repository**

git clone https://github.com/your-repo/savannah-photo-app.git

cd savannah-photo-app

**💡 2. Backend Setup**

cd backend

npm install

Create a .env file in the backend/ folder with:

MONGO\_URI=mongodb+srv://your-mongodb-uri

JWT\_SECRET=your-secret-key

CLOUDINARY\_CLOUD\_NAME=your-cloudinary-name

CLOUDINARY\_API\_KEY=your-cloudinary-api-key

CLOUDINARY\_API\_SECRET=your-cloudinary-secret

PORT=5000

Start the backend:

npm run dev

**Backend runs on:**  
http://localhost:5000

**💡 3. Frontend Setup**

cd frontend

npm install

Create a .env.local file in frontend/ with:

NEXT\_PUBLIC\_API\_URL=http://localhost:5000/api

Start the frontend:

npm run dev

**Frontend runs on:**  
http://localhost:3000

**📌 Deployment**

**💡 Backend Deployment (Render)**

1. Push backend code to GitHub
2. Create a **new Web Service** on [Render](https://render.com/)
3. Set **Build Command**:
4. npm install
5. Set **Start Command**:
6. npm start
7. Add environment variables (MONGO\_URI, JWT\_SECRET, CLOUDINARY\_\*)
8. **Deploy!**

Backend URL Example:

https://your-backend.onrender.com

**💡 Frontend Deployment (Vercel)**

1. Push frontend code to GitHub
2. Go to [Vercel](https://vercel.com/) and create a new project
3. Select the **frontend repository**
4. Add environment variables (NEXT\_PUBLIC\_API\_URL)
5. **Deploy!**

Frontend URL Example:

https://your-frontend.vercel.app

**📌 API Reference**

All API endpoints are **documented in Postman**.  
👉 **View API Documentation Here** → [Postman API Collection](https://team-maabara-bunifu.postman.co/workspace/Savannah-Photo-App-Workspace~2d13895d-8ac1-406d-b9db-13b1b82467b6/collection/28493118-19b95d3d-2bca-42f6-8af8-b300b0dd6b07?action=share&creator=28493118)

**📌 Docker Setup**

You can run the entire application using Docker.

**💡 1. Build and Run Everything with Docker Compose**

docker-compose up --build

Frontend: http://localhost:3000  
Backend: http://localhost:5000/api

**📌 Security & Best Practices**

* **Environment Variables:** Store API keys, JWT secrets, and database credentials in .env files.
* **CORS Policy:** Only allow frontend to communicate with backend via CORS settings.
* **Validation:** Ensure inputs are sanitized before storing in the database.

**📌 Conclusion**

🚀 **Savannah Photo App** is a complete **MERN-stack** solution with authentication, image uploads, and social features.

* ✅ Successfully **deployed backend on Render** and **frontend on Vercel**.
* ✅ Uses **MongoDB Atlas for database storage** and **Cloudinary for image uploads**.
* ✅ API documentation is available on **Postman**.

**Enjoy using Savannah Photo App!** 🚀🔥