Secure Photo App: Secure, Cloud-Based Photo Management

10.08.2025

Overview:

The **Secure Photo App** is a full-stack web application designed to provide users with a safe and seamless experience for uploading, storing, and managing photos.

Built with a **React frontend** and a **Python backend (FastApi)**, the app leverages modern authentication, cloud storage, and a robust database to ensure both **usability** and **security**.

Key Features:

- 🔐 **User Authentication**: Utilizes **Auth0** for secure login and access control.
- 📤 **Photo Uploads**: Users can upload images in various formats, which are stored securely in **Azure Blob Storage**.
- ☁️ **Cloud Storage**: Integration with Azure ensures scalable and reliable storage for user photos.
- 🗄️ **Database Management**: **PostgreSQL** is used to store metadata and manage user data efficiently.
- 📱 **Responsive UI**: The React-based frontend offers a modern, mobile-friendly interface.
- 🔗 **API-Driven Architecture**: The backend exposes RESTful APIs for all core operations.

Architecture:

Frontend

- **Framework**: React
- **Structure**: Organized into components, views, assets, and utilities for maintainability.
- **Authentication**: Communicates with Auth0 for user login/logout and profile management.
- **API Integration**: Interacts with the backend via RESTful endpoints for uploads, user data, and more.

Backend

- **Language**: Python
- **Modules**: Handles authentication, database operations, Azure Blob interactions, and file uploads.
- **Authentication**: Auth0 integration using **JWT** and **RS256** algorithms for secure API access.
- **Database**: PostgreSQL stores user and photo metadata.
- **Cloud Storage**: Azure Blob Storage for scalable photo storage.

Configuration

The backend .env file configures critical services:

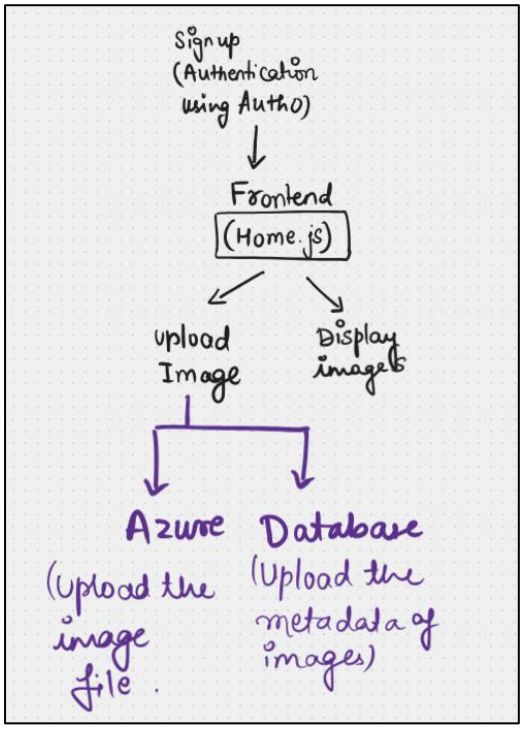
```
# Auth0 Configuration
AUTH0_DOMAIN=xxxxxxxxxxxxxxx.us.auth0.com
API_IDENTIFIER=https://myapi.example.com
ALGORITHMS=.....

# PostgreSQL Configuration
DATABASE_URL=postgresql://username:password@localhost/secure_photo_app

# Azure Storage
AZURE_STORAGE_CONNECTION_STRING=....
AZURE_CONTAINER_NAME=.....
```

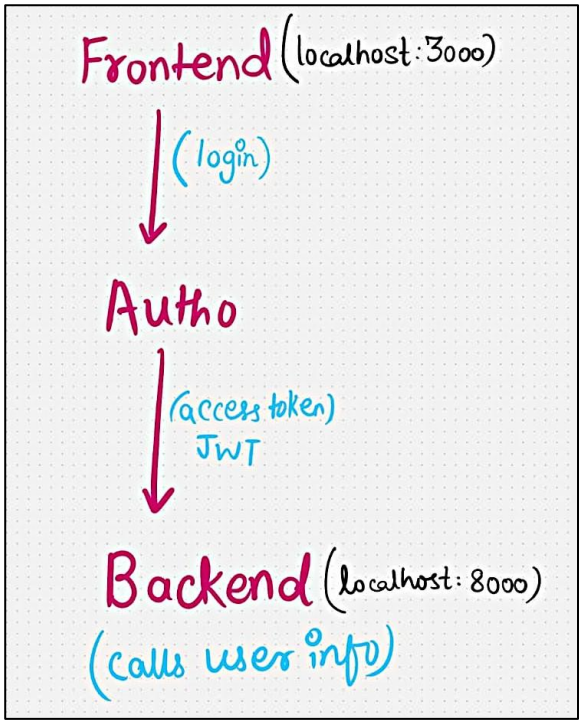
- **Auth0**: Handles authentication and authorization for API endpoints.
- **PostgreSQL**: Stores user and photo information.
- **Azure Storage**: Manages photo files in the cloud.

How It Works

1. **User Registration & Login**
   Users sign up or log in via Auth0, ensuring secure access.
2. **Photo Upload**
   Authenticated users can upload photos through the frontend. The backend receives the file, stores metadata in PostgreSQL, and uploads the image to Azure Blob Storage.

3. **Photo Management**
   Users can view, download, or delete their photos. All operations are protected by authentication.
4. **Security**
   JWT tokens and secure algorithms (**RS256**) ensure only authorized access to APIs and user data.





Getting Started

Backend

1. Set up Python environment and install dependencies from requirements.txt.
2. Configure .env with Auth0, PostgreSQL, and Azure credentials.
3. Run the backend server.

- uvicorn main:app -reload

Frontend

1. Install Node.js dependencies:

- npm install

2. Start the React development server:

- npm run dev

3. Configure API endpoint in frontend/.env.

Use Cases

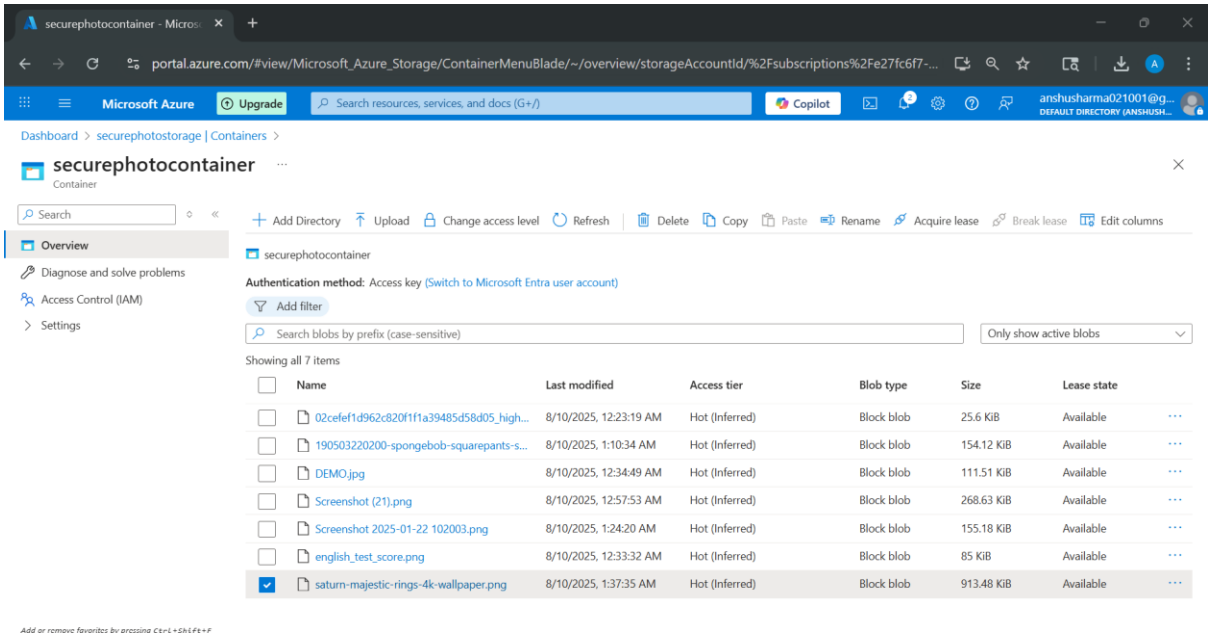
- **Personal Photo Vault**: Safely store and manage personal photos.
- **Enterprise**: Extend for secure document/image management in organizations.
- **Cloud Migration**: Demonstrates best practices for cloud storage integration.

Security Considerations

- All sensitive credentials are stored in environment variables.
- Auth0 provides robust authentication and authorization.
- Azure Blob Storage ensures data durability and privacy.
- PostgreSQL is used with secure connection strings.

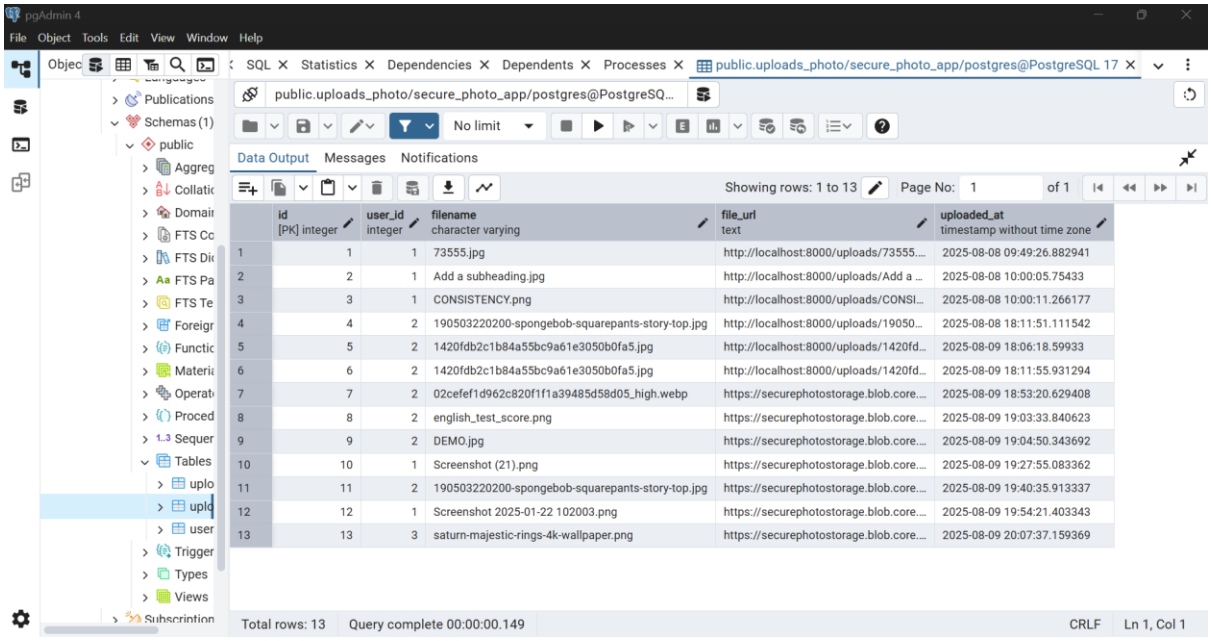
Results:

Azure Blob Storage:

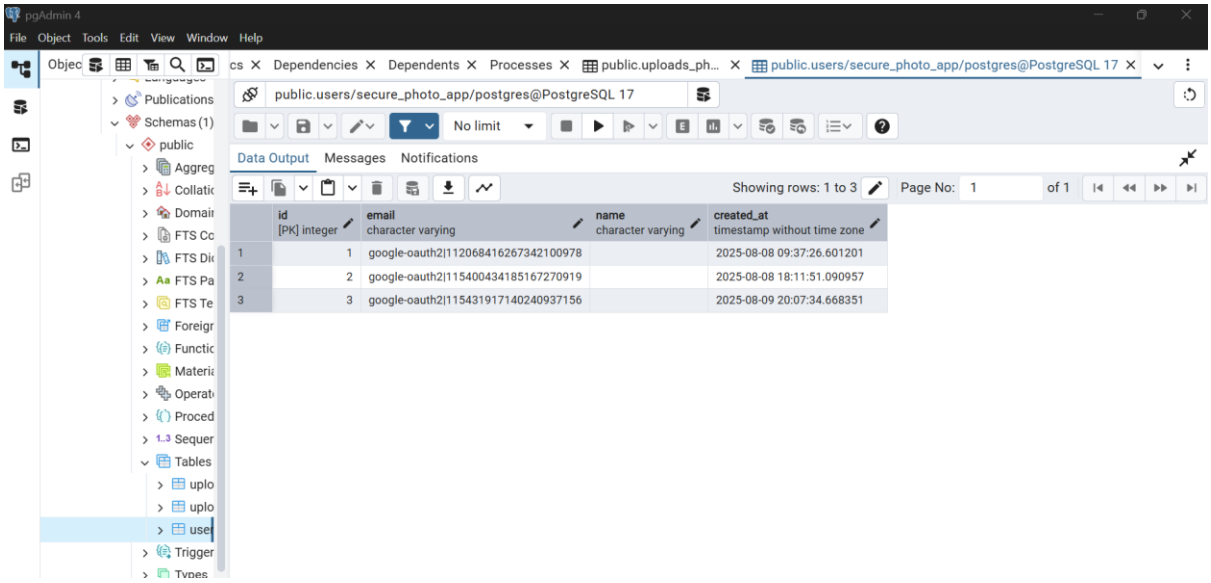


PostgreSql:

User_upload table:



User table:



Conclusion:

The **Secure Photo App** is a modern, secure solution for photo management, combining **best-in-class authentication**, **cloud storage**, and a **user-friendly interface**.

Its modular architecture makes it easy to extend, scale, and maintain whether for personal use or enterprise deployment.