BlogHub Application Architecture Documentation

# 1. Overview

BlogHub is a full-stack blogging platform that includes user authentication, blog management, and email verification. It is built with:  
- Backend: Node.js + Express REST API  
- Database: MongoDB (Atlas)  
- Frontend: React.js (served statically in production)  
- Authentication: JWT-based with OTP verification

# 2. System Architecture

## Backend (Node.js/Express)

| Component | Description |  
|--------------------|-----------------------------------------------------------------------------|  
| Server | Entry point (server.js) that sets up Express, middleware, and routes. |  
| Routes | Modular routes (/api/auth, /api/blogs) handled by controllers. |  
| Models | MongoDB schemas (User, Blog) using Mongoose for data management. |  
| Middleware | Authentication (authenticate middleware) and authorization (authorize middleware). |  
| Utils | Helper functions (e.g., sendEmail for OTP via Nodemailer). |

## Frontend (React)

| Component | Description |  
|--------------------|-----------------------------------------------------------------------------|  
| Static Build | Served from backend/public in production (express.static). |  
| API Calls | API requests via Axios to /api/\* endpoints. |

## Database (MongoDB Atlas)

- Stores users (email, hashed password, role) and blogs (title, content, and reference to the author).  
- Utilizes Mongoose for schema definition and querying.

## Authentication Flow

1. Signup: The user submits their email → The backend sends an OTP to the email → OTP is verified → User is created.  
2. Login: On successful login, a JWT is issued.  
3. Access Control: The admin role is required for deleting blogs.

# 3. Key Features & Flow

## A. User Authentication

1. Signup with OTP  
 - POST /api/auth/signup: Generate OTP → Send via email → Temporarily store OTP.  
 - POST /api/auth/verify-otp: Validate OTP → Create user in database.  
  
2. Login  
 - POST /api/auth/login: Verify credentials → Return JWT.

## B. Blog Management

1. Create Blog  
 - POST /api/blogs (Admin-only): Save blog to DB with author: req.user.id.  
  
2. Fetch Blogs  
 - GET /api/blogs: Returns all blogs with the author's details (populated via Mongoose).  
  
3. Delete Blog  
 - DELETE /api/blogs/:id (Admin-only): Deletes the blog from the database.

## C. Email Service

- OTPs are sent via Nodemailer using Gmail SMTP.  
- Environment variables (EMAIL\_USER, EMAIL\_PASS) are configured in the platform's settings (e.g., Render).

# 4. Deployment (Render)

- Backend: Node.js service with environment variables configured in Render's dashboard.  
- Frontend: Static files are served from backend/public in production mode.  
- Database: MongoDB Atlas connection through the MONGO\_URI environment variable.

# 5. Error Handling

- JWT Errors: Return 401 Unauthorized for invalid or expired JWT.  
- Role Errors: Return 403 Forbidden for unauthorized access attempts.  
- OTP Errors: Return 400 Bad Request for expired or invalid OTP.

# 6. Security

- Passwords: Stored as securely hashed values using bcryptjs.  
- Sensitive Data: All secrets are stored in environment variables (no hardcoded secrets).  
- Routes: Protected by authentication (authenticate) and role-based authorization (authorize) middleware.

# 7. Future Improvements

- Implement rate-limiting for OTP requests.  
- Add refresh tokens to extend JWT sessions.  
- Containerize the app using Docker for easier deployments and scalability.

# System Diagram

[Frontend] → [Backend (Node/Express)] → [MongoDB Atlas]   
 ↑   
 [Nodemailer (OTP)]