Project Title: Cloud-Based Multi-Tenant Blogging Platform

Overview

A cloud-native blogging platform built using **Next.js** for the frontend and serverless backend services from **AWS**. The application supports multi-tenant architecture, where users can create accounts, start blogs, publish posts, and customize their themes. It includes features such as markdown editor, user authentication, image uploads, SEO-friendly pages, and analytics.

Target Users

- Individual bloggers
- Organizations needing custom blogging instances
- Admins managing blog subscriptions

Tech Stack

- **Frontend**: Next.js 14 (App Router, TypeScript, TailwindCSS)
- **Authentication**: Amazon Cognito
- **Database**: Amazon DynamoDB (for blog metadata, users, posts)
- **Storage**: Amazon S3 (for images)
- **API**: AWS Lambda + API Gateway
- **Analytics**: Amazon Pinpoint or custom via CloudWatch logs
- **CI/CD**: GitHub Actions + AWS Amplify or Vercel
- **Infrastructure as Code**: AWS CDK (or Terraform)

Features

1. User Accounts & Authentication

- Register/login/logout via Cognito
- Email verification and password reset

2. Multi-Tenant Blog Management

- Each user can manage multiple blogs
- Subdomain support: `user.yourdomain.com`
- Blog theme selection and customization

3. Post Creation

- Markdown-based rich editor

- Autosave draft feature
- Post scheduling and publishing

4. Image Management

- Upload and embed images from S3
- Optimize image delivery with S3 + CloudFront

5. Blog Viewing

- Static generation (SSG/ISR) for public posts
- SEO metadata (OpenGraph, meta tags)

6. Analytics & Dashboard

- Post views per day/week
- Geographic distribution
- Top posts by views

7. Admin Dashboard

- Manage users and blogs
- Moderate flagged content
- Adjust subscription tiers

Pages Overview (Next.js App Router)

- `/` — Landing page
- `/auth/login` & `/auth/register`
- `/dashboard` — User blog dashboard
- `/dashboard/blogs/:id` — Blog settings
- `/dashboard/blogs/:id/posts/new` — Create post
- `/dashboard/blogs/:id/posts/:postId/edit` — Edit post
- `/:username/:slug` — Public blog post page
- `/:username` — Blog homepage

Database Schema (DynamoDB)

Users Table

| PK (userId) | email | username | createdAt | subscription |

Blogs Table

| PK (blogId) | userId | name | subdomain | theme | createdAt |

Posts Table

| PK (postId) | blogId | title | slug | content | status | createdAt | updatedAt |

Media Table

| PK (fileId) | userId | blogId | url | uploadedAt |

Deployment Plan

1. Set up AWS CDK infrastructure:
2. Cognito User Pool
3. DynamoDB tables
4. S3 bucket for media
5. API Gateway + Lambda functions
6. CloudFront distribution
7. Build Next.js frontend and deploy:
8. With ISR/SSG support
9. Auth integration with Cognito
10. CI/CD pipeline via GitHub Actions:
11. Test and linting
12. Auto deploy to Vercel or Amplify

Extensions (Optional for Future)

- Newsletter support (via SES)
- Comment system (moderated, optionally using Disqus or custom)
- Monetization options (ads, Stripe integration)
- Offline editing (PWA)
- Theme marketplace

Security Considerations

- Protect API endpoints with Cognito JWT validation
- Use fine-grained IAM permissions for S3 and Lambda
- Enable logging and monitoring for all services
- Rate limiting and anti-abuse filters

Deliverables

- Source code (frontend & backend)
- AWS CDK infrastructure code
- Deployment guide
- Demo video
- README documentation

Timeline (Example: 8 Weeks)

- Week 1: Project setup and planning
- Week 2–3: User auth + blog management
- Week 4–5: Post editor + media upload
- Week 6: Public blog views + SEO
- Week 7: Analytics dashboard + admin tools
- Week 8: Testing, deployment, and documentation

Success Criteria

- A user can register, create a blog, publish a post, and view it publicly
- Admin can moderate users/blogs
- Images load quickly via S3 + CloudFront
- Posts render SEO-friendly pages
- Platform is serverless and scalable