Specscart Full Stack Developer Assessment

Title: Next.js Blog Platform with Dynamic Custom Blocks

Create a full-featured blogging platform using Next.js (App Router) where users can create, read, and manage blog posts. The posts support dynamic {{block}} tags that are parsed and rendered as custom UI components.

Tech Stack

- Framework: Next.js 14+ (App Router)
- Language: TypeScript
- Styling: TailwindCSS (optional but preferred)

- Rendering: Server-Side Rendering (SSR) where appropriate
- Database: In-memory, JSON file, SQLite, or MongoDB
- Optional: Auth (JWT / NextAuth.js), Markdown parser

Core Features

1. Home Page (/)
• Server-rendered list of blog posts

• Each blog card should include: Title, Author name, Short snippet (first 200 characters or summary), Cover image (if available), Published date

2. Post Detail Page (/posts/[slug])
• SSR page showing full blog content

• Parse content for {{block ...}} tags and render corresponding UI components

Example Block:
{{block name="Top Picks" image="/top-products.png" products="SKU123,SKU456"}}

3. Create/Edit Blog Page (/create, /edit/[id])
• Form with fields: Title, Cover image URL, Author name, Blog body

• Client-side validations for required fields and valid image URL

• Save blog post to local file or in-memory DB

4. API Routes (under /api)

- GET /api/posts - Fetch all posts
- GET /api/posts/[id] - Fetch post by ID
- POST /api/posts - Create new post
- PUT /api/posts/[id] - Edit a post
- DELETE /api/posts/[id] - Delete a post

Block Parsing Logic

1. Use regex to detect {{block ...}} in the content

2. Extract attributes (name, image, products) from the tag

3. Replace the tag with a React component rendered via SSR

Mock Product Data

```
export const MOCK_PRODUCTS = [
```

```
  { sku: "SKU123", name: "Mechanical Keyboard", price: "$99", image: "/keyboard.png" },
  { sku: "SKU456", name: "Gaming Mouse", price: "$49", image: "/mouse.png" },
  { sku: "SKU789", name: "Monitor", price: "$199", image: "/monitor.png" },
];
```

Bonus Features (Optional but appreciated)

- Comment System - Allow comments under each post
- Dark Mode - Toggle between light/dark using Tailwind/CSS vars
- Markdown Support - Use marked or remark to support markdown
- SEO-Friendly Slugs - Use /posts/your-post-title instead of IDs
- Deployment - Deploy using Vercel, Fly.io, or Render
- Pagination/Infinite Scroll - Load More or infinite scroll
- Toast Notifications - Show toast for actions like post creation
- Animations - Enhance UI with Framer Motion or CSS transitions
- Search/Filter - Filter posts by keyword, author, or tags

Deliverables

• GitHub repository containing:

- App using app/ directory (App Router)
- API routes under /api/
- /data/products.ts with mock data
- Clean, maintainable codebase

• README.md with: Setup instructions, Tech stack used, Live demo link (if deployed)

Evaluation Criteria

- Functionality: All required features implemented
- SSR Usage: Appropriate use of server-side rendering
- Code Quality: Modular, DRY, maintainable code
- UI/UX: Clean design, responsiveness, and usability
- Block Parsing: Correct parsing and rendering of {{block}} components
- Bonus: Markdown, dark mode, animations, comments, etc.