Persona:

Anda adalah seorang pengembang perangkat lunak Full-Stack ahli dengan spesialisasi dalam membangun aplikasi web modern, cepat, dan skalabel. Anda sangat mahir dalam menggunakan React untuk frontend, Hono.js untuk backend, dan PostgreSQL untuk database. Anda menulis kode yang bersih, modular, dan memiliki komentar yang jelas.

Tujuan Utama:

Buat aplikasi web lengkap bernama "MLBB Random Build Generator" berdasarkan spesifikasi yang diberikan. Aplikasi ini harus memiliki frontend yang interaktif, backend yang cepat, database yang terstruktur, sistem autentikasi, dan fitur untuk menghasilkan gambar build.

Spesifikasi Teknis & Fitur:

1. Tumpukan Teknologi (Tech Stack):

- **Frontend:** React (menggunakan Vite) dengan Tailwind CSS untuk styling.
- **Backend:** Hono.js yang berjalan di lingkungan Node.js.
- **Database:** PostgreSQL.
- **Generator Gambar:** Menggunakan HTML Canvas API di sisi frontend untuk membuat gambar build secara dinamis.

2. Struktur Database (PostgreSQL):

Buat skema database dengan tabel-tabel berikut. Gunakan tipe data ARRAY untuk menyimpan daftar ID.

- roles:
    - id (SERIAL PRIMARY KEY)
    - role_name (VARCHAR(50), UNIQUE, NOT NULL) -- *Isi dengan 'user', 'admin', 'super-admin'*
- users:
    - id (UUID PRIMARY KEY, default: gen_random_uuid())
    - email (VARCHAR(255), UNIQUE, NOT NULL)
    - password_hash (VARCHAR(255), NOT NULL)
    - role_id (INTEGER, REFERENCES roles(id), NOT NULL)
    - created_at (TIMESTAMP, default: CURRENT_TIMESTAMP)
- heroes:

    - id (SERIAL PRIMARY KEY)
    - name (VARCHAR(100), NOT NULL)
    - lane (VARCHAR(50), NOT NULL) -- *Contoh: 'exp', 'mid', 'roam', 'jungle', 'gold'*
    - icon_url (TEXT)
- spells:
    - id (SERIAL PRIMARY KEY)
    - name (VARCHAR(100), NOT NULL)
    - icon_url (TEXT)
- items:
    - id (SERIAL PRIMARY KEY)
    - name (VARCHAR(100), NOT NULL)
    - category (VARCHAR(50), NOT NULL) -- *Contoh: 'physical', 'mage', 'tank'*
    - icon_url (TEXT)
- builds:
    - id (UUID PRIMARY KEY, default: gen_random_uuid())
    - user_id (UUID, REFERENCES users(id))
    - hero_id (INTEGER, REFERENCES heroes(id))
    - spell_id (INTEGER, REFERENCES spells(id))
    - item_ids (INTEGER[], NOT NULL) -- *Gunakan tipe data ARRAY*
    - image_url (TEXT) -- *URL ke gambar build yang disimpan (opsional)*
    - created_at (TIMESTAMP, default: CURRENT_TIMESTAMP)

3. API Backend (Hono.js):

Buat endpoint API berikut dengan routing yang jelas.

- **Autentikasi:**
    - POST /api/auth/register: Mendaftarkan pengguna baru.
    - POST /api/auth/login: Login pengguna dan mengembalikan JWT.
- **Data Game:**
    - GET /api/heroes: Mendapatkan semua data hero.

    - GET /api/spells: Mendapatkan semua data spell.
    - GET /api/items: Mendapatkan semua data item.
- **Generator & Galeri:**
    - POST /api/builds/generate: Menerima filter (lane, item category), menghasilkan build acak, menyimpannya ke database, dan mengembalikan data build. Endpoint ini harus dilindungi (memerlukan login).
    - GET /api/builds: Mendapatkan daftar build untuk galeri (dengan paginasi).
    - GET /api/builds/:id: Mendapatkan detail satu build.
    - DELETE /api/builds/:id: Menghapus build (hanya bisa dilakukan oleh pemilik build atau admin).

4. Fitur Frontend (React):

- **Halaman Utama (Generator):**
    - Tampilkan filter untuk lane hero dan kategori item.
    - Tombol "Generate" yang memanggil API /api/builds/generate.
    - Setelah mendapatkan hasil, tampilkan build (hero, spell, 6 item) secara visual.
    - Gunakan komponen <canvas> untuk menggambar hasil build menjadi sebuah gambar.
    - Tambahkan tombol "Unduh Gambar" untuk menyimpan isi canvas sebagai file PNG.
- **Halaman Galeri:**
    - Tampilkan semua build yang sudah dibuat dalam bentuk grid thumbnail.
    - Implementasikan fitur filter berdasarkan hero.
    - Setiap thumbnail bisa diklik untuk melihat detail build.
- **Halaman Login & Register:**
    - Formulir sederhana untuk login dan registrasi.
    - Arahkan pengguna yang belum login ke halaman ini jika mereka mencoba mengakses fitur yang dilindungi.
- **Manajemen State:** Gunakan React Context atau Zustand untuk manajemen state global (seperti status autentikasi pengguna).

5. Alur Kerja & Instruksi Tambahan:

1. **Mulai dengan Backend:** Siapkan proyek Hono.js, koneksi ke database PostgreSQL, dan buat skema tabel.
2. **Data Awal (Seeding):** Buat script untuk mengisi tabel heroes, spells, dan items dengan data placeholder (cukup 5-10 data per tabel untuk awal).
3. **Implementasi API:** Buat semua endpoint API yang telah didefinisikan, termasuk logika untuk autentikasi JWT dan middleware untuk proteksi rute.
4. **Lanjutkan dengan Frontend:** Buat proyek React dan rancang komponen UI sesuai fitur yang dijelaskan.
5. **Integrasi Frontend-Backend:** Hubungkan komponen React ke endpoint API Hono.js untuk mengambil data dan mengirim permintaan.
6. **Implementasi Canvas:** Buat logika untuk menggambar hasil build ke dalam elemen <canvas> setelah data diterima dari backend.
7. **Responsif:** Pastikan seluruh antarmuka pengguna (UI) sepenuhnya responsif dan terlihat bagus di perangkat mobile maupun desktop.
8. **Komentar:** Berikan komentar yang jelas pada bagian-bagian kode yang kompleks (logika bisnis, fungsi canvas, middleware).