PRD for VegaStack

Perfect 👍 Since you want a **developer-facing PRD**, I'll expand the document into something closer to a **system specification**:

- **API contracts** with request/response examples
- **Database schema (tables & relationships)**
- **Flow diagrams (where needed)**
- **Validation rules**
- **Error handling patterns**

📄 Developer-Facing PRD – SocialConnect

1. Introduction

SocialConnect is a social networking application built with **Next.js**, **Supabase (Postgres + Storage + Realtime)**, and **JWT authentication**.

It allows users to:

- Create accounts & profiles
- Post content (text + image)
- Interact socially (follows, likes, comments)
- Consume a personalized feed
- Receive real-time notifications
- Be moderated by admins

---

2. System Architecture

2.1 High-Level

- **Frontend**: Next.js + TailwindCSS (UI + API calls)
- **Backend**: Next.js API routes (REST-style endpoints)
- **Database**: Supabase PostgreSQL
- **Storage**: Supabase Storage (images)
- **Realtime**: Supabase Realtime subscriptions (notifications)
- **Auth**: JWT (access/refresh)

---

3. Database Schema

3.1 Users Table

```
CREATE TABLE users (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  email VARCHAR(255) UNIQUE NOT NULL,
  username VARCHAR(30) UNIQUE NOT NULL,
  password_hash TEXT NOT NULL,
  first_name VARCHAR(50),
  last_name VARCHAR(50),
  bio VARCHAR(160),
  avatar_url TEXT,
  website TEXT,
  location TEXT,
  role VARCHAR(10) DEFAULT 'user', -- 'user' or 'admin'
  is_active BOOLEAN DEFAULT TRUE,
  created_at TIMESTAMP DEFAULT NOW(),
  updated_at TIMESTAMP DEFAULT NOW()
);
```

3.2 Posts Table

```
CREATE TABLE posts (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
```

```
  author_id UUID REFERENCES users(id) ON DELETE CASCADE,
  content VARCHAR(280) NOT NULL,
  image_url TEXT,
  category VARCHAR(20) DEFAULT 'general',
  like_count INT DEFAULT 0,
  comment_count INT DEFAULT 0,
  is_active BOOLEAN DEFAULT TRUE,
  created_at TIMESTAMP DEFAULT NOW(),
  updated_at TIMESTAMP DEFAULT NOW()
);
```

3.3 Follows Table

```
CREATE TABLE follows (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  follower_id UUID REFERENCES users(id) ON DELETE CASCADE,
  following_id UUID REFERENCES users(id) ON DELETE CASCADE,
  created_at TIMESTAMP DEFAULT NOW(),
  UNIQUE(follower_id, following_id)
);
```

3.4 Likes Table

```
CREATE TABLE likes (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id UUID REFERENCES users(id) ON DELETE CASCADE,
  post_id UUID REFERENCES posts(id) ON DELETE CASCADE,
  created_at TIMESTAMP DEFAULT NOW(),
  UNIQUE(user_id, post_id)
);
```

3.5 Comments Table

```
CREATE TABLE comments (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  post_id UUID REFERENCES posts(id) ON DELETE CASCADE,
  author_id UUID REFERENCES users(id) ON DELETE CASCADE,
  content VARCHAR(200) NOT NULL,
  is_active BOOLEAN DEFAULT TRUE,
  created_at TIMESTAMP DEFAULT NOW()
);
```

3.6 Notifications Table

```
CREATE TABLE notifications (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  recipient_id UUID REFERENCES users(id) ON DELETE CASCADE,
  sender_id UUID REFERENCES users(id) ON DELETE CASCADE,
  notification_type VARCHAR(20) CHECK (notification_type IN ('follow', 'like',
'comment')),
  post_id UUID REFERENCES posts(id),
  message VARCHAR(200),
  is_read BOOLEAN DEFAULT FALSE,
  created_at TIMESTAMP DEFAULT NOW()
);
```

---

4. API Endpoints

4.1 Authentication

Register – `POST /api/auth/register`

Request

```
{
  "email": "test@example.com",
  "username": "test_user",
  "password": "StrongPass123",
  "first_name": "John",
  "last_name": "Doe"
}
```

Response (201)

```
{
  "message": "User registered. Please verify email."
}
```

Login – POST /api/auth/login

Request

```
{
  "username_or_email": "test_user",
  "password": "StrongPass123"
}
```

Response (200)

```
{
  "access_token": "jwt-access-token",
  "refresh_token": "jwt-refresh-token",
  "user": {
    "id": "uuid",
    "username": "test_user",
    "email": "test@example.com"
```

```
  }
}
```

4.2 Users & Profiles

Get Profile – `GET /api/users/{id}`

Response

```
{
  "id": "uuid",
  "username": "test_user",
  "bio": "Tech enthusiast",
  "avatar_url": "https://cdn.supabase.co/.../avatar.png",
  "followers_count": 120,
  "following_count": 80,
  "posts_count": 50
}
```

Update Profile – `PUT /api/users/me`

Request

```
{
  "bio": "Updated bio",
  "website": "https://mysite.com"
}
```

4.3 Posts

Create Post – `POST /api/posts`

Request

```
{
  "content": "Hello world! 🚀",
  "category": "general",
  "image_url": "https://cdn.supabase.co/post.png"
}
```

Response

```
{
  "id": "uuid",
  "content": "Hello world! 🚀",
  "author": { "id": "uuid", "username": "test_user" },
  "created_at": "2025-08-27T12:00:00Z"
}
```

4.4 Follows

- Follow: `POST /api/users/{id}/follow`
- Unfollow: `DELETE /api/users/{id}/follow`
- Followers list: `GET /api/users/{id}/followers`
- Following list: `GET /api/users/{id}/following`

4.5 Likes & Comments

- Like Post: `POST /api/posts/{id}/like`
- Unlike Post: `DELETE /api/posts/{id}/like`
- Comment: `POST /api/posts/{id}/comments`
- Get Comments: `GET /api/posts/{id}/comments`

4.6 Feed

Personalized Feed – `GET /api/feed?page=1`

Response

```
{
  "page": 1,
  "posts": [
    {
      "id": "uuid",
      "content": "My first post",
      "author": { "id": "uuid", "username": "user123" },
      "like_count": 5,
      "comment_count": 2,
      "liked_by_user": true,
      "created_at": "2025-08-27T10:00:00Z"
    }
  ]
}
```

4.7 Notifications

Get Notifications – `GET /api/notifications`

Response

```
[
  {
    "id": "uuid",
    "type": "like",
    "sender": { "id": "uuid", "username": "jane_doe" },
    "message": "Jane liked your post",
    "is_read": false,
    "created_at": "2025-08-27T10:05:00Z"
  }
```

```
]
```

4.8 Admin Endpoints

- `GET /api/admin/users` → list users
- `POST /api/admin/users/{id}/deactivate` → deactivate user
- `GET /api/admin/posts` → list all posts
- `DELETE /api/admin/posts/{id}` → delete post
- `GET /api/admin/stats` → total users, posts, daily actives

5. Validation Rules

- Username: 3–30 chars, alphanumeric + `_` only
- Password: ≥8 chars, must include uppercase, lowercase, number
- Post Content: ≤280 chars
- Comment Content: ≤200 chars
- Image Upload: JPEG/PNG, ≤2MB

6. Error Handling

- **400** – Validation failed (e.g., username too short)
- **401** – Unauthorized (invalid/missing token)
- **403** – Forbidden (not owner/admin)
- **404** – Resource not found
- **500** – Internal server error

Error Format:

```
{
  "error": "ValidationError",
```

```
  "message": "Username already taken"
}
```