Backend Assignment: Build the Backend for Linktr.ee / Bento.me with Referral System

Objective:

Build the backend for a platform similar to Linktr.ee or Bento.me, incorporating user registration, login, referral system, and other essential functionalities. The backend should be secure, efficient, and scalable.

Requirements:

User Registration & Authentication:

Implement a user registration system where users can sign up using email, username, and password. Include validation for email format, password strength, and duplicate usernames/emails.
Implement a login system for user authentication using email/username and password.
Use JWT (JSON Web Tokens) or OAuth 2.0 for secure user authentication and token management.
Implement a password reset system with email verification and token expiration.

Referral System:

Implement a referral system where existing users can refer new users to sign up.
For each referral, generate a unique referral link for the existing user (e.g., https://yourdomain.com/register?referral=USER_ID).
Track the number of successful sign-ups from each referral link.
Optionally, implement a reward system for the referrer (e.g., credits, free premium features, etc.), and track the reward logic in the database.

Password Management:

Use a secure hashing algorithm (e.g., bcrypt or Argon2) for storing user passwords.
Ensure password recovery is handled via secure tokens and email.

API Endpoints:

Create RESTful API endpoints for the following actions:
POST /api/register: Register a new user, accepting email, username, password, and referral code (if applicable).
POST /api/login: Authenticate user credentials and return a JWT token.
POST /api/forgot-password: Handle password recovery requests.
GET /api/referrals: Fetch the list of users referred by the logged-in user (if needed).
GET /api/referral-stats: Retrieve statistics related to the referral system (e.g., number of successful sign-ups from the user's referrals).

Data Validation & Error Handling:
Validate all user inputs (e.g., email format, password length, referral code validity).
Provide detailed error messages for invalid inputs or failed actions (e.g., “Email already in use”, “Invalid referral code”).
Handle unexpected errors gracefully with appropriate HTTP status codes (e.g., 500 for server errors).

Database Design:
Users Table: Store user data including id, username, email, password_hash, referral_code, referred_by, created_at.
Referrals Table: Track the referrals by storing referrer_id, referred_user_id, date_referred, and status (pending, successful, etc.).
Create relationships between tables, linking referred users to their referrers and ensuring the integrity of the referral system.
Optionally, add a Rewards Table to track the rewards or incentives earned by referrers (if implementing rewards).

Security:
Implement protections against common vulnerabilities like SQL injection, XSS (Cross-Site Scripting), and CSRF (Cross-Site Request Forgery).
Use rate limiting on the login and registration endpoints to prevent brute-force attacks.
Store JWT tokens securely and use HttpOnly cookies to prevent XSS vulnerabilities.

Session Management:
Use JWT or sessions for maintaining user authentication across pages. Ensure tokens are stored securely (e.g., in HttpOnly cookies).

Performance & Scalability:
Ensure the backend can handle a large number of concurrent users, especially for the referral system, which can lead to high traffic spikes.
Implement caching for frequently accessed data (e.g., user details, referral stats) to improve performance.
Consider horizontal scaling or load balancing if necessary to handle growth.

Testing:
Write unit and integration tests for API endpoints, particularly for registration, login, and referral system logic.
Test edge cases for the referral system, such as invalid referral codes, users trying to refer themselves, and tracking referral counts accurately.
Ensure proper validation for user input and handle edge cases (e.g., already registered users, expired referral tokens).