Gemini CLI Prompts with Corresponding Changes

1. Analyze this Next.js API route and identify possible causes for 500 internal server errors.

Changes Made:

- Wrapped all API routes (chat, events, restaurants, wallet, user) in try/catch blocks.
- Added proper error logging using console.error() for debugging.
- Checked for session existence using getServerSession or auth() to avoid unauthorized errors causing 500s.
- Added validation for route parameters like id in [id]/route.js files.

2. Check the NextAuth integration in this Next.js project for misconfigurations or session issues.

Changes Made:

- Ensured NEXTAUTH_URL and NEXTAUTH_SECRET are set in .env.
- Replaced invalid imports in login (signInasPost) with correct signIn("credentials").
- Added checks in APIs to return 401 Unauthorized if the session is missing.
- Updated login and signup routes to handle NextAuth responses properly.

3. Identify potential causes for form submission failures in this React + Next.js frontend and suggest fixes.

Changes Made:

- Updated /user/complete-profile and /user/update-profile routes to handle formData() correctly.
- Added validation using zodUserDetailsSchema to ensure correct types (e.g., phone as number).
- Checked for missing or invalid fields before saving to the database.
- Added proper success/error responses for frontend consumption.

4. Review this file upload logic for the Seller API and suggest ways to handle large images safely and efficiently.

Changes Made:

- Updated POST /api/Seller to convert formData image to Base64 and upload to Cloudinary.
- Checked upload success and returned proper error if it failed.
- Validated input using Zod schemas (zodFlatInputSchema, zodStationaryInputSchema).
- Ensured only valid Seller users can create entries.

5. Provide recommendations for improving validation and error responses for registration and login APIs.

Changes Made:

- Added Zod validation (zodLoginSchema, zodUserSchema) for login/signup data.
- Used fromZodError to return readable validation messages.
- Added conflict checks for existing email/username during signup.
- Standardized error responses with errorResponse() and success responses with successResponse().

6. Suggest all database schemas required for my Next.js project with user authentication, events, chats, restaurants, wallets, and seller products.

Changes Made:

- Created **User schema** with fields: username, email, role, wallet, registrationCompleted, isVerified.
- **Event schema** with status, createdBy, startDate, endDate.
- **Chat schema** with sender, receiver, entityId, entityType, messages, lastMessage.
- **Restaurant schema** with cuisine, rating, location, status, createdBy, college.
- **Seller product schemas**: Flat and Stationary.

- **Request schema** for seller fetch queries.

7. Design endpoints for the entire app including authentication, user management, events, chats, restaurants, wallets, and seller product management.

Endpoints Implemented:

- **Auth & Registration:** api/register/[requestType]/route.js → POST (login & signup)
- **User Management:**
  - api/user/complete-profile/route.js → POST
  - api/user/update-profile/route.js → GET, PUT
- **Events:** api/events/[id]/route.js → GET (admin/creator permission check)
- **Chats:** api/chat/route.js → GET, POST, PATCH
- **Restaurants:**
  - api/restaurants/route.js → GET, POST
  - api/restaurants/[id]/route.js → GET
- **Wallet:** api/wallet/route.js → GET, POST
- **Seller Products:**
  - api/Seller/route.js → POST (add product)
  - api/Seller/[fetchItem]/route.js → GET (fetch by model)

8. Design app architecture for a full-stack Next.js project with MongoDB, authentication, file uploads, and admin approval flow.

Changes Made:

- **Folder Structure:**
  - app/ → pages and API routes
  - api/ → REST API endpoints
  - lib/ → DB connection, models, helpers
  - auth/ → authentication config

    - components/ → reusable UI
- **Backend Architecture:**
    - MongoDB for persistence
    - NextAuth for authentication
    - Cloudinary for image uploads
    - Admin approval workflow for Seller products
- **Frontend Architecture:**
    - Next.js pages consuming API routes
    - Conditional rendering based on roles and registration status