Prompt for AI

> Review the **entire Pandiyatra project** including frontend React app, Django backend, Docker setup, and PostgreSQL database. Perform the following tasks carefully:

---

1. Frontend (React) Review

- Analyze folder structure and file organization. Suggest missing folders/files based on clean React architecture (components, pages, services, utils, hooks, assets, env files, etc.).
- Check all API calls to the backend. List all used endpoints. Highlight missing API calls or incorrectly implemented calls.
- Check environment files (`.env`) and configuration.
- Suggest improvements for code readability, structure, and best practices.

---

2. Backend (Django + DRF) Review

- Review folder structure, Django apps, models, serializers, views, urls, and settings.
- List **all API endpoints currently implemented**, including:
    - HTTP method (GET, POST, PUT, DELETE)
    - URL/path
    - Purpose/description
    - Connected model (if applicable)
- Identify **missing APIs** or incomplete endpoints required for the frontend tasks.
- Review environment settings (`.env`) for database, secret keys, and other sensitive data.
- Ensure **clean Django architecture**, proper model relationships, and use of serializers.

- Suggest any refactoring for code cleanliness and maintainability.

---

3. Database (PostgreSQL via Docker)

- List **all current database tables** along with columns, data types, and relationships.
- Identify missing tables/fields required by backend models.
- Provide SQL scripts or Django migrations to create missing tables.
- Review Docker Compose file or Docker setup and ensure proper port mapping and volumes.

---

4. Docker Review

- Ensure Docker files (backend Dockerfile, db container) are correct.
- Check Docker Compose configuration for correct services, ports, dependencies, and volume mounts.
- Suggest any optimizations.

---

5. Full API Flow Verification

- For each backend API endpoint, verify if it is connected properly with the frontend.
- List endpoints by feature or folder.
- Identify missing endpoints or incorrect implementations.
- Suggest corrections and improvements.
- Prepare a **full API mapping table**, including:
    - Endpoint URL

    - HTTP method
    - Purpose
    - Frontend usage
    - Model/table involved

---

6. Agile / Project Management Integration

- Organize tasks and endpoints as per sprint workflow.
- Prepare a **Gantt chart structure** for backend and frontend tasks based on API development, database setup, and frontend integration.

---

7. Deliverables

- **Full list of API endpoints** (current and missing) with HTTP methods.
- **Full list of database tables** with columns, data types, and relationships.
- Missing frontend and backend files/folders with suggested structure.
- Clean Docker setup and PostgreSQL database tables.
- Recommendations for clean code architecture for React frontend and Django backend.
- Suggested task list for sprints, aligned with Gantt chart.

Instruction: Review the project carefully **file by file and folder by folder**. Suggest **exact code/files/migrations to add** wherever missing. Provide instructions for testing APIs with Postman. Ensure the final project is clean, complete, and fully functional.

1️⃣ Full Database Tables (PostgreSQL)

User

Column	Type	Constraints	Description
user_id	UUID / SERIAL PK	PRIMARY KEY	Unique user ID
user_name	VARCHAR(100)	NOT NULL	User’s full name
user_email	VARCHAR(100)	UNIQUE, NOT NULL	Email for login
user_phone	VARCHAR(15)	UNIQUE, NOT NULL	Phone number
user_address	TEXT		Address
user_role	VARCHAR(20)	NOT NULL	Role: USER/PANDIT/ADMIN
timezone	VARCHAR(50)		Timezone of user
profile_pic_url	TEXT		Profile picture URL
password	VARCHAR(128)	NOT NULL	Hashed password

PanditProfile

Column	Type	Constraints	Description
pandit_id	UUID / SERIAL PK	PRIMARY KEY	Unique pandit ID
user_id	UUID	FK → User(user_id)	Linked user account
experience_years	INT		Experience in years
languages	TEXT[]		List of languages
bio	TEXT		Short bio
certificate_url	TEXT		Verification certificate
is_verified	BOOLEAN	DEFAULT FALSE	Verified or not

rating	FLOAT		Average rating

Occasion

Column	Type	Constraints	Description
occasion_id	UUID / SERIAL PK	PRIMARY KEY	Unique ID
occasion_name	VARCHAR(100)	NOT NULL	Name of occasion
occasion_description	TEXT		Details
base_price_npr	NUMERIC(10,2)	NOT NULL	Price in NPR
base_price_usd	NUMERIC(10,2)	NOT NULL	Price in USD
image_url	TEXT		Image

Booking

Column	Type	Constraints	Description
booking_id	UUID / SERIAL PK	PRIMARY KEY	Unique booking ID
user_id	UUID	FK → User(user_id)	Customer
pandit_id	UUID	FK → PanditProfile(pandit_id)	Assigned pandit
occasion_id	UUID	FK → Occasion(occasion_id)	Occasion booked
datetime_user	TIMESTAMP	NOT NULL	User’s local datetime
datetime_nepal	TIMESTAMP	NOT NULL	Nepal timezone datetime
timezone	VARCHAR(50)		User timezone
status	VARCHAR(20)	DEFAULT 'PENDING'	Booking status
total_npr	NUMERIC(10,2)		Total price in NPR

total_usd	NUMERIC(10,2)	Total price in USD
currency	VARCHAR(5)	'NPR'/'USD'

Samagri

Column	Type	Constraints	Description
samagri_id	UUID / SERIAL PK	PRIMARY KEY	Unique ID
name	VARCHAR(100)	NOT NULL	Item name
price_npr	NUMERIC(10,2)	NOT NULL	Price in NPR
price_usd	NUMERIC(10,2)	NOT NULL	Price in USD
stock	INT	DEFAULT 0	Quantity
image_url	TEXT		Image URL
occasion_id	UUID	FK → Occasion(occasion_id)	Related occasion

Book

Column	Type	Constraints	Description
book_id	UUID / SERIAL PK	PRIMARY KEY	Unique book ID
title	VARCHAR(100)	NOT NULL	Book title
price_npr	NUMERIC(10,2)	NOT NULL	Price in NPR
price_usd	NUMERIC(10,2)	NOT NULL	Price in USD
borrow_days	INT		Duration for borrowing
pdf_url	TEXT		PDF download link
type	VARCHAR(50)		Type: EBOOK/PHYSICAL
stock	INT	DEFAULT 0	Quantity available

CartItem

Column	Type	Constraints	Description
cart_item_id	UUID / SERIAL PK	PRIMARY KEY	Unique item in cart
user_id	UUID	FK → User(user_id)	Owner of cart
booking_id	UUID	FK → Booking(booking_id)	Linked booking (optional)
samagri_id	UUID	FK → Samagri(samagri_id)	Purchased item (optional)
book_id	UUID	FK → Book(book_id)	Purchased book (optional)
quantity	INT	DEFAULT 1	Quantity
type	VARCHAR(20)		'BOOK'/'SAMAGRI'/'BOOKING'

Payment

Column	Type	Constraints	Description
payment_id	UUID / SERIAL PK	PRIMARY KEY	Unique payment ID
user_id	UUID	FK → User(user_id)	Paid by user
gateway	VARCHAR(50)		Stripe/Khalti
amount	NUMERIC(10,2)	NOT NULL	Amount paid
currency	VARCHAR(5)	NOT NULL	'NPR'/'USD'
status	VARCHAR(20)		SUCCESS/FAILED/PENDING
transaction_id	VARCHAR(100)	UNIQUE	Gateway txn ID

VideoRoom

Column	Type	Constraints	Description
room_id	UUID / SERIAL PK	PRIMARY KEY	Unique video room
booking_id	UUID	FK → Booking(booking_id)	Linked booking

room_url	TEXT	Video call URL
recording_url	TEXT	Recording URL
participants	INT	Number of participants
start_time	TIMESTAMP	Start time
end_time	TIMESTAMP	End time

ChatMessage

Column	Type	Constraints	Description
message_id	UUID / SERIAL PK	PRIMARY KEY	Unique message
booking_id	UUID	FK → Booking(booking_id)	Related booking
sender_id	UUID	FK → User(user_id)	Sender
text	TEXT		Message content
image_url	TEXT		Optional image
timestamp	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Message timestamp

---

Review

Column	Type	Constraints	Description
review_id	UUID / SERIAL PK	PRIMARY KEY	Unique review
booking_id	UUID	FK → Booking(booking_id)	Linked booking
rating	INT		1–5 rating
comment	TEXT		Feedback text
timestamp	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Review time

Kundali

Column	Type	Constraints	Description
kundali_id	UUID / SERIAL PK	PRIMARY KEY	Unique kundali
user_id	UUID	FK → User(user_id)	User linked
dob	DATE	NOT NULL	Date of birth
time	TIME	NOT NULL	Time of birth
place	TEXT		Birthplace
planets	JSON		Planet positions
predictions	TEXT		Generated predictions
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Created timestamp

---

AdminRole

Column	Type	Constraints	Description
role_id	UUID / SERIAL PK	PRIMARY KEY	Unique role ID
user_id	UUID	FK → User(user_id)	Assigned admin
name	VARCHAR(50)		Role name
permissions	JSON		Permissions list

Alert

Column	Type	Constraints	Description
alert_id	UUID / SERIAL PK	PRIMARY KEY	Unique alert
pandit_id	UUID	FK → PanditProfile(pandit_id)	Alert for pandit
type	VARCHAR(50)		Type of alert
message	TEXT		Alert content

language	VARCHAR(10)		Alert language
timestamp	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Alert time

2️⃣ Full API List (REST)

Authentication

Method	Endpoint	Description	Request Body
POST	/api/auth/register/	Register user	{ userName, userEmail, userPhone, password }
POST	/api/auth/login/	Login (JWT)	{ userEmail, password }
POST	/api/auth/forgot-password/	Request password reset	{ userEmail }
POST	/api/auth/change-password/	Change password	{ oldPassword, newPassword }
POST	/api/auth/send-otp/	Send OTP	{ userPhone }
POST	/api/auth/verify-otp/	Verify OTP	{ userPhone, otp }
GET	/api/users/me/	Get logged-in user	JWT required
PUT	/api/users/me/	Update profile	{ userName, userPhone, userAddress, profilePicURL }

Pandits

Method	Endpoint	Description	Body / Query
GET	/api/pandits/	List all verified pandits	Query: language, experienceYears
GET	/api/pandits/<id>/	Get pandit detail	-
POST	/api/pandits/	Create pandit profile	{ experienceYears, bio, certificateURL, languages: [] }

PUT	/api/pandits/<id>/	Update pandit profile	Same as POST
DELETE	/api/pandits/<id>/	Delete pandit	Admin only

Occasions

Method	Endpoint	Description	Body / Query
GET	/api/occasions/	List occasions	-
GET	/api/occasions/<id>/	Occasion detail	-
POST	/api/occasions/	Add occasion	{ name, description, imageURL }
PUT	/api/occasions/<id>/	Update occasion	Same as POST
DELETE	/api/occasions/<id>/	Delete occasion	Admin only

Samagri & Books

Method	Endpoint	Description
GET	/api/samagri/	List samagri (filter: occasion)
GET	/api/books/	List books (filter: type)
GET	/api/samagri/<id>/	Samagri detail
GET	/api/books/<id>/	Book detail
POST / PUT / DELETE	/api/samagri/ & /api/books/	Admin only

Bookings & Cart

Method	Endpoint	Description
POST	/api/bookings/	Create booking
GET	/api/bookings/	List user bookings
GET	/api/bookings/<id>/	Booking detail

PUT	/api/bookings/<id>/	Update booking
DELETE	/api/bookings/<id>/	Cancel booking
POST	/api/cart/	Add cart item
GET	/api/cart/<orderID>/	Get cart
DELETE	/api/cart/<cartItemID>/	Remove cart item
POST	/api/orders/	Create order (cart + booking)
GET	/api/orders/	List user orders
GET	/api/orders/<id>/	Order detail
PUT	/api/orders/<id>/	Update order status
DELETE	/api/orders/<id>/	Cancel order

Payments

Method	Endpoint	Description
POST	/api/payments/	Make payment { orderID, gateway, amount, currency }
GET	/api/payments/<id>/	Payment status
GET	/api/orders/<id>/payments/	Payments for an order

Video & Chat

Method	Endpoint	Description
POST	/api/videorooms/	Create room
GET	/api/videorooms/<bookingID>/	Get room info
PUT	/api/videorooms/<roomID>/	Update room (recording)
POST	/api/videoparticipants/	Add participant
GET	/api/videoparticipants/<roomID>/	List participants

POST	/api/chat/	Send message
GET	/api/chat/<bookingID>/	Get messages

Review, Kundali & Alerts

Method	Endpoint	Description
POST	/api/reviews/	Add review for booking
GET	/api/reviews/<bookingID>/	List reviews
POST	/api/kundali/	Create kundali
GET	/api/kundali/<userID>/	Get user kundalis
POST	/api/alerts/	Create alert (admin/pandit)
GET	/api/alerts/<panditID>/	List alerts

✅ This table covers **all database tables + full REST API endpoints** mapped to models.

Day	Tasks
Day 1	Finalize **database schema** with tables, columns, relationships. Generate Django models & migrations.
Day 2	Implement **authentication** APIs (register, login, JWT, OTP/email). Test with Postman.
Day 3	Implement **Pandit profile** APIs & **Occasions/Services** endpoints. Connect with serializers.
Day 4	Implement **Booking & Cart** APIs, including linking bookings to users & services.
Day 5	Implement **Payments** integration (Stripe/Khalti) + API endpoints. Test payments workflow.
Day 6	Implement **Video Rooms & Chat** APIs + Reviews + Kundali endpoints.

Day 7	Full **integration testing**, generate Postman collection, finalize Docker setup, ensure frontend can connect.