

Complete Authentication & User API Endpoints

1 User Registration

Method: **POST**

Endpoint: **/api/auth/register/**

Description: Registers a new user.

Request Body:

```
{  
  
  "userName": "JohnDoe",  
  
  "userEmail": "johndoe@example.com",  
  
  "userPhone": "1234567890",  
  
  "userAddress": "123 Main St",  
  
  "password": "strongpassword"  
}
```

Response:

```
{  
  
  "message": "User registered successfully"  
  
}
```

2 User Login

Method: **POST**

Endpoint: **/api/auth/login/**

Description: Logs in a user and returns a JWT token.

Request Body:

```
{  
  
  "userEmail": "johndoe@example.com",  
  
  "password": "strongpassword"  
  
}
```

Response:

```
{
```

```
    "token": "jwt_token_here"
  }
```

③ Get Logged-in User Profile

Method: **GET**

Endpoint: **/api/users/me/**

Description: Retrieves the logged-in user's profile details.

Authentication: Requires JWT token in the Authorization header.

Request Header:

Authorization: Bearer <JWT_TOKEN>

Response:

```
{
  "userName": "JohnDoe",
  "userEmail": "johndoe@example.com",
  "userPhone": "1234567890",
```

```
"userAddress": "123 Main St"

}
```

4 Update User Profile

Method: **PUT**

Endpoint: **/api/users/me/**

Description: Updates the logged-in user's profile details.

Authentication: Requires JWT token in the Authorization header.

Request Body:

```
{

  "userName": "JohnDoeUpdated",

  "userPhone": "9876543210",

  "userAddress": "456 New St",

  "profilePicURL":
    "http://example.com/profilepic.jpg"

}
```

Response:

```
{  
  
  "message": "Profile updated successfully"  
  
}
```

5 Forgot Password

Method: POST

Endpoint: /api/auth/password/forgot/

Description: Requests a password reset by sending an OTP or reset link to the user's email.

Request Body:

```
{  
  
  "userEmail": "johndoe@example.com"  
  
}
```

Response:

```
{  
  
  "message": "Password reset link sent to email"  
  
}
```

6 Reset Password Using OTP/Token

Method: **POST**

Endpoint: **/api/auth/password/reset/**

Description: Resets the password using the OTP/token sent to the user's email.

Request Body:

```
{  
  
  "token": "otp_token_here",  
  
  "newPassword": "newstrongpassword"  
  
}
```

Response:

```
{  
  
  "message": "Password reset successfully"  
  
}
```

7 Send OTP for Verification

Method: **POST**

Endpoint: **/api/auth/otp/send/**

Description: Sends an OTP to the user's email for verification.

Request Body:

```
{  
  
  "userEmail": "johndoe@example.com"  
  
}
```

Response:

```
{  
  
  "message": "OTP sent to email"  
  
}
```

```
}
```

8 Verify OTP

Method: **POST**

Endpoint: **/api/auth/otp/verify/**

Description: Verifies the OTP sent to the user's email.

Request Body:

```
{  
  
  "userEmail": "johndoe@example.com",  
  
  "otp": "123456"  
}
```

Response:

```
{  
  
  "message": "OTP verified successfully"  
}
```

9 Change Password

Method: **POST**

Endpoint: **/api/auth/password/change/**

Description: Allows a logged-in user to change their password by providing the current password and a new password.

Authentication: Requires JWT token in the Authorization header.

Request Body:

```
{  
  
  "currentPassword": "old_password_here",  
  
  "newPassword": "new_password_here",  
  
  "confirmPassword": "new_password_here"  
}
```

Response (Success):

```
{  
  
  "message": "Password changed successfully"  
}
```

```
}
```

Response (Error - Current Password Incorrect):

```
{
```

```
  "error": "Current password is incorrect"
```

```
}
```

Response (Error - New Password Mismatch):

```
{
```

```
  "error": "New password and confirm password do  
not match"
```

```
}
```

Response (Error - Password Does Not Meet Security Requirements):

```
{
```

```
"error": "New password does not meet security requirements"
```

```
}
```

1 Authentication & Users

Method	Endpoint	Description	Request Body / Params
POST	/api/auth/register/	Register new user	{ "userName", "userEmail", "userPhone", "userAddress", "password" }
POST	/api/auth/login/	Login user (JWT/Token)	{ "userEmail", "password" }

GET	<code>/api/users/me/</code>	Get logged-in user profile	Auth required
PUT	<code>/api/users/me/</code>	Update profile	<code>{ "userName", "userPhone", "userAddress", "profilePicURL" }</code>
POST	<code>/api/auth/password/forgot/</code>	Request password reset	<code>{ "userEmail" }</code>
POST	<code>/api/auth/password/reset/</code>	Reset password using OTP/token	<code>{ "token", "newPassword" }</code>

POST	/api/auth/otp/send/	Send OTP for verification	{ "userEmail" }
POST	/api/auth/otp/verify/	Verify OTP	{ "userEmail", "otp" }

2 Roles & Pandits

Method	Endpoint	Description	Request Body / Params
GET	/api/pandits/	List all verified pandits	Optional query: language, experienceYears

GET	<code>/api/pandits/<id>/</code>	Get pandit details	-
POST	<code>/api/pandits/</code>	Create pandit profile	<code>{ "experienceYears", "bio", "certificateURL", "languages": [] }</code>
PUT	<code>/api/pandits/<id>/</code>	Update pandit profile	Same as POST
DELETE	<code>/api/pandits/<id>/</code>	Delete pandit profile	Admin only

3 Occasions, Samagri & Books

Occasions

Method	Endpoint	Description	Request Body / Params
GET	<code>/api/occasions/</code>	List all occasions	-
GET	<code>/api/occasions/<id>/</code>	Occasion detail + price	-
POST	<code>/api/occasions/</code>	Add occasion	<code>{ "occasionName", "occasionDescription", "imageUrl" }</code>
PUT	<code>/api/occasions/<id>/</code>	Update occasion	Same as POST

DELETE	<code>/api/occasions/<id>/</code>	Delete occasion	Admin only
--------	---	-----------------	------------

Samagri & Books

Method	Endpoint	Description
GET	<code>/api/samagri/</code>	List all samagri (optional filter by occasion)
GET	<code>/api/books/</code>	List all books (optional filter by type)
GET	<code>/api/samagri/<id>/</code>	Samagri detail + price
GET	<code>/api/books/<id>/</code>	Book detail + price
POST/PUT/DELETE	<code>/api/samagri/</code> & <code>/api/books/</code>	Admin only, standard CRUD

4 Orders, Cart & Booking

Orders

Method	Endpoint	Description	Request Body / Params
POST	<code>/api/orders/</code>	Create new order (cart items + optional booking)	<pre>{ "userID", "items": [{ "itemType", "itemID", "quantity" }], "bookingID": optional }</pre>
GET	<code>/api/orders/</code>	List user orders	-
GET	<code>/api/orders/<id>/</code>	Order detail including bookings & cart items	-

PUT	/api/orders/<id>/	Update order status / payment confirmation	{ "status", "paymentStatus" }
DELETE	/api/orders/<id>/	Cancel order	-

Cart

Method	Endpoint	Description	Request Body / Params
POST	/api/cart/	Add item to cart	{ "orderId", "itemType", "itemID", "quantity" }
GET	/api/cart/<orderId>/	Get cart	-

		details	
DELETE	/api/cart/<cartItemID>/	Remove item	-

Booking

Method	Endpoint	Description	Request Body / Params
POST	/api/bookings/	Create booking (pandit + occasion, optional orderID)	{ "userID", "panditID", "occasionID" , "datetime", "orderID": optional }
GET	/api/bookings/	List user bookings	-

GET	<code>/api/bookings/<id>/</code>	Booking details (video room info included)	-
PUT	<code>/api/bookings/<id>/</code>	Update booking (reschedule/status)	<code>{ "datetime", "status" }</code>
DELETE	<code>/api/bookings/<id>/</code>	Cancel booking	-

5 Payments (Khalti / Stripe)

Method	Endpoint	Description	Request Body / Params
--------	----------	-------------	-----------------------

POST	/api/payments/	Make payment	{ "orderId", "gateway", "amount", "currency" }
GET	/api/payments/<id>/	Payment status	-
GET	/api/orders/<id>/payments/	Payments for an order	-

6 Video Rooms & Chat

VideoRoom

Method	Endpoint	Description	Request Body / Params
GET	/api/videorooms/<bookingID>/	Room URL &	-

		participants	
POST	/api/videorooms/ /	Create room for booking	{ "bookingID", "panditID" }
PUT	/api/videorooms/ /<roomID>/	Update room / add recording URL	{ "recordingURL", "endTime" }

Video Participants

Method	Endpoint	Description	Request Body / Params
POST	/api/videoparticipants/	Add participant	{ "roomID", "userID" }

GET	/api/videoparticipants/<roomID>/	List participants	-
-----	----------------------------------	-------------------	---

ChatMessage

Method	Endpoint	Description	Request Body / Params
POST	/api/chat/	Send message	{ "bookingID", "senderID", "text", "imageUrl" }
GET	/api/chat/<bookingID>/	Fetch chat for booking	-

7 Review, Kundali & Alerts

Review

Method	Endpoint	Description	Request Body / Params
POST	<code>/api/reviews/</code>	Add review for booking	<code>{ "bookingID", "rating", "comment" }</code>
GET	<code>/api/reviews/<bookingID>/</code>	List reviews for a booking	-

Kundali

Method	Endpoint	Description	Request Body / Params
POST	<code>/api/kundali/</code>	Create Kundali	<code>{ "dob", "time", "place", "userID" }</code>

GET	/api/kundali/ /<userID>/	Get user Kundalis	-
-----	-----------------------------	-------------------	---

Alert

Method	Endpoint	Description	Request Body / Params
GET	/api/alerts/ <panditID> /	List alerts for pandit	-
POST	/api/alerts/	Create alert (admin/pandit)	{ "panditID", "type", "message", "language" }

✓ Notes for Exam

1. Follow **REST principles**: resources = nouns, HTTP verbs for actions.

2. Use **JWT token authentication**.
3. Support **pagination & filtering** on large lists (Pandits, Orders).
4. **Booking optionally linked to orderID** → allows buying items only.
5. **Payments linked to orderID** → unified checkout for books, samagri, and pandit booking.