Sequence Diagrams and Flow Details

1) Subscription Flow (User + Mobile Client + Backend + Stripe)  
Description: Covers new user registration, trial application, entering payment method, subscription creation and webhook handling.

Step-by-step:  
1. User registers on mobile app (email/password).  
2. Mobile calls POST /auth/register -> Backend creates user and sets trial\_start/trial\_end (7 days).  
3. User proceeds to subscribe (before or after trial) -> mobile collects card using Stripe SDK and receives payment\_method\_id.  
4. Mobile sends payment\_method\_id to backend: POST /stripe/create-subscription {user\_id, payment\_method\_id, plan\_id}.  
5. Backend creates or retrieves Stripe Customer, attaches PaymentMethod, and creates Subscription with desired price ID and trial\_period\_days (if required).  
6. Stripe returns subscription object. Backend stores stripe\_subscription\_id and current\_period\_end.  
7. Stripe sends webhooks (invoice.paid, payment\_intent.succeeded, invoice.payment\_failed) to /webhook/stripe.  
8. Backend verifies webhook signature, updates subscription/payment records and notifies user via push/email.

ASCII Sequence (simple):  
User -> Mobile App: register()  
Mobile App -> Backend: POST /auth/register  
Backend -> Database: create user, trial\_start/end  
User -> Mobile App: enter card (Stripe SDK)  
Mobile App -> Stripe: create PaymentMethod (client-side)  
Stripe -> Mobile App: returns payment\_method\_id  
Mobile App -> Backend: POST /stripe/create-subscription (payment\_method\_id)  
Backend -> Stripe: create customer/attach PM/create subscription  
Stripe -> Backend: returns subscription  
Stripe -> Webhook -> Backend: invoice.paid / payment\_intent.succeeded  
Backend -> Database: update subscription status  
Backend -> Mobile App: push notification / email confirmation

PlantUML (copy into an online PlantUML editor to visualize):  
@startuml  
actor User  
participant "Mobile App" as Mobile  
participant "Backend API" as Backend  
participant "Stripe API" as Stripe  
database DB

User -> Mobile: Register / Start Trial  
Mobile -> Backend: POST /auth/register  
Backend -> DB: create User + set trial\_start/trial\_end  
Backend --> Mobile: 201 Created + JWT

User -> Mobile: Enter payment details  
Mobile -> Stripe: create PaymentMethod (client)  
Stripe --> Mobile: payment\_method\_id

Mobile -> Backend: POST /stripe/create-subscription {payment\_method\_id, plan\_id}  
Backend -> Stripe: create Customer / attach PM / create Subscription (trial\_period\_days=7)  
Stripe --> Backend: subscription object  
Backend -> DB: store subscription info  
Stripe -> Backend: webhook (invoice.paid / payment\_intent.succeeded)  
Backend -> DB: update payment & subscription status  
Backend --> Mobile: notify subscription active  
@enduml

2) Booking Flow (Search events + Book + Confirmation)  
Description: User browses events (momence), books an event, backend validates capacity and confirms booking.

Step-by-step:  
1. Mobile requests GET /momence/events -> Backend proxies to Momence API and returns event list.  
2. User selects event and taps Book.  
3. Mobile sends POST /bookings {user\_id, event\_id} to Backend.  
4. Backend checks capacity (Momence event capacity or internal record) and eligibility (membership/credits/subscription active).  
5. If OK, Backend calls Momence API to create booking or stores booking and marks seat reserved.  
6. Backend sends confirmation to user and updates calendar/booking list.  
7. Backend may send email confirmation and push notifications (reminders).

ASCII Sequence:  
User -> Mobile: view events  
Mobile -> Backend: GET /momence/events  
Backend -> Momence API: GET Events?hostId=6663&token=...  
Momence --> Backend: events list  
Backend --> Mobile: events list  
User -> Mobile: select event -> Book  
Mobile -> Backend: POST /bookings {event\_id}  
Backend -> Momence: check capacity / create booking  
Momence --> Backend: booking confirmed  
Backend -> DB: store booking  
Backend --> Mobile: booking confirmation  
Backend -> Notification Service: schedule reminders

PlantUML:  
@startuml  
actor User  
participant "Mobile App" as Mobile  
participant "Backend API" as Backend  
participant "Momence API" as Momence  
database DB

User -> Mobile: Browse Events  
Mobile -> Backend: GET /momence/events  
Backend -> Momence: GET /Events?hostId=6663&token=...  
Momence --> Backend: events list  
Backend --> Mobile: events list

User -> Mobile: Book Event  
Mobile -> Backend: POST /bookings {event\_id}  
Backend -> DB: check capacity & user eligibility  
alt capacity available  
 Backend -> Momence: POST /Bookings (or internal reservation)  
 Momence --> Backend: confirmation  
 Backend -> DB: store booking  
 Backend --> Mobile: booking confirmation  
else capacity full  
 Backend --> Mobile: error (capacity full)  
end  
@enduml