documentation

Smart Service Booking Management System — Documentation

1) Entities, Relationships, and Fields (DB Lock)

1.1 User

Fields

- `firstName` (String, required)
- `lastName` (String, required)
- `email` (String, required, unique)
- `password` (String, required, minLength 6)
- `role` (enum: `ADMIN | PROVIDER | CUSTOMER`, default `CUSTOMER`)
- `isApproved` (Boolean, default false)
- `createdAt`, `updatedAt` (timestamps)

Registration rules (already implemented)

- Users can register only as `PROVIDER` or `CUSTOMER`
- `PROVIDER` → `isApproved=false`
- `CUSTOMER` → `isApproved=true`

1.2 Service

Collection: `services`

Fields (exactly these, no more / no less)

- `title` (String, required)
- `description` (String, required or optional — pick one and keep consistent; recommended: optional)

- `price` (Number, required)
- `duration` (Number, required) *(store in minutes for simplicity)*
- `providerId` (ObjectId ref `users`, required)
- `createdAt`, `updatedAt` (timestamps)

Relationships

- `User(PROVIDER)` 1 → many `Service`
- Each service belongs to exactly one provider

1.3 Booking

Collection: `bookings`

Fields (exactly these, no more / no less)

- `customerId` (ObjectId ref `users`, required)
- `serviceId` (ObjectId ref `services`, required)
- `providerId` (ObjectId ref `users`, required)
- `date` (Date, required) *(booking date/time)*
- `status` (String, required, enum)
  - Recommended enum (minimal): `PENDING | CONFIRMED | CANCELLED`
- `createdAt`, `updatedAt` (timestamps)

Relationships

- Customer 1 → many bookings
- Service 1 → many bookings
- Provider 1 → many bookings (via providerId)

Consistency rule

- `booking.providerId` **must equal** `service.providerId` at time of booking.

---

2) Exact Build Scope (API + Frontend) — Minimal and matches your flows

A) API to build (exact endpoints)

Auth

- `POST /auth/register`
- `POST /auth/login`
- `GET /auth/me`

Admin Dashboard API

> All admin routes must be protected with requireAuth + requireRole(ADMIN).

1. `GET /admin/users?role=ALL|CUSTOMER|PROVIDER`

- Returns:
  - `_id, firstName, lastName, email, role, isApproved`

1. `PATCH /admin/users/:userId/approval`

- Body: `{ "isApproved": true|false }`
- Only applicable if target user is a PROVIDER

1. `DELETE /admin/users/:userId`

- Deletes provider or customer

Provider Dashboard API

> Protected with requireAuth + requireRole(PROVIDER)
>
> Optional: block access if `isApproved=false` (recommended).

Provider → View My Services

1. `GET /provider/services`

- List all services where `providerId = currentUserId`

Provider → Edit Service (modal save)

1. `PATCH /provider/services/:serviceId`

- Editable fields (only from lock):
  - `title`, `description`, `price`, `duration`

Provider → Delete Service

1. `DELETE /provider/services/:serviceId`

Provider → View All Bookings

1. `GET /provider/bookings`

- List bookings where `providerId = currentUserId`

Provider → Unbook booking

1. `DELETE /provider/bookings/:bookingId`

Customer Dashboard API

Customer → Browse/Search services

1. `GET /services?search=term`

- Search across `title` (and optionally `description` if you want)

Customer → Book service

1. `POST /bookings`

- Body: `{ "serviceId": "...", "date": "ISO_STRING" }`
- Backend sets:
  - `customerId` from token
  - `providerId` from service.providerId
  - `status` default (recommend: `PENDING` or `CONFIRMED`—pick one and lock it)

Customer → View my bookings

1. `GET /bookings/my`

- List bookings where `customerId = currentUserId`

B) Frontend interfaces to build (exact dashboard views)

1) Admin Dashboard UI

One page with:

- Role filter: All / Customers / Providers
- Table with these columns exactly:
  1. Name
  2. Email
  3. My Services column
     - Customer: “No services available”
     - Provider: link to provider’s services page (admin view)
  4. Approved status (providers only)
  5. Actions
     - Update approval (providers only)
     - Delete (customer/provider)

---

2) Provider Dashboard UI

Provider Home screen

- Two buttons:
  - View All Bookings
  - View My Services

View All Bookings screen

- Table of bookings (provider's)
- Action column: Delete/Unbook

View My Services screen

- List provider services
- Click a service → Modal:
  - Show details
  - Editable fields (from lock): title, description, price, duration
  - Buttons: Save, Delete Service

---

3) Customer Dashboard UI

Browse Services screen

- Search bar
- List all services
- Click service → Modal:
  - Show details
  - Book Service button

View My Bookings screen

- List customer bookings

---

3) Implementation Notes (to prevent scope creep)

- **No extra fields** beyond the locked ones above.
- Booking `providerId` is stored because your lock says so.
- Provider approval:
  - Providers exist even if unapproved

- Unapproved providers should be prevented from provider dashboard actions (recommended).
