1-erd.md

■erDiagram
HOTEL ||--o{ ROOM : "has many"
ROOM ||--o{ BOOKING : "has many"

HOTEL {
int HotelId PK
string Name
string Location "optional"
string Description "optional"
}

ROOM {
int RoomId PK
int HotelId FK "→ Hotel.HotelId"
string RoomType "single | double | deluxe"
int Capacity
}

BOOKING {
int BookingId PK
string BookingReference "unique"
int HotelId FK "→ Hotel.HotelId (denormalised)"
int RoomId FK "→ Room.RoomId"
int GuestCount
date StartDate "DateOnly"
date EndDate "DateOnly, exclusive"
}

2-projectstructure.md

■
CODE (SOLUTION) DESIGN:

HotelBookingApi
■■■ Controllers
■■■ Domain
■ ■■■ Hotel.cs
■ ■■■ Room.cs
■ ■■■ Booking.cs
■■■ Persistence
■■■ Services
■■■ DTOs
■■■ Program.cs
■■■ Startup.cs
■■■ appsettings.json

DOMAIN-DRIVEN DESIGN (DDD):

HotelBookingApi/

■■■ Domain/ <-- Core business entities & rules
■ ■■■ Hotel.cs
■ ■■■ Room.cs
■ ■■■ Booking.cs
■■■ DTOs/ <-- Request/Response shapes for API
■ ■■■ BookingRequestDto.cs
■ ■■■ BookingResponseDto.cs
■■■ Persistence/ <-- EF Core DbContext, configurations
■■■ Services/
■■■ Controllers/

DOCUMENTATION:

HotelBookingApi/
Controllers/
Domain/
DTOs/
Persistence/
Configurations/
Services/
Program.cs
Dockerfile
Docs/
*.md <-- Set of Documentation Files

3-restfulapi-endpoints.md

RESTful API Endpoints
(to ensure proper DTOs and mapping to domain models).

Endpoint Verb Description
/api/hotels?name=X GET Search hotel
/api/rooms/available?start=...&end;=...&guests;=N GET Available rooms
/api/bookings POST Book a room
/api/bookings/{reference} GET Retrieve booking
/api/data/seed POST Seed data
/api/data/reset POST Reset database

4-demo-script.md

■FINAL DEMO PLAN:

POST /api/v1/admin/maintenance/seed

GET /api/v1/hotels/search?name=Sun → note hotelId

GET /api/v1/hotels/{hotelId}/rooms/available?roomType=double&guests;=2&startDate;=2030-01-10
&endDate;=2030-01-12

POST /api/v1/bookings (body in README)

GET /api/v1/bookings/{reference}

Re-check availability → one fewer room

(Optional) POST /api/v1/admin/maintenance/reset

5-Test-ideas.md

■CONCRETE TEST IDEAS:
(Map to Business Rules)

Availability

Empty set when all rooms booked; non-empty when at least one free.

Edge: end == start + 1 (1-night stay).

Edge: end == other.StartDate (no overlap).

Edge: start == other.EndDate (no overlap).

Capacity

Reject booking guests > capacity.

Accept booking guests == capacity.

Uniqueness

Booking reference format BKG-XXXXXXXX.

References are unique across many creates (100–1000).

No room switching

Attempt to book across dates that would need a split → ensure the API approach (single room search) prevents this by design.

Validation

All FluentValidation rules return 400 with field errors.

Faults

Missing hotel → 404 (if you expose GET /hotels/{id}).

Invalid room type → 400.

Concurrency (optional advanced)

Parallel bookings on same room/type/dates → exactly one succeeds; others 409/400.

Validators
// BookingRequestValidator
RuleFor(x => x.HotelId).GreaterThan(0);
RuleFor(x => x.RoomType).NotEmpty().Must(t => new[]
{"single","double","deluxe"}.Contains(t.ToLower()));
RuleFor(x => x.GuestCount).GreaterThan(0);
RuleFor(x => x.StartDate).LessThan(x => x.EndDate);

// RoomAvailabilityRequestValidator
RuleFor(x => x.HotelId).GreaterThan(0);
RuleFor(x => x.RoomType).NotEmpty().Must(t => new[]
{"single","double","deluxe"}.Contains(t.ToLower()));
RuleFor(x => x.Guests).GreaterThan(0);
RuleFor(x => x.StartDate).LessThan(x => x.EndDate);

// HotelSearchValidator
RuleFor(x => x.Name).NotEmpty().MinimumLength(2);

6-swagger-test.md

■Run the “happy-path” flow in Swagger

Seed data

POST /api/v1/admin/maintenance/seed

Try it out → Execute → expect 200 OK.

Search hotel

GET /api/v1/hotels/search?name=DMI

Expect 200 OK, note hotelId.

Check availability

GET /api/v1/hotels/{hotelId}/rooms/available

roomType=double, guests=2, startDate=2030-01-10, endDate=2030-01-12

Expect 200 OK with matching rooms.

Create booking

POST /api/v1/bookings

Body:

{
"hotelId": 1,
"roomType": "double",
"guestCount": 2,

"startDate": "2030-01-10",
"endDate": "2030-01-12"
}

Expect 201 Created with a BookingReference.

Get booking

GET /api/v1/bookings/{reference}

Expect 200 OK with booking details.

7-invalid-inputs-test.md

■TRIGGER VALIDATION
(to see FluentValidation at work in Swagger)

INVALID INPUTS WITH ERRRS:

Empty/short hotel search

GET /api/v1/hotels/search?name=s → 400

Error: Name must be at least 2 characters.

Bad availability request

GET /api/v1/hotels/{hotelId}/rooms/available?roomType=vip&guests;=0&startDate;=2030-01-12&endDate;=2030-01-10

400 with errors for RoomType, Guests, and StartDate < EndDate.

Bad booking

POST /api/v1/bookings with guestCount=0 or endDate <= startDate

400 with exactly the messages you wrote in rules.