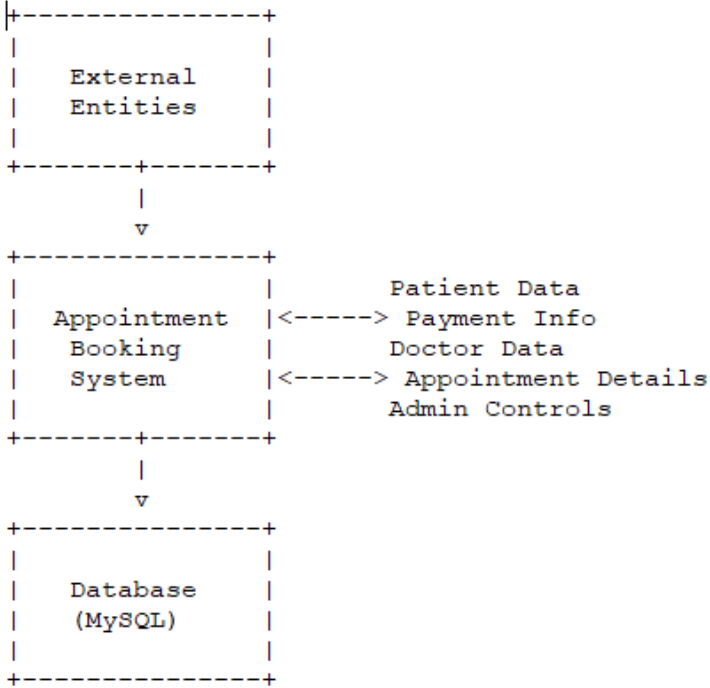


DFD Level 0



Logical Statement :

IF external entity is Patient THEN system must:

- Accept registration data (name, email, password)
- Provide appointment booking interface
- Process payment transactions

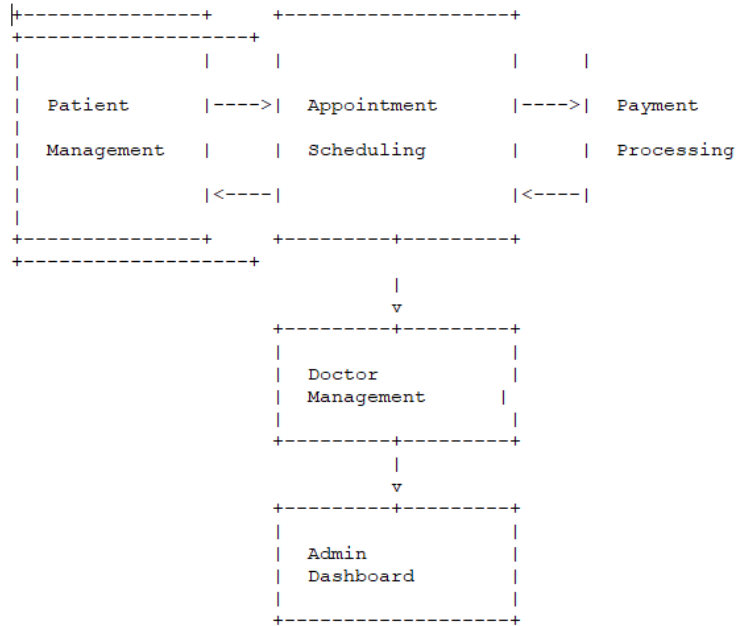
IF external entity is Doctor THEN system must:

- Accept profile information (specialty, schedule, rates)
- Provide appointment management interface
- Display earnings reports

IF external entity is Admin THEN system must:

- Provide user management controls
- Enable system configuration
- Generate analytical reports

DFD Level 1



Logical Statement:

Patient Management:

IF new patient registration THEN:

- Validate email uniqueness
- Hash password
- Create patient record
- Send confirmation email

IF patient login THEN:

- Verify credentials
- Generate session token
- Redirect to dashboard

Appointment Scheduling:

IF patient requests appointment THEN:

- Verify doctor availability
- Check time slot conflict
- Reserve temporary slot (15-min hold)
- Return confirmation prompt

Payment processing

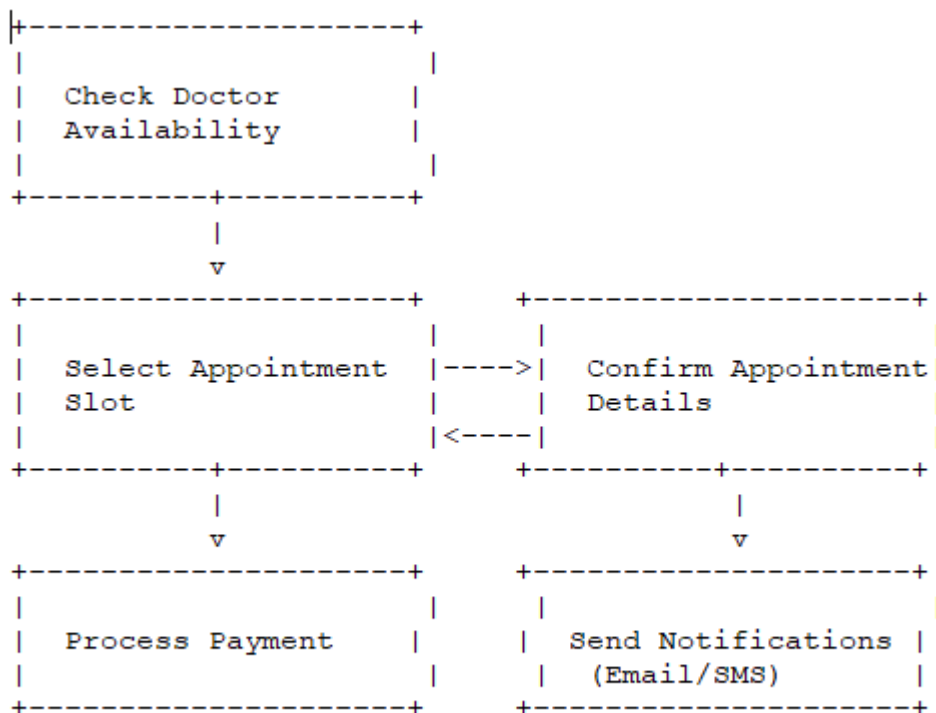
IF payment initiated THEN:

- Validate card details
- Charge amount = doctor_rate + service_fee
- IF payment success THEN:
 - * Confirm appointment
 - * Update doctor earnings
 - * Generate receipt

ELSE:

- * Release time slot
- * Return error message

DFD Level 2



Logical Statement:

Slot availability:

IF payment initiated THEN:

- Validate card details
- Charge amount = doctor_rate + service_fee

- IF payment success THEN:

- * Confirm appointment
- * Update doctor earnings
- * Generate receipt

ELSE:

- * Release time slot
- * Return error message

Payment Validation:

IF payment_amount < doctor.minimum_rate THEN

REJECT with "Below minimum charge"

ELSE IF payment_amount > system.max_limit THEN

FLAG for manual review

ELSE

PROCESS payment

ER Diagram



Logical Statement:

Patient Entity:

CONSTRAINT:

email MUST BE UNIQUE AND VALID FORMAT

phone MUST MATCH `^[+][0-9]{10,15}$` regex

password MUST BE 8+ CHARS WITH 1 SPECIAL CHAR

Appoinment Entity:

CONSTRAINT:

status MUST BE IN ['pending', 'confirmed', 'completed', 'cancelled']

date_time MUST BE FUTURE DATETIME

patient_id AND doctor_id MUST REFERENCE VALID RECORDS

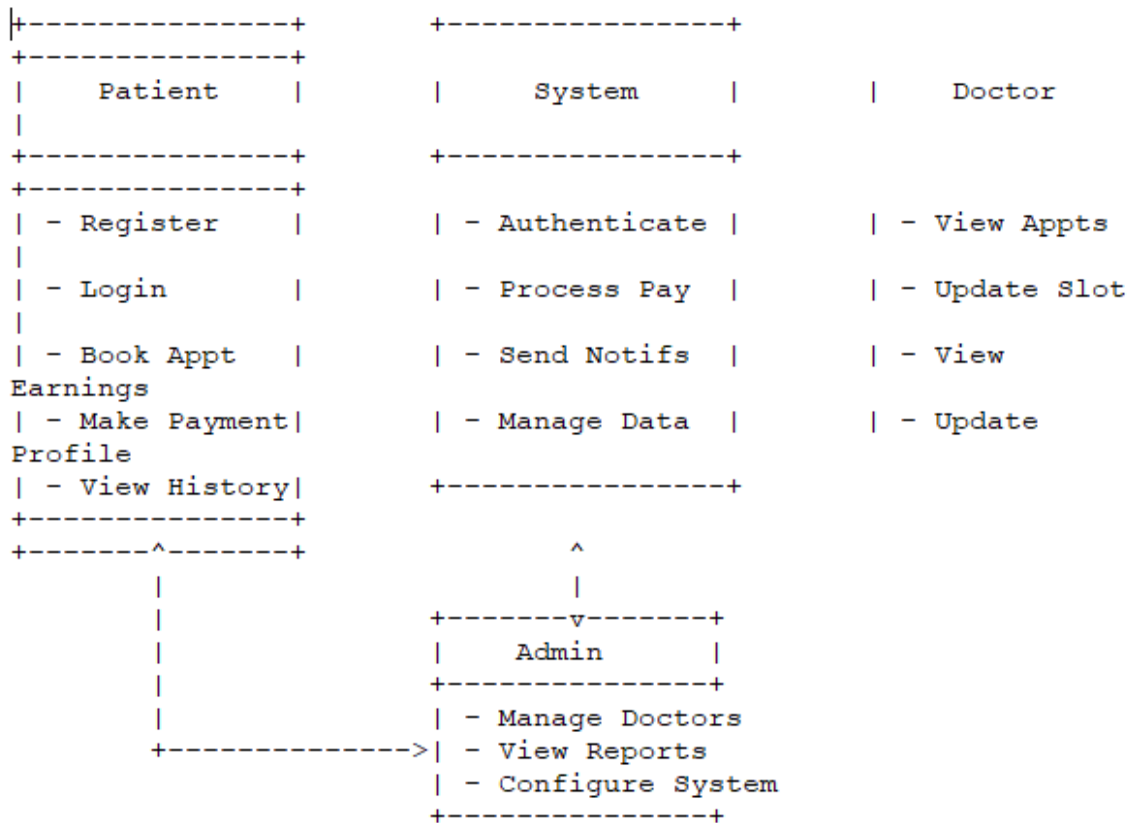
Doctor Patient relationship

FOREACH appointment:

CANCEL IF doctor OR patient account deleted

NO CASCADE ON DELETE (archive instead)

UseCase Diagram



Logical Statement:

Patient book appointment:

PRECONDITION:

- Patient logged in
- Valid payment method on file

POSTCONDITION:

- New appointment created
- Payment recorded
- Notifications sent

EXCEPTIONS:

- IF no available slots THEN suggest alternative dates
- IF payment fails THEN allow 2 retries before locking slot

Doctor view Earnings:

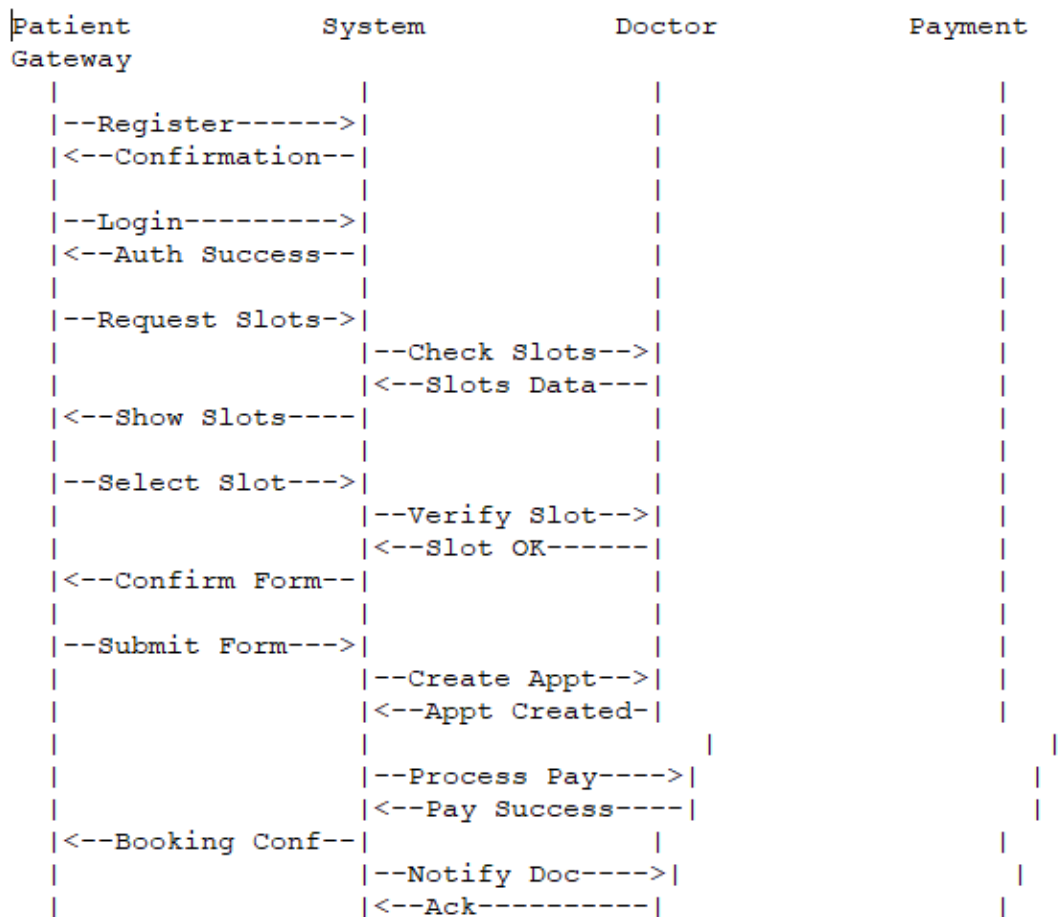
CALCULATION:

total_earnings = SUM(appointments.payment_amount WHERE status=completed)

upcoming_payments = SUM(appointments.payment_amount WHERE status=confirmed)

FILTERABLE BY date_range (default: current month)

Sequence Diagram



Logical Statements:

Successful booking flow:

1. Patient submits credentials → System verifies:

IF credentials_valid THEN proceed

ELSE return error_code 401

2. System checks slots → Doctor's calendar:

IF no conflicts THEN return available_slots[]

ELSE return nearest_available

3. Payment processing:

WHILE attempts < 3 AND payment_status != success:

PROCESS payment

IF gateway_timeout THEN retry_after(5sec)

4. Notification rules:

FOR EACH successful booking:

SEND email AND SMS to patient

PUSH notification to doctor app

IF high_priority_specialty THEN also call admin