Lab 3- Advantis Dental Surgeries System – Domain Model UML Class Diagram and problem Statement.

Summary:

1. System Overview

Advantis Dental Surgeries (ADS) manages a network of dental clinics across multiple cities. The company needs a **web-based information system** to manage its day-to-day operations including registering dentists and patients, scheduling appointments, managing surgery locations, and handling billing activities.

The system serves three main users:

- **Office Manager:** Registers dentists and patients, books and manages appointments, and handles billing.
- **Dentist:** Views scheduled appointments and patient details.
- **Patient:** Requests, cancels, or reschedules appointments and views their assigned dentist and surgery details.

The system enforces key business rules:

1. A dentist cannot have more than **five appointments per week**.
2. A patient with **outstanding unpaid bills** cannot book new appointments.

This software will streamline administrative work, ensure accurate scheduling, and improve communication between ADS staff, dentists, and patients.

2. Entities (Domain Objects)

SN	Entity	Description
1	**OfficeManager**	Represents the administrative user who manages dentists, patients, and appointments.
2	**Dentist**	Represents a dental professional in the ADS network, with unique contact and specialization details.
3	**Patient**	Represents a registered client seeking dental services, with contact info and demographics.
4	**Appointment**	Represents a booking between a patient and a dentist at a specific surgery location, with date and time.
5	**Surgery**	Represents a physical dental surgery location (clinic) with address and phone details.
6	**Bill**	Represents a financial record linked to an appointment, indicating the amount and payment status.

3. Functional Requirements

ID	Requirement Description
FR1	The system shall allow the Office Manager to register new dentists.
FR2	The system shall allow the Office Manager to register new patients.
FR3	The system shall allow patients to request appointments online or via phone.
FR4	The Office Manager shall be able to schedule, update, or cancel appointments.
FR5	The system shall automatically send appointment confirmation emails to patients.
FR6	Dentists shall be able to log in and view all their appointments.
FR7	Patients shall be able to log in to view, cancel, or reschedule appointments.
FR8	The system shall display details of surgery locations for each appointment.
FR9	The system shall generate and store a bill for every appointment.
FR10	The system shall prevent a dentist from being assigned more than five appointments per week.
FR11	The system shall prevent a patient with unpaid bills from booking new appointments.
FR12	The system shall allow the Office Manager to update bill payment statuses.

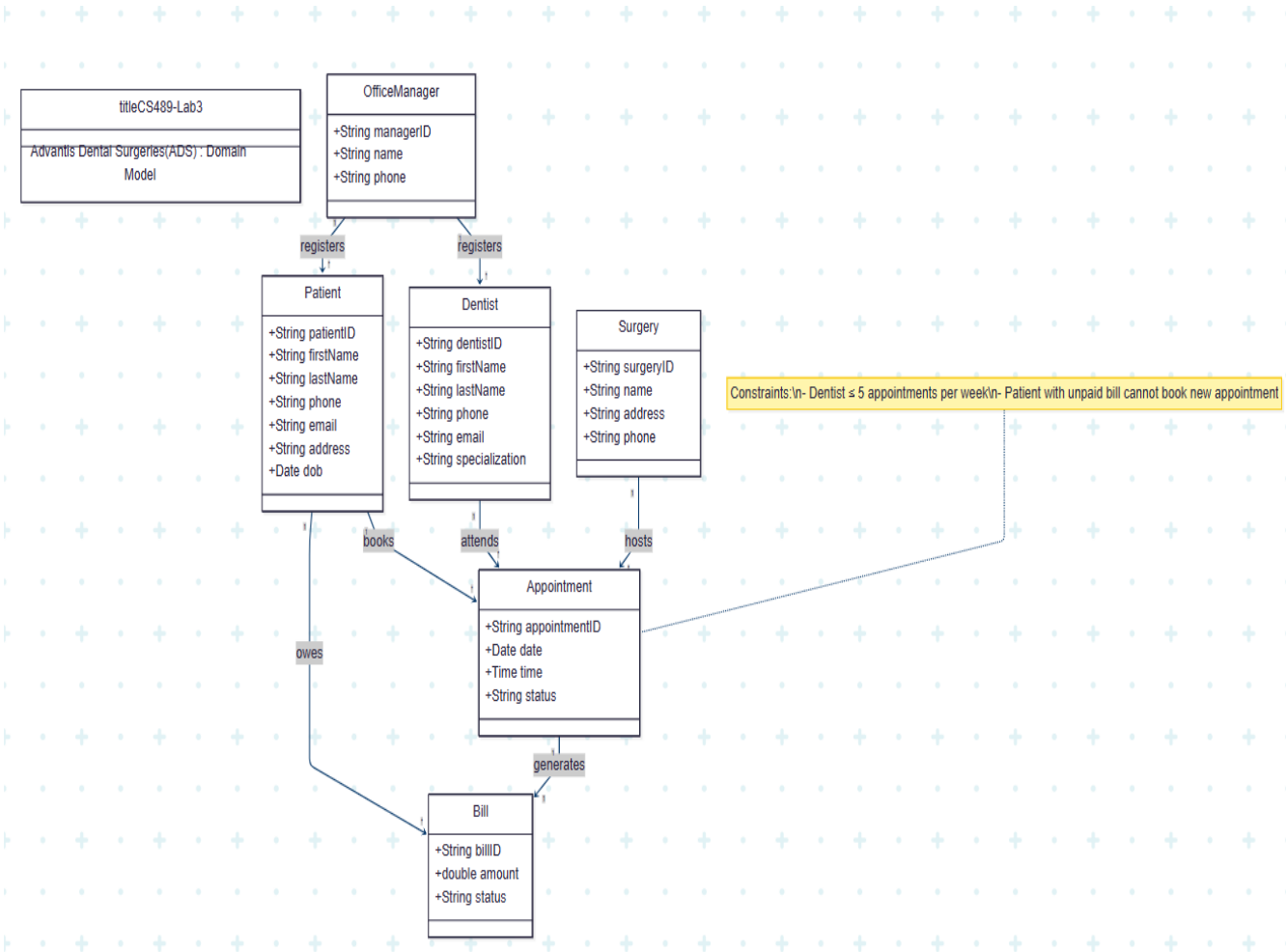
4. Non-Functional Requirements

Category	Requirement Description
Usability	The system shall provide a user-friendly web interface accessible on desktops and tablets.
Performance	The system shall respond to user requests (login, booking, viewing appointments) within 2 seconds.
Security	Only authorized users (manager, dentist, patient) shall access the system using login credentials.
Scalability	The system shall support multiple dental clinics and up to 10,000 users concurrently.
Reliability	The system shall maintain data integrity during concurrent operations and recover from crashes.
Maintainability	The system shall be modular and follow a layered architecture to simplify updates and debugging.
Availability	The system shall maintain 99% uptime during working hours.
Portability	The system shall be deployable on both cloud and on-premise environments.
Auditability	The system shall log all major actions (registrations, bookings, billing updates).

5. Domain Model UML Class Diagram

Diagram Description

- **OfficeManager** registers **Dentists** and **Patients** (1 → *).
- **Patient** books **Appointments** (1 → *).
- **Dentist** attends **Appointments** (1 → *).
- **Surgery** hosts **Appointments** (1 → *).
- Each **Appointment** generates one **Bill** (1 → 1).
- Each **Patient** owes one or more **Bills** (1 → *).
- Constraints:
    - A dentist can have a maximum of 5 appointments per week.
    - A patient with unpaid bills cannot create a new appointment.



6. Entity Relationship Summary

From Entity	To Entity	Relationship Name	Multiplicity	Description
Office Manager	Dentist	registers	1 → *	One manager can register multiple dentists.
Office Manager	Patient	registers	1 → *	One manager can register multiple patients.
Patient	Appointment	books	1 → *	Each patient can book multiple appointments.
Dentist	Appointment	attends	1 → *	Each dentist can attend multiple appointments (⩽5 per week).
Surgery	Appointment	hosts	1 → *	Each surgery can host multiple appointments.
Appointment	Bill	generates	1 → 1	Each appointment generates exactly one bill.
Patient	Bill	owes	1 → *	Each patient can have multiple bills.

7. Use Case

Actor OfficeManager as OM
Actor Dentist as D
Actor Patient as P
Office Manager Actions
(Register Dentist) as UC1
(Register Patient) as UC2
(Book Appointment) as UC3
(Cancel / Reschedule Appointment) as UC4
(Send Confirmation Email) as UC5
(Generate Bill) as UC6
(Update Bill Status) as UC7
Dentist actions
(View Appointments) as UC8
(View Patient Details) as UC9
Patient actions
(Request Appointment) as UC10
(View Appointments) as UC11
(Cancel / Change Appointment) as UC12
(View Bill Status) as UC13

Summary of the use case

Diagram Explanation

Actor	Role in System	Key Use Cases
Office Manager	Manages all core operations — dentist registration, patient onboarding, appointment scheduling, billing, and communication.	Register Dentist, Register Patient, Book Appointment, Send Confirmation Email, Generate Bill, Update Bill Status
Dentist	Views assigned appointments and patient details to prepare for consultations.	View Appointments, View Patient Details
Patient	Interacts with the system to request, view, and manage appointments and bills.	Request Appointment, View Appointments, Cancel/Change Appointment, View Bill Status

8. High-Level Architecture Overview

Microservice-Based Architecture for Advantis Dental Surgeries (ADS)

Each major domain (from your UML) becomes an independent **bounded context** / **microservice**, communicating through REST APIs and Kafka events.

Microservices Breakdown

Microservice	Responsibility	Main Entities	Event Topics
Patient Service	Manage patients, profiles, and billing linkages	Patient, Bill	patient.created, bill.updated
Dentist Service	Manage dentist registration, specialization, and schedules	Dentist	dentist.created, appointment.limit.check
Appointment Service	Handle all appointment creation, updates, cancellations	Appointment	appointment.created, appointment.canceled
Billing Service	Generate, track, and update payment status for appointments	Bill	bill.generated, bill.paid
Surgery Service	Manage surgery locations and schedules	Surgery	surgery.updated
Notification Service	Send appointment confirmation and cancellation emails	(No entity, uses Kafka consumer)	Consumes: appointment.created, appointment.canceled

Microservice Supporting Components

Component	Purpose
API Gateway (Spring Cloud Gateway)	Central entry point for routing requests to backend services (authentication, rate limiting, load balancing).
Kafka Broker	Enables asynchronous, event-driven communication between services.
Resilience4j / Circuit Breaker	Ensures system stability during service failures or high latency.

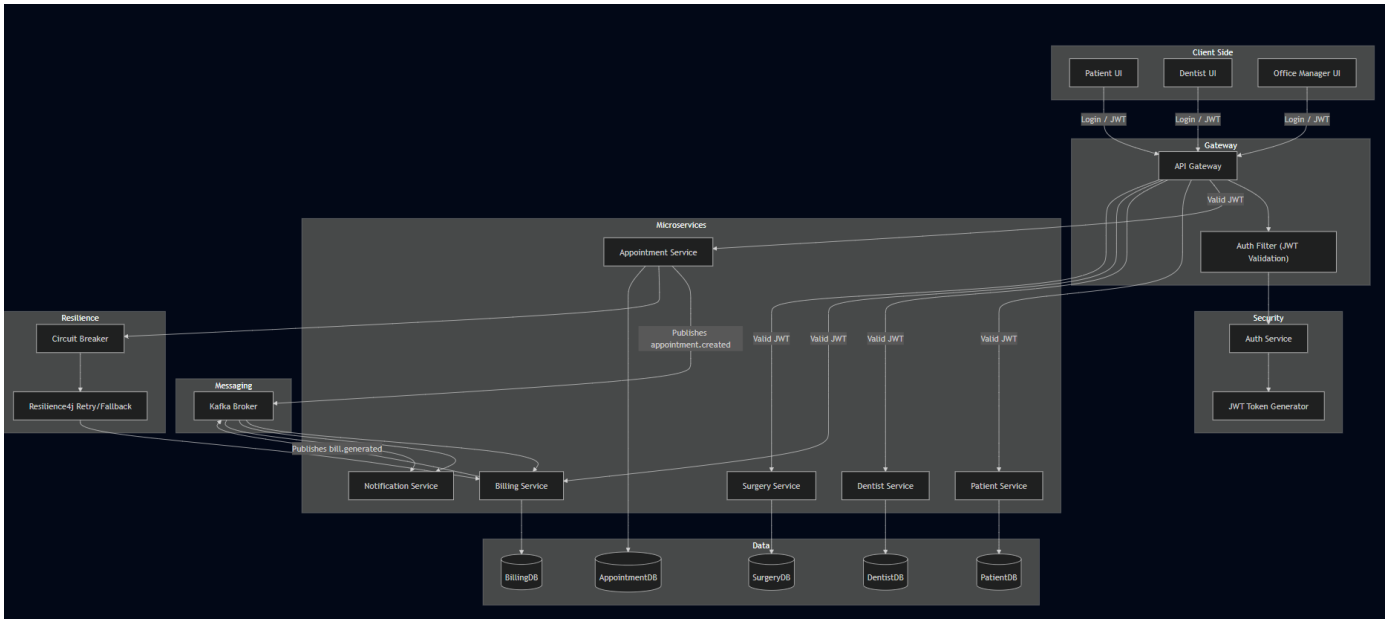
Component	Purpose
Eureka Discovery Server (Optional)	Service registry for dynamic discovery and routing.
Config Server (Optional)	Centralized management of service configurations.

Port Assignment

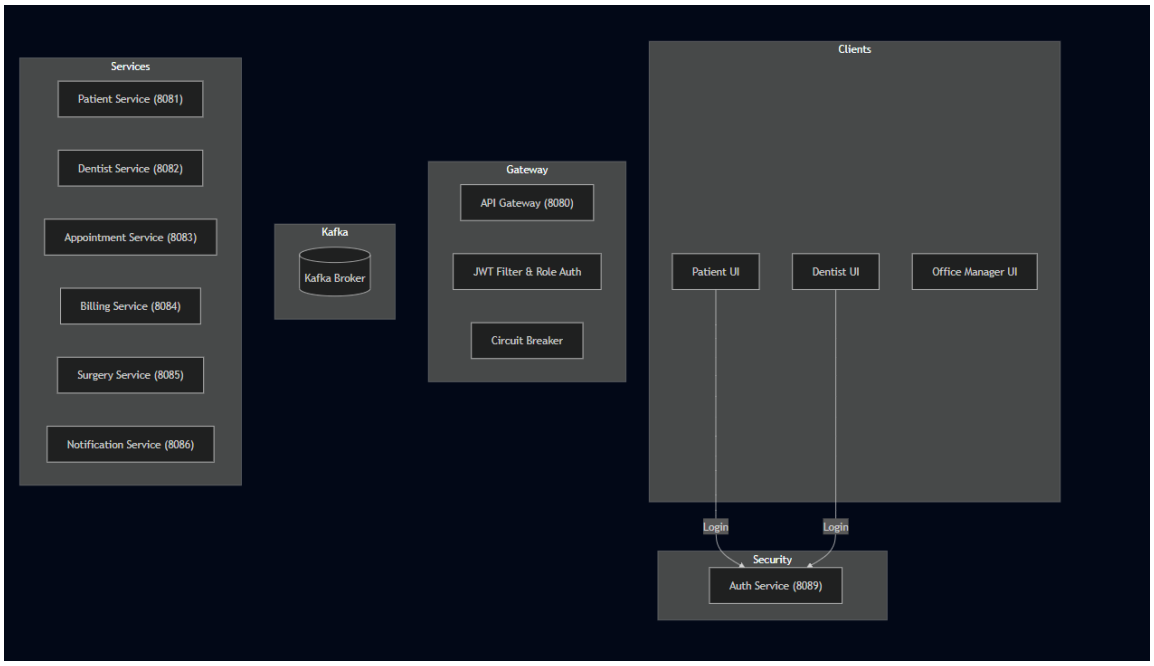
Service	Port	Description
API Gateway	**8080**	Entry point, routing & JWT filter
Patient Service	**8081**	Manages patient and bill linkage
Dentist Service	**8082**	Manages dentist data & schedule
Appointment Service	**8083**	Manages appointments
Billing Service	**8084**	Handles billing generation/payment
Surgery Service	**8085**	Manages surgery locations
Notification Service	**8086**	Sends emails via Kafka events
Auth Service	**8089**	JWT login/signup authentication
Kafka Broker	**9092**	Message broker for events

Scalable System Architecture Diagram

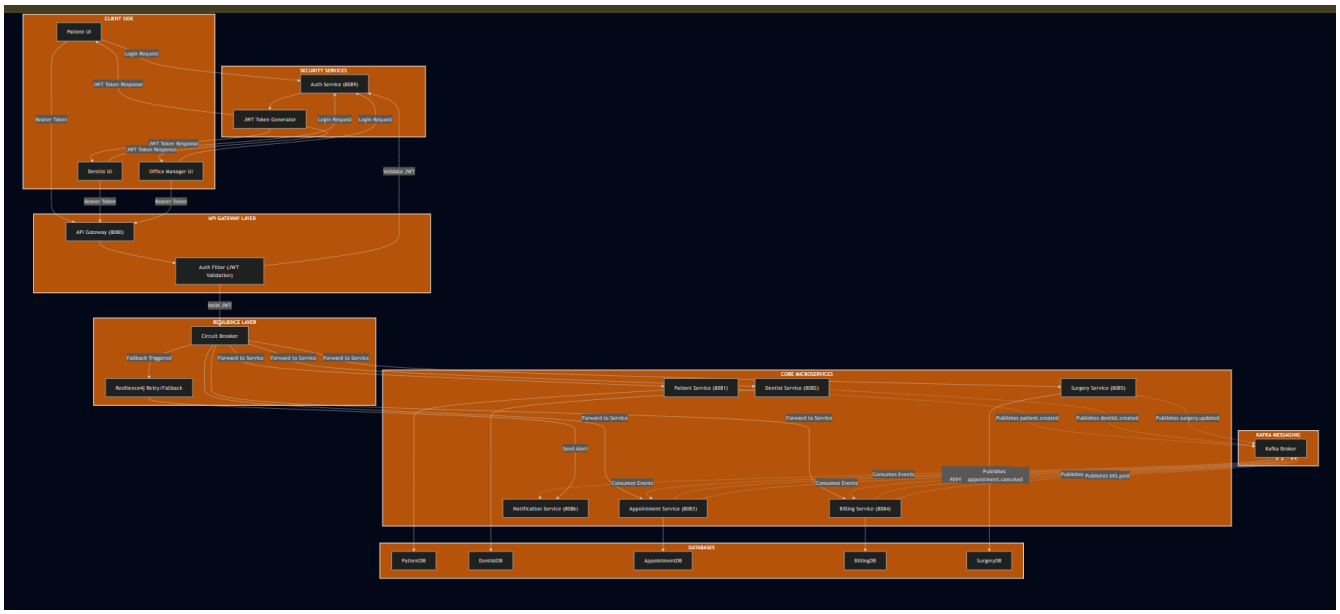
Without Security



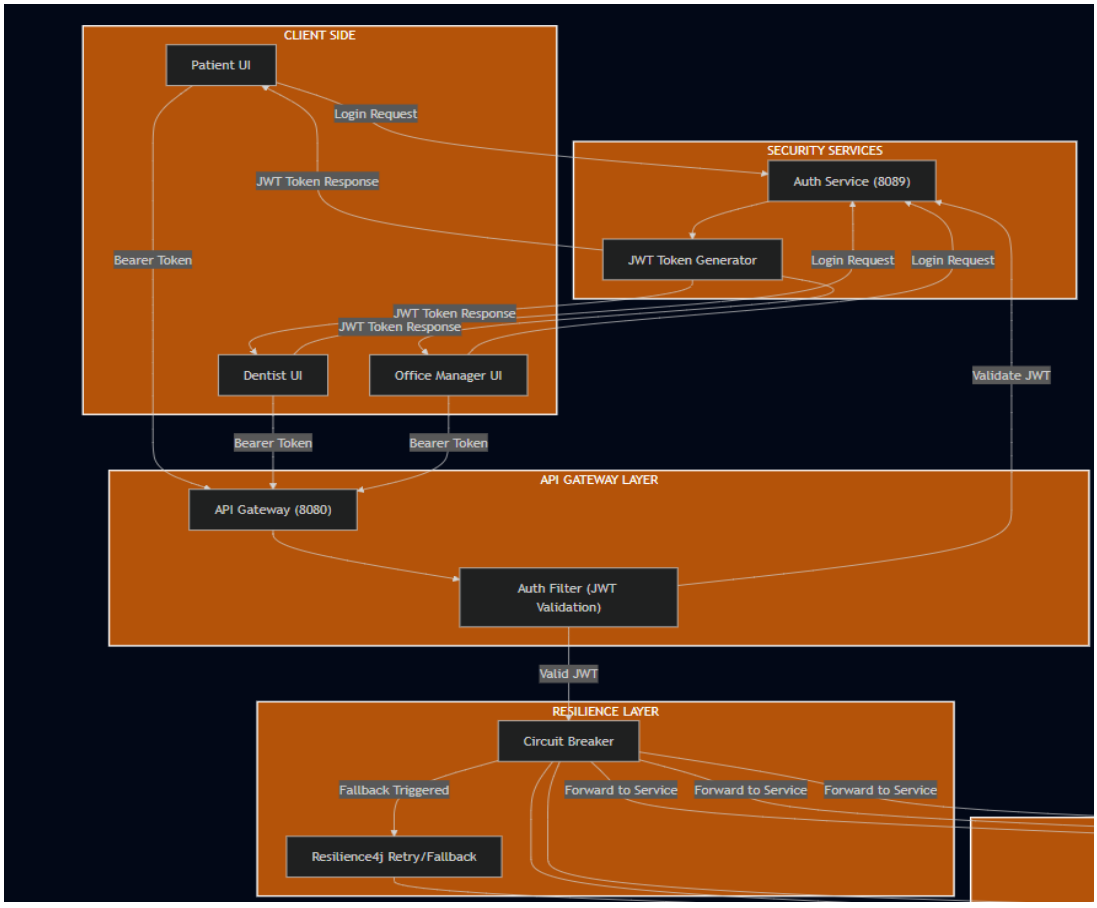
Break down with security



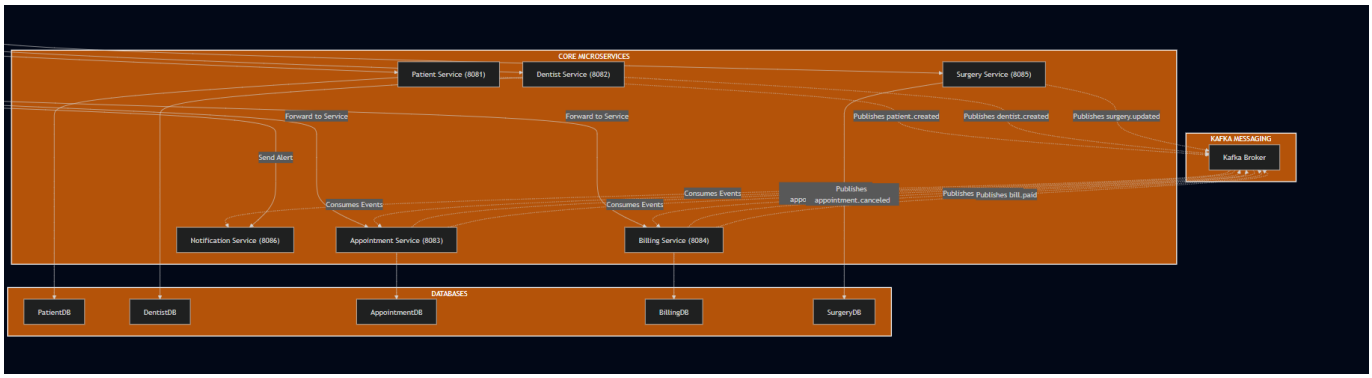
Overall Architecture



Client side



MicroServices side


CORE MICROSERVICES
Patient Service (8081)
Dentist Service (8082)
Surgery Service (8085)
Forward to Service
Forward to Service
Publishes patient.created
Publishes dentist.created
Publishes surgery.updated
KAFKA MESSAGING
Kafka Broker
Send Alert
Consumes Events
Consumes Events
Consumes Events
Publishes appo appointment.canceled
Publishes
Publishes bill.paid
Notification Service (8086)
Appointment Service (8083)
Billing Service (8084)
DATABASES
PatientDB
DentistDB
AppointmentDB
BillingDB
SurgeryDB