PART I: Software Architecture Style Selection

Chosen Software Architecture Style: Microservices Architecture

---

A. Justification Based on Component Granularity

__Definition of Microservices Architecture__

- System is divided into **independently deployable services**
- Each service handles a **specific business capability**
- Services communicate via **APIs (REST/HTTP)**

__Granularity of AssistX Components__

AssistX is logically divided into the following independent services:

1. **User Management Service**
    - Handles user registration, login, authentication
    - Manages roles (Customer / Agent / Admin)
2. **AI Query Processing Service**
    - Processes user queries
    - NLP-based intent detection
    - Generates automated responses
3. **Ticket Management Service**
    - Creates, updates, assigns support tickets
    - Tracks ticket status (Open / Pending / Resolved)
4. **Escalation Service**
    - Transfers unresolved tickets to human agents
    - Handles priority-based routing
5. **Notification Service**
    - Sends email/SMS updates
    - Alerts agents about new tickets
6. **Database Layer (Per Service or Shared DB)**
    - Stores tickets, user data, logs

    - Can be independently scaled

Why This Matches Microservices

- Each module performs a **single business function**
- Services can be:
    - Developed independently
    - Tested independently
    - Deployed independently
- Communication occurs through:
    - REST APIs
    - JSON data exchange

Thus, AssistX components are **fine-grained and loosely coupled**, which aligns with **Microservices Architecture**.