SSRMS UML Planning and Backend Architecture

Planning with UML Diagrams

UML diagrams are excellent tools for visualising the system structure and behaviour before writing code. Based on the sources, here are key diagrams you should consider creating:

1. Class Diagram:

- Represents the static structure of the system, focusing on the data stored in the Database Layer and the core entities involved in the Application and Service layers.
- Key classes to include would directly map to the data stored:
  * User (perhaps with subclasses or roles like ShopOwner, Customer, Supplier, Official)
  * Shop (linked to ShopOwner and potentially Customer via reviews/complaints, stores Verification Status)
  * Supplier (linked to Order and ShopOwner, stores Verification Status and Reputation Score)
  * Review (linked to Customer and Shop)
  * Rating (linked to Customer and Shop)
  * Complaint (linked to Customer and Shop, includes image uploads, processed by AI)
  * Order (linked to ShopOwner and Supplier, includes Products)
  * Product (linked to Supplier and Order)
- Define attributes for each class based on the information provided (e.g., Shop might have name, location, registration status; Complaint might have description, date, status, image URL).
- Show relationships between these classes. Include multiplicities (e.g., one shop can have many reviews).

2. Sequence Diagrams:

- Useful for illustrating dynamic behaviour, showing interaction between objects in specific use

cases.

- Create sequence diagrams for key backend workflows:

  * Shop Registration

  * Submitting a Complaint

  * Shop Owner Placing an Order

  * Government Official Approving a Registration

- These help clarify order of operations, method calls, and data flow.

3. Component Diagram:

- Illustrates physical components of your backend system and their relationships.

- Represent the different layers as components (Application, Service, API, Data Access, Database, Security).

- Include external components: AI Complaint Filtering microservice, Image Storage, Notification Services.

- This helps understand overall structure and dependencies.

Structured Backend Planning Approach:

1. Deep Dive into Requirements

2. Data Model Design

3. API Design

4. Business Logic Specification

5. Service Implementation Plan

6. Security Implementation Plan

7. Integration Planning

8. Technology Choice & Setup

9. Deployment Strategy