Here's a structured overview of the code architecture, covering the main components, their roles, and interactions:

1. Core Components

- Controller/Handler: Manages incoming requests and routes them to the appropriate services. Responsible for coordinating responses to clients.

- Service Layer: Contains the core business logic. Services perform processing, enforce business rules, and handle data validation before interacting with the data layer.

- Repository/Data Access Layer: Manages interactions with the database or data sources. This layer abstracts data operations (e.g., CRUD operations) to ensure consistency and encapsulate database logic.

- Models: Define data structures that represent entities within the system (e.g., User, Order). Models are often mapped to database schemas or external API structures.

- Utilities/Helpers: Provide reusable functions for tasks like logging, error handling, or data transformation. These functions are often generic and can be called across multiple components.

2. Interactions and Data Flow

- Request Handling: The Controller receives incoming requests, performs initial validation, and passes the request to the Service Layer.

- Business Logic Processing: The Service Layer performs business logic on the request data and, if necessary, calls the Repository to interact with the database.

- Data Storage/Retrieval: The Repository executes database queries, retrieves or stores data, and returns results to the Service Layer.

- Response Composition: The Service Layer processes retrieved data, applies business rules, and returns the data to the Controller.

- Response Dispatch: The Controller formats the final response and sends it back to the client.

3. Additional Components

- Middleware (if applicable): Handles cross-cutting concerns like authentication, logging, and request/response transformation before reaching the main processing flow.

- Error Handling: Ensures proper exception management, logging, and returns user-friendly error messages without exposing internal system details.

- Configuration: Centralizes environment-specific settings like database credentials, API keys, and other configurations.

4. Key Interactions

- Synchronous Calls: Services call repositories and other services synchronously to ensure immediate data processing and response.

- Asynchronous Processing (if applicable): For tasks that don’t require an immediate response, such as notifications or background jobs.

- API Interactions: In systems requiring external API calls, a separate API Client component encapsulates and handles all external interactions.

This architecture provides a modular, maintainable structure, where each component is independent yet interacts seamlessly to fulfill application requirements.

Output ScreenShots -   
  




