2. Rancang dan jelaskan konsep-konsep Tactical Pattern dari Domain Driven Design dalam rancangan sistem Aplikasi E-Bidder:

1. Entity, Value Object, and Domain Service:

Entity

An Entity is a domain object that has a distinct identity that runs through time and different states.

- **Member**:
  Represents a user of the system who can act as a seller or buyer. Identified by memberId and can create listings, place bids, and ask questions.
- **Listing**:
  Represents an item being auctioned. Identified by listingId, contains item details, auction period, current bids, and related questions.
- **Bid**:
  Represents a bid placed by a member on a listing. Includes bid amount, timestamp, and bidder identity.
- **Question**:
  Represents a question asked by a potential buyer about a listing. Contains the question, an optional answer, and status (open/closed).

Value Object

A Value Object is a domain object that has no identity. It is defined entirely by its attributes and is immutable.

- **AuctionPeriod**:
  Encapsulates the start and end date/time of the auction.
- **PaymentMethod** & **ShippingMethod**:
  Specify how the item can be paid for or delivered.

Domain Service

A Domain Service contains domain logic that doesn't naturally belong to any one entity or value object.

- **AutoBidService**:
  Responsible for managing automatic bidding behavior.
- **AuctionAuditService**:
  Responsible for analyzing and replaying bid history to reconstruct the final state of an auction listing.

2. Aggregate:

An Aggregate is a cluster of domain objects that are treated as a single unit for data changes. One entity acts as the Aggregate Root and controls access to the objects inside the aggregate.

The Listing is the Aggregate Root, it encapsulates and governs access to:

- ItemInfo (title, description, category) –> Value Object
- AuctionPeriod –> Value Object
- PaymentMethod and ShippingMethod –> Value Objects
- Bids –> a list of Bid entities
- Questions –> a list of Question entities

→ Only the Listing entity can determine whether a bid is or whether a question can be asked or answered.

Example:

If currentDate > auctionEndDate -> rejectBid()

3. Factory:

A Factory is used to encapsulate the creation logic of complex aggregates. Instead of having client code manually instantiate and assemble objects, the factory handles it and ensures that all business rules and invariants are enforced during creation.

ListingFactory is responsible for creating a valid Listing aggregate.

Example:

ListingFactory.createListing(member, itemInfo, auctionPeriod, paymentMethod, shippingMethod)

→ ListingFactory checks that the startDate < endDate, verifies that required payment and shipping methods are provided, and initializes bid history and question list to empty.

4. Repository

A Repository provides access to aggregates as if they were in memory, abstracting away the data source. It manages the persistence and retrieval of aggregate roots.

- ListingRepository
- BidRepository
- MemberRepository

Example:

ListingRepository.findById(listingId): Listing

ListingRepository.save(listing: Listing)

Repositories only manage aggregate roots. Internal components (Bid or Question) are accessed through their parent Listing.

5. Domain Event:

A Domain Event is an object that captures something that has happened in the domain and is important to domain experts. They help decouple parts of the system and allow other components to react to changes.

- BidPlaced
- AuctionEnded
- QuestionAsked
- QuestionAnswered

Each event includes:

- Timestamp
- Relevant data
- Reference to the affected entity

Use Cases:

- Send notifications to sellers when a new bid is placed.
- Update analytics or logs.
- Trigger auction closing when the end time is reached.

6. Event Sourcing:

Event Sourcing is a pattern where state is not stored directly, but derived by replaying a sequence of domain events.

- BidPlaced events
- AuctionEnded events

Example:

1. BidPlaced(memberId=101, amount=500)

2. BidPlaced(memberId=102, amount=600)

3. QuestionAsked(memberId=103, message="Is it new?")

...

(Replay these to get the current state of the listing)