Lab 2: Iterative Requirements & Domain Modeling

Objective

The primary goal of this lab is to apply iterative techniques for requirements gathering and domain modeling, leveraging Unified Modeling Language (UML) and Craig Larman's principles. This involves identifying domain concepts and refining them into a cohesive domain model through class diagrams.

Domain Context: Online Auction System

For this lab, we will focus on the domain of **Online Auction Systems**. This domain offers a complex interplay of roles, interactions, and rules, making it suitable for exploring advanced modeling techniques.

Tasks

1. Problem Statement

Develop a domain model for an Online Auction System that facilitates:

- **Auction Listings**: Sellers list items for auction, specifying details like minimum bid, duration, and item description.
- **Bidding Process**: Registered users place bids on active auctions.
- **Payment and Fulfillment**: Winning bidders complete transactions, and sellers arrange item delivery.
- **User Management**: Registration, role assignment (buyer/seller), and account management.

2. Requirements Gathering (Role-Played Stakeholder Interviews)

- Conduct role-play interviews with peers assuming the roles of buyers, sellers, and auction platform administrators.
- Elicit functional and non-functional requirements, focusing on system constraints, user interactions, and workflows.
- Prioritize requirements using MoSCoW (Must Have, Should Have, Could Have, Won't Have) analysis.

3. Domain Modeling Using UML Class Diagrams

- **Identify Core Domain Objects**:
  Use Larman's "Conceptual Class Category List" to extract potential classes from the problem statement. Examples include **Auction**, **User**, **Bid**, **Item**, and **Payment**.
- **Define Relationships**:
  - Associations: E.g., a **User** "places" a **Bid** on an **Auction**.
  - Multiplicities: E.g., A **User** may place multiple bids, but each bid belongs to exactly one auction.
  - Generalizations: E.g., Different user types like **Seller** and **Buyer** inherit from a parent **User** class.
- **Apply UML Syntax**:
  - Represent classes with attributes and methods.
  - Illustrate relationships using connectors with appropriate multiplicities and annotations.

4. Refinement Using GRASP Patterns

- Assign responsibilities to classes based on GRASP principles:

    - **Information Expert**: Assign responsibility for bid tracking to the **Auction** class.
    - **Creator**: Assign the creation of **Bid** objects to the **User** class since users initiate bids.
- Simulate dynamic requirement changes (e.g., support for proxy bidding or auction extensions) and adapt the model.

5. Iterative Updates

- Refine the model based on peer feedback.
- Incorporate new requirements or identify gaps in the initial design.
- Maintain traceability of requirements to ensure alignment with the domain model.

Deliverables

1. Domain Model (UML Class Diagram)

A polished class diagram that captures the key concepts, relationships, and behaviors in the Online Auction System domain.

2. List of Key Use Cases

Provide brief descriptions for critical use cases, such as:

- **Create Auction**: A seller lists an item for auction with required details.
- **Place Bid**: A registered buyer places a bid on an active auction.

- **Determine Winner**: The system identifies the highest bidder at the end of the auction duration.
- **Process Payment**: A winning bidder completes payment for the auctioned item.

3. Reflection Report

Document how the requirements evolved during the lab, emphasizing the impact of iterative processes and peer feedback on the final domain model. Include:

- Initial assumptions.
- Key insights from stakeholder interviews.
- Challenges faced during modeling.
- Final changes implemented and their rationale.

Expected Learning Outcomes

By completing this lab, participants will gain:

- Practical experience in iterative requirements gathering and domain modeling.
- A deeper understanding of applying GRASP patterns for responsibility assignment.
- Enhanced skills in adapting domain models to dynamic requirements, aligning with real-world software engineering practices.