Software Engineering- UML Diagrams

# Project Topic (e.g. Online Grocery Store Management System)

Group Members

|  |  |  |
| --- | --- | --- |
| Member Name | Member Surname | Student ID |
| Alican | Kaya | 49429 |
| Ege Bora | Güzel | 58351 |
| İrfan | Bag | 49279 |
| Cenker | Sule | 58566 |
| Efe Berk | Tancı | 59115 |
| Berke | Turk | 58169 |

# Introduction

This document provides a complete set of UML diagrams for the Hotel Booking and Management Platform. It visualizes the system’s functional requirements, static structure, and dynamic behaviors, offering clear guidance for implementation and review:

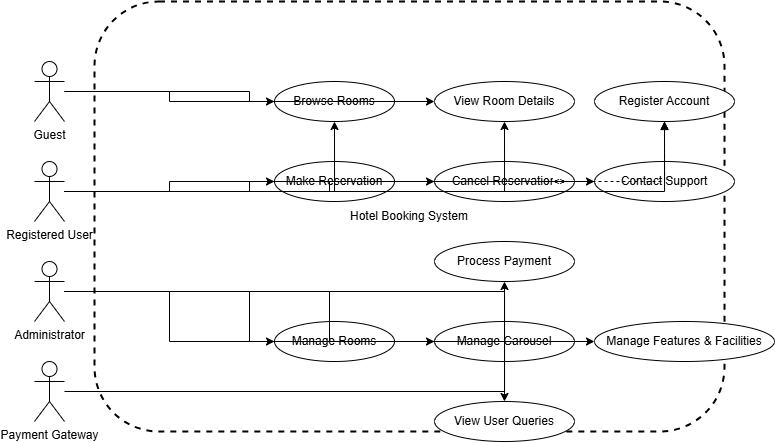
* Use Case Diagrams capture interactions between actors (guests, registered users, administrators, payment gateway) and the system.
* Class Diagram defines the core domain model (User, Room, Reservation, Payment, Feature, Facility, CarouselItem, Query) and their relationships.
* Sequence Diagrams illustrate key scenarios: room search and booking, payment processing, reservation cancellation, and admin operations.
* State Diagram shows the lifecycle of a reservation entity as it moves between states (Available → Pending → Confirmed → Cancelled).
* Activity Diagram outlines the end-to-end reservation workflow from searching rooms through confirmation.

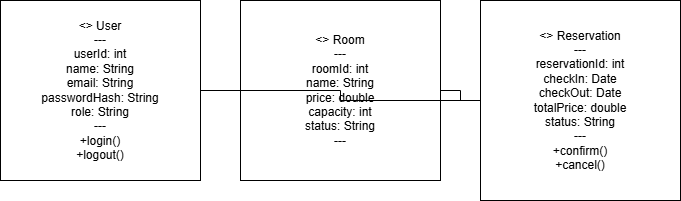
# Use case diagrams (preferably one diagram per actor)

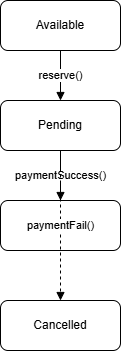
* Actors:
* Guest (anonymous visitor)
* Registered User
* Administrator
* Payment Gateway (external)

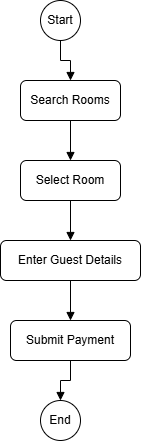
Primary Use Cases:

* Browse Rooms
* View Room Details
* Register Account
* Login / Logout
* Make Reservation
* Cancel Reservation
* Contact Support
* Process Payment
* Manage Rooms (Admin)
* Manage Carousel (Admin)
* Manage Features & Facilities (Admin)









* View User Queries (Admin)

# Class diagram

* Key Classes and Attributes:
* User: userId, name, email, passwordHash, role
* Room: roomId, name, price, capacity, status
* Reservation: reservationId, checkInDate, checkOutDate, totalPrice, status
* Payment: paymentId, amount, paymentDate, status
* Feature, Facility: featureId, title, icon; facilityId, title, icon
* CarouselItem: itemId, imageUrl, caption, order
* UserQuery: queryId, name, email, message, createdAt

Relationships:

* User 1..\* ↔ 0..\* Reservation
* Reservation 1 → 1 Payment
* Room 1..\* ↔ 0..\* Reservation
* Room 1..\* ↔ 0..\* Feature (via RoomFeature)
* Room 1..\* ↔ 0..\* Facility (via RoomFacility)

# Sequence diagrams (one per person in the group)

* Scenarios:

1. Room Booking Flow (Guest → Reservation → Payment):
   * Guest searches rooms → selects room → fills reservation form → invokes payment → receives confirmation.
2. User Registration and Login:
   * New user submits registration → system validates and creates account → user logs in → session established.
3. Reservation Cancellation:
   * Registered User views reservations → selects cancel → system updates reservation status → sends notification.
4. Admin Carousel Management:
   * Admin logs in → navigates to carousel section → adds/edits/deletes carousel items via AJAX.

# State and Activity Diagrams (one of each type per person in the group)

* Entity: Reservation

States & Transitions:

1. Available (initial) → on "Reserve" → Pending
2. Pending → on "Payment Success" → Confirmed
3. Pending → on "Payment Failure" → Cancelled
4. Confirmed → on "Cancel Request" → Cancelled
5. Cancelled → (final)

## Activity Diagram

Workflow: Hotel Reservation Process

1. Start
2. Search Rooms
3. Select Room
4. Enter Guest Details
5. Validate Availability
6. Submit Payment
7. Confirm Reservation
8. Send Confirmation Email
9. End

## Evaluation Criteria: Use Case Diagram (max 8 points)

**1. Content Accuracy (4 points)**

This category assesses the correctness and completeness of the actors, use cases, and their relationships.

* **Actors (1 point)**
  + *Completeness (0.5 points)*: Identify all relevant actors (e.g., users, external systems) that interact with the system.
  + *Accuracy (0.5 points)*: Correctly label actors using appropriate names or roles (e.g., "Customer," "Admin").
* **Use Cases (1.5 points)**
  + *Completeness (0.5 points)*: Include all necessary use cases based on the project’s functional requirements.
  + *Naming (0.5 points)*: Use appropriate, action-oriented names for use cases (e.g., "Place Order," "View Profile").
  + *Descriptions (0.5 points)*: Provide brief descriptions or annotations for each use case, if included, to clarify their purpose.
* **Relationships (1.5 points)**
  + *Associations (0.5 points)*: Correctly associate actors with the use cases they interact with.
  + *Include/Extend (0.5 points)*: Properly use "include" and "extend" relationships where applicable (e.g., for shared or optional behaviors).
  + *Correctness (0.5 points)*: Avoid incorrect or redundant relationships that do not add value or misrepresent interactions.

**2. Diagram Quality (2 points)**

This category evaluates the clarity, organization, and adherence to UML standards in the diagram.

* **Clarity and Readability (1 point)**
  + *Layout (0.5 points)*: Arrange elements logically (e.g., actors on the left, use cases inside the system boundary) to enhance understanding.
  + *Labeling (0.5 points)*: Ensure clear, non-overlapping labels and minimal crossing of lines for better readability.
* **System Boundary (0.5 points)**
  + Clearly define the system scope with a labeled boundary box that contains all use cases.
* **Notation Standards (0.5 points)**
  + Adhere to UML standards for use case diagrams (e.g., stick figures for actors, ovals for use cases, proper arrows for relationships).

**3. Alignment with Project Requirements (1 point)**

This category ensures the diagram accurately reflects the project’s functional requirements.

* **Consistency with Requirements (1 point)**
  + *Coverage (0.5 points)*: Include use cases that address all functional requirements specified in the project.
  + *Relevance (0.5 points)*: Avoid extraneous use cases that do not align with the project’s scope or stakeholder needs.

**4. Timely Submission (1 point)**

* Submit the task by the specified deadline to earn this point. Late submissions, even if high-quality, will not receive this point.

## Evaluation Criteria: Class Diagram (max. 8 points)

The Class Diagram must accurately represent the static structure of the system, showing its classes, attributes, methods, and relationships, while aligning with the previously defined Component Diagram. The evaluation is based on the following categories:

**1. Correctness and Completeness (2 points)**

* **Classes (1 point)**: All necessary classes are identified based on the system's requirements. Each class should have a clear purpose, and no unnecessary or redundant classes should be included.
* **Attributes and Methods (1 point)**: Classes include appropriate attributes and methods that reflect the system's data and operations. Attributes should be relevant to the class's role, and methods should cover key functionalities.

**2. Relationships (2 points)**

* **Identification (1 point)**: Relationships between classes (e.g., associations, aggregations, compositions, inheritance) are correctly identified based on the system's design.
* **Representation (1 point)**: Relationships are properly represented with correct multiplicity (e.g., 1..\*, 0..1), roles, and association names where applicable.

**3. Alignment with Component Diagram (1 point)**

* **Consistency (0.5 points)**: Classes are logically grouped or associated with the components defined in the Component Diagram, reflecting the system's architecture.
* **Structural Integrity (0.5 points)**: The Class Diagram does not introduce elements that contradict the structure or functionality outlined in the Component Diagram.

**4. Clarity and Presentation (1 point)**

* **Naming Conventions (0.5 points)**: Classes, attributes, and methods use clear, meaningful, and consistent names (e.g., "Customer" instead of "Class1").
* **Layout (0.5 points)**: The diagram is organized to avoid overlapping elements and crossing lines, making it easy to read and understand.

**5. Adherence to UML Standards (1 point)**

* **Notation (0.5 points)**: Classes, attributes, and methods follow UML notation (e.g., classes as rectangles with three compartments, attributes as "-age: int", methods as "+getAge(): int").
* **Relationship Types (0.5 points)**: Relationships use correct UML line types (e.g., solid for associations, diamond for aggregations) and stereotypes (e.g., <<interface>>) where applicable.

**6. Timely Submission (1 point)**

* Submit the task by the deadline to earn this point. Late submissions will not receive this point, regardless of quality.

## Evaluation Criteria: Sequence Diagram (max 8 points)

The evaluation is based on the following categories:

**1. Correctness and Completeness (2 points)**

* **Scenario Coverage (1 point)**: The Sequence Diagram must accurately represent a specific use case or scenario from the project. All relevant objects (e.g., actors, system components) and interactions necessary to depict the scenario should be included.
* **Interaction Details (1 point)**: Messages must correctly reflect the sequence of operations, including method calls, returns, and any asynchronous communications. The interaction should be logically complete, covering the scenario from start to finish.

**2. Alignment with Component Diagram (1 point)**

* **Consistency (0.5 points)**: Objects (e.g., lifelines) in the Sequence Diagram must correspond to components or instances of components defined in the previously prepared Component Diagram.
* **Interface Adherence (0.5 points)**: Interactions (e.g., messages) must respect the interfaces and dependencies outlined in the Component Diagram, ensuring architectural consistency.

**3. Clarity and Presentation (1 point)**

* **Organization (0.5 points)**: The diagram should be neatly arranged, with lifelines and messages organized to minimize crossing lines and enhance readability.
* **Labeling (0.5 points)**: All elements, including lifelines and messages, must be clearly labeled with descriptive names or actions (e.g., "User clicks 'Submit'") to ensure the diagram is easy to understand.

**4. Adherence to UML Standards (2 points)**

* **Notation (1 point)**: The diagram must use correct UML notation, including lifelines for objects/actors, solid arrows for synchronous calls, dashed arrows for returns, and proper arrow styles for asynchronous messages.
* **Control Structures (1 point)**: If applicable, control structures such as loops, alternatives (e.g., alt fragments), or parallel processes (e.g., par fragments) must be included and notated accurately.

**5. Timely Submission (1 point)**

* The assignment must be submitted by the deadline to earn this point. Late submissions will not receive this point.

### Additional Requirements and Notes

* **Individual Contribution**: Each group member must create at least one Sequence Diagram, and these diagrams should cover distinct use cases or scenarios. The group is responsible for ensuring compliance, and contributions should be clearly attributed in the submission (e.g., labeled with the creator’s name). Failure to meet this requirement may result in deductions at the instructor’s discretion.
* **Project Relevance**: Diagrams must be tailored to the specific project—generic or unrelated interactions will lead to point deductions.
* **Legibility**: Diagrams should be legible, whether hand-drawn or created with a digital tool. Digital tools are recommended for clarity, but neat hand-drawn submissions are acceptable.
* **Component Diagram Integration**: All interactions and objects should align with the system’s architecture as defined in the Component Diagram.

## Evaluation Criteria for State and Activity Diagrams

The evaluation is based on the following categories:

* **State Diagrams (3 points)**
  + **Technical Accuracy (1 point)**  
    The states and transitions are correctly identified and logically connected, accurately reflecting the behavior of the objects or components being modeled. Diagrams should align with the system’s design as defined in prior work.
  + **Coverage (1 point)**  
    The set of State Diagrams collectively covers all critical objects or components with significant state-dependent behavior, as identified in the Class Diagram and Use Case Diagram.
  + **Presentation and UML Standards (1 point)**  
    Diagrams are clear, well-organized, and use proper UML notation for State Diagrams (e.g., correct symbols for states, transitions, initial/final states). Labels should be concise and readable.
* **Activity Diagrams (3 points)**
  + **Technical Accuracy (1 point)**  
    Activities, decisions, and flows are accurately modeled, correctly representing the system’s processes or workflows as outlined in the Use Case Diagram.
  + **Coverage (1 point)**  
    The set of Activity Diagrams collectively covers all major workflows or processes in the system, as defined in the Use Case Diagram.
  + **Presentation and UML Standards (1 point)**  
    Diagrams are clear, well-organized, and adhere to UML notation standards for Activity Diagrams (e.g., correct use of activity nodes, decision nodes, forks/joins). Labels should be precise and legible.
* **Consistency with Previous Diagrams (1 point)**  
  Both State and Activity Diagrams are consistent with each other and with the previously prepared Component, Use Case, and Class Diagrams. For example:
  + Activities in the Activity Diagrams should correspond to the use cases from the Use Case Diagram.
  + States in the State Diagrams should reflect the behavior of classes defined in the Class Diagram.
  + The diagrams should integrate logically with the system structure outlined in the Component Diagram.
* **Timely Submission (1 point)**  
  The assignment is submitted by the specified deadline. Late submissions will not receive this point.

### Additional Notes

* **Group Contribution**: Each group member must contribute at least one State Diagram and one Activity Diagram. The submission should clearly indicate which diagrams were created by which member to ensure fair participation.
* **Project Relevance**: Diagrams must be specific to the project being modeled. Generic or unrelated content will result in point deductions.
* **Legibility**: Diagrams should be professional and easy to read. Digital tools (e.g., Lucidchart, Draw.io) are recommended, but neat, hand-drawn diagrams are acceptable if scanned clearly.